Thesis Proposal: Situation based approach for virtual crowd simulation

Mankyu Sung
University of Wisconsin, Madison
Advisor: Michael Gleicher

April 21, 2004

1 Introduction

Crowds are an important feature of the real world, hence high quality crowd simulation is vital in many virtual environment applications for education, training, and entertainment. At the same time, there are several conflicting goals that make crowd animation a difficult task. First, simple characters are more efficient to evaluate, but complex characters can capture more realistic crowd behaviors. Second, we would also like to control over the actions of the crowd, but we don't want to control every agent individually, therefore we have to come up with a level of autonomy that satisfies user controllability and automation at the same time. Third, the real-time performance should be maintained even though a large crowd is simulated. Finally, the crowd behavior in virtual environment must be visually and semantically plausible.

However, up to now, there has been no solution that solves these conflicting goals. For instance, many researchers have proposed cognitive agent architectures [13, 34, 41, 1] that are made up of a knowledge representation, algorithms to learn new knowledge, and algorithms for planning actions based on the knowledge. These systems are not scalable because their complexity grows at least as rapidly as the overall environmental complexity. At the other extreme, physics inspired approaches, such as a social force model [17]

or a particle system [16], can show realistic crowd flow but are only applicable to situations such as emergency evacuation, in which crowd behavior is homogeneous (everyone does the same thing) and interactions with the environment are minimal. Numerous robotics algorithms can be used in a wide range of situations, but they are not intended to generate human-looking motion and are hard to control [2, 25, 11].

In this research, we set three specific demands of crowd simulation that are able to solve the conflicting goals described above. These demands are scalability, convincingness and controllability. The more specific definition of these demands are following :

- **Scalability**: Scalability is the major contribution of this research. This demand makes it possible that an increasingly complex environment, where a lot of complex behaviors are needed, can be created without a corresponding increase in the complexity of the characters. The required information that a character needs to respond to an environment gradually increases or decreases depending on how many situations the character is in. Also, it makes the crowd adopt to various kinds of environments automatically without changing character architecture. For instance, in our approach, a crowd for shopping mall environment can be used in a theater environment without changing the character's architecture. The more specific demands require the following:

  - **Scalable character memory** : The amount of memory for a character does not increase proportionally as the complexity of an environment increases.

  - **Performance scalability** : The overall simulation performance (frame rate) and the amount of memory required does not increase proportionally as the complexity of an environment increases.

- **Convincingness**: This demands that simulation maintains visually and semantically convincing behavior. The visually convincing behavior corresponds to realistic motion of crowds while the semantically convincing behavior means that the behavior should be reasonable for the given situation. We can assume the "realistic" motion by applying animation techniques that are proven to show realistic human character motion in crowds. For semantically convincing behavior, we may be needed additional information about the environment to determine the behaviors of crowds. For instance, depending on the traffic sign at a crosswalk, a crowd should determine behaviors for crossing or waiting.

- **Controllability**: The demand of controllability means that we should be able to specify behaviors at a particular space at a particular time easily while satisfying user specified constraints. In particular, the constraints in our approach are, a scenario for controlling crowd flow and a density map of crowd for the environment. The scenario specifies the order of situations that the crowd is going to meet. For example, in the theater, the scenario might say that people must buy a ticket at a ticket booth and then enter the lobby and hang around there before beginning a movie, and finally enter a movie room and watch the movie. In the same manner, the density map controls how many people are in a typical situation at a particular time. The density map can be associated with a scenario so that the scenario can control the number of people in a situation as well. The controllability is closely related to the user interface problem because it needs authoring tools to specify situations in the environment and special data structures for controlling crowds at a particular time.

For these demands, we present a novel *situation based* approach for simulating crowds for a complex virtual environment that meets these demands. In this approach, the convincing behaviors and controllability for characters are maintained and t the simplicity of the characters and the complexity of controlling them grows slowly as the environment becomes more complex. **It's my thesis that the situation based approach is able to achieve the demand of scalability, controllability and convincingness for crowd simulation.**

In the situation based approach, the crowd is simulated in two levels(Figure 1). At the high level, we adopt a distributed control mechanism called *situation* that gives each character in a crowd specific details about how to react at any given moment based on its local environment. Beyond that the situations control local behaviors of crowd for a particular circumstance, our approach makes the situations be composed to emulate a complex circumstance where several situations influence the character simultaneously. By using the situation technique, the scalability is improved because characters are under only a few situations at the same time and the number of situations, which has fixed size, can be increased as the environment grows without increasing the complexity of characters. In addition, for controllability, the situations are connected as a graph structure to represent an order that crowd is going to meet.

At the low level we use a probability scheme which computes probabilities over state transitions and then samples to move the simulation forward.

The probability scheme synthesizes complex behavior by composing multiple simple behaviors, which helps to create semantically convincing behaviors. In addition, we use a *Snap-Together-Motion* [15] process to maintain visually convincing behaviors. The STM ensures that there are no artifacts as motions are concatenated together over time.

The remainder of this proposal is divided as follows. Section 2 presents an overview of the proposed research. It describes how the situation based mechanism and probability scheme work, how to achieve demands from these methods and explains how our methods will be validated. Section 3 reviews related work. Finally, Section 4 summarizes some results to date and Section 5 presents an outline of how the remaining research will be approached.

2 Research Overview

2.1 Research Overview

The aim of our simulation is to produce the moment to moment actions of individuals in a crowd. At a given instant of time, an individual may perform any number of actions, but in our system the set is limited to those that we are able to animate. This constraint cames from the observation of real people in real world. Even though most people can do numerous actions at any given time, there are only small number of "reasonable" actions. This fact is also matched up our motion capture based human animation. Because we use motion synthesis based on connecting clips of existing motion (Snap Together Motion [15]), the individuals have a finite repertoire of actions (the available clips), and therefore at any instant they will have a discrete set of choices for what action to do next. The sequence of such discrete choices represents the behavior of an character over time.

The Snap-Together-Motion constrains the set of transitions for a character from one state to the next (for now, consider a state only contains a position and orientation) to the small set of movements that can plausibly follow the current state. The set of actions forms a directed graph where the nodes represent *states* where different choices are available, and the edges represent the actions that can be performed when it is in that state. The use of a directed graph, or a finite state machine, is a common mechanism for synthesizing the movements of characters in interactive systems.

The *behavior* of an individual character is the sequence of actions that it takes. This sequence stems from a set of choices: in each state, the character must choose which state to move to next by choosing one action. The mech-

anism for making the choice is what gives characters their behaviors. Our challenge is to create an state transition mechanism that works for crowd simulation that meets our goals of controllable behaviors in a convincing and controllable way.

The key feature behind our action selection mechanism is that when individuals are anonymous their specific actions may appear somewhat random. Consider people crossing a busy city street at a particular instant. There are many actions they may choose: they might continue walking across the street; they might turn around and walk back the way they came from; they might glance to the side to check if a car is disobeying the traffic signals. If we knew the individual, their choice might be clear: we might know that a particular person is prone to remembering that she forgot something in her office while crossing the street. For an individual we know little about (e.g. a member of a crowd), we cannot say for certain. We can only model the action choice probabilistically. A person crossing the street is more likely to keep crossing, but there is some chance that they may turn around and walk back.

We therefore use a *probability scheme* as our action selection mechanism. At any instant in time, we consider the choices that the character has for its next state, create a probability distribution as to which choice is likely to be taken, then sample from this distribution to determine the course of action. The challenge, hence, becomes determining the probability distributions such that the sequence of action choices leads not just to plausible behavior of the individual, but also the desired behavior of the entire crowd. We call a mechanism for determining a probability distribution a *behavior function.*

Complex behavior (and a correspondingly complex distribution function) is often made from a number of simpler pieces. For example, a person walking in the city will be avoiding others, trying to move towards a goal, trying to obey traffic laws, etc. Each of these simple behaviors gives rise to a behavior function. Because we can *compose* probability distributions, we are able to combine the simple behavior functions into more complex aggregate behaviors.

A behavior function may depend on many things. For example, it may depend on the location of the individual (the middle of the street is not a good place to stop), on what the character is able to "sense" (wait until the signal says "walk" before crossing), or even on an aggregate controller that attempts to regulate the crowd (if there are too many people on one side of the street, it is more likely for an individual to cross so that things are better balanced).

Our situation-based approach determines which behaviors are currently

influencing a character. When a character enters a situation, such as crossing the street, our system extends the character such that it is able to behave appropriately. Most obviously, it adds temporary behavior functions to the character that will be composed with the character's existing ones, allowing it to choose the character's actions more wisely. Situations also may add actions to the character - an individual needs not know how to look both ways unless it is crossing the street. In fact, even the character's ability to sense its environment can be adapted to the situation. Only when in the street crossing situation does the character need to know how to sense the status of the crossing signal. When the character leaves the situation, all of the situation-specific extensions are removed.

The main inspiration for our situation based approach is the anonymity offered by crowds. When we look at a crowd, we care only about *what* is happening, not *who* is doing it. This has two implications. First, the actions of the crowd should be driven by situations – what is happening – and not by individuals. This motivates our situation-based strategy. Second, viewers of a crowd cannot individually identify and track characters. Rather, viewers are attentive to overall statistics of the crowd: the direction that it's moving, the apparent agitation of its members, the number of people waiting for the bus. Hence, the actions of an individual matter only in its short-term contribution to the crowd's behavior, and not in its long-term planning. This motivates our probabilistic approach to simulation.

In situation based approach, a character is never in more than a few situations at any given time. This limits the set of behaviors a character requires at any given moment. For example, while watching a movie a character needs to know how to sit in a theater, but it doesn't need to remember how it bought the ticket, nor how it uses the bathroom. We exploit this with *composable* behaviors. A character starts off with simple behavior models. When it enters a situation, such as approaching a ticket booth, it is augmented with the machinery necessary to purchase tickets. Once a ticket is purchased, the machinery is removed again. In this manner, the amount of memory for a character is limited only by how many situations the character is under, which makes scalability property be preserved. In addition, since we *compose* new behaviors with old behaviors rather than replacing existing behaviors with new behaviors, the existing and new behaviors are able to be co-existed inside the character. This fact provides non-deterministic feature in behavior selection.

In terms of controllability, the situation also plays important role. That is, by connecting situations, we can control the order that crowd is going to meet. The connection between situations can be represented as a graph

structure called *situation graph* where nodes are situations and edges are connection between two situations. Depending on the scenario, by using this graph structure, we can control what the next situation is given current situation and how many people move from one this situation to others, which is very useful for controlling crowd.

To make the virtual environment more convincing, we should populate the environment as many characters as possible. To meet this demands, we should have a smarter way to simulate them. For efficient simulation, the crowd might be simulated in multiple levels depending on the distance from camera or other factors. For example, if the characters are occluded by some objects, then the characters should be eliminated from simulation loop. Even though they are visible, if they are too small on image plane because they are located in too far away place, they don't need to be simulated in detail. Instead, less expensive behaviors and animations can be used for improving overall performance.

The first advantage of this approach is ease of authoring; it breaks the problem of character design into the design of local activities, rather than one monolithic system. Second, re-use is enhanced; the core behavior and actions can be used in any environment and the situation specific modules can be shared wherever situations are shared. Third, efficiency is improved; at any given moment characters only have information for the situation that they are in – not all of the information for the entire environment. Fourth, the distributed control mechanism makes crowd flow be controlled efficiently; we can transfer crowd from one situation to another easily. Finally, de-centralized control of characters makes overall performance scalable, especially when a large crowd is simulated; each situation takes care of only a small number of characters.

The situation-based approach based on probability scheme puts an emphasis on environment design because overall crowd movement depends on the situations present in the space and where the situation are located. For specifying situations, we adopt a familiar *painting interface*. It allows the user to specify a particular situation by drawing it on the environment directly. Situations can be composed easily by overlaying several situations in one area.

3 Related Work

In this section, we are going to overview related researches briefly and illustrates how their approaches are different from our approach and why their

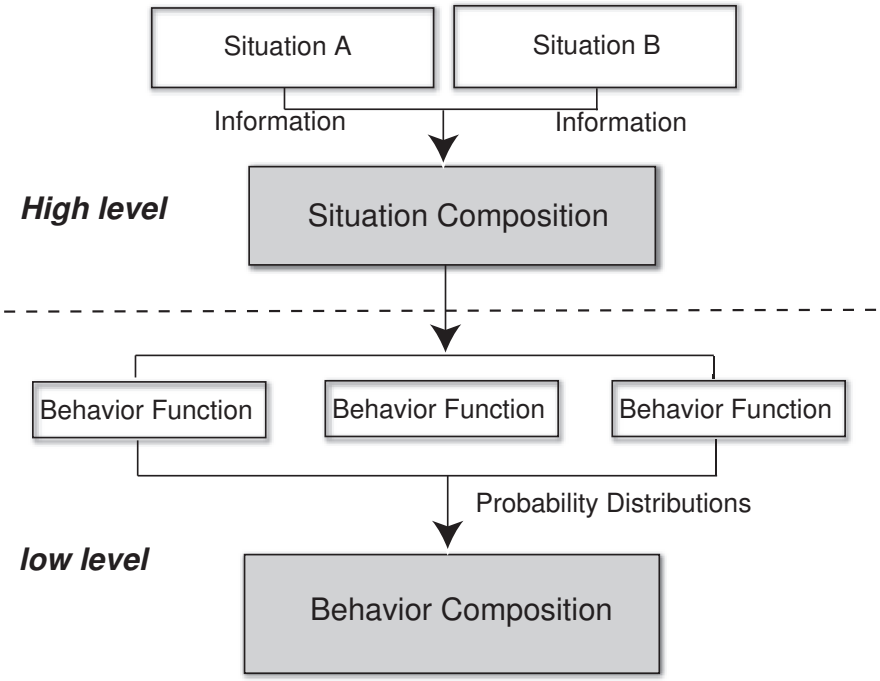

Figure 1: 1: An overview of our two-level character architecture. At the high level, situations control the events that characters respond to, and provide mechanisms for handling the events. At the low level, the behaviors resulting from multiple events are composed with a probabilistic technique to choose the next action..

approaches do not meet our three demands.(scalability, controllability and convincingness)

3.1 Smart Environments

To reduce the complexity of controlling crowds, several systems [9, 34, 10, 36] have attached information to the environment to guide the characters within it. Simple examples include driving and pedestrian simulators that embed lane or pathway information in the model. Thomas and Donikian [36] proposed a similar method that offers required information about the environment to characters. One difference is that the information is organized hierarchically based on GIS(Geographical Information System) for efficiency. Michael and Chrysanthou [26] simulate characters by adopting Gibson's theory which is presented in [38]. According to the Gibson's "natural movement" theory, when characters are walking on the street, they will simply guide themselves by moving toward further available walkable surfaces. Since it's hard to implement virtual sensing capability for characters to check walkable surfaces, they pre-compute visibility information at junctions of road and provide them to characters when they are there. Since all these approaches provide only geometric information about the environment, the behaviors of the crowd do not have convincingness. For same reason, the crowd lacks controllability. Our approach is similar to the above approaches in that we also embed information, such as planned paths, into the environment, but in addition we include the *behaviors*, needed *states* and even *sensors* to interpret that information. This additional information works to produce convincing crowd behavior. For controllability, the situation that provides this additional information can be set on the environment easily through our user interface.

The most commercially successful system of this type is a computer game, *The Sims* [12], in which objects advertise services, such as "satisfy hunger", and define a procedure that is run, in the characters name space, when the character responds to the service. *The Sims* highlights the authoring advantage of a rich environment: part of the game's appeal is the ability to add new objects easily and make characters respond to them. *The Sims* add behaviors by enforcing a specific, linear *plan* on the interacting characters. If the desired object-specific behavior is not amenable to a simple plan, the approach breaks down. For instance, a "mingle with a crowd" behavior could not be added, which is a evidence of lack of convincing behaviors. Such interactions must be part of the characters innate behavior, increasing its complexity. In contrast, our method adds "behaviors" that interact on

equal terms with the existing behaviors and have all the power of rule-based state machines. This allows for much simpler underlying characters and more complex novel extension behaviors.

Similar to *The Sims*, Kallmann and Thalmann [21] describe *smart objects*: objects that provide a plan for their use. The character, upon approaching an object, is told to execute a specific sequence of steps. For instance, an elevator informs a character to push the button, wait, then enter when the door opens. This approach is similar to our situation based approach in a sense that the characters are provided with all necessary information from the environment directly. But, in our approach, the situation is a more general concept that includes non physical effects such as friendship among characters. Moreover, in our approach, rather than indicating a specific behavior to a character at a particular time, the situation proposes a composable behavior that can be combined with others. One result of this is that characters in our system do not always respond the same way to the same object, just as real people behave, which increases the convincingness of the behaviors.

All of the above existing approaches are able to provide the proper information about the environment to crowds so that they can react to the environment reasonably. However, they failed in satisfying demands of convincingness and controllability.

3.2 Character animation

For convincing character animation, several researches have proposed many animation techniques. In this section, we are going to overview these techniques briefly and explain which technique we choose for our crowd simulation.

3.2.1 human animation

Most group behavior algorithms represent entities like simple creatures such as points or one-legged robots. The primary reason for this is that it's easy to parameterize velocity and acceleration to animate the characters in a realistic way. On the other hand, human have much higher DOF(Degree of Freedom) than these simple creatures, each of which must be specified in every frame of animation. Moreover, people have proven adept at recognizing the subtleties of human motion; even small inaccuracies in an animation can be quite apparent. This demand has recently caused motion capture data to evolve into a solution for animating human characters. Basically, for ani-

mating human characters, we need good motion data. Many example-based motion synthesis techniques based on motion blending have been proposed recently [40, 31, 23]. Basically, these techniques interpolate several motions to synthesize a desired new motion. To get a walking motion with a particular speed, for example, they search for similar walking motions in the motion set, and interpolate them to get the desired walking motion. In our system, on the other hand, we use appropriate motions directly without any interpolation because we assume that a behavior can be represented with a series of motions. Therefore, to simulate a particular behavior, we only need to figure out what motions are necessary and *select* a motion in an appropriate way, rather than creating a new motion. Performance improvement is a major advantage in this approach because we do not use search and interpolation. The thing that has to be checked is that there is no artifact when two motions are connected. For this purpose, we use distilled motions called *Snap-Together-Motions* that guarantee quality of motion even though we connect several motions at the same time [15, 24].

Alternatives to motion-captured data for crowds include image-based techniques. Loscos et al [33] describe a method that simulates relatively large crowds in real-time. However, the realism of image-based approaches is limited by the amount of imagery that must be stored in order to handle arbitrary viewing conditions.

3.2.2 Non-Human creature animation

Several techniques have been proposed to simulate large numbers of creatures other than crowds. Reynolds [30] describes a distributed behavior model for simulating the motion of a flock of birds. He demonstrates that realistic animations of group formations can be created by applying simple rules to determine the behavior of the individuals in the flock. Although his model is similar to our approach in that both approaches are targeted to simulate multiple creatures, Reynolds' model does not scale beyond the behaviors for simple creatures like birds. Thus the complex behaviors that a human crowd show are not possible to simulate. Tu and Terzopoulos [37] simulate artificial fish by using synthetic vision. Since they are using reactive behaviors for the simulation, the fishes show the same behavior for same stimuli, which is not the case for humans.

3.3 Intelligent Agents

The creation of virtual humans capable of behaving and interacting realistically in a 3D virtual environment requires specific agent architecture [35, 1, 41]. In particular, in order to make agents behave believably in a range of different situations, the *role-passing system* has been studied recently. Inspired by the idea from [19], Carol O'Sullivan et al adopts the *role-passing system* into intelligent virtual agent for a virtual environment. [**?**, **?**] This technique is similar to our *situation based approach* in that both try to make characters as generic as possible, and make them respond to various kinds of environments. However, there are significant differences between two approaches. First, in our system, all necessary information is encoded in a special data structure called "situation" and attached to the environment directly because we do not care what roles a particular individual has. In the role-passing system, on the other hand, each individual is specified by a particular role depending on the situation. Therefore, the number of roles is different from person to person. In a sense, our situation based approach encompasses this role-based system because we are also able to specify a role to agent with spatial situation called *non-spatial situation*. Second, in the role-passing system, the roles are associated with a set of motivations which drive the agent to perform particular tasks. The activation levels of motivation are related to attributes of agents which define personality. So among all possible motivations, the most dominant motivation is selected and the behavior associated with the motivation is performed. In this system, the Fuzzy Cognitive Maps(FCM) algorithm is used for determining the next action given the internal states of agents. However, in our situation, we don't keep any attributes in the agents. The main reason is that the attributes of agents and the resulting final actions can be resolved by using the probability scheme. Third, in roll-passing architecture, each agent can have at most one role at a time. However, in our situation based approach, each agent can be in several situations at the same time through situation composition.

In summary, the roll-passing technique has a benefit in scalability, but their architecture does not have a mechanism for convincingness and controllability.

3.4 Crowd Modelling

The classical rule-based system is the most popular approach for crowd modelling. Musse and Thalmann [27] explain crowd behavior in terms of a sociological relations among individuals and present two level collision avoidance

methods. Later, they deal with a hierarchy composed of crowd, group, and characters, where the group is the most complex structure tha t contains information to be distributed among the individuals, and controls each member with scripts, behavioral rules, or external controls. In addition, they endow crowds with different levels of autonomy. [28]. Blue and Adler [3] model pedestrian flow with *Cellular Automata(CA)* technology and demonstrate that this model produces acceptable fundamental flow patterns. This model has an advantage in estimating crowd density and speed of crowd flow, but it is not possible to show individually realistic behavior because it only focuses on the overall movement of crowds. Jiang et al [20] uses a *space syntax* technique to explain relationship between crowd density and the topological structure of environment. However, space syntax does not take account of the real difference in the size of blocks or varying perceptions of time and distance in different spatial configurations. One common problem of all these crowd models is a lack of controllability because they do not have any method for specifying a particular behavior to a particular space or a particular person.

At the other extreme, physically based approaches such as fluid dynamics [18], Particle system [7, 14] and social force model [17, 16] have been proposed. These methods and our approach have something in common: both put an emphasis on aggregative crowd behavior. Though, they are only effective in showing the flow and an density of a crowd. Thus, they are only applicable to typical situation such as emergency or pedestrians on the street because they focus only on the locomotive behavior of crowd. Brogan and Hodgins [4] proposed an algorithm for modelling group behaviors, in which each individual is forced to move toward a position computed based on group velocity, neighbors, and visible obstacles. They applied their algorithm to the simulation of Olympic bicycle racing [5]. The physically based approaches can show visually convincing behaviors of multiple characters, but they are hard to control and more semantically convincing behaviors such as "check a traffic signal and cross the street" are not possible to simulate.(there is the lack of controllability and convincingness).

In the robotics community, on the other hand, there has been some research that apply robotics algorithms to animating multiple characters and flocks. Bayazit [2] uses a global road map to set paths for multiple characters without collisions, and similar methods have been introduced by Furtney [11]. Tsi-Yen Li et al, proposed the Leader-Follower model [25], the main goal of which is to generate collision-free paths for characters. Since their focus is on path planning for multiple characters, complex behaviors such as "find an empty seat and sit down, then watch a movie" could not be simulated, this represents an lack of convincingness. In our approach, collision avoidance is

done by a behavior function which penalizes input states which may cause a collision by giving low probability to them.

4 Work to Date

Significant progress has been made toward our research goals. This section summarizes the results to date; details may be found in [32].

4.1 Scalability

Basically, the demand for scalability is achieved through our special concept called *situations*. Up to now, we have constructed the architectures of situation and then test them on several sample environments. Following sections will explain basic concept of situation and its components.

4.1.1 Situation based approach

Our basic assumption is that an environment consists of a set of different situations and only a few characters are under the same situation at the same time. A situation is distinguished from other situations by what typical behaviors the crowd can show. That is, the situation in our system controls the behaviors of a local group of characters. This provides distributed control over the crowd because we can give situation-specific behaviors to characters only when they need them. Major reason is that we decouple the character architecture from a particular environment. All control mechanism is encoded on environments and control information are added to characters only when they need it. In addition, our approach provides composability among multiple situations so that characters can respond to the complex scenario in which several different situations are combined.

A situation can be any circumstance that has typical local behaviors. From observations of real people, we easily see that one of the most important factors that affects character's behavior is its spatial location. For instance, a behavior for getting on or off a bus can be seen only at the bus stop. Therefore, we can set the "bus stop" situation around the bus stop. Other than spatial location, the social relationship between people is another important factor. Friends usually walk together when they move. Therefore, this "walking-together" behavior is a typical behavior that can be seen among people with friendship. It means that the "friendship" can be another situation. From this observation, we categorize all situations into two different kinds.

- **Spatial situation**: The situation has a region that affects in the environment (e.g, bus stop, ticket booth, ATM machine). This situation cannot be moved after definition (which can be at run-time) but can be deleted at run-time.

- **Non-spatial situation**: This situation (e.g friendship, group) does not have any region in the environment because it is attached to the crowd directly, not to the environment. Therefore, this situation can only be set at run time and characters are explicitly added or removed from the situation by the user.

The structure of situation is shown in Figure 2. In our system, all behaviors are implemented by special function called *behavior functions*. The behavior functions get a set of states as input and then computes their probabilities with a respect of a criteria. In the context of situation based approach, the behavior functions implement behaviors specific to the situation. Thus, the behavior functions in a situation are only reasonable for the situation. Other than behavior functions, the situation includes sensors and states, and event rule components. The sensors provide sensing capability for characters to catch events, states are state transitions (motion clips) that are used only for the local behaviors, and the event rule is a way to relate events to specific behaviors. When a character is under a situation, all these components are added to its representation at the same time. For a spatial situation, the components are added when the character first enters the situation's region of influence. For non-spatial situations, the character is affected when the user assigns the situation to it(the character). The components are removed from the character when it leaves the situation's region or the situation is otherwise removed. This dynamic adding and removal of behaviors enables us to achieve scalable characters.

The first thing a situation does to an character under its influence is to extend its states by adding new transitions to the characters current state transition graph. We refer to this as *extending* the graph. At the same time, this increases the number of states available for selection by the probabilistic selection mechanism. An example of an extended graph is shown in Figure 3. In essence, each situation has its own small state transition graph that is grafted onto the characters existing graph by connecting new transition edges. Each situation knows where in the character's default graph its own small state graph should be linked to. There can be multiple transitions between the existing graph and the new portion.

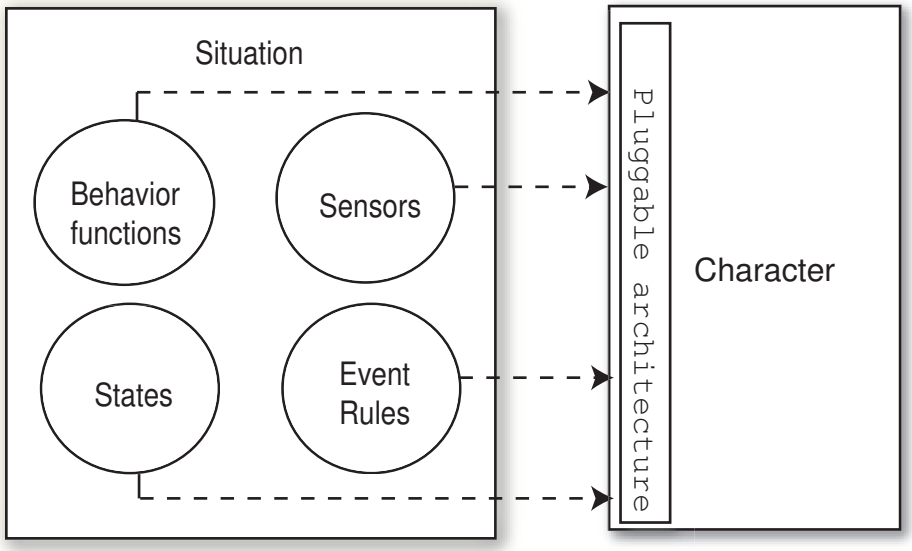

Figure 2: When an character is under a situation, it gets all the information from the situation directly. This includes states, behavior functions, sensors and event rules.

4.1.2 Sensors and memory

The events control the action of local behaviors in a situation. That is, depending on what happens in the environment, different events are sent and these change the character's behaviors. The events are captured by virtual sensors which are attached to the character by a situation. The captured events are stored in a list in character's virtual memory structure. The more complex structured memory architecture was discussed in [8]. Unlike structured memory architecture, we use simple list structure for simplicity and ease for manipulation. When a situation attaches a sensor, the sensor creates an entry in this list and updates it whenever the character uses the sensor. The contents of virtual memory are wiped out when the character is no longer in the situation.

In our system, we use four different sensors to catch four different events.

- **Empty sensor** : This sensor checks for the presence of any characters in a particular position in the environment.
- **Proximity sensor** : This sensor maintains the distance from a character to a particular position in the environment.

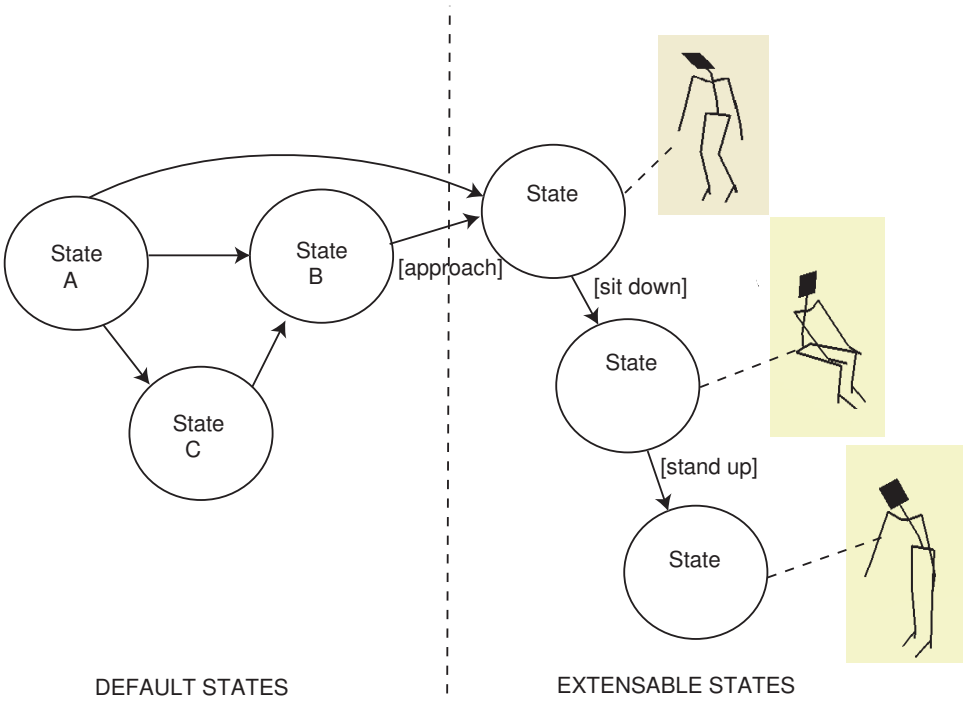

Figure 3: The states are organized as a graph structure and the graph is extended by adding a new sub-graph when the character is in some situation.

- **Signal sensor** : This sensor checks whether or not a signal in the environment is on.
- **Character sensor** :This sensor checks a particular character's motion and behavior including position and orientation.

Given events captured from sensors, the event rules inform the character which behavior functions to evaluate and compose, and can provide arguments to behavior functions that depend on the sensor. For example, suppose a character is in a "crosswalk situation", then the situation attaches a signal sensor to the character so that it catch the traffic sign. The associated behavior rule checks the sensor and applies a proper behavior function suited either to waiting (if the light says don't walk) or to crossing the street. These dynamic behaviors are composed with any others to determine the character's actions.

4.1.3 Situation Composition

In general, characters are under several situations at the same time. This is especially true of non-spatial situations such as group membership that must be composed with behaviors for fixed areas like a bus stop. In our system, the composition of two situations is similar to taking their set union: any state belonging to either is extended onto the graph, any behavior function required by either is added, and so are any sensors.

Some situations must prevent certain behaviors from occurring. For example, in a bench situation, to make an character sit down at a specific position, the "don't turn" default behavior should be ignored in composing the behaviors. This is achieved through event rules that specifically delete the undesirable behavior from the composition.

In this manner, the situation based approach provides a way for crowd to be adopted in various kinds of environment efficiently without making character architecture complicated.

4.1.4 Situation implementations

The situations are implemented on the *Python* with *SWIG* interfaces. The python is a simple interpreted, interactive and object-oriented programming language that can easily interact with existing popular languages like C or C++. The actual situation is encoded as a single python script file and be brought into a main environment file by user interface. Thus, adding a new situation is just adding a text python file to the main environment file. Once

all python scripts are entered into the main simulator which is built with C++, they are interpreted inside the simulator and executed. Besides situations, the event rules are also implemented with python language. Since the python script can access the agent's internal memory structure that contains information gathered through sensors, we can easily specify what kinds of behavior functions should be composed for given events inside the event rule script. A situation has a link to a event rule(python script) and brought in together to main environment file at running time.

4.2 Convincingness

The convincing demands are achieved from two basic techniques. First, the visual convincingness is achieved by using distilled motion data called *snap-together-motion*. Second, the semantical convincingness is obtained from our new underlying framework called *probability scheme* which provides a basic way for creating complex behaviors by composing multiple behaviors. Following sections describe these two techniques.

4.2.1 Snap-Together-Motion

A crowd is animated with motion capture data. By its definition, the motion captured data guarante the visual quality of motion because they records the actual movement of live-performer. However, at the same time, motion capture has an important limitation:it's hard to use in real applications such as a virtual environment because the motion capture data are just a stream of data. For virtual environment where controlling character is a major problem, new motion technique called STM(Snap-Together-Motion) is proposed by Gleicher et al. [15]. This technique preprocesses a set of short clips of motion that can be concatenated to make continuous stream of motion without causing any artifacts. In our system, all behaviors are generated by a set of actions and each action has a corresponding motion clip that is obtained by STM process. Thus, the use of STM guarantees the visual quality of behaviors.

4.2.2 Probability scheme

As described in Section 2, the behavior of the character comes from its choices about which actions to take. At each time step of the simulation, the character has a state $s = \{t, \mathbf{p}, \theta, \alpha, \mathbf{s}^-\}$, where t is the time, $\mathbf{p}$ is a 2D position vector, θ is an orientation, α is an action and $\mathbf{s}^-$ is a set of previous states.

The position and orientation indicate the spatial disposition of the character. The action is directly linked to a motion clip and determines which clip should be played for the current frame. The past states are used by the behavior mechanism to give some correlation in the character's behavior over time. Without some previous state information it is difficult to enforce behaviors like "walk in a straight line". The aim of the probabilistic state selection mechanism is to choose a next state given the current state.

The link between actions and motions means that we have a finite set of possible choices for the next action – it has to be one of the available motion clips. Each potential next state has an associated probability representing the chance of being selected. Our behaviors modify these probabilities on the fly, as well as the set of potential states. Note that this is conceptually similar to probabilistic finite state machines typically used for character AI [39], but in our approach both the state graph and the transition matrix are modified at run time. Our state transitions also represent lower level behavior compared to traditional finite state machines.

4.2.3 Behavior functions and behavior composition

When examining the movement of crowd in real life, we easily see that there are many different factors that influence the behavior of an character. For example, depending on whether or not there is a person or a object nearby, people might change their route. Or, if they have a target place that they are moving toward, they usually go as straight as possible. To simulate crowd behavior realistically, we need a way to take account of these various kinds of factors and synthesize a complex behavior that reflects them. More specifically, given the possible next states, we need a way to judge these states from the point of view of different factors and then compose transition probabilities to reflect all factors.

Each influence on the character is encoded in a *behavior* and special functions called *behavior functions* perform the task of transforming a behavior into a set of transition probabilities on states. For example, the "overlap behavior function" takes care of avoiding other characters. The "Don't turn behavior function" checks if a state transition causes too much change in orientation. The behavior functions judge the potential state transitions with their own rules independently and return the probabilities.

Suppose the set of possible next states is $\mathbf{S} = \{s_1, s_2, ..., s_n\}$, where s_i is a particular state. A behavior function evaluates all states in S and calculates their probability. A prototype behavior function is shown below.

```
Behavior function (States S[], Prob P[])
{
  For state s in set of S do:
        x = evaluate(s)
        v = sigmoid(x, α)
        P[s] = v
}
```

The *evaluate* routine inside a behavior function is a general function that can use any information available to it, such as the state of various features of the environment or the past state of the character. The *evaluate* function returns any real value, with a value of 0 meaning a 0.5 probability. The value is pushed through a *sigmoid* function, which normalizes the values to a range $(0, 1)$. We do this to make composition of behaviors more stable. The sigmoid function *f(x)* is,

$$f(x) = \frac{1}{1 + e^{-\alpha x}}$$

where α is a constant value chosen differently for each behavior.

The primary reason for using behavior functions is that we are able to compose several component behaviors to synthesize more complex behaviors that embody all the influences of the component behaviors. Composition of behaviors is simply the multiplication of the probabilities they produce (see Figure 4). That is,

$$P'(s) = \prod_{k=1}^{n} P_k(s)$$

where n is the number of composed behavior functions, s is a possible next state, the $P_k(s)$ is a probability distribution from k^{th} behavior function and $P'(s)$ is an unnormalized probability distribution function. We use multiplication because it allows one behavior to veto a state transition (by setting its probability to 0). Finally, re-normalization should be performed on the final composed probability distribution to ease sampling.

$$P(s) = \frac{P'(s)}{\sum_{k=1}^{m} P_k(s)}$$

where m is the number of states in S and s is the a particular state in S.

Once we have a final probability distribution, the simulation algorithm chooses the particular state transition to perform by sampling according to

$P(s)$. The sampling allows the character to choose not only states with high probability but also states with low probability, even though it's not frequent. This gives a randomness feature to the crowd.

4.2.4 Default states and behaviors

If an character is not in any specific situation we would still like it to exhibit certain default actions. This relieves the user from having to specify a situation for every point in the world. The default state transitions in our system are the identity transition (which does not change the character's state), five walking actions with different turn angles and six turning in place actions with different angles. Each transition takes a character from one position and orientation in the environment to another.

Several different default behavior functions combine to produce a sequence of states appropriate for a character wandering through an environment.

- ***ImageLookup* Behavior** : This behavior gets a bitmap describing the obstacles in the environment, and gives zero probability to states that cause the character to enter a place where some objects are located. Otherwise, it gives high probability.

- ***TargetFind* Behavior** : If the character has a goal position, this behavior gives high probability to states that make the character move toward the position. Otherwise, it gives low probability. Basically, this behavior is for moving characters from one place to another through path planning. We use the Probabilistic Road Map (PRM) method [22, 29, 2] in which many way-points are distributed randomly in the environment and linked together with Dijkstra's all-pairs shortest path algorithm including visibility. Since we compute all-pairs paths between two way points in a preprocessing step, at run-time we can find any path we want in real time.

- ***Overlap* Behavior** : This behavior gives zero probability to states that cause a collision between characters. Otherwise, it gives high probability. The collision detection algorithm used in our system is a simple adaptive spatial subdivision.

These default behaviors are always composed unless a character is in a situation that specifically ignores them (see Section 4.1.3).

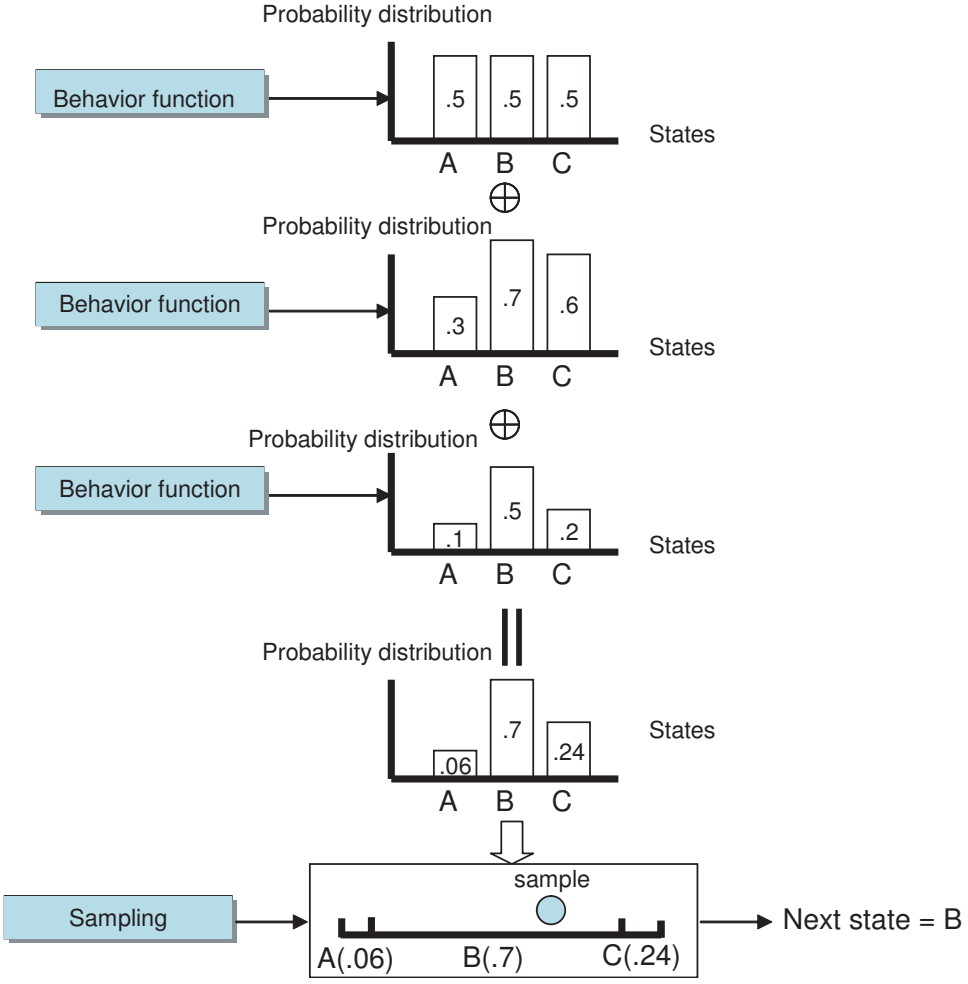

Figure 4: The behavior functions compute the probability of input states independently. The probability distributions that are computed from behavior functions are then composed. Finally, The next state is selected through sampling on the composed probability distribution

4.3 Controllability

The works for demand of controllability includes user interface for specifying behaviors of crowds. The use of situation-based approach puts an emphasis on environment design because the crowd's behavior depends on where a particular situation is located (for the case of spatial situation) and what situations the characters are in. For specifying a particular situation on the environment, we adopt a *painting interface*. This allows us to specify situations easily by drawing their regions on the environment directly like drawing a picture on the canvas. Painting is particularly useful for spatial situations. For non-spatial situations, we use standard techniques to select the characters to whom the situation should apply.

The painting interface is based on a layered structure where each layer represents a region for a situation and is saved as a bitmap file. The layered structure makes situation composition easy because it is done by overlapping several layers. Figure 5 shows the painting interface for spatial and non-spatial situations. The non-spatial situation also can be specified easily inside painting interface. By using grouping tool, we can specify which characters are going to be under the situation. And then, we can adding situation-specific behaviors on the group at run time or preprocessing time.

4.4 Validation

We will implement the technique described in this section and demonstrate through a variety of sample environments. That is, we will produce a "proof of concept" system demonstrating the workability of the methods. In order to prove scalability, we will measure the performance based on the frame-rate and memory use and then plots on graphs to check the trend.

5 Research Plan

Following is a tentative schedule for completing the proposed research and a brief description of how each step will be approached.

5.1 Smarter situation

Tentative Completion Date: June 15, 2004

Problem Statement. In simulating crowds for a specific environment, we might have a scenario that we want to see from the crowds. For example, we might want the crowd to go through some door and enter a room and

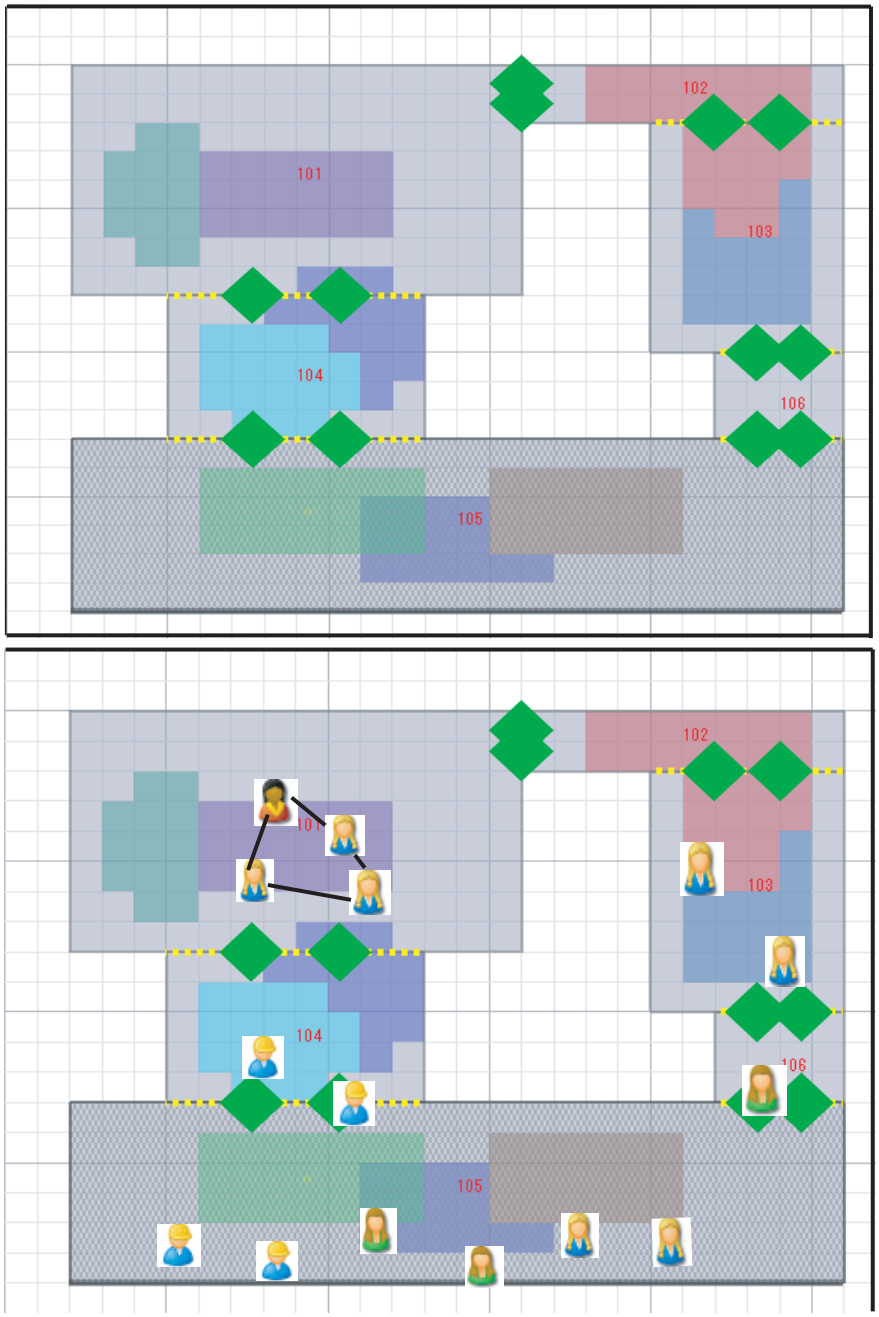

Figure 5: **Top:** The spatial situations can be set easily by drawing the situations on the environment directly. Also, the situation composition can be specified by overlaying situations **Bottom:** The non-spatial situation can be set on the crowd by grouping required people.

hang around for some amount of time. Or, we want to specify the order of rooms the crowd is going to move.

Research Objective. For this purpose, we want to build a special data structure for controlling crowd so that they are able to follow a specific scenario. This data structure should have a functionality that reads a text script containing scenario and interpret it or interactively control the crowd by setting scenario at running time. The most proper structure for this goal is a graph data structure where nodes are *situations* and edges represent connections between situations. The edges should have direct edges to indicate the order of situations that the crowd is going to be in.

5.2 LOD(Level of Detail) of behaviors

Tentative Completion Date: September 15, 2004

Problem Statement. For a large virtual environment like a virtual city populated by human-like characters, only a relatively small proportion of characters are relevant to viewer's interactions at any given moment [6]. The majority of characters will be distant from the viewer, and can be generated at much less expense. As the viewer moves through the environment, the importance of any one character will rise and fall depending on how close the viewer to the character or other factors. This provides a cue for simulating and animating large crowds without degrading performance.

Research Objective. Our goal is to develop a multi-resolution character that offers a range of behavior, animation and rendering models for a given characters, so that an individual character can switch behaviors at run-time to produce the required fidelity at the cheapest cost. Assuming that current behaviors are base, depending on the distance from the camera or other factors, By using this LOD technique, the simulation will seamlessly switch resolution to provide the user with a realistic, interactive, real-time experience. The situation structure might have a control for switching behaviors and manage all necessary information.

5.3 Hierarchical situation

Tentative Completion Date: December 1, 2004

Problem Statement. In current system, all situations are flat, which means that there is no hierarchy among situations. However, in some case, the situations can have hierarchy so that the common information for several situations are able to be shared at upper level of the hierarchy. For example, a queue situation for making crowd stand in a line has a hierarchy that puts

a generic queue situation on the upper level and vertical queue situation and horizontal queue situation at lower levels. The upper generic queue situation contains the common information that needs for all lower situations while the lower situation only have their specific information. When one of the lower situations are specified on the environment(spatial situation), the upper level situations is automatically inherited. In this manner, the hierarchical situation make it possible to organize situations efficiently and enhance the reusability.

Research Objective. Our research goal is to change current flat situation structure into the hierarchical one so that there exist hierarchical relations among situations. For this purpose, We have to figure out what components are commonly shared among the situations, and what relations one situation has to the others. The underlying data structure of situation might need to be modified to reflect the hierarchy.

December and early January will be reserved for submitting results to SIGGRAPH.

5.4 Adjustment of discrete action choices

Tentative Completion Date: July 1, 2004

Problem Statement. Since current system assume that all characters are given by discrete set of action choices for next state, sometimes the characters are not able to achieve their needed goals. For example, suppose they have a goal position in the environment, they might not able to reach the exact spot of goal position because of lack of actions to reach there, though they can be very close to the goal through path planning. This exact spot is important for the case when the character have to interact with virtual object such as bench on the spot.

Research Objective. For this purpose, we want to have a functionality that adjusts discrete choices to continuous choices so that the character can reach to exact spot on the environment. This demands requires on-line motion editing technique with minimizing overloading for real time performance.

5.5 Writing the Thesis

Tentative Completion Date: May 15, 2005

The thesis is to be finalized in time for a defense at the end of the 2004-2005 academic year.

References

[1] R. Aylett and M. Cavazza. Intelligent virtual environment-a state of the art report. In *Proceedings of Eurographics 2001 STARs*, 2001.

[2] O. B. Bayazit, J. Lien, and N. Amato. Better group behaviors in complex environments using global roadmaps. In *Proceedings of the 2002 Artificial Life (ALIFE)*, December 2002.

[3] V. J. Blue and J. L. Adler. Emergent fundamental pedestrian flows from cellular automata microsimulation. In *Transportation Research Record 1644*, pages 29–36, 1998.

[4] D. Brogan and J. Hodgins. Group behaviors for system with significant dynamics. 4(1):137–153, 1997.

[5] D. Brogan, R.A. Metoyer, and J. Hodgins. Dynamically simulated characters in virtual environments. 15(5):58–69, 1998.

[6] S. Chenney. Multi-resolution characters in large virtual environment. NSF grant proposal for 2004.

[7] Bouvier. E and Cohen. E. From crowd simulation to airbag deployment:particle system, a new paradigm of simulation. In *Journal of Electronic imaging*, Annual Conference Series, pages 94–107, 1997.

[8] T. Figueiredo Evers and S. Raupp Musse. Building artificial memory to autonomous agents using dynamic and hierarchical finite state machine. In *Proceeding of the Computer Animation Conference 2002*. The Computer Graphics Society (CGS) and the IEEE Computer Society, 2002.

[9] N. Farenc, R. Boulic, and D. Thalman. An informed environment dedicated to the simulaion of virtual humans in urban context. In *Proceedings of EUROGRAPHICS 1999*, Annual Conference Series, 1999.

[10] N. Farenc, S. R. Musse, and E. Schweiss. One step towards virtual human management for urban environment simulation. In *Proceedings of the ECAI Workshop on Intelligent User Interfaces*, 1998.

[11] F. Feurtey. Simulating the collision avoidance behavior of pedestrians. In *M.S thesis*. Dept. of EE, the Univ. of Tokyo, 2000.

[12] Kenneth D. Forbus and Will Wright. Some notes on programming objects in *The Sims*, 2001. http://www.cs.northwestern.edu/∼forbus/c95-gd/index.html.

[13] J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling:knowledge, reasoning and planning for intelligent character. In *Proceedings of ACM SIGGRAPH 99*, Annual Conference Series. ACM SIGGRAPH, 1999.

[14] P. G. Gipps and B. Marksjo. A micro-simulation model for pedestrian flows. In *Mathmatics and Computer in Simulation*, 1985.

[15] M. Gleicher, H. Shin, and L. Kovar. Snap-together-motion. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation 2002*. ACM SIGGRAPH, 2002.

[16] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamics feature of escape panic. In *Nature*, pages 487–490, 2000.

[17] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. In *Physical Review*, pages 4282–4286, May 1995.

[18] L. Henderson. On the flow mechanics of human crowd model. In *Transportation Research, 8*, pages 509–515, 1974.

[19] I. Horswill and R. Zubek. Robot architectures for believable game agents. In *Proceedings of the 1999 AAAI Spring Symposium on Artificial Intelligence and Computer Games*. AAAI techincal report SS-99-02, 1999.

[20] B. Jiang, C. Claramunt, and B. Klarqvist. An integration of space syntax into gis for modelling urban spaces. In *International Journal of Applied Earth Observation and Geoinformation*, pages 161–171, 2000.

[21] M. Kallmann and D. Thalmann. Modeling objects for interaction tasks. In *Proceeding of Eurographics Workshop on Animation and Simulation 1998*, pages 73–86, 1998.

[22] L. Kavraki and J.-C Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE Int. Conf. Robotics and Automation*, pages 2138–2145, 1994.

[23] L. Kovar and M. Gleicher. Flexible automatic motion blending with registration curves. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2003*, July 2003.

[24] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proceedings of ACM SIGGRAPH 2002*, Annual Conference Series. ACM SIGGRAPH, July 2002.

[25] T. Li, Y. Jeng, and S. Chang. Simulating virtual human crowds with a leader-follower model. In *Proceedings of Computer Animation Conference 2001*. The Computer Graphics Society (CGS) and the IEEE Computer Society, 2001.

[26] B. MacNames, S. Dobbyn, P. Cunningham, and C. O'Sullivan. Simulating virtual human across diverse situations. In *Proceeding of the Intelligent Virtual Agents(IVA) 2003(short paper)*, 2003.

[27] D. Michael and Y. Chrysanthou. Automatic high level avavar guidance based on affordance of movement. In *Proceedings of Eurographics 2003*, 2003.

[28] S. R Musse and D. Thalmann. A model of human crowd behavior: Group inter-relationship and collision detection analysis. In *Proceeding of Computer Animation and Simulations 97, Eurographics workshop*, pages 39–51, 1997.

[29] S. R Musse and D. Thalmann. Hierarchical model for real time simulation of virtaul human crowds. In *IEEE Transaction on Visualizeaton and Computer Graphics*, pages 152–164, 2001.

[30] C. O'Sullivan, J. Cassell, H. Vilhjalmsson, J. Dingliana, S. Dobbyn, B. McNamee, and C.Peter and. Level of detail for crowds and groups. *Graphics Forum*, 21(4), 2002.

[31] M.H. Overmars and P. Svestka. A probabilistic learning approach to motion planning. In *Proc. Workshop on Algorithmic Foundations of Robotics*, pages 19–37, 1994.

[32] C. Reynolds. Flocks, herds, and schools: A distributed behavior model. In *Proceedings of ACM SIGGRAPH 87*, Annual Conference Series. ACM SIGGRAPH, 1987.

[33] C. Rose, M. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Application*, 18(5):32–40, 1998.

[34] M. Sung and S. Chenney M. Gleicher. Scalable behavior for crowd simulation. In *In review*, 2004.

[35] F. Tecchia and Y. Chrysanthou. Real-time rendering of densely populated urban environment. In *Eurographics Rendering Workshop*, 2000.

[36] F. Tecchia, C. Loscos, R. Conroy, and Y. Chrysanthou. Agent behavior simulator(abs): A platform for urban behavior develpment. In *Proceedings of GTEC 2001*, 2001.

[37] D. Thalmann. The foundations to build a virtual human society. In *Proceeding of the Intelligent Virtual Agents(IVA) 2001*, 2001.

[38] G. Thomas and S. Donikian. Modelling virutal cities dedicated to behavior animation. In *Proceedings of Eurographics 2000*, 2000.

[39] X Tu and D. Terzopoulos. Artificial fishes:physics, locomotion, perception, behavior. In *Proceedings of ACM SIGGRAPH 94*, Annual Conference Series. ACM SIGGRAPH, 1994.

[40] A. Turner and A. Penn. Encoding natural movement as an agent-based system:an investigation into human pedestrian behavior in the built environment. In *Environment and Planning B:Planning and Design*, pages 473–490, 2002.

[41] A. Watt and F. Policarpo. *3D Games:Real time rendering and software Technology*. Addison-Wesley, 2001.

[42] D. Wiley and J. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Application*, 17(6):39–45, 1997.

[43] R. Wray, R. Chong, J. Phillips, S. Rogers, and B. Walsh. A survey of cognitive and agent architecture. http://ai.eecs.umich.edu/cogarch0/.