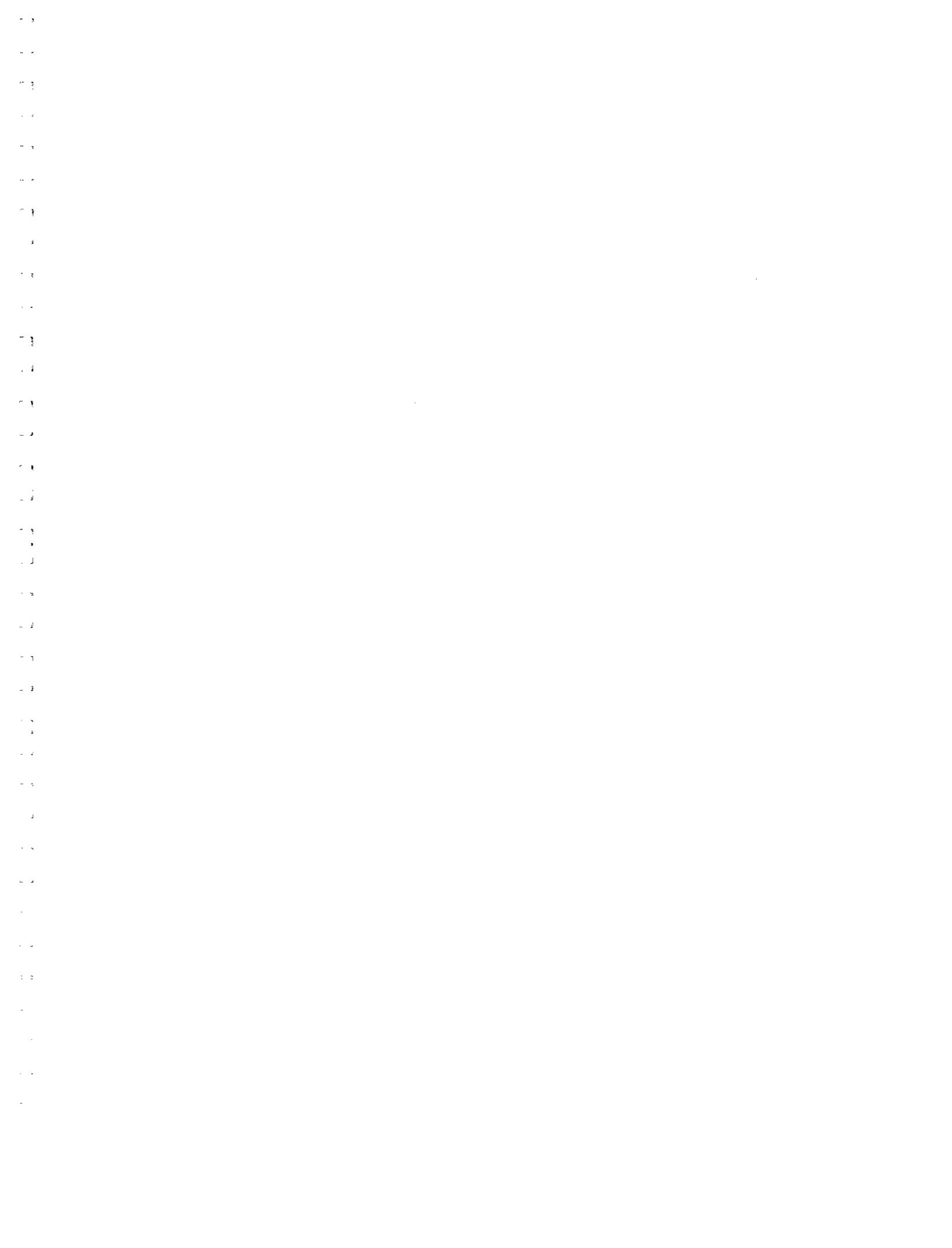


**Multi-resolution Animation and Behavior
in Densely Populated Scenes**

by

Dom Sithikorn Bhuphaibool



**Multi-resolution Animation and Behavior
in Densely Populated Scenes**

by

Dom Sithikorn Bhuphaibool

B.A., SUNY Geneseo, 1995

Thesis

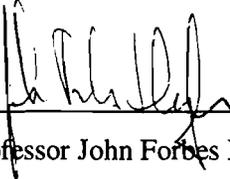
**Submitted in partial fulfillment of the requirements for the Degree of Master of Science
in the Department of Computer Science at Brown University**

Providence, Rhode Island

May 1999

This thesis by Dom Sithikorn Bhuphaibool
is accepted in its present form by the Department of
Computer Science as satisfying the
thesis requirement for the degree of Master of Science

Date April 30, 1999



Professor John Forbes Hughes, Advisor

Approved by the Graduate Council

Date _____

Peder J. Estrup
Dean of Graduate School and Research

Abstract

In computer graphics, we seldom see interactive scenes that contain a large population of characters. For example, the settings for most computer games are indoors where dungeons and rooms provide a physical limitation to the number of characters that can be present. Furthermore, scenes that do incorporate a large population are usually not interactive, because of the inherent computation required to simulate them. This is more apparent as the complexity of each character increases. In addition to the individual complexity, if we allow the characters to interact with each other pairwise, the computation required to simulate them may grow at a quadratic rate with respect to the number of characters in the scene. We have developed multi-resolution techniques in geometry, animation, and behavior. Whereas most multi-resolution research has focused on simplifying geometry, animation, or behavior separately, we allow our system to leverage information from the three different areas. Our research objective is to increase the population threshold of a crowded simulation while maintaining its interactivity. We apply these multi-resolution techniques to a Ballroom Dance simulation where the couples are performing the American Waltz.

Acknowledgements

First and foremost, I would like to thank my advisor John Hughes for his insights, guidance, and invaluable feedback. Thanks to Steve Dollins for helping me choose my thesis topic and for many intriguing discussions. Thanks to Jeff White for his enthusiasm and his relentless effort in propelling me to pursue my thesis. Thanks to Cindy Grimm for her careful review and for helping me in my thesis writing. Lastly, I would like to thank Nancy Pollard for her inspiration, guidance, discussions, and overall support throughout my thesis writing and implementation.

Table of Contents

ABSTRACT	III
ACKNOWLEDGEMENTS	IV
TABLE OF CONTENTS	V
LIST OF TABLES	VIII
LIST OF ILLUSTRATIONS	IX
1.0 INTRODUCTION	1
1.1 MOTIVATION	1
1.2 RESEARCH OBJECTIVE	1
1.3 RESEARCH NOVELTY	2
1.4 IMPLEMENTATION	2
1.5 OVERVIEW	3
2.0 RELATED WORK	5
2.1 LEVELS OF DETAIL IN GEOMETRY	5
2.2 LEVELS OF DETAIL IN ANIMATION	6
2.3 LEVELS OF DETAIL IN BEHAVIOR	7
2.4 COST / BENEFIT HEURISTICS	8
2.5 AUTONOMOUS AGENT SYSTEMS	9
2.6 OUR SYSTEM	10
2.6.1 GEOMETRY	10
2.6.2 ANIMATION	10
2.6.3 BEHAVIOR	11
2.6.4 IMPORTANCE ASSESSMENT	11
2.6.5 SIMULATION OF AUTONOMOUS AGENTS	11
2.6.6 SUMMARY	12
3.0 BALLROOM DANCING	13
3.1 AMERICAN WALTZ	13
4.0 SIMULATION SYSTEM	15

4.1 ARCHITECTURE OVERVIEW	15
4.2 MODEL	17
4.3 ANIMATION.....	20
4.4 BEHAVIOR.....	23
5.0 MULTI-RESOLUTION.....	25
5.1 OVERVIEW	25
5.2 STOCHASTIC EXPERIMENT	25
5.3 MULTI-RESOLUTION GEOMETRY	27
5.4 MULTI-RESOLUTION ANIMATION.....	27
5.4.1 KINEMATIC CHAIN	28
5.4.2 JOINT RANGE CLAMPING.....	29
5.4.3 JOINT ELIMINATION CAVEAT.....	31
5.5 MULTI-RESOLUTION BEHAVIOR.....	32
5.5.1 DANCE BEHAVIOR SIMPLIFICATION.....	33
5.5.1.1 COLLISION AVOIDANCE	33
5.5.1.2 DANCE FLOOR FLOW BEHAVIOR	35
5.5.1.3 DANCE FLOOR CONSTRAINT BEHAVIOR.....	35
5.5.1.4 AMERICAN WALTZ BEHAVIOR	35
5.5.1.5 POLITE FRAME.....	36
5.5.1.6 PROMENADE	36
5.5.1.7 SPIN TURN.....	37
5.5.2 DANCE BEHAVIOR SUMMARY.....	37
6.0 IMPORTANCE ASSESSMENT	39
6.1 FORMULATION OF IMPORTANCE	39
6.2 VISUAL IMPORTANCE	39
6.3 SEMANTIC IMPORTANCE	40
6.4 LOCAL IMPORTANCE.....	40
6.5 GLOBAL IMPORTANCE	40
7.0 RESULTS.....	42

7.1 STATIC ENVIRONMENT	42
7.2 DYNAMIC ENVIRONMENT.....	43
8.0 FUTURE WORK.....	50
8.1 ANIMATION SYSTEM	50
8.2 MULTI-RESOLUTION TECHNIQUES	50
8.3 EXTENDING MULTI-RESOLUTION TECHNIQUES TO OFF-LINE ENGINES.....	51
9.0 CONCLUSION	51
APPENDIX A	54

List of Tables

TABLE 1: GEOMETRIC AND ANIMATION SPECIFICATION OF BASE MODELS.....	18
TABLE 2: SUMMARY OF DANCE BEHAVIORS.....	37
TABLE 3: SIMPLIFYING DIFFERENT TYPES OF BEHAVIORS.....	37
TABLE 4: ATTRIBUTES PERTAINING TO VISUAL IMPORTANCE.....	40
TABLE 5: FRAME RATES OF MULTI-RESOLUTION IN A STATIC ENVIRONMENT.....	42
TABLE 6: PERCENTAGE OF IMPROVEMENT OF MULTI-RESOLUTION IN A STATIC ENVIRONMENT.....	43
TABLE 7: FRAME RATES OF MULTI-RESOLUTION IN A DYNAMIC ENVIRONMENT.....	45
TABLE 8: PERCENTAGE OF IMPROVEMENT OF MULTI-RESOLUTION IN A DYNAMIC ENVIRONMENT.....	45

List of Illustrations

FIGURE 1: A BIRD'S EYE VIEW OF THE DANCE FLOOR.....	3
FIGURE 2: A CLOSE-UP VIEW OF COUPLES PERFORMING THE WALTZ	4
FIGURE 3: ANOTHER BIRD'S EYE VIEW OF THE DANCE FLOOR	4
FIGURE 4: TRAVELLING WITH A LIMITED SELECTION OF STEPS	14
FIGURE 5: SYSTEM ARCHITECTURE	16
FIGURE 6: BASE MODELS AND CROSS CONFIGURATION MODEL	19
FIGURE 7: ANIMATION ARCHITECTURE	20
FIGURE 8: USING INVERSE KINEMATICS TO ASSIST IN PIVOT TURNS	21
FIGURE 9: BEHAVIOR ARCHITECTURE	24
FIGURE 10: GRAPH OF THE RATIO OF CHARACTERS IN THE FOV AND THE RATIO OF CHARACTERS IN HI- RESOLUTION VERSUS LOW-RESOLUTION.....	26
FIGURE 11: GRAPH OF THE NUMBER OF CHARACTERS IN HI-RESOLUTION VERSUS THE NUMBER OF CHARACTERS IN LOW-RESOLUTION.	27
FIGURE 12: JOINT RANGE CLAMPING TECHNIQUES	30
FIGURE 13: COMPUTING FINAL TRAVEL DIRECTION.....	34
FIGURE 14: DANCE FLOOR FLOW AND DANCE CONSTRAINT VECTOR FIELDS.....	38
FIGURE 15: 100 COUPLES STANDING STILL IN HIGH RESOLUTION	46
FIGURE 16: 100 COUPLES STANDING STILL WITH LEVELS OF DETAIL IN GEOMETRY, ANIMATION, AND BEHAVIOR	47
FIGURE 17: 100 COUPLES DANCING IN FULL RESOLUTION.....	48
FIGURE 18: 100 COUPLES DANCING WITH LEVELS OF DETAIL IN GEOMETRY, ANIMATION, AND BEHAVIOR ..	49

1.0 Introduction

1.1 Motivation

In computer graphics, we seldom see interactive scenes that contain a large population of characters. For example, the settings for most computer games are indoors where dungeons and rooms provide a physical limitation to the number of characters that can be present. Furthermore, scenes that do incorporate a large population are usually not interactive, because of the inherent computation required to simulate them. This is more apparent as the complexity of each character increases. In addition to the individual complexity, if we allow the characters to interact with each other pairwise, the computation required to simulate them may grow at a quadratic rate with respect to the number of characters in the scene. To allow more characters in a scene, we can either increase our computational resources or apply multi-resolution techniques to simplify the computation needed to perform our simulation. Since there is always a physical bound to how much computational power is available, it is more practical and beneficial to develop methods to simplify our computation.

We borrow the term *multi-resolution* from the geometric simplification community. The idea of creating levels of detail to simplify computation is not particularly new and was first mentioned by Clark in 1976 [1]. Geometric multi-resolution is achieved by creating hierarchical data structures to represent models of a single object at different resolutions. Lower resolution models require less computation than higher models. When an object is rendered onto the screen, if its lower-resolution representation is visually indistinguishable from its higher-level one, then the lower-resolution model should be used. An example of this is an object that requires several hundreds of polygons to model. If this object is so distant that when projected onto the screen it only occupies a couple of pixels, then we can approximate this object by a few polygons.

1.2 Research Objective

Our research objective is to increase the population threshold of a crowded simulation while maintaining its interactivity. We are interested in virtual environments where characters dominate the computational requirements of the simulation. We define characters to be entities that exhibit geometry, animation, and behavior. The computational load for simulating the virtual environment should, in the ideal case, only be dependent on what is relevant to the current view. Therefore, if the population of the virtual environment increases but the additional characters do not alter the current view in any way, then the computational load should not increase. On the

other hand, if the additional characters do affect the current view, then the computational load of the simulation should only increase in proportion to the importance of the additional characters. We will discuss our definition of importance and its assessment for characters in section 6.0.

1.3 Research Novelty

Character complexity can be broken down into three areas: geometry, animation, and behavior. The geometry of a character describes its physical appearance. The animation describes how characters move their various body parts. The resulting motion may have been pre-computed (motion captured or key-framed) or it may be computed at runtime (e.g., using inverse kinematics to correct a motion). Behavior describes high level goals for characters and may be arbitrarily complex. Some examples of behaviors are avoiding collision while dancing, travelling counterclockwise on a dance floor, and staying within the bounds of the dance floor.

Whereas most multi-resolution techniques have focused on simplifying geometry, animation, or behavior separately, we allow our system to leverage information from the different areas according to our heuristics. We believe that by allowing interaction between these different areas (i.e., geometry, animation, and behavior), we can increase our performance on multi-resolution analysis. We give an example of this by simplifying and approximating the arm movement of a walking character at a distance. Multi-resolution analysis must be initiated in at least one of these areas and we choose to start with the animation. The shoulder joint contributes much more to the hand's displacement and the arm motion than does the elbow joint. As the character travels farther away from the viewer, some motion information may become lost. This is because the projected motion displaces fewer pixels than when the character is closer to the viewer. When the animation of the elbow joint does not visibly alter the overall arm motion, we can eliminate the joint. The geometry level can then leverage from this information and combine the limbs (i.e., upper arm and forearm) into one. The character may have previously possessed a behavior in which he scratches his head each time he turns the same direction four consecutive times. The behavior level may cull this behavior since when the elbow joint is missing, the character can no longer scratch his head.

1.4 Implementation

We implemented a system that integrates multi-resolution techniques from the three areas (i.e., geometry, animation, and behavior). The architecture of this system is discussed in more detail in

section 4.1. We formulated heuristics on when and how to apply these multi-resolution techniques, and applied it to our simulation system in the Ballroom Dancing context. Performance results were gathered by applying different combinations of multi-resolution techniques. We concluded that significant performance increase was achieved. The results of our findings are discussed in further detail in section 7.0.

1.5 Overview

We discuss previous work in section 2 and summarize the differences between our system and previous research systems. In section 3, we describe our chosen application and the motivation in selecting it. Section 4 details our simulation system, its architecture, and our character model. Multi-resolution techniques in the three areas (i.e., geometry, animation, and behavior) are then discussed in section 5. The discussion of importance assessment follows in section 6. Our results are presented in section 7. We then discuss future work in progress and present a conclusion in section 8 and 9 respectively.

Figure 1: A bird's eye view of the dance floor

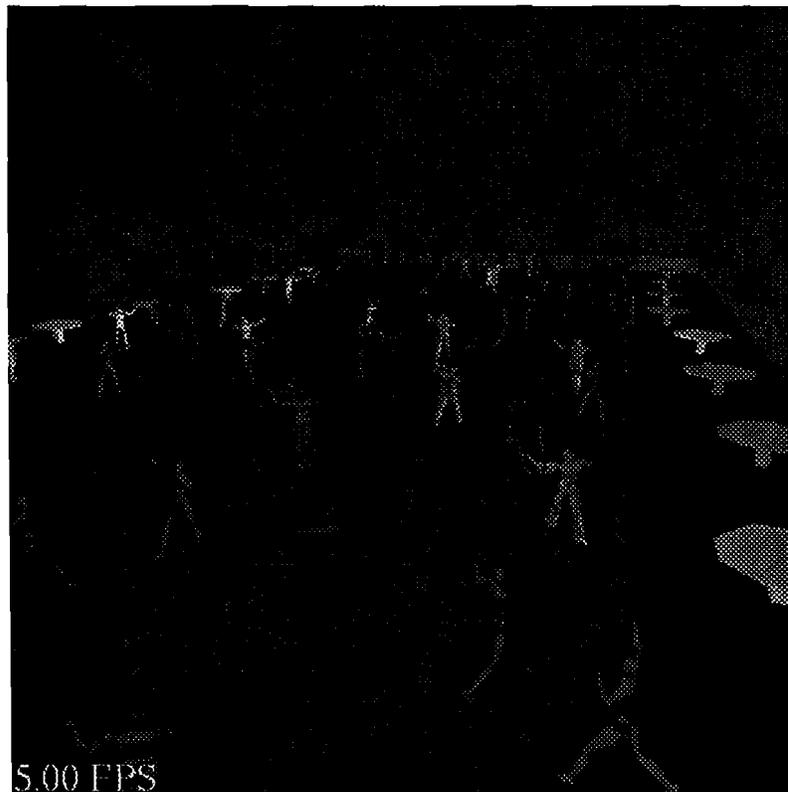


Figure 2: A close-up view of couples performing the Waltz

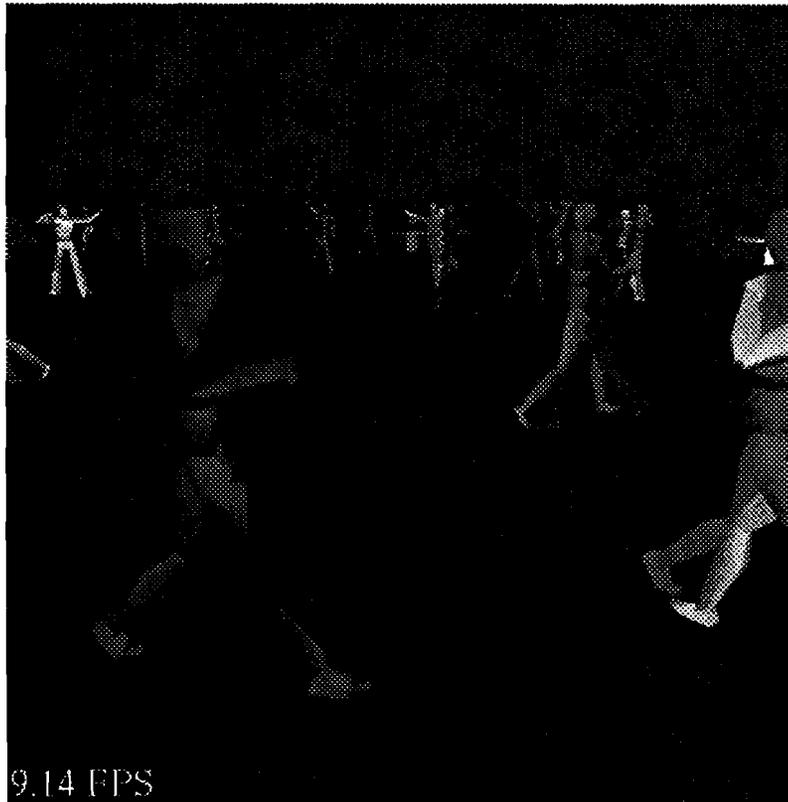
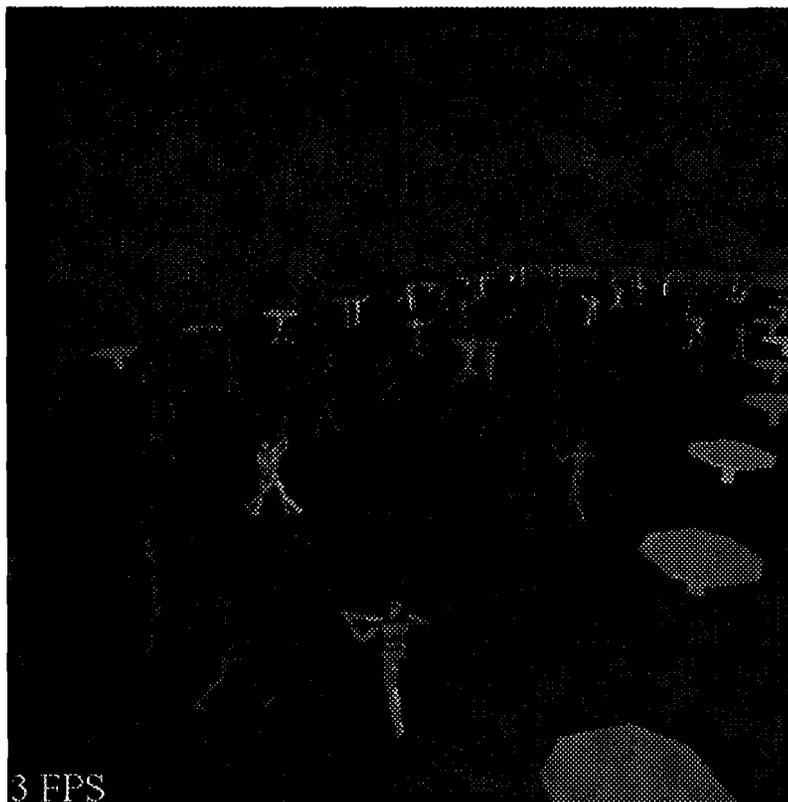


Figure 3: Another bird's eye view of the dance floor



2.0 Related Work

Researchers in the past have concentrated on simplifying geometry, animation, or behavior separately. In this section, we summarize the previous work that has been done in generating levels of details in each area. We also discuss related research where levels of detail were used to achieve a uniform frame rate through a cost/benefit heuristic. In developing strategies to generate levels of detail, we explore different autonomous agent systems to which these simplification methods may apply. We then conclude the section with a discussion of the differences between our system and the previous work.

2.1 Levels of Detail in Geometry

Most of the pioneering works in multi-resolution techniques dealt with simplifying just geometry. Heckbert and Garland present a survey of various techniques for geometric levels of detail [2,3]. The goal behind most geometric simplification techniques is to increase rendering performance by eliminating or approximating computations that contribute to excessive geometric details. We can define excessive details to be geometric details that affect an area smaller than one pixel in screen space. An example of this is an object constructed from several hundreds of polygons, which is so distant that it only takes up a few pixels when projected onto the screen. We can decrease rendering computation time by approximating this object with just a few polygons to generate the same image.

Geometric multi-resolution techniques were developed according to the underlying model they modified. The simplest model is a curve consisting of a sequence of points or line segments. Next in complexity is a mesh model which consists of a network of polygons forming a single, continuous surface. The most complex model is a polyhedral model, which allows arbitrary topology. Different multi-resolution techniques have been developed to leverage certain attributes from each model. Some of these techniques take advantage of volume data [4], while others concentrate on 3D meshes like vertex decimation [5,6], vertex clustering [7], and edge contraction[8,9,10,11]. Each multi-resolution technique has its advantages and disadvantages. For example, edge contraction provides good quality and offers smooth transitions between the different resolution models but is slower than vertex clustering.

motion blur decreases its behavioral influence. This is because the behaviors of a fast moving object are generally less noticeable.

A behavior's predictability may vary according to its motion. Cyclic behaviors are generally more predictable than non-cyclic behaviors. An example of this is a shuttle bus at the airport, which travels from the terminal to the car rental office. The time of its arrival can be roughly predicted provided that we know the duration of each trip and its time of departure. Behavioral predictability is ultimately influenced by time. If the viewer spends too much time away from the behavioral spatial bound, he/she may not have witnessed the departure of the shuttle bus and will be less likely to accurately predict its next time of arrival.

Lastly, each behavior has a semantic importance associated to it, which may be author defined. The behavioral importance is a subjective and context dependent quality. Using the example above, the shuttle bus is much more important than random birds, which may be flying above in the sky.

2.4 Cost / Benefit Heuristics

Funkhouser and Sequin [17] took a completely different approach in applying multi-resolution techniques to complex scenes. Their application was a walkthrough of virtual environments. Whereas most of the previously mentioned work used visual accuracy and perception information to dictate the resolution models of each object in the scene, Funkhouser and Sequin concentrated on maintaining a consistent interactive frame rate. To achieve this goal, they were willing to sacrifice the visual accuracy of the scene. A cost/benefit heuristic was developed to determine the importance of each object and hence its resulting resolution level. An optimization problem was constructed by maximizing the benefit in the scene and subjecting it to a cost constraint, which is defined by the maximum time available to render the scene. Attributes that may affect the benefit value of an object are semantics, focus, motion blur, and hysteresis. Some objects are inherently more important than others, and therefore have a higher benefit value according to their semantic. Focus increases the benefit value of an object if it is closer to the middle of the screen. Objects that are moving quickly may be rendered as motion blur, which allows the benefit value to decrease. Lastly, to avoid hysteresis, each object has a tendency to stay within their current resolution level over successive frames.

2.5 Autonomous Agent Systems

In applying multi-resolution techniques to a simulation in the context of a virtual world, we needed to develop a system that supports autonomous agents. Perlin and Goldberg [18] developed an authoring tool/system called Improv, which is used to create real-time, behavior-based animated actors. Their system supports automatic transition between various motions through motion blending. Action compositing, which is the ability to layer local motions over global motions, is also supported. An example of action compositing is layering a hand waving motion over a walking motion to produce a motion in which a character is walking while waving his/her hand. Perlin et al. used coherent noise to make repetitive motions look realistic. An example of this is the random blinking of the eye, the direction of a wandering gaze, and the weight shifting of a character standing in one spot. One of the attractive features of the Improv system is its support for scripting. Authors can control characters within the scene by creating motion scripts and behavior scripts. Since Improv was designed to be a "Desktop Theater" [23] where the concentration was in directing actors, it does not have a strong support for character interactions. Actors in Improv interact with each other through a shared blackboard, which is cumbersome and slow but provides a standard interface for character interactions. Another drawback of the Improv system is its lack of control and coordination for groups.

Blumberg and Galyean [19] developed a very elegant architecture to support behavior-based autonomous creatures which are directable at three levels: motivational level, task level, and direct motor level. These three levels of directability allow the user to influence the characters at many different conceptual levels. At the motivational level, a user can influence an autonomous agent by suggesting emotions or mental states. At the task level, the user can direct the autonomous agent to perform a specific task. Lastly, at the direct motor level, the user may force the autonomous agent to perform a specific action. One attractive feature of Blumberg's system is the ability to arbitrate conflicting behaviors. Autonomous agents often are performing several behaviors concurrently, and many times it is not possible to satisfy all active behaviors due to their conflicting nature. Therefore, it is important to have a method of arbitration so that execution of behaviors does not oscillate between one another.

These two systems are very similar in concept with subtle differences. Whereas Perlin et al. attempt to make directable characters autonomous, Blumberg et al. attempt to make autonomous creatures directable. Blumberg et al. focused more attention on the behavioral aspect of the

system while Perlin et al. divided their attention equally among the animation and behavioral issues.

2.6 Our System

In this subsection we will discuss the similarities and differences of our system to previous work. We also discuss which aspects of previous research we decided to incorporate into our system.

2.6.1 Geometry

Because our model is constructed from very simple geometry (i.e., very low polygon count), we did not implement any of the sophisticated geometric multi-resolution techniques outlined in the survey by Heckbert and Garland [2,3]. However our system was designed to be easily extended to incorporate these techniques should the need arise. We do apply level of details to geometry by combining polyhedrons, which will be discussed in further detail in section 5.3. We use geometric multi-resolution research as our inspiration and borrow some ideas from this community. In analogy to edge contraction [8,9,10,11], we wish to eliminate joints to simplify the overall motion when possible. In generating levels of detail for animation and behavior, we seek the attractive feature of smooth transitions between the different resolution models, which exists in the edge contraction method.

2.6.2 Animation

In eliminating joints, we allow our model's body parts to be in different resolution levels concurrently. This is a departure from the technique used by Granieri et al. [13]. Due to their model's animation and geometry being tightly coupled, it is more difficult for Granieri et al. to exhibit a smooth transition between models of different resolutions. This is because the transition distance (i.e., the distance before the transition can be made) is governed by the most complex local motion. An example of this is a character walking while frantically waving his right arm. Due to the more complex motion of his right elbow, the distance at which one can switch resolution models for his right arm is farther than that for his left arm. But because Granieri et al. resolution models are static in that all body parts share the same resolution level, they cannot take advantage of being able to switch to lower resolution levels earlier for specific body parts. We exploit this fact (i.e., different body parts can switch to lower resolution levels at different time) by allowing our model to assume different resolution levels among its various body parts. Because of this, we must adapt our motion dynamically and our method of doing so is discussed

in further detail in section 5.4. Granieri et al. precompute lower resolution motions from the highest resolution by mapping and interpolating key points, and then storing the motion information for later playback. One negative characteristic of their motion playback system is that their motions appear to be sterile and synthetic. Since we are already adapting our motion dynamically, we can add coherent noise to the overall motion to rectify this situation.

Although our animation system utilizes key-framed motion, we designed it to be both flexible and extensible so that other forms of motion, such as motion-captured data or dynamic simulation used by Hodgins [12,20,21], can be easily incorporated.

2.6.3 Behavior

We use a similar approach as White in generating levels of detail for behavior. We restrict our class of behaviors to those related to ballroom dancing and identify attributes common to each. We then analyze and generate levels of details specifically for each behavior.

2.6.4 Importance Assessment

Funkhouser et al. used the cost/benefit heuristic to assign geometric resolution models to each object. We can apply the same ideas to determine the resolution levels of animations and behaviors. Our goal differs in that we are not sacrificing image quality to achieve a pre-specified frame rate. We can make a lower bound guarantee of our image quality and we approximate geometry, animation, and behavior to increase the population threshold for a given scene. We also focus on maintaining user interactivity within the simulation.

2.6.5 Simulation of Autonomous Agents

Our simulation architecture is very similar to those of Perlin's Improv system and Blumberg's system. At the animation level, we use forward kinematics to offset characters as their joints move. We control motion by explicitly assigning values to various degrees of freedom of the associated joint. Hence, our motion scripts are very similar to those of Improv. Similar to the Improv system, we apply coherent noise to slightly alter the final motion. We extend the Improv system by applying inverse kinematics to correct certain motions so that they adhere to physical constraints in the environment. An example of this is not allowing the feet to penetrate the floor. We do not support Improv's motion transition nor do we support any sort of mesh deformation to create facial expressions.

Bloomberg's system has a one-to-one correspondence to our system in architectural design. Our animation level is almost identical to Blumberg's direct motor level. Bloomberg et al. further divides their motor system into a Controller layer, Motor Skill layer, and a Degrees of Freedom layer. Our motions and degrees-of-freedom controllers can be directly mapped to Bloomberg's Motor Skill layer and Degrees of Freedom layer respectively.

Bloomberg's task level is equivalent to our behavior level. Although we do not perform any automatic arbitration of behaviors, we allow our behavior system to be easily extensible so that behavior arbitration methods, such as Bloomberg's, can be incorporated into our system.

In the context of autonomous agent system, we extended these systems so that it may support largely populated scenes. We believe that richer stories can be told in the context of densely populated scenes, where lots of dynamic interactions exist.

2.6.6 Summary

In summary, we improved upon some multi-resolution techniques in the context of our Ballroom simulation. We also applied multi-resolution techniques in different areas (i.e. geometry, animation, and behavior) and showed how they can interact with each other. We improved upon recent autonomous agent systems and used it to simulate our ballroom dancers in a virtual environment.

3.0 Ballroom Dancing

To test our multi-resolution techniques we need to develop a simulation with qualities we perceived to be important to our algorithm. We define four criteria that must be fulfilled in choosing our multi-resolution simulation:

- Ability to accommodate an arbitrarily large population.
- Spatial constraint on the population.
- Global movements and lots of animation details.
- High interactions between the characters in the scene.

The first criterion is the ability to accommodate an arbitrarily large population. This is required so that we may test different levels of scene complexity. The second criterion is spatial constraint so that we may make some assumptions about the scene in correlation with the environment. With spatial constraint, we can also control the density of the overall scene. The third criterion is that there must be global movements within the simulation so that the scene remains in a very dynamic state. We define the scene to be dynamic if the characters are continually changing their positions. Visual flow techniques may be used to quantify how dynamic a scene is. The last criterion is interaction. We desired a simulation that requires a great deal of character interaction.

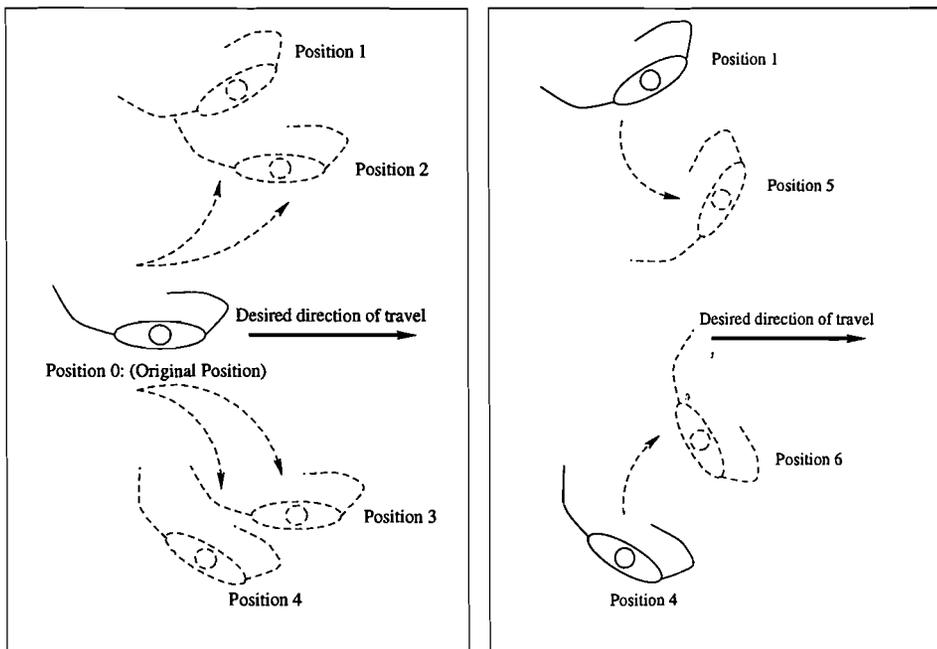
Ballroom dancing fulfills all four requirements. The dance floor may be arbitrary large and hence can hold an arbitrary number of dancers. Characters in the ballroom dancing simulation are constrained to the dance floor, which satisfy our second criterion. Ballroom dancing dictates that couples must travel along the dance floor in a counter clockwise direction. This attribute produces global movements in the scene and hence makes the scene more dynamic and interesting. Interactions among the characters are simulated through collision avoidance.

3.1 American Waltz

We selected the American Waltz above other ballroom dances because it allows the couples enough freedom to navigate the dance floor. We defined navigational freedom by two attributes. The first is *step commitment*, which is defined by the length of time and the distance a couple must travel to execute the chosen step. The second attribute is *directional freedom*, which is the ability to travel in a desired direction by choosing legal sequences of steps. Compared to other ballroom dances, the American Waltz possesses a moderate *step commitment* and a moderate *directional freedom*. In all smooth ballroom dances, travelling to a specific destination on the

dance floor may be a bit more complicated than anticipated. Often, the desired direction of travel cannot be achieved from the current legal selection of steps. Rather, the couple must execute a step to place themselves in an orientation that will allow them to move in the desired direction using the next legal selections of steps as seen in figure 1. This nature of ballroom dancing makes collision avoidance a bit more complicated. One example is if there is only one direction that a couple can travel to avoid collision but that direction cannot be achieved from the available selections of steps. To account for this situation, each couple must perform some sort of look ahead in their path-planning algorithm.

Figure 4: Travelling with a limited selection of steps



Overhead view of a man in dance position without a partner. The circle represents his head while the ellipse represents his body. The bold arrow represents the direction the character wishes to travel. The character is stepping with his left foot first, and therefore has four selections of steps to choose from (Frame 1). Each selection of steps modifies the character's position and orientation. It should be noted that none of the available selections of steps allow the character to travel in the desired direction (i.e., the direction of the bold arrow). Therefore, the character must choose a step which will place him in an orientation so that the next selection of steps will allow the character to travel in the desired direction (Frame 2). With this knowledge in mind, the character can choose either the steps that will place him in Position 1 or Position 4 as shown in Frame 1. From Position 1, the character may then choose a step to allow him to travel in the desired direction while travelling backwards. From Position 4, the character may then choose a step to allow him to travel in the desired direction travelling forward (Frame 2). If our only constraint is to have the character travel in the desired direction regardless of his final orientation, then Position 1 or Position 4 would suffice. But if we constrain his final orientation, then he may have to choose between Position 1 or Position 4 as his first step in the selection process.

4.0 Simulation System

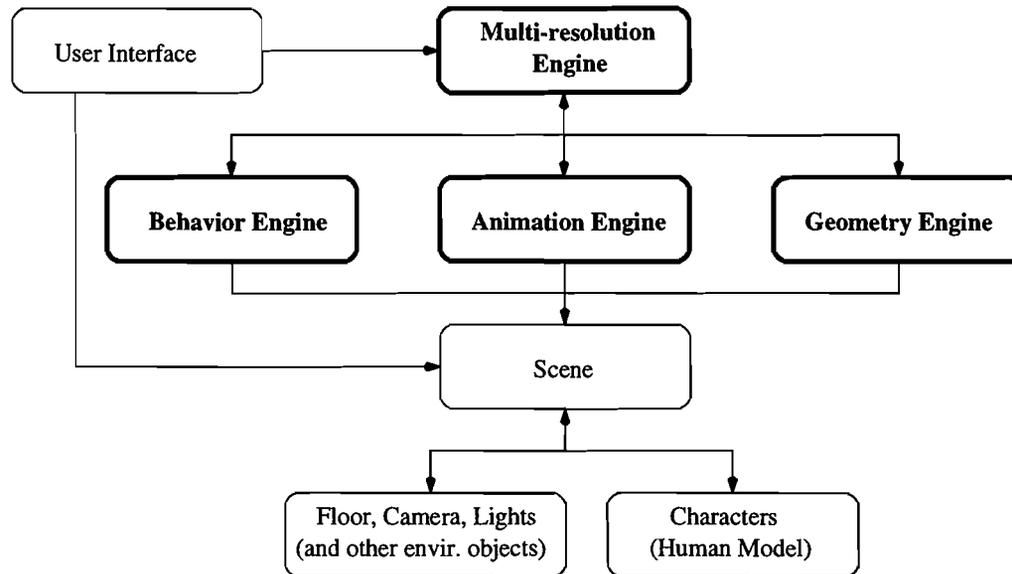
4.1 Architecture Overview

Our architecture consists of four separate and modularized systems:

- Geometry
- Animation
- Behavior
- Multi-resolution

This separation is made both at the software component level and at the system architectural level. At the software component level, these systems were implemented as Java packages. The separate components provide building blocks for the human models. They also provide interfaces to allow different systems to interact with each other. At the system architectural level, execution engines were implemented for each system to maintain the internal states of characters' geometry, animation, and behavior as shown in Figure 2. Each engine may be executed independently and the frequency of its execution may be dependent on the machine's architecture and application's requirements. It is obvious that the Geometry Engine must be executed at every frame (but not more often than the system refresh rate). The Animation Engine and Behavior Engine control how frequently each character's motion and behavior is updated. This may be application-specific and may also be machine-specific. For example, if one's machine possesses a very fast CPU and the behaviors are very CPU intensive, then behaviors can be executed more frequently because the architecture allows more efficient processing.

Figure 5: System Architecture



This figure shows how each engine interacts with other components of the Ballroom Dance simulation. Execution engines are shown in bold. The multi-resolution engine possess the most intelligence and it assigns execution policies to the other engines. The Behavior, Animation, and Geometry engine modifies the scene. The scene consists of characters, floor, lights, and other objects. The user interface can manipulate the scene. One example is changing the character population within a scene. The user interface can also be used to control the Multi-resolution engine.

The functions of each engine are as follows:

- **Geometry Engine** - Responsible for rendering everything in the scene. It provides shapes and transforms which may be manipulated by the Animation Engine over time.
- **Animation Engine** - Manipulates the geometry to produce coordinated motion. Motions are defined by changes in various joints which are further broken down into degrees of freedom. The Animation Engine also provides various noise filters to apply to the current motion.
- **Behavior Engine** - Controls how a character behaves and reacts to the environment. High level goals and tasks are implemented at the behavior level and are executed stochastically.
- **Multi-resolution Engine** - Interacts with all three engines to assign different resolution models for each character in the scene.

4.2 Model

Our highest resolution human model contains 18 joints and 41 degrees of freedom (of which, 6 are the overall position and orientation). The dimensions of each body part are taken from anthropometric data to ensure that the models have correct proportions. Male and female models should also be scaled appropriately relative to one another. From the highest resolution model, we derive two lower resolution models containing 10 joints, 28 degrees of freedom and 6 joints, 24 degrees of freedom respectively as shown in Table 1 and Figure 3. It should be noted that the specified degrees of freedom in the lower resolution models include 6 DOFs that account for the overall position and orientation of the model. We call these three resolution models *base models*.

The base models are further broken down into separate kinematic chains. We define a kinematic chain to be a sequence of rigid links separated by rotational joints. We allow our character to share different resolutions of kinematic chains at any given instance. For example, suppose a character is not utilizing his left arm but making an important gesture with his right arm while slowly walking. This character may have a configuration in which his left arm is in low resolution, his legs in medium resolution, and his right arm in high resolution. We call this cross-configuration and show an example of it in the second portion of Figure 3.

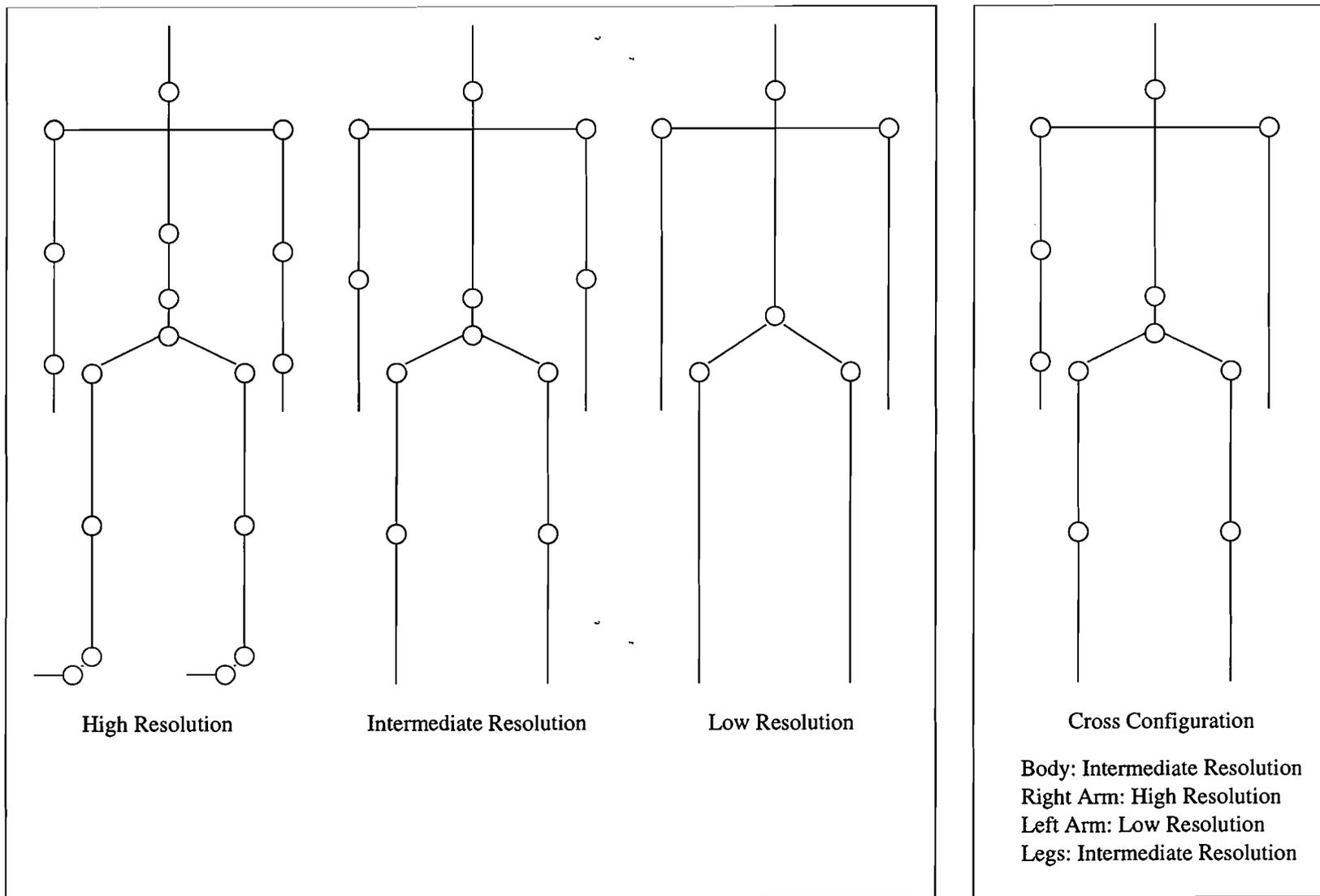
Since the lower resolution models have fewer joints they also have fewer rigid segments connecting the joints. The lengths of these rigid segments vary from one model to another. Because of this, the joints of the lower resolution model corresponding to the ones in the higher resolution models may be in different locations. An example of this is the location of the elbow joint in the high resolution model and the intermediate resolution model as seen in Figure 3. Therefore, we cannot simply reuse the higher resolution motion by applying it to the lower resolution model. Since we also allow different parts of the body to be in different resolution levels, it would not be practical to pre-compute a mapping from the higher resolution model to the lower resolution model as was done by Granieri et al. [13]. This is because the number of possible resolution configurations may grow very fast with respect to the number of body parts and resolution levels allowed. Rather, we adapt the base motion in real-time to the current resolution configurations. We do, however, preprocess the motion to identify which part of it can be simplified and under what conditions. This will be discussed in further detail in the Multi-resolution section (section 5.0).

Table 1: Geometric and Animation Specification of Base Models

	Geometry (polygons)	Joints	Degrees of Freedom	Motion Frequency
High Resolution	158	18	41	30 Hz
Medium Resolution	106	11	32	10 Hz
Low Resolution	66	6	24	3 Hz

- Geometry – The number of polygons used to construct the specified model.
- Joints – The number of joints a specified model possesses.
- Degrees of Freedom – The total number of rotations in X, Y, and Z axes in each joints.
- Motion Frequency – The number of time the animation of the model is updated per second (Hertz).

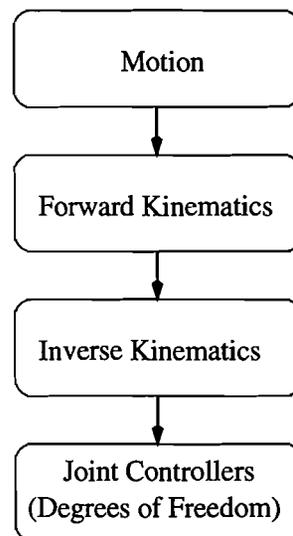
Figure 6: Base Models and Cross Configuration Model



4.3 Animation

The animation architecture is made up of Motions, Forward Kinematics, Inverse Kinematics, and Joint Controllers as seen in figure 4. Motions provide an atomic element for animating characters. We provide an example of a motion script in appendix A. Examples of motions are walking, running, performing an American Waltz Left Basic Step, or getting into dance position by providing your partner with a frame. Motions describe how each rotational joint behaves over time. Not all joints need to be included, but rather a subset of joints can be declared to participate in a motion. Motions can be layered to produce more complex motions. Motions are layered by placing more global motions at the bottom of the motion stack and placing more local motions at the top of the motion stack. Local motions override global motions. An example of this is layering a local hand waving motion over a global walking motion to produce a motion where a character walks while waving his/her hand.

Figure 7: Animation Architecture



Component Layers for a Motion

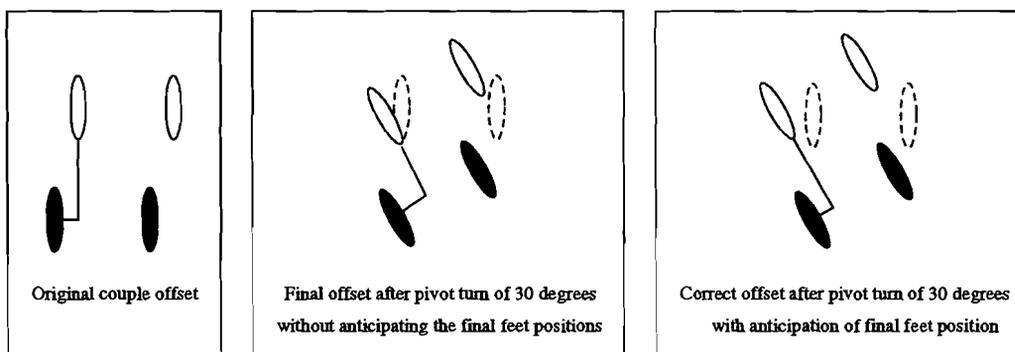
We use forward kinematics to offset the root position as the joints rotate. For example if the right leg is in contact with the floor and the right hip rotates, then the root position will be adjusted so that the right foot remains in contact with the floor at the same exact position prior to the rotation of the hip joint.

Motions animate and manipulate geometry without any knowledge of the environment.

Therefore, below the Motions layer we provide an Inverse Kinematics layer to alter the motion dynamically. The Inverse Kinematics layer places the character's end effectors (i.e. the hands and

feet) at the desired position. This is done by iteratively solving the problem of what joint angles are needed to move the end effector a small amount towards the desired location. An example of a situation which benefits from inverse kinematics is keeping the man's left hand in contact with the woman's right hand while the couples are dancing. Another example where inverse kinematics is useful is when a couple performs a progressive turn where each person is pivoting in a different location. For the couple to line up correctly after the turn, their pivoting feet must initially be placed in the correct relative positions as shown in figure 5.

Figure 8: Using Inverse Kinematics to Assist in Pivot Turns



Overhead view of the foot positions and the relative offset of the woman to the man. The man's feet are in blue while the woman's feet are in yellow. The man and woman are standing facing each other. The line forming a right angle in frame 1 from the man's left foot to the woman's right foot shows the correct offset when the man and woman are in dance position. If the man turned counter-clockwise while pivoting on his left foot and the woman turned counter-clockwise while pivoting on her right foot, the final positional offset of the couple will be incorrect as shown in frame 2. To correct this offset error, we must anticipate how many degrees the couple is turning, and place the woman's right foot in the correct position before pivoting and turning as seen in frame 3.

At the lowest level of the animation system are the joint controllers. Joint controllers provide an abstraction of how joints are allowed to rotate. Certain joints have fewer degrees of freedom than other joints. Key-framed joint angle interval information is also stored at the Joint Controller level. Motion curves can be converted and stored within the Joint Controller level. Joint controllers modify the degree of freedom (DOF) values of the joints they are assigned to. These values are ultimately used to transform and manipulate the geometry, and thus produce the desired coordinated motion over time.

We control our motions through motion scripts, which specify how joint angles change over time and which noise filter each joint use. As mentioned earlier, appendix A contains a sample motion script.

We give an example to illustrate the steps taken in producing a motion. First a motion is instantiated from a motion script. A sample motion may be Right-Leg-Forward motion, which describes how the right hip, knee, ankle, and ball of foot rotate in a prespecified time interval or key frames. This motion is then associated with an action. In our example, the Right-Leg-Forward motion can be associated with a Walk action. When the Walk action is triggered, the joint angles associated with the motion are set according to the current progress in the timeline. Since the right leg is assumed to be free and not anchored to the ground, the forward kinematics layer does not alter the root position. The inverse kinematics layer then adjusts these degrees of freedom so that the model adheres to physical constraints. For example, if the character was walking uphill, his right foot may have penetrated the floor when it was rotated forward. The inverse kinematics layer is responsible for raising the right foot so that it is exactly above the floor. The joint controller layer then takes these degree of freedom values and transforms the associated geometry.

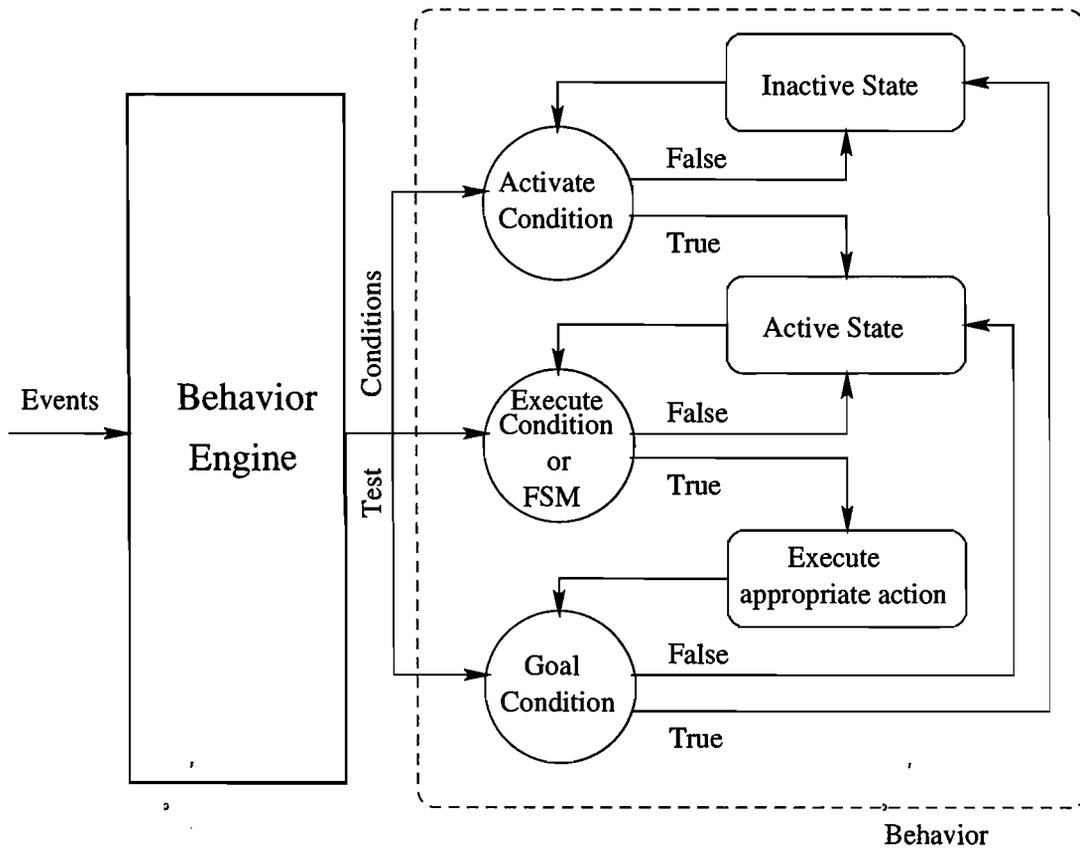
4.4 Behavior

A behavior represents a set of rules a character uses to achieve a high level goal. These goals may be modeled as internal variables or states within the character. Characters satisfy high level goals by executing the corresponding actions, which modify the appropriate internal variables and states. Some examples of behaviors are searching for food, following another person, or dancing the American Waltz. The corresponding motivations behind these behaviors are hunger, attraction, and dancing to music respectively. Behaviors in our system may be added or deleted dynamically.

Figure 6 demonstrates the high level architecture of a behavior and how it changes its states. Behavioral states are depicted by rectangular boxes with rounded corners and conditions are depicted by circles. At any given time, a character's behavior may be either active or inactive. All initially added behaviors assume the inactive state and are considered dormant until an event triggers the Behavior Engine to test the active-condition of the behavior. If the active-condition is satisfied, the behavior then changes to an active state. Once active, the behavior is then executed according to its frequency policy. If the execution-condition is satisfied, the appropriate action is triggered. Complex behaviors may replace the execution-condition with a finite state machine (FSM). The finite state machine controls and manages internal FSM states, which correspond to different actions. Once an action is performed, a goal-condition is tested. This goal-condition represents the ultimate goal of the behavior. If the goal-condition is satisfied, the behavior returns to the inactive state.

We demonstrate the flow of behavioral states in the context of an example of the Floor Constraint behavior. The Floor Constraint behavior first starts out in an inactive state. When the character's distance from the edge of the dance floor falls below a certain value, an event is generated with the character's id and sent to the behavior engine. The behavior engine then activates the Floor Constraint behavior of the corresponding character. Once active, the behavior engine tests the execute-condition of the Floor Constraint behavior. The execute-condition of the Floor Constraint behavior is whether or not the character will continue to travel towards the edge of the dance floor. If so, the behavior performs an action which suggests to the character to select a dance step that will take him/her away from the edge of the dance floor. The goal-condition for this behavior is whether or not the character's distance increases above a certain value. If it does, then the goal-condition is met and the behavior returns to the inactive state. If the goal-condition is not met, then the behavior remains active and its execute-condition is repeatedly tested over time.

Figure 9: Behavior Architecture



5.0 Multi-resolution

5.1 Overview

To motivate our multi-resolution techniques, we first discuss the results of an experiment we performed. Assuming a certain factor of computational savings we computed the field of view half angle in which our resolution techniques will excel to the maximum capacity. We then discuss various techniques we used to generate levels of detail for geometry, animation, and behavior.

5.2 Stochastic Experiment

In applying multi-resolution techniques to our simulation, we performed a stochastic experiment to determine which view angle maximizes the performance gain. Our simulation performance is directly proportional to the number of people in the view frustum. It is also proportional to the ratio of people in high-resolution versus people in low-resolution. Therefore, the product of these two attributes will determine the target view angle.

We used a Monte Carlo method by uniformly distributing 500 people within a unit circle. For each camera half-angle between 5 and 90 degrees with an increment of 5 degrees, we perform 10,000 iterations to determine how many of these people are in view and among those in view, how many of them are in high-resolution versus low-resolution. From running our Ballroom Dance simulation, our heuristic suggests that people at a distance greater than 0.8 unit from the camera should be in low resolution and people at a distance less than 0.8 unit from the camera should be in high resolution. The camera position and orientation are randomly chosen for each one of the 10,000 iterations. We expect that as the camera angle increases, the ratio of the people whom are in view would also increase. On the other hand, of the people whom are in view, the ratio of those in high resolution versus low resolution should decrease as the camera angle increases.

With visual correctness aside, our intent in performing this experiment is to find out at what camera angle would our system benefit the most (i.e., when the two curves intersect). This is shown in figure 7. Although the curves in figure 7 intersect at around 40 degrees half-angle, we would like to take into consideration the number of characters in high resolution versus the number of characters in low resolution. Figure 8 shows a graph of the number of characters within the field of view who are in high resolution and those who are in low resolution for various

camera half-angles. As expected, as the camera half-angle increases, the number of characters in low resolution increases. The number of characters in high resolution, on the other hand, increases and then decreases with the maximum value at 30 degrees half-angle. In analyzing the graph of figure 7 and figure 8, we would like to limit the number of characters in the field of view (preferably below 20 percent of the total population) and lower the ratio of characters in high-resolution versus characters in low-resolution. At a camera half-angle of 30 degrees, about 17 percent are in view and 83 percent of the population can be culled out. Of the 17 percent that are in view, only 68 percent will need to be in high resolution whereas the others can be approximated at lower resolution.

Figure 10: Graph of the Ratio of Characters in the FOV and the Ratio of Characters in Hi-Resolution versus Low-Resolution

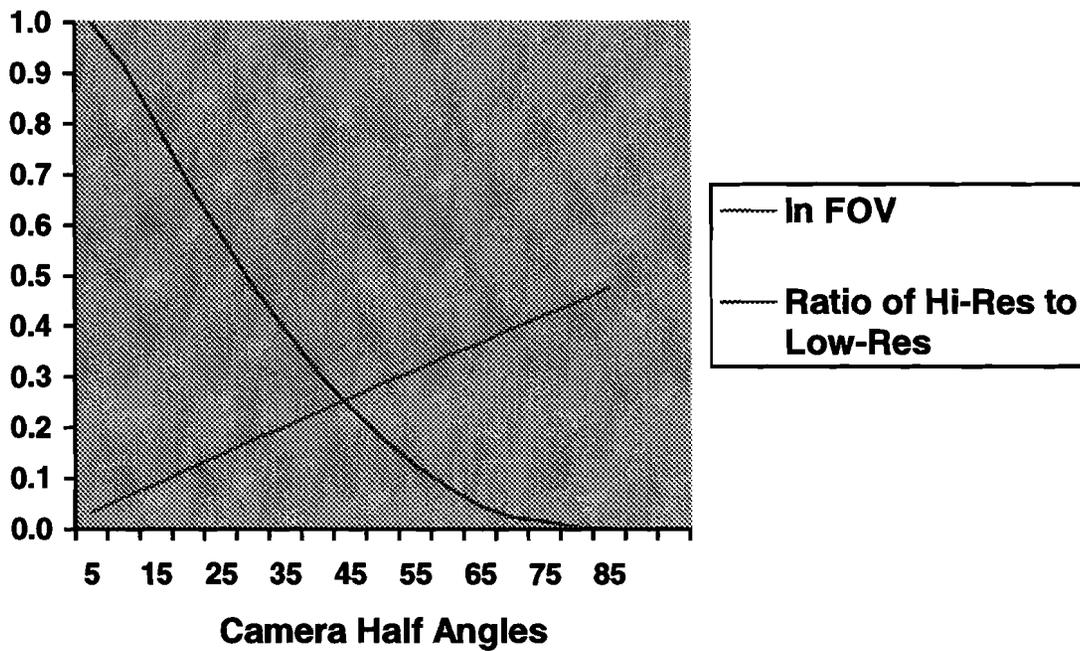
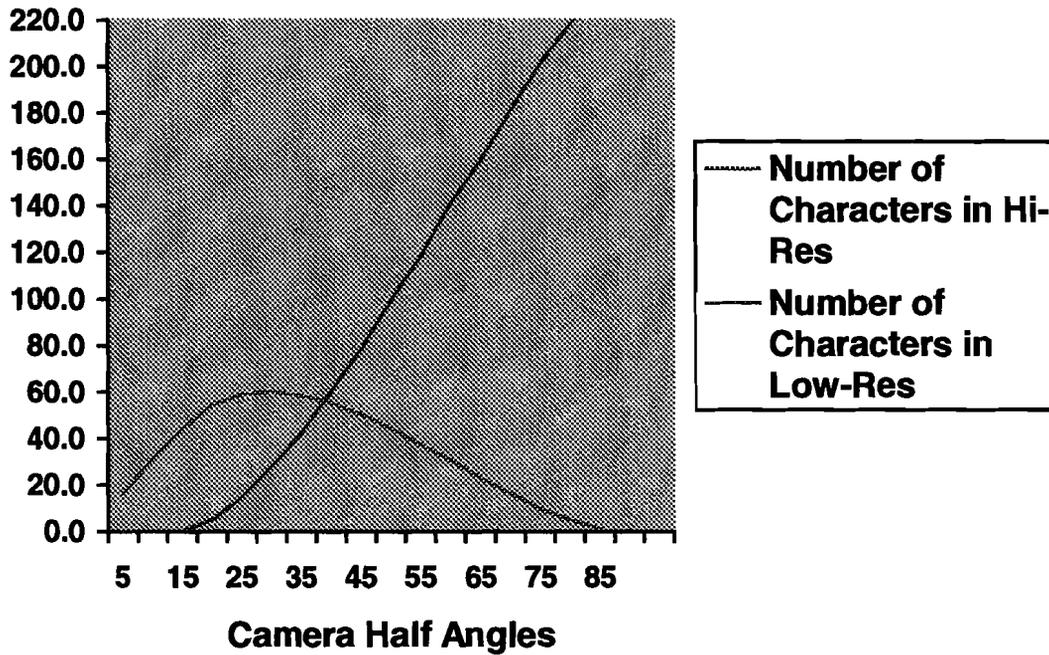


Figure 11: Graph of the Number of Characters in Hi-resolution versus the Number of Characters in Low-resolution.



5.3 Multi-resolution Geometry

We generate levels of detail in geometry by combining limbs from the highest resolution model. Each limb is modeled by a polyhedron. Limbs may be combined automatically by the system or by a method specified by the user. In using the system to automate the process, we only permit those with similar geometry or shapes to be combined. Limbs with different shapes may be combined if the author specifies the resultant approximated geometry. Our system combines a model's limbs when the joint connecting the two limbs are eliminated.

We do not incorporate any sophisticated mesh simplification techniques or any other techniques specified in the survey by Heckbert and Garland [2,3]. We did however, design our geometry system to be flexible and extensible so that these multi-resolution simplification methods may be easily incorporated.

5.4 Multi-resolution Animation

We developed techniques to eliminate fine motion detail when it cannot be discerned from the overall motion. One essential characteristic of realistic motion is small and subtle movements.

These movements add quality to a character's animation. As characters move farther away from the viewer, finely detailed and subtle movements become less distinguishable from the overall motion. We can therefore cull out the computation used to generate these subtleties when they cannot be discerned from the overall motion. An example of this in ballroom dancing context is the rise-and-fall characteristic of the American Waltz. Dancers rising onto their toes on the second beat and falling back to their heels at the end of the third beat generate the rise-and-fall motion. This motion may be eliminated when couples are so far away from the viewer that the rise-and-fall motion cannot be discerned. Another example is the body sway of the American Waltz. Dancers' bodies sway opposite to the direction they are moving. A dancer travelling directly to the right sways his/her body to the left. Body sway is a very subtle motion and does not need to be simulated when the viewer cannot distinguish the movement.

We use the animation level to drive both multi-resolution geometry and multi-resolution behavior. Our motivation in eliminating joints is to simplify motion computations as well as to allow us to combine geometry where it is applicable. Since all of our motions are joint based (or degrees-of-freedom based), by eliminating a joint, we can cull out unnecessary computation (animation computation and rendering transformation) of a joint which may not contribute to the final motion. Furthermore, joint elimination greatly reduces the numerical computation of iterative inverse kinematics.

5.4.1 Kinematic Chain

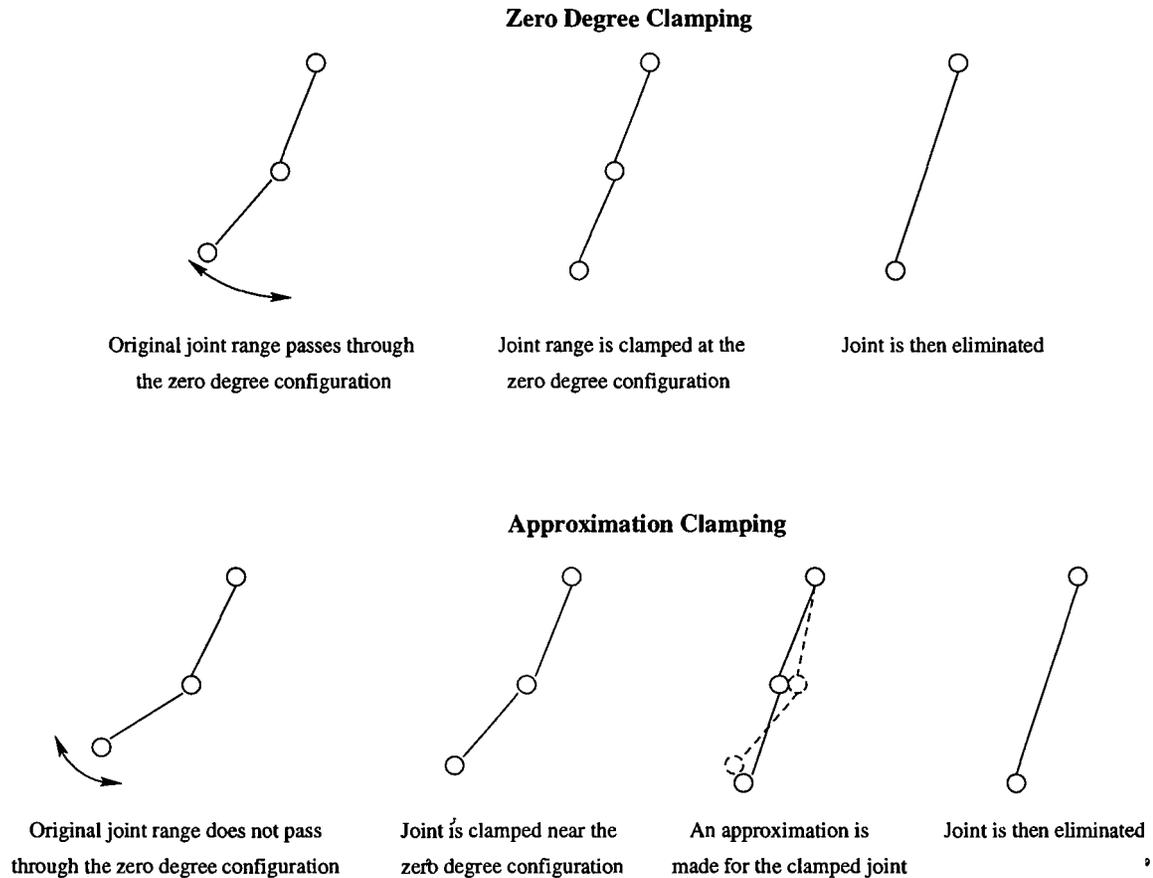
All joints are associated with a hierarchical chain, which we call a *kinematic chain*. A *kinematic chain* is a sequence of rotational joints connected by rigid segments. For example, the leg of our highest resolution model is constructed with four rotational joints. The root joint is the hip. Next are the knee, ankle, and lastly the ball of the foot. In analyzing which joints we should eliminate to simplify a particular motion, we evaluate the motion of each kinematic chain and how each joint affects the displacement of the end effector. Joints closer to the root of the chain are less likely to be eliminated because rotational changes in them contribute more displacement to the end effector than do rotations at joints that are more distant from the root. Root joints are never eliminated, as they connect different chains together.

5.4.2 Joint Range Clamping

We eliminate joints from a character by applying two techniques: *Zero Degree Clamping* and *Approximation Clamping*. Both are joint-range-clamping techniques. To eliminate a joint, we progressively reduce the range until it reaches zero. We call this action *range-clamping*. The purpose in clamping a joint's range is to eliminate small and subtle movements from motions when characters become less important. Our premise is that small and subtle details in motion that are not noticeable by the viewer should not be simulated and their computations can be culled.

The main difference between *Zero Degree Clamping* and *Approximation Clamping* is that *Zero Degree Clamping* attempts to return the joint to its initial configuration (i.e., its zero degree configuration). The zero degree configuration of a joint is defined to be the joint state where there is no rotation in any degree of freedom. If the elbow joint, for example, were in a zero degree configuration, then the result would be a straight arm. *Approximation Clamping*, on the other hand, sets the joint relatively close to its initial configuration. *Approximation Clamping* is used on joints that do not pass through the zero degree configuration. See figure 9 for an example.

Figure 12: Joint Range Clamping Techniques



Certain criteria must be met before a joint can be eliminated. These criteria are checked both at the start of the application and at run-time. At the start of the application, the Multi-resolution Engine is responsible for preprocessing all motions to determine which joints become candidates for elimination (for a given motion). The preprocessing criteria are as follows:

- The joint range must be below a predefined threshold.
- If the joint range passes through the zero degree configuration, then it becomes a candidate for *Zero Degree Clamping*.
- If the joint range does not pass through the zero degree configuration but the range is close enough to the zero degree configuration, then it becomes a candidate for *Approximation Clamping*.

After the preprocessing phase for each motion, each joint is tagged if it is a candidate for elimination. During run-time, the Multi-resolution Engine checks a run-time condition to determine whether a candidate joint's range should be clamped and eventually eliminated. The

run-time condition checks whether the distance of the joint from the viewer exceeds a certain threshold. The distance threshold is derived from the following two criteria:

- The displacement of the end effector caused by the joint must be below a certain pixel range.
- Body parts connected to the candidate joint must be below a certain screen size threshold.

In eliminating a joint, its range is progressively clamped according its distance from the camera. During the preprocessing step, we calculate the *grace distance*, *progressive clamp distance range*, and *culling distance*. *Grace distance* is defined to be the distance from the camera in which joint elimination will commence. The *progressive clamp distance range* is the range of distance in which the candidate joint's range is being progressively clamped to a certain configuration. The *culling distance* is the distance from the camera in which the candidate joint will be eliminated completely. The geometry connecting the joint may also be combined at this point or at a distance greater than the culling distance.

5.4.3 Joint Elimination Caveat

A character's joints cannot be eliminated blindly. Although we target specific joints for elimination, we must also take into account whether the joint we are trying to eliminate is a part of a *support kinematics chain*. We define a *support kinematics chain* to be any kinematics chain in which changes to any joint angle of the chain displaces the root position. For example, both legs are *support kinematics chains* with each chain consisting of the pelvic joint, knee joint, ankle joint, and ball of foot joint. Changes in any of these joints will offset the character's root position by means of forward kinematics. Once we start to perform motion clamping on a joint within a support chain, we cannot continue to use forward kinematics to offset the root position. This is because the free leg, which will eventually become the next support chain, may penetrate the floor due to the clamping effect. To compensate for this, when a character reaches a certain distance from the camera, it switches from using forward kinematics to using global motion approximation, which is the linear interpolation of the root position between key frames, to offset its root position. Multi-resolution techniques are then applied to the joints within the *support kinematics chain* in the same manner.

5.5 Multi-resolution Behavior

In generating levels of detail for behaviors, we evaluate each behavior individually and develop specific simplification techniques for that behavior. Since behaviors may be arbitrarily complex, we anticipate that the most computational savings will result from our ability to approximate highly complex behaviors.

We have divided our dance-related behaviors into two categories: *probability driven behaviors* and *authored behaviors*. As mentioned in the architecture section, once a behavior becomes active, its execute-condition is repeatedly tested according to its frequency policy. The difference between the two types of behaviors is that *authored behaviors* have fixed frequency policies. The frequency at which the behavior is executed, once it is active, is constant. *Probability driven behaviors*, on the other hand, have stochastic frequency policies. The frequency at which a *probability driven* behavior is executed varies within a prespecified range. Environmental conditions and character internal variables may affect the frequency at which *probability driven* behaviors are executed.

Authored behaviors usually encompass those behaviors that are essential for an author to convey a story. Most *authored behaviors* become active the moment they are added to a character. This is accomplished by sending an “activate behavior” event to the Behavior Engine. Some examples of *authored behaviors* are travelling down the line of dance, staying confined within the dance floor, and avoiding other couples. Each of these behaviors has an execution-condition that will cause it to trigger the appropriate action if the condition is met. The execution-condition of the “line of dance” behavior is always satisfied, which causes the direction-of-travel vector of the character to be modified accordingly. The execution-condition of the “dance floor constraint” behavior is only satisfied if the couple’s distance from the edge of the dance floor falls below a predefined threshold. Lastly, the execution-condition of the “collision avoidance” behavior is satisfied if there are other couples within the local radius.

Probability driven behaviors, on the other hand, add life to the characters. These behaviors contribute to the subtle differences between the characters. Some examples of *probability driven behaviors* are the promenade behavior and the spin turn behavior. These behaviors become active when the couple’s distance to the camera falls below a certain threshold. Once active, the execution-condition is tested periodically by the Behavior Engine and at a frequency according to the chosen policy. The execution-condition for the promenade behavior depends upon the amount

of dance floor available to the left side of the man. If this condition is satisfied and there is enough room to perform the promenade, then the system determines stochastically whether the motion should be performed or not. This probability may be influenced by the couple's position and orientation. For example, if the promenade would take the couple in a direction directly away from the camera, then the probability of performing the step should be decreased. This is because the motion becomes less interesting when we cannot observe the majority of the movement. Similar conditions hold for the spin turn behavior.

5.5.1 Dance Behavior Simplification

Although simplifying a character's behaviors may provide us the most computational savings, it is not clear how we can automate this process. One approach might be to classify plausible behaviors into categories and create a behavior-taxonomy. But this may be difficult to do because behaviors cover a broad spectrum. Furthermore, behaviors may be arbitrarily complex, so even if a taxonomy did exist, it is not apparent how we could automate the behavior simplification process. Therefore, we evaluate all dance behaviors individually and generate levels of detail specific to each behavior. In the following sections we discuss collision avoidance, dance floor flow, dance floor constraint, American Waltz, polite frame, promenade, and spin turn behaviors in further detail.

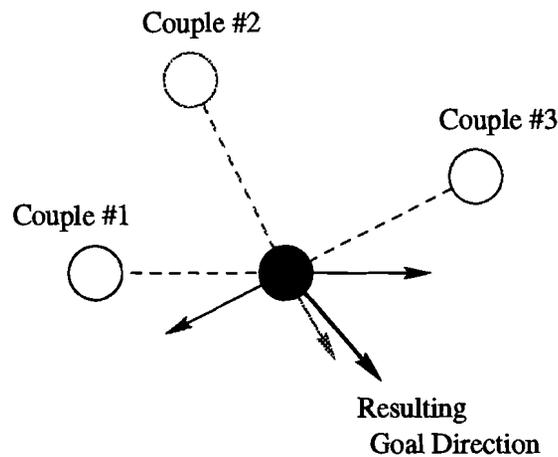
5.5.1.1 Collision Avoidance

Using a naïve approach and performing a pair-wise collision test, the computational complexity of collision avoidance is quadratic in the number of characters in the scene. To lower the computational cost, we simply use a local algorithm by partitioning the dance floor into sections. We allow the sections to overlap to preserve the continuous characteristic of the dance floor. The collision avoidance algorithm is then performed among the population in each section. We can further decrease our collision avoidance complexity by implementing a space-time bound algorithm similar to that of Hubbard's [22]. Performance gains would result from eliminating fixed time steps and using adaptive time steps and by predicting the time of the next possible intersection.

Our collision avoidance algorithm is performed in two phases. Each couple's goal direction is computed during the first phase. This is accomplished by having couples generate a repulsion vector from one another. The magnitude of the repulsion vector is inversely proportional to the

distance between the characters. The repulsion vectors are summed and normalized. In summing up the repulsion vectors, we record the largest magnitude and use it to scale the final vector. Scaling is performed because the final vector will be used to interact with other repulsion vectors, such as those generated from obstacles and those generated from the dance floor constraint behavior. Figure 10 shows how we compute the final vector. The second step is to stochastically select a legal step given the couple's orientation. We then perform a pair-wise collision test among the couples in the local area. If we detect a possible collision, we select a different step and re-execute the collision detection test. If all possible steps have been exhausted, then we allow the couple to stand still during the next measure of music.

Figure 13: Computing Final Travel Direction



The multi-resolution technique used for collision avoidance is to decrease the accuracy and frequency of execution. Our highest resolution collision avoidance model computes the direction of the greatest floor space after each couple has moved. In computing this direction, if the couple in question has not selected a step, we predict their step according to their current orientation. The intermediate resolution level computes the direction of travel as the sum of the repulsion vectors and selects a plausible motion where collision would not occur. At the lowest resolution level, couples select steps according to the final goal direction disregarding any potential collision. As mentioned in section 3.1, not all direction-of-travel goals can be satisfied with the available legal selection of steps. Therefore at the lowest resolution level, collisions may still occur due to the constraint of travel direction.

5.5.1.2 Dance Floor Flow Behavior

The Dance Floor Flow behavior sets the goal direction of the character according to its current location on the dance floor. This behavior simulates the flow of smooth ballroom dances where couples always travel counterclockwise on the dance floor. We implement a vector field on the dance floor to guide the couples along the dance floor.

The multi-resolution form of this behavior consists of decreasing the frequency in which it computes the flow vector given a position. Our dance floor is rectangular and hence, it is very simple to compute the flow vector as seen in Figure 11. But if we were to implement a more elaborate dance floor where there are obstacles placed strategically, judges, etc., or simply construct the floor from different geometric layouts, then we might wish to develop a multi-resolution technique to approximate the flow vector for lower resolution models.

5.5.1.3 Dance Floor Constraint Behavior

The Dance Floor Constraint behavior keeps the couples on the dance floor. This is implemented through repulsion vectors. When characters travel near the edge of the dance floor, a repulsion vector is generated to direct the couple away from the edge. The magnitude of this repulsion vector is inversely proportional to the distance of the couple to the edge of the dance floor. This behavior has an area of influence which presides near the edge of the dance floor. Only couples who come into this area of influence are affected by the behavior. The Dance Floor Constraint behavior also prevents couples from crossing the middle of the dance floor. This is shown in Figure 11. This behavior is always executed in high resolution (i.e., there is no approximation for this behavior) as we want to strictly enforce couples to remain within the dance floor at all time.

5.5.1.4 American Waltz Behavior

The American Waltz behavior determines what the next step should be given the couples' current state such as the next foot to step on, the desired direction of travel, the current location on the dance floor, etc. As mentioned in the collision avoidance section, the most detailed model computes the direction with the most available floor space and tries to perform a step to take the couple in that direction. The corresponding behavior model for the American Waltz allows all possible step selections. The only restriction is in the next foot to step on. As the viewer gets farther away, we approximate the American Waltz behavior by limiting the selection of available steps given the couple's current state. The savings comes in the collision avoidance check which

couple were oriented such that the man is facing the camera, then the likelihood of executing the promenade motion would be very high.

5.5.1.7 Spin Turn

One interesting motion in dancing is a turn, especially when it is performed very quickly. A spin turn is not a legal step in the American Waltz syllabus, but we added the motion because it is interesting and it gives the animation a little flare. We implemented spin turn as a *probability driven* behavior. Its multiple resolutions are like those of the promenade behavior. We assign it an influence bound, stochastically determine when it will check for available space, and stochastically determine whether or not it will trigger the spin turn motion according to the couple's current position and orientation.

5.5.2 Dance Behavior Summary

We summarize the dance behaviors by categorizing them as either authored behaviors or probability driven behaviors. We also list the behavior simplification technique associated to each dance behavior as shown in table 2. We summarize the multi-resolution technique used for each category of dance behaviors and list them in table 3.

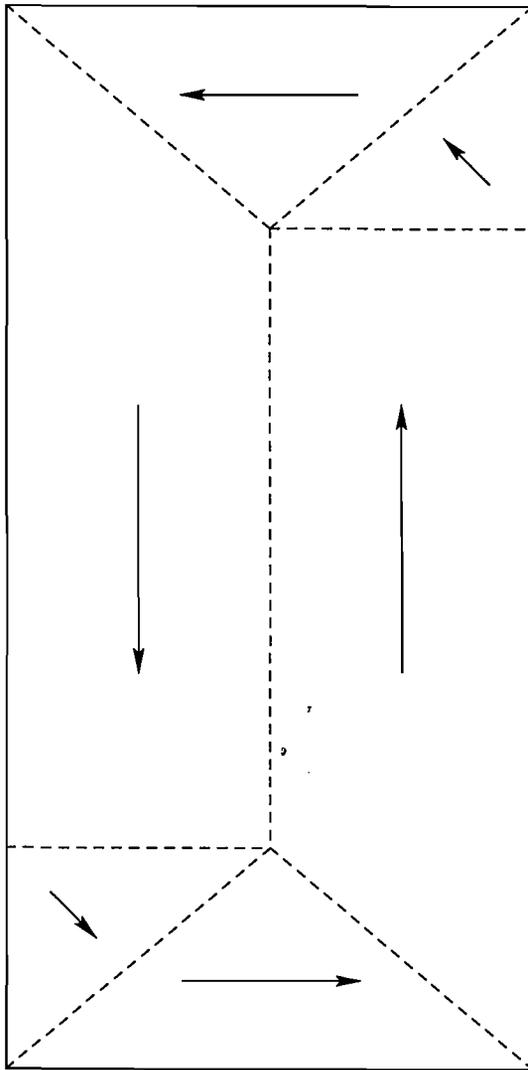
Table 2: Summary of Dance Behaviors

Behaviors	Category	Multi-resolution Technique
American Waltz	Probability	Decrease Accuracy
Dance Floor Flow	Authored	Decrease Frequency
Dance Direction (Collision Avoidance)	Authored	Decrease Frequency
Polite Frame	Authored	Prohibit Execution
Promenade	Probability	Decrease Probability
Dance Floor Constraint	Authored	None (Always in Full Res.)
Spin Turn	Probability	Decrease Probability

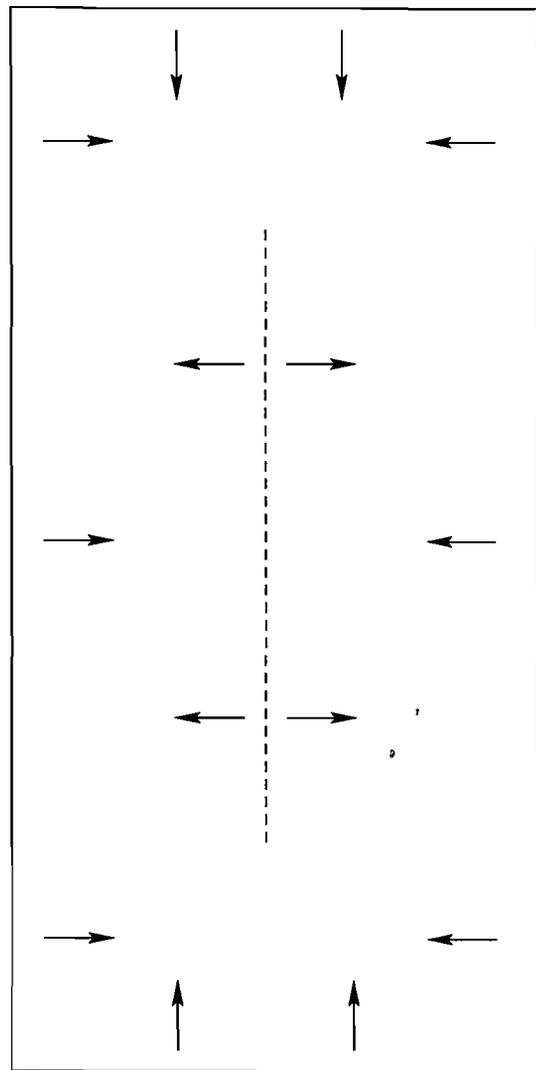
Table 3: Simplifying different types of behaviors

Authored Behaviors	Probability Driven Behaviors
Decrease Accuracy	Decrease Probability
Decrease Frequency	
Prohibit Execution	

Figure 14: Dance Floor Flow and Dance Constraint Vector Fields



Dance Floor Flow



Dance Floor Constraint

6.0 Importance Assessment

A character's importance determines the resolution levels it assumes in each of the three areas (i.e., geometry, animation, and behavior). Importance is an application specific quality. A dynamic simulation interested in realistic collisions may consider characters to be important if they may potentially collide, regardless of their visibility. For our ballroom simulation, we define importance to be visual information as well as semantic importance. In addition to the relative position of a character to the viewer, active animations and behaviors may also influence a character's importance. We define *local importance* to be the actions and behaviors that are associated to a character. In determining the final importance value of a character, we also take into account the importance value of surrounding characters. We call this *global importance* and discuss it in more detail in section 6.5.

6.1 Formulation of Importance

The following equation shows how we compute the overall importance value of a character. This final importance value is used to determine the resolution level of the character.

$$I = \frac{\alpha \cdot V + \beta \cdot L}{\gamma \cdot G} + \delta \cdot S$$

where $\alpha, \beta, \gamma, \delta$ are coefficients

and I = Overall Importance
 V = Visual Importance
 L = Local Importance
 G = Global Importance
 S = Semantic Importance

Visual importance, semantic importance, local importance, and global importance are discussed in further detail in the subsequent sections.

6.2 Visual Importance

The importance value of a visible character may vary according to its distance from the viewer, its focus in the view frustum, and its speed of movement. These attributes are similar to those defined by Funkhouser et al. [17]. Characters farther from the viewer are less important than closer ones. Characters closer to the center of the view frustum become more important. The speed at which characters move alters their importance. Faster moving characters are less

important than slower ones because less visual information can be attained. We show the order of importance in the following table.

Table 4: Attributes pertaining to visual importance

Importance	Visibility	Proximity to View Angle	Distance from Viewer	Focus	Speed
1	No	Far	N/A	N/A	N/A
2	No	Near	N/A	N/A	N/A
3	Yes	N/A	Far	N/A	Fast
4	Yes	N/A	Far	N/A	Slow
5	Yes	N/A	Medium	No	Fast
6	Yes	N/A	Medium	Yes	Fast
7	Yes	N/A	Medium	No	Slow
8	Yes	N/A	Medium	Yes	Slow
9	Yes	N/A	Near	N/A	Fast
10	Yes	N/A	Near	N/A	Slow

6.3 Semantic Importance

All characters possess an inherent importance value, which is defined by the author. This represents the fact that protagonist and antagonist are more important than supporting characters. A character with high value is more likely to stand out within a given scene. The semantic importance allows the author to override the system's multi-resolution evaluation process.

6.4 Local Importance

Each motion and behavior has an interest value associated with it. This value may be author defined or may be computed dynamically according to a set of rules. For example, we may define an action to be important if the motion of the kinematic chains displaces the end effectors by a large amount. An example of a motion with a high interest value is a spin turn. The animation and behavior's interest value are used to modify a character's importance value when the corresponding motion or behavior is active.

6.5 Global Importance

In determining a character's final importance value, we also take into account the animation and behavior of surrounding characters. For example, we may define a spin turn action to be very

important because it is a visually interesting motion. We may also define a character standing around to be significantly less interesting. Suppose that all characters within view were performing the spin turn except for one, who is just standing still. The motionless character stands out more in the scene and should be considered more important. We use this heuristic to modify the importance value of each character according to the motions and behaviors of surrounding characters.

The interest value of a motion corresponds to how much overall movement it produces. This interest value is then used to evaluate differences in local action versus global actions. Behaviors' interest values are author defined and are used in a similar way to evaluate differences in local behaviors versus global behaviors.

7.0 Results

The platform used in retrieving the results was a high-end Intel Workstation with 400 MHz Intel Pentium II Xeon processor, AGP Intergraph 3420T graphics card, and 256 MB of RAM. Our application was written entirely in Java (JDK 1.2) utilizing Magician 1.1, which is a wrapper for OpenGL.

7.1 Static Environment

We first apply our resolution techniques to a static environment to confirm that performance gains can be achieved. We define a static environment to be one in which characters generate no motion and there is no interaction between the characters (i.e., characters are just standing still). Since there is no interaction among the characters, we do not apply multi-resolution techniques to behaviors. Table 5 shows the difference in frame rates by applying multi-resolution techniques to animation, and applying multi-resolution techniques to both geometry and animation. Table 6 shows the percentage of improvement in applying multi-resolution techniques to different areas.

Table 5: Frame Rates of Multi-resolution in a Static Environment

Population	View Frustum Culling	Base Frame Rate	Frame Rate using Multi-res Animation	Frame Rate using Multi-res Geometry & Animation
25 couples	No	17.0	19.0	25.0
25 couples	Yes (68% in-view)	21.0	23.0	31.5
50 couples	No	9.0	10.0	13.0
50 couples	Yes (62% in-view)	12.0	13.0	17.0
100 couples	No	5.0	5.0	6.9
100 couples	Yes (70% in-view)	6.0	6.0	8.0
100 couples	No	5.0	6.0	7.0
100 couples	Yes (52% in-view)	7.0	8.0	10.0

Because of the static environment, applying multi-resolution to the animation system did not result in significant performance increase. As expected, significant performance increase was achieved by applying multi-resolution to the geometry system.

Table 6: Percentage of Improvement of Multi-resolution in a Static Environment

Population	View Frustum Culling	Base Frame Rate	Pct. Improvement using Multi-res Animation	Pct. Improvement using Multi-res Geometry & Animation
25 couples	No	17.0	11.7%	47.0%
25 couples	Yes (68% in-view)	21.0	9.5%	50.0%
50 couples	No	9.0	11.1%	44.4%
50 couples	Yes (62% in-view)	12.0	8.3%	41.6%
100 couples	No	5.0	0.0%	38.0%
100 couples	Yes (70% in-view)	6.0	0.0%	33.3%
100 couples	No	5.0	20.0%	40.0%
100 couples	Yes (52% in-view)	7.0	14.0%	42.8%

7.2 Dynamic Environment

Next we allow the characters to perform the American Waltz and take a bird's eye view of the dance floor looking down towards the long end. A little more than half of the dance floor can be seen from this angle. Since characters are moving about, the percentage of people in the view frustum constantly varies. Therefore we take the average frame rate during which the percentage of the people in the view frustum is in the specified range. Table 7 shows the difference in frame rates by applying multi-resolution techniques to animation, animation & geometry, and animation, geometry & behavior in a dynamic environment. Table 8 shows the same scenario but with percentage of improvement instead of frame rates.

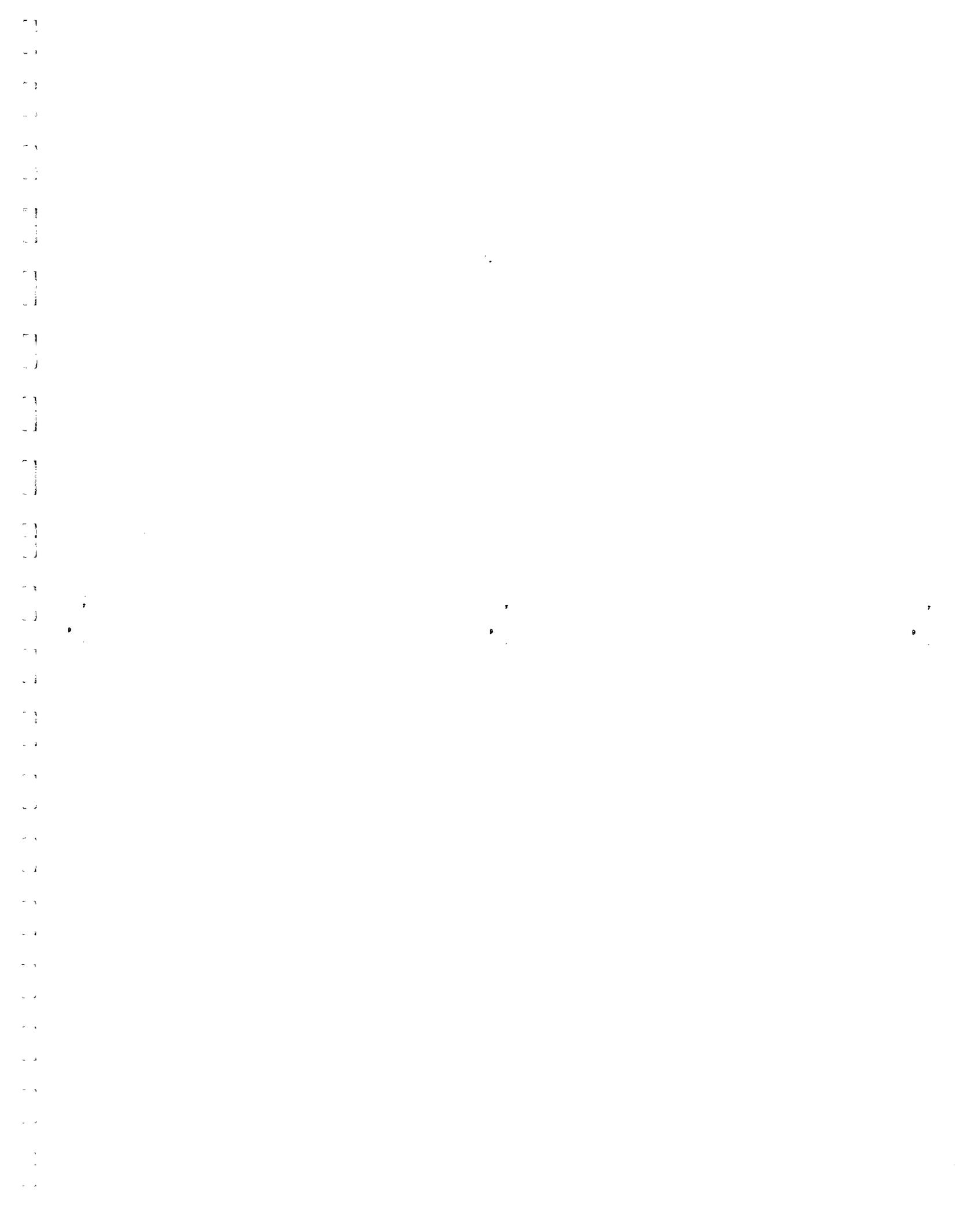


Table 7: Frame Rates of Multi-resolution in a Dynamic Environment

Population	View Frustum Culling	Base Frame Rate	Frame Rate using Multi-res Animation	Frame Rate using Multi-res Geometry & Animation	Frame Rate using Multi-res Geometry, Animation, & Behavior
25 couples	No	11.5	13.5	15.0	15.0
25 couples	Yes (60%)	14.0	17.0	19.5	19.5
50 couples	No	6.5	9.0	11.0	11.2
50 couples	Yes (50-52%)	10.0	14.0	17.0	17.6
100 couples	No	3.6	5.0	6.0	6.2
100 couples	Yes (50-52%)	4.8	7.0	8.3	8.7
100 couples	No*	3.9	5.2	5.7	5.9
100 couples	Yes (20%)*	13.0	16.5	16.8	17.5

*Camera is at floor level looking diagonally across the short end of the dance floor.

Table 8: Percentage of Improvement of Multi-resolution in a Dynamic Environment

Population	View Frustum Culling	Base Frame Rate	Pct. Improvement using Multi-res Animation	Pct. Improvement using Multi-res Geometry & Animation	Pct. Improvement using Multi-res Geometry, Animation, & Behavior
25 couples	No	11.5	17.4%	30.4%	30.4%
25 couples	Yes (60%)	14.0	21.4%	39.3%	39.3%
50 couples	No	6.5	38.4%	69.2%	72.3%
50 couples	Yes (50-52%)	10.0	40.0%	70.0%	76.0%
100 couples	No	3.6	38.8%	66.7%	72.2%
100 couples	Yes (50-52%)	4.8	45.8%	72.9%	81.2%
100 couples	No*	3.9	33.3%	46.2%	51.2%
100 couples	Yes (20%)*	13.0	27.0%	29.2%	34.6%

*Camera is at floor level looking diagonally across the short end of the dance floor.

Figure 15: 100 Couples standing still in high resolution

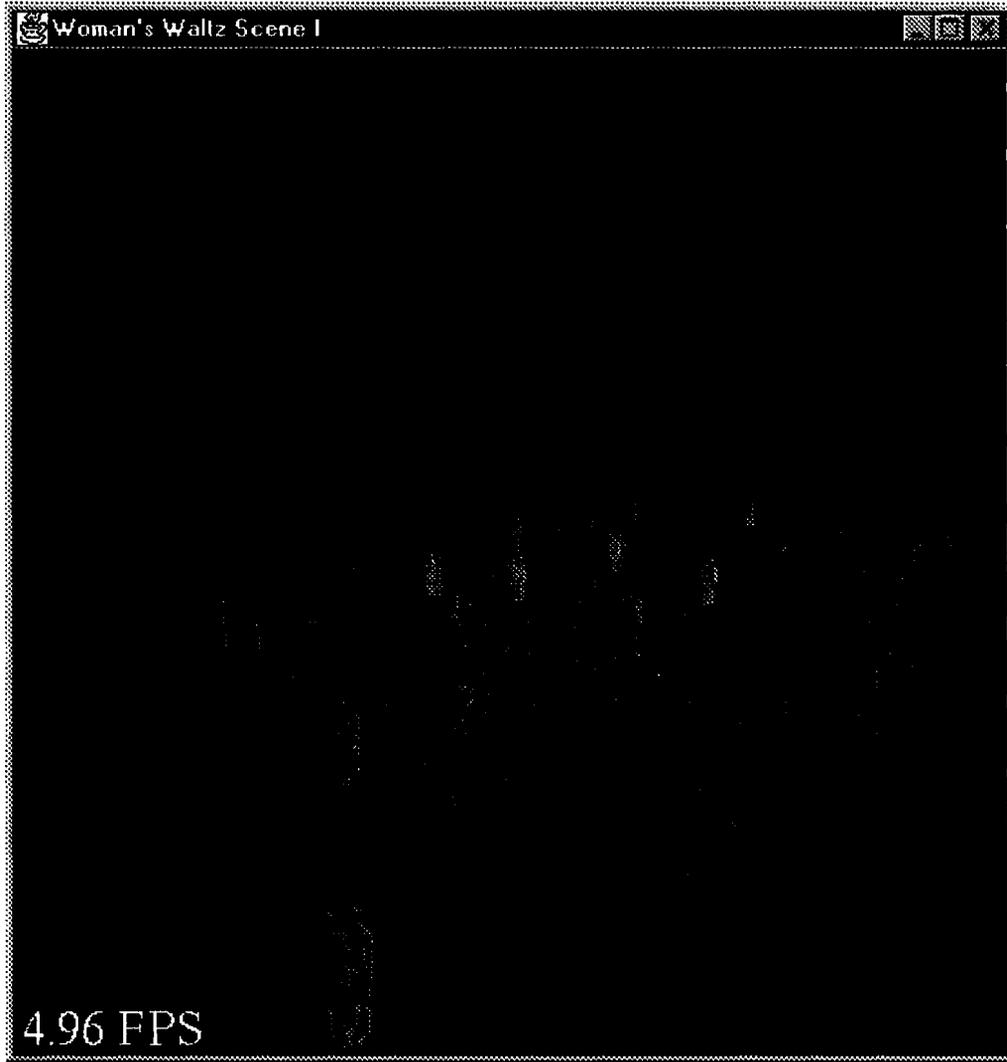


Figure 16: 100 Couples standing still with levels of detail in Geometry, Animation, and Behavior

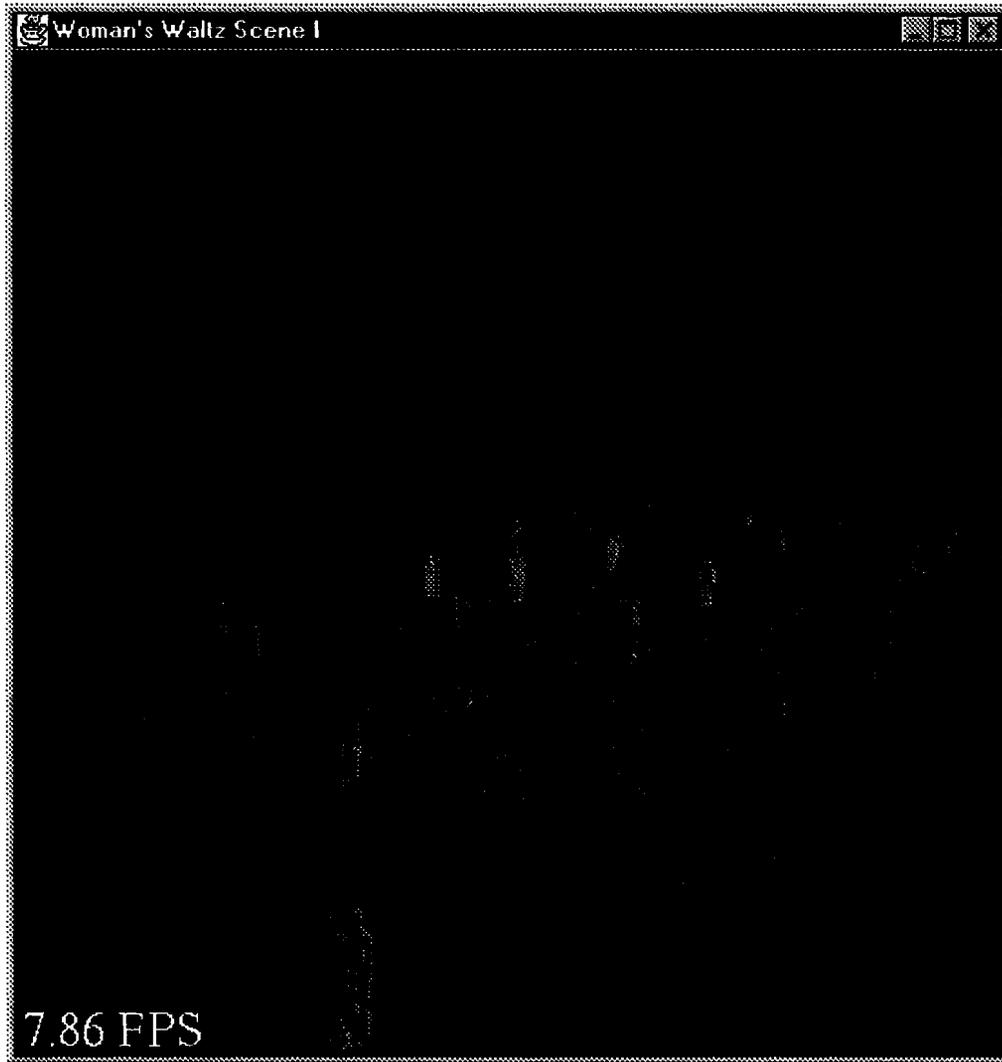


Figure 17: 100 Couples dancing in full resolution



Figure 18: 100 Couples dancing with levels of detail in Geometry, Animation, and Behavior



8.0 Future Work

8.1 Animation System

Specifying motions through degrees of freedom is not very intuitive. We need higher level input in creating motions, such as foot placement relative to the body, joint placement as a percentage of the overall motion (body sway), etc. Also, with our current system, key-frames must be evenly placed within the timeline. The current implementation places key-frames at the end of each beat. We need an adaptable key-framing system where there is no constraint in the placement of key-frames.

We currently target specific joints to eliminate. As mentioned in the multi-resolution section, joints closer to the end-effector are more likely to be eliminated. For our dance motions, the joints that are closer to the end-effector are mostly modeled as one degree of freedom joints. We would like to extend our joint range clamping technique to be able to smoothly clamp three degrees of freedom joints. Given an arbitrary articulated model, the joints closer to the end-effector may have more than one degree of freedom.

We are exploring methods to automate the animation simplification process. Given an arbitrary articulated model with various DOFs, the system should be able to develop a strategy to simplify the model's motion.

Motion capture data are very convincing because of the subtleties and gracefulness inherent in human motion. We attempted to simulate the subtleties of motion by adding coherent noise to our key framed motion. We are exploring methods to simulate the gracefulness of dance motion by categorizing similar motions in ballroom dancing. Different dances have different motion characteristics. For example, the Tango is characterized by slow and sharp movements, which accompanies the staccato in the music. Waltzes seem to have a flowing characteristic to it. Steps taken in smooth dances usually start out slow, then quick, and finally slow down at the end of the beat. A sine square curve can better approximate this motion than a linear interpolation of between the key frames. We call these motion filters artistic-filters.

8.2 Multi-resolution Techniques

We are exploring methods to utilize occlusion more effectively. In very densely populated scenes, most of the characters within the view frustum may not be visible due to occlusion. Currently, we

take advantage of occlusion by formulating a heuristic in that characters who are beyond a certain distance from the camera will most likely be occluded and we can assign a lower resolution model to them. We also assume that occlusion will deter observers from seeing artifacts in switching between different animation levels of detail. We would like to develop methods that take advantage of occlusion directly as opposed to just using our indirect approach.

8.3 Extending Multi-resolution Techniques to Off-line Engines

We would like to extend our technique to the domain of feature films. The context in which we apply our multi-resolution techniques is an interactive virtual environment. Interaction imposes a soft real-time constraint and therefore visual realism is dependent upon the machine architecture our application is running on. Feature films, on the other hand, do not require an interactive display rate. Therefore, the problem we are trying to solve in this domain is completely different. It is terribly inefficient to have artists animate each character in a crowd scene. A different approach is to animate several characters, create multiple copies of them, and place them strategically to form a crowd scene. The problem with this method is that it is visually unsatisfying. Animation for the crowd scene has to be cyclic and if the observer looks closely, it can be easily recognized. We can extend our technique to simulate a crowd scene which may not be interactive but is much less computation intensive. This helps eliminate the tedious task of having animators animate each character in the crowd scene.

9.0 Conclusion

We have improved upon previous multi-resolution techniques in the context of a Ballroom Dance simulation. We have also developed new multi-resolution techniques for animation and behavior. In applying levels of detail to our simulation, we have shown how the three areas (geometry, animation, and behavior) can interact with each other. Although, most of our multi-resolution techniques are not automated, we have shown that significant performance increase can be achieved. It is not clear how we can quantify visual accuracy in approximating a Ballroom Dance simulation. However under a dense environment in real-time, our results seems indistinguishable from the fully simulated counterpart.

References

- [1] James H. Clark. Hierarchical Geometric Models for Visible Surface Algorithms. *CACM* Volume 19, Number 10, October 1976, pages 547-554.
- [2] Paul S. Heckbert and Micheal Garland. Multiresolution modeling for fast rendering. In *Proceedings of Graphics Interface '94*, pages 43-50, Banff, Alberta, Canada, May 1994. Canadian Information Processing Society.
- [3] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. *Multiresolution Surface Modeling Course. SIGGRAPH '97*.
- [4] Taosong He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel-based object simplification. *Visualization '95 Proceedings*. IEEE Computer Science Press, 1995.
- [5] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics (SIGGRAPH '92 Proceedings)*, Volume 26, Number 2, pages 65-70.
- [6] Marc Soucy and Denis Laurendeau. Hierarchical surface triangulation of range data. *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, September 1991.
- [7] Jarek Rossignac and Paul Borrel. Multi-resolution (3D) approximations for rendering complex scenes. *Modeling in Computer Graphics: Methods and Applications*. Springer-Verlag, Berlin, 1993, pages 455-465.
- [8] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *SIGGRAPH '93 Proceedings*, pages 19-26, August 1993.
- [9] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96 Proceedings*, pages 99-108, August 1996.
- [10] Hugues Hoppe. View-dependent refinement of progressive meshes. In *SIGGRAPH '97 Proceedings*, pages 189-198.
- [11] Jovan Popovic and Hugues Hoppe. Progressive Simplicial Complexes. In *SIGGRAPH '97 Proceedings*, pages 217-224.
- [12] Deborah A. Carlson and Jessica K. Hodgins. Simulation levels of details for real-time animation.
- [13] J. P. Granieri, J. Crabtree, and N. I. Badler. Production playback of human figure motion for 3d virtual environments. In *VRAIS '95 Conference Proceedings*, pages 127-135, 1995.
- [14] C. A. Chrislip and J. F. Ehlert, Jr. Level of detail models for dismounted infantry in NPSNET-IV.8.1 Master's thesis, Naval Postgraduate School, 1995.
- [15] S. Cheney and D. Forsyth. View dependent culling of dynamic systems in virtual environments. In *Proceedings of 1997 Symposium on Interactive 3D Graphics*, 1997.

- [16] Jeff White. Improving behavior efficiency in virtual worlds. Master's Thesis, Brown University, September 1999.
- [17] T. A. Funkhouser and C. H. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 247-254, August 1993.
- [18] Ken Perlin and Athomas Golberg. Improv: A System for Scripting Interactive Actors in Virtual Worlds. *Computer Graphics (SIGGRAPH '96 Proceedings)*.
- [19] Bloomberg and Gaylean. Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments. *Computer Graphics (SIGGRAPH '95 Proceedings)*.
- [20] J. K. Hodgins, W. L. Wooten, D. C. Brogan, J. F. O'Brien. Animating human athletics. *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 71-78.
- [21] J. K. Hodgins and N. S. Pollard. Adapting simulated behaviors for new characters. *Computer Graphics (SIGGRAPH '97 Proceedings)*.
- [22] Philip M. Hubbard. Space-time bounds for collision detection. Master's Thesis, Brown University, February 1993.
- [23] S. Strassman, *Desktop Theater: Automatic Generation of Expressive Animation*, PhD thesis, MIT Media Lab, June 1991

Appendix A

A sample motion script for an American Waltz Left Basic Step performed by a man:

Dance "American Waltz"

Beats 3

Motion "Man Forward Left Basic Step"

Begin Motion

RightThigh	0.0 : 15.0	*	0.0 : 0.0	SIN_SQR
	15.0 : 0.0	*	0.0 : -15.0	SIN_SQR
	0.0 : 0.0	*	-15.0 : 0.0	SIN_SQR
LeftThigh	0.0 : -40.0	*	0.0 : 0.0	SIN_SQR
	-40.0 : 15.0	*	0.0 : 15.0	SIN_SQR
	15.0 : 0.0	*	15.0 : 0.0	SIN_SQR
RightShin	0.0 : 37.62	*	*	SIN_SQR
	37.62 : 0.0	*	*	SIN_SQR
	0.0 : 0.0	*	*	SIN_SQR
LeftShin	0.0 : 25.0	*	*	SIN_SQR
	25.0 : 0.0	*	*	SIN_SQR
	0.0 : 0.0	*	*	SIN_SQR

...

End Motion

The first line declares a new dance. The second line tells the system that there are 3 beats per measure in this dance. After a motion has been declared, a "Begin Motion" line signifies the system to start registering motion details. The first column declares the joint to be modified. The second, third and fourth column represents the rotation in X, Y, and Z-axis respectively. The values are specified in degrees and the range is separated by colons. The last column signifies the noise filter to be used when interpolating between the range values. There are three lines for each joint with each line representing the associated beat (e.g., first line represents the first beat, second line represents the second beat, etc.).

