Abstract

Manually control each individual's behaviour in a crowd scene is very difficult. Manually modelling and animating each individual in a virtual environment is also time-consuming and impractical. A work process for crowd simulation generally consists of framework design, behaviour design and crowd visualization. Our work mainly contributes to the first two parts of a process: framework design, behaviour design. And we also give an overview research on the crowd visualization in crowd simulation area. Therefore our project is divided into three parts: First, we give a comprehensive background investigation on crowd simulation in each industry and provide different perspective of its research area based on different classification system. Secondly, we discuss about our behaviour design in detail. This is core skeleton of crowd model. A hierarchy-based model is introduced in this section and FSM is applied in our behaviour data structure. Also technical issues of crowd behaviour are emphasized with the concern of social, psychological factors. Finally, crowd variety and details of crowd representation are discussed including imposter, LOD, etc.

• Our work designs a behaviour model for crowd spectator simulation, specifically, on making spectator crowd have reactions when particular events in a sports game happen.

• We design a convenient production tool for animators to avoid duplicative work in simulating spectator crowd.

• A comprehensive research on crowd design, crowd behaviour and crowd representation is involved.

Table of content

Chapter1. Introduction and Context

1.1 Aims and objectives

Nowadays, with the rapid development of film and game industries, there is a need for in-house or commercial software which helps animators create a crowd scene, especially for those animators without programming experience. Although modelers are able to model precise human templates and animators can produce perfect animation for each human model, they still meet many obstacles when making a crowd scene with hundreds, thousands of people or even more. Therefore our work is not about modelling precise human templates or making good character animation. Our aim is producing a crowd simulation production tool which mainly has two tasks: (1) Setting up the behaviour model for crowd spectator simulation; (2) Producing an easy animation tool for animators to populate spectator crowd more efficiently. These are where our original ideas come from and why we start this project. Based on the research about crowd's behavioural and social factors, we provide a framework of crowd spectator simulation for sports games. And also we describe several techniques that allow us to achieve interactive behaviours of crowd in sports game.

The core aim of our project is the interactive crowd scene which is created based on implementing suitable behaviours for the spectator crowd. In our work, we introduce a hierarchy-based behaviour model which including three levels: (1) an external control is created which allows the user to be able to initialise crowd behaviours and a friendly user interface for external behaviour control is designed; (2) group control helps manipulate different teams of crowd and those teams have different reaction and different goals; (3) individual control which helps crowd react in different emotional ways is based on each character's emotion attributes. All of these three points will be discussed in detail in chapter 3.

Chapter2. Background and related work

2.1 Background

2.1.1 What is an agent?

Macal and North (2005) lists several definitions of agent which are proposed by former researchers [11]. They indicate that agents should have five features: identifiable, situated, flexible, goal-directed, and autonomous (self-directed). Figure 1 gives the definition of an agent. Four reasons are given by them to explain why ABS (agent-based system) is become more and more popular on a worldwide level: Firstly, the interdependency among various industries leads to traditional modelling method reaching its limit. Secondly, no matter artificial intelligence, economic market or other areas, most modern research areas are becoming more and more complex because many realistic models are added into these researches. Some traditional assumptions and data are becoming obsolete to some extent. Thirdly, modern research always involves more finer-level data. Lastly, hardware and software development makes many "impossible" model implementations designed several years ago feasible.

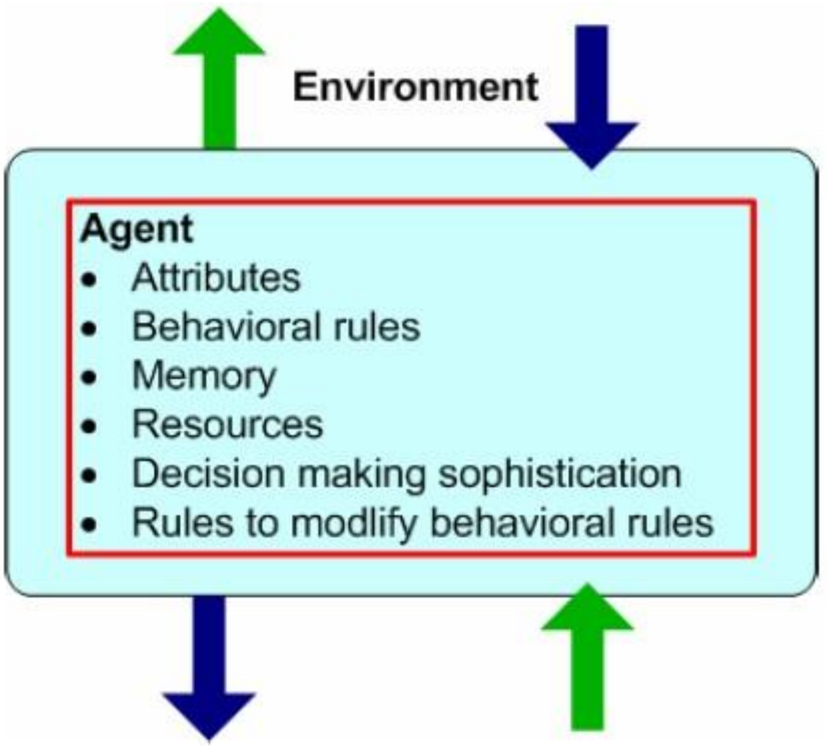

Environment
Agent
Attributes
Behavioral rules
Memory
Resources
Decision making sophistication
Rules to modlify behavioral rules

Figure 1. Definition of an agent. Source: Ref. [11].

Agent-based crowd is originated from the creation of ABS (or ABMS). The theory of ABS was first originated from the design of Game of Life (Gardner 1970). It is the simplest model of agent-based system. Crowd simulation is only one branch of ABS application. Now it has become a complex subject which involves with not only computer science, but also system dynamics, social and management science and complexity and management science [11]. They indicate that the goal of agent-based system is simulating individual behaviours and social rules. In sum, there is still no uniform definition of agent in this field. Table 1 provides a summarization of agent-based modelling technique applications.

Business and Organizations	**Society and Culture**
• Manufacturing	• Ancient civilizations
• Consumer markets	• Civil disobedience Terrorism
• Supply chains	• Social determinants
• Insurance	• Organizational networks
Economics	**Military**
• Artificial financial markets	• Command & control
• Trade networks Infrastructure	• Ecology
• Electric power markets	• Animal group behaviour
• Hydrogen economy	• Cell behaviour
• Transportation crowds	• Sub cellular molecular behaviour
• Human movement	
• Evacuation modelling	

Table 1. Agent-based modelling applications in each industry. Source: Ref. [11].

2.1.2 Overview of crowd simulation

Nowadays, crowd control has been applied in many industries such as military, architecture, sociology, etc. Figure 2 shows an overview of crowd control application in each industry. Different crowd applications have different design models, script languages and algorithms [17]. Some industries focus on the crowd's "explicit features" like appearance, relative position and quality of model templates [4]. Some others dig deep into how the crowd's psychological and social behaviour change over time under different situations [13]. Generally, the differences between crowd research areas are: crowd model design, behaviour concern, model template and rendering techniques. Although computer crowd simulation is a new research area which starts in the middle and late 90s, mass and crowd behaviour has been studied since the end of 19th century. Moreover, although there are a wide range of areas where crowd simulation has been applied into, the research in one area is often isolated from another – their interdisciplinary communication is not enough [10].

Behaviour of crowd has been studied a lot in the psychological and social area. The main idea for behaviour of crowd is that each individual will get rid of his own individuality to adopt the trend of a crowd, which is completely different as that of being alone [10]. The motion of crowd can be defined as *homogeneous* and *complicated*: *homogeneous* means synchronised or united revealed in the crowd motion, and *complicated* means crowd motion involves many parameters to control crowd behaviours.

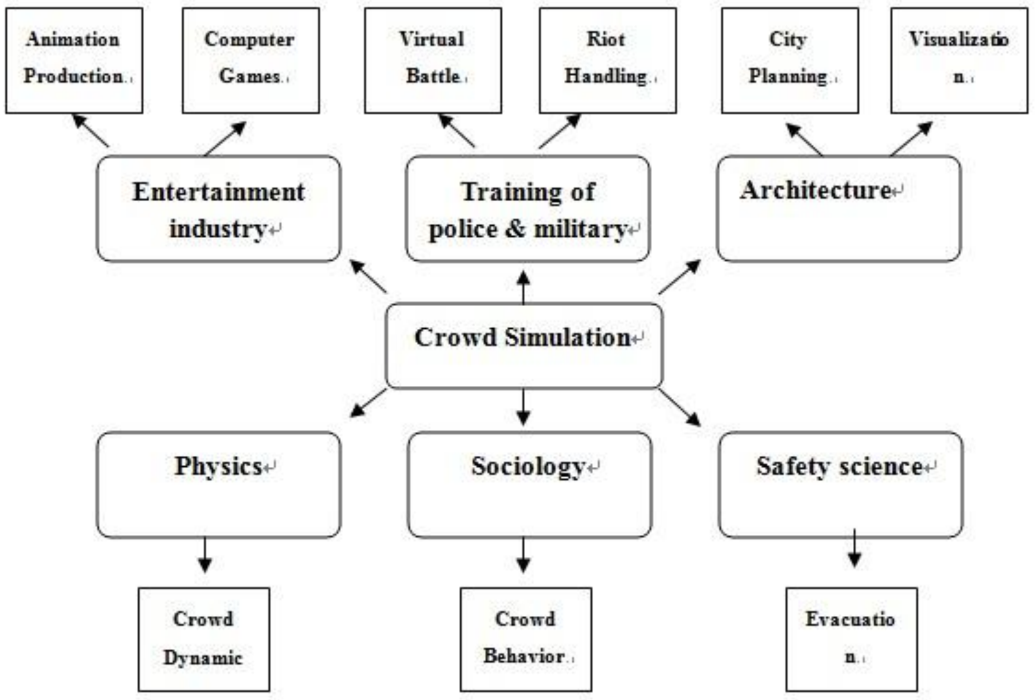


Figure 2. Overview of crowd application in each industry. Source: Ref. [2].

A crowd, which refers to an uncertain number of individuals, is ubiquitous and constantly-changing no matter in real world or virtual world. The method of implementing crowd simulation is different from single agent simulation. Benesh (1986) and Roloff (1981) defined a crowd as: "*A large group of individuals in the same physical environment sharing a common goal and may act in a different way than when they are alone"* [4]. Crowd simulation involves a comprehensive knowledge of each area: For example, many physical, psychological and social factors [6] were added into crowd control. In addition, variety needs for agent's appearance, pose, behaviour, animation and reaction to events also have to be concerned. Soraia Raupp Musse, Branislav Ulicny and Amaury Aubel (2004) propose another perspective of crowd simulation [10]. They classify crowd simulation according to the goals of different crowd. There are four goals representing the mainstream applications of crowd simulation: Entertainment, crowd motion simulation, populating collaborative virtual environments and behaviour modelling of crowds. Suiping Zhou, Dan Chen et al. (2010) also gives us a detailed, deep analysis of crowd classification based on crowd size and time scale [6]. For example, our project on sports games (Entertainment industry) belongs to 'short-term' zone and its crowd size ranges from "small-medium" (tens to hundreds) to "huge" (thousands or more). Figure 3 shows the classification of crowd model under above criteria.

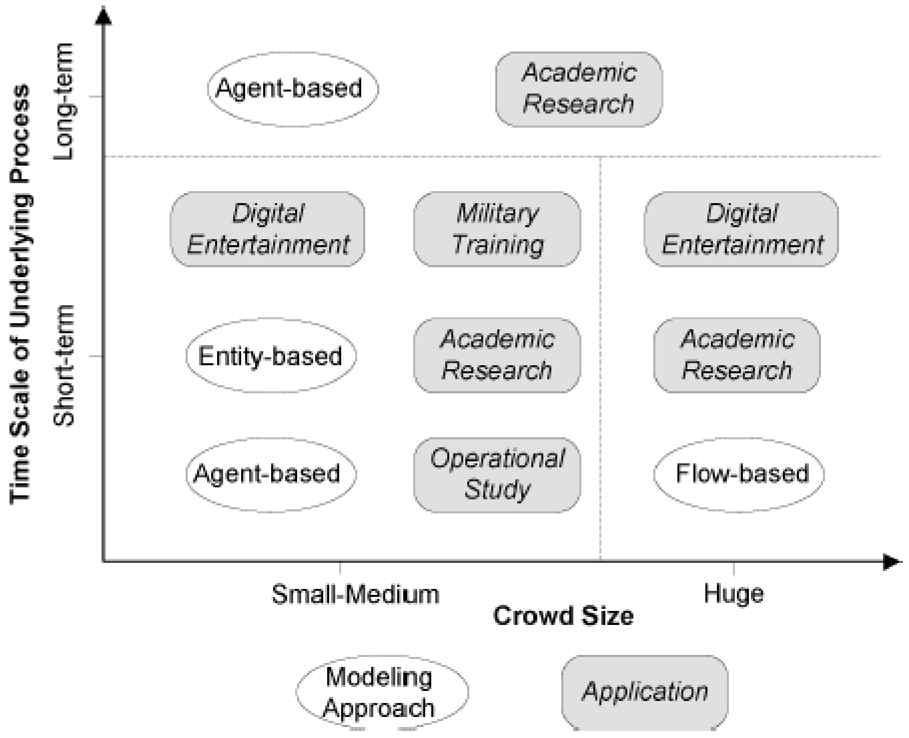


Figure 3. Classification of crowd models. Source: Ref. [6].

Crowd research has been categorized by model design approaches. There are two main directions [6, 10]. One research field, which mainly works on the movement and flowing trend of crowd, treats a crowd as an object of study. Chenney, S. et al discuss the representation work of two mainstream models in this area [14, 15]. One is defined as flow-based approach. The other is particle-based approach. These two approaches both involve physical and mathematical calculation. The principle of [14] is that although each virtual human is treated as an individual model, they are still implemented homogenously. The other work [15] further indicates that crowd movements and actions are controlled by mathematical, physical parameters. In general, these works can not be defined as "autonomous" or "intelligent" model design. The other research field is agent-based system (ABS). It is dominant at present crowd research area and has gained much progress with increasing development of CPU and GPU. Agent-based model involves with various behaviour rules (path planning, collision avoidance, navigation) and autonomous reactions to different events. In addition, many complex cognitive features are being developed including memory, stress and consciousness [2].

2.1.3 Four model approaches

The two fields mentioned above can be classified into four crowd modelling approaches. And each approach provides a fundamental framework for crowd simulation. After a comparison of different approaches, we will gain a basic understanding of how to apply appropriate model and algorithm to crowd simulation in our project. Four major approaches, Flocking system, Particle system, Hybrid system and Agent-based system are discussed below:

Flocking system, where a crowd is processed as a whole, is based on physical models. It was first applied into a computer evacuation program EVACNET 4 (Francis and Kisko 1984). And then Flow tiles [14], based on a velocity-field design, provides a better way to solve flocking simulation. Figure 3 gives a basic idea about velocity field and one application used in simulating stream. Fluid design is appropriate for simulating environment with hundreds and thousands of people. However, fluid design is hard to simulate specific details as desired [16]. It is mainly contributed to dense and large-scale crowd flow.

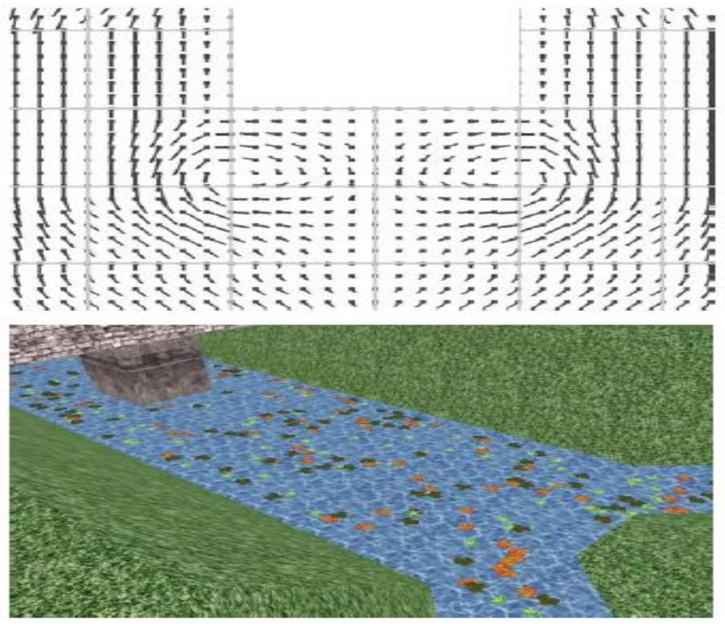


Figure 4. An example of divergence-free velocity fields. (up)
Flow embedded in an application, used to drive leaves
in the river. (down) Source: Ref. [14].

Particle system treats each agent in the crowd as homogeneous. It was first introduced to solve aggregate motion of animals like a flock of birds, a school of fish, etc [24]. In this system by setting up distributed behaviour model and physics rules, each unit (a particle) has its own perception in a dynamic scene. This non-hierarchy structure for crowd modelling is designed under force field and mathematical rules [15]. With the development of crowd simulation, now the boundary between particle system and agent-based model becomes more obscure [6]; Figure 5 gives an illustration of the simulation of boid flocks in particle system.

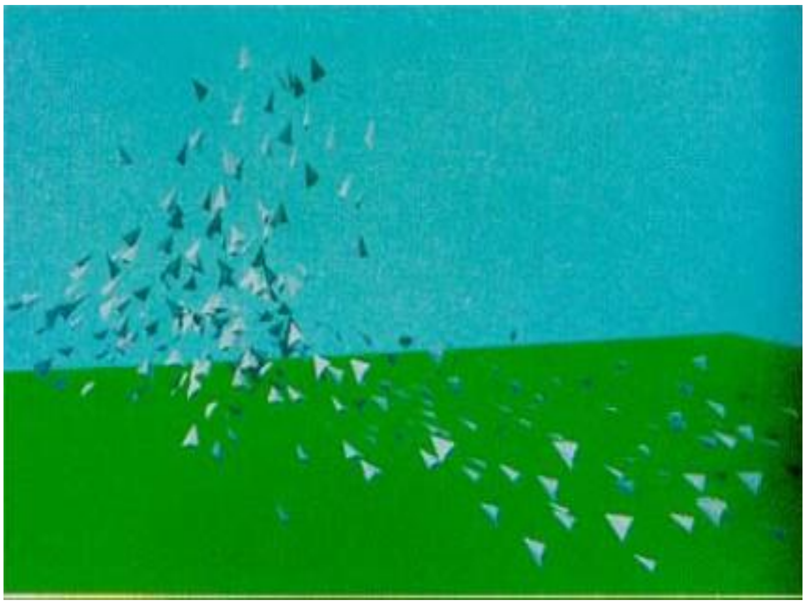


Figure 5. An example of boid flocks which is originated from particle system.

Source: Ref. [24].

Hybrid system is an integration of flocking system and particle systems. It involves both physical forces (flow fluids) and pre-defined rules [6].

Agent-based system is the most advanced and cutting-edge research area of crowd modelling and simulation in recent years. Many traditional tools can not satisfy developers' increasing needs on crowd simulation. The main difference between agent-based system and other systems is the adaptive algorithm and complex behaviour rules applied in agent-based model which makes each individual more intelligent and autonomous. Figure 6 is a detailed description of agent-based model used in electric power market [11].

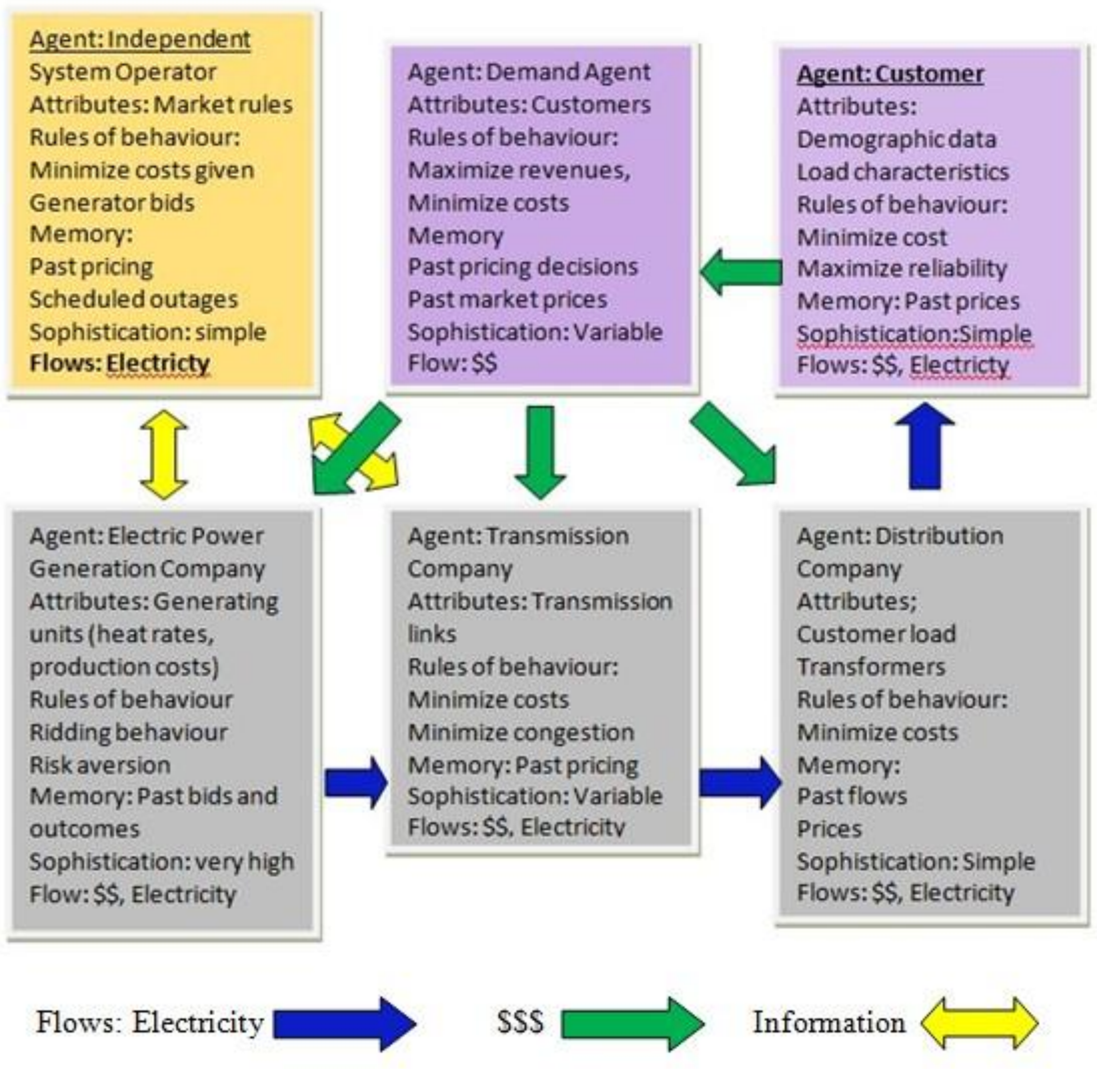


Figure 6: Agent-based model used in Electric power market. Source: Ref. [11].

Several labs like Crowd Dynamics, MRL@NYU, VMASC, CHMS@UPenn, are focusing on different areas of agent-based system. VRlab is one of the leading labs in real-time virtual human research. Entertainment industry has adopted many crowd simulation technologies from VRlab [6]. Vicrowd which is the main work of agent-based framework is also from VRlab.

2.2 Content of the project

2.2.1 Behaviour model design

In our project, there are three fundamental levels in behaviour model: crowd, group and individual. Our model is a hierarchy-based model. Each character has its own behaviour and emotion. Without any trigger event or external control, an agent is still able to behave on its own autonomously. But this does not mean that they don't have group behaviours. When the system finds events triggered or receives external instruction from user, it will distribute related behaviour into each character according to behavioural rules. Although each agent is independent, they can still form a team, a group to have collaborative behaviours.

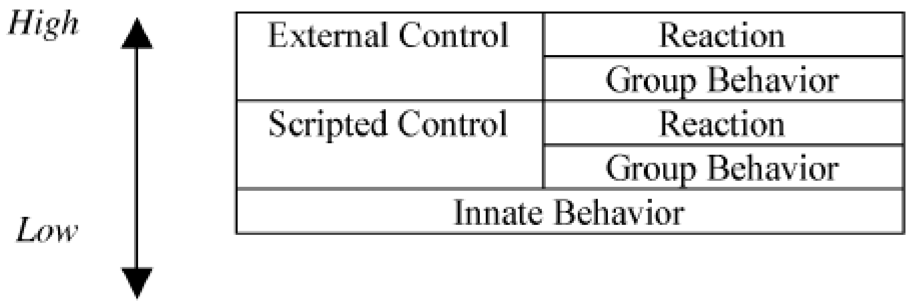
Generally, on a crowd level, behavioural rules are often set in advance, which forms a basic structure for model simulation. This makes it possible for group with various sizes to get control parameters from crowd. A group can be represented by many ways like: (1) a leader could be designed whose behaviour was followed by others [1]. (2) there is no leader but people in the same group have same team label. Autonomous group can be guided by external control which helps agents react to events. Individual, an agent, who stores his innate behaviour, also adopts the behaviour of its group behaviours. Based on this feature it is feasible to model different groups like families, lovers, etc. Such a flexible control over crowd allows us to generate behaviours according to different application and constrains, which means this structure, can be used as an extension or improvement of other crowd simulation project. This model offers different degrees of behaviour control including: pre-defined behaviours, rules control and guided control. Pre-defined behaviours are programmed (scripted) behaviours which were embedded in crowd model before simulation. Rules are those script functions which were trigged by specified event. Guided control emphasizes external control. This type of behaviour-author-tool control is extensible and flexible for crowd deign.

Moreover, this type of model can also reduce conflicts between different levels of control when multiple orders were sent to agents. For example, when scripted behaviours (programmed behaviours) control an agent who is clapping for a nice goal, guided control is able to allow the character not to clap without closing or constraining scripted behaviours (e.g. looking around, keep quiet, idle). Another example is: A spectator has some innate behaviour like watching games and talking to others. When a team wins (event occurs), he can still talk (innate behaviours) associated with standing up (reaction of event) [1].

2.2.2 Behaviour Research

Behaviour, namely innate attribute of each agent, is part of important implementation of in our crowd simulation. Generally, it can be divided into three parts: physical, social, psychological factors [6]. In shooting films and making crowd for background in games, physical factors like orientation, walking speed and gesture needs to be considered. In recent years, many games like Assassins Greed: brother hood [18], L.A. Niore [2011] are adding social, psychological factors into Non-player-characters (e.g. pedestrians, creatures) to provide more interaction for players with virtual environment.

Then physical factor which involves with different actions, path finding and collision avoidance is discussed based on VICROWD model. It is feasible to control the talk, chatting and any basic behaviour of spectator by using scripted control. And any reaction and triggered event related to sports (applaud, angry, lose, explosive) can also be controlled by external control method. In another word, external control has a higher priority than scripted control. Both ways directly influence the reaction and group behaviour. Innate behaviour holds the lowest priority. Figure 9 shows the VICROWD behaviour priority. Some parts of this flexible strategy have been applied into our project.



External Control	Reaction
	Group Behavior
Scripted Control	Reaction
	Group Behavior
Innate Behavior	

Figure 7. Behaviour priority. Source: Ref. [1].

2.3 Related work

[26] gives an overview of the differences between linear and interactive animation and provides a detailed flow work of animation production. In this work a detailed explanation of a FSM provides a clearer view of how to embed FSM into animation application. [23] provides a behaviour model which is based on virtual agents in ancient Roma. External and individual behaviours are created for these agents. Agents in this model could react to both actors and the play. And their emotion state, social class are taken into consideration. [10] provides a basic group-based behaviour model which is created based on observing 7,405 informal groupings of people and 1458 people in various conditions. Its result gives a practical reference to simulate a crowd: 71% of all groups contain 2 individuals, 21% contains 3 individuals, 6% contains 4 individuals and only 2% contains 5 or more people. Soraia Raupp Musse and Daniel Thalmann introduced a hierarchical model for crowd simulation, VICROWD [1]. The feature of VICROWD is its group-based structure, involving with intelligence, memory, goals and cognitive sense. Hierarchical model is the core part of framework design. But it is still only one part of the whole project. Crowd simulation involves many parts of implementation [4, 9]: modelling, variation, behaviour setting, animating, rending, etc. [3] gives a detailed method of visualizing a large-scale number of pedestrian crowd and gives a clear framework of crowd simulation pre-processing including model templates, material database and motion database. This work provides a modified model which aims to improve the efficiency of simulating pedestrians in large-scale environment (thousands or more) in real-time applications. The work indicates that making different models and then sharing the same data with proper variations at run time is an efficient way to reduce memory cost. In this work the data of each character is 17 MB including model template, motion and material database. A point-based method accelerating technique is introduced in [3] for rending issue. Furthermore, there are still many other works like 2D-image based solutions in [4, 19, 20]. But 3D visualization issues are still universal. [25]'s algorithm searches through each state of FSM. High-level state is represented as a library of character poses. A global planning technique helps arrange the sequence of a collection of motions. [4] utilizes the LOD(level of detail) and ROI(region of interest) to reduce calculation times. By calculating the distance from vertex to the camera, LOD indexes which represent different human template resolution are labelled on each individual. ROI, region of interest, applies different motion planning methods to different templates. The higher the ROI value of the character is, the more complicated planning algorithm is applied to the character. In the contrast, lower ROI value saves more memory usage during simulation but the character has a more simple behaviour.

MASSIVE [22] is originated from WETA digital studio. It was first created for the epic battle scenes in Lord of the Rings Trilogy and then was used in many films [12]. Common techniques like A* algorithm and FSM [17] are adopted in simulation [6]. Two main crowd simulation products are provided by MASSIVE: Massive Prime^{TM} and Massive Jet^{TM} (Simplified Version). They both provide friendly user interfaces like Motion Tree Editor, Brain Editor, Cloth Editor and Action Editor to facilitate artists and visual effects professionals. Some main plug-ins in MASSIVE like Motion Tree Editor and Action Editor are still good references to my project.

LEGION SDUDIO, an agent-based pedestrian simulation software, owns a huge database over 1000 hours video footage with more than 8 million pedestrian movements and spanning 14 cities over the world. It is very powerful for simulating pedestrians especially on path finding and collision avoidance. Furthermore, SIMULEX, which is designed for simulating evacuation from specific building, is an agent-based software from Integrated Environmental Solutions (IES) Ltd. A.I PLANT, from PRESAGIS, also involving with path-planning and collision avoidance, provides agent-based crowd simulation in virtual environment. The way to collect data of above software is a good for simulating spectator crowds in my project.

Chapter 3 Project Design

3.1 Overview of framework

The core aim of our project is to design an efficient and easy behaviour model for production tools which help animators set up any spectator scene in sports. Except the core aim, we also produce a simple production tool which is embedded in Maya. The friendly user interface with few external controls is very easy to learn. An animator without programming experience can also manage it quickly. The only thing he has to do is just clicking on some option buttons. This flexible and easily-control system is easy to be maintained and embedded into other crowd populating works. As shown in Figure 8, our project mainly includes three parts:

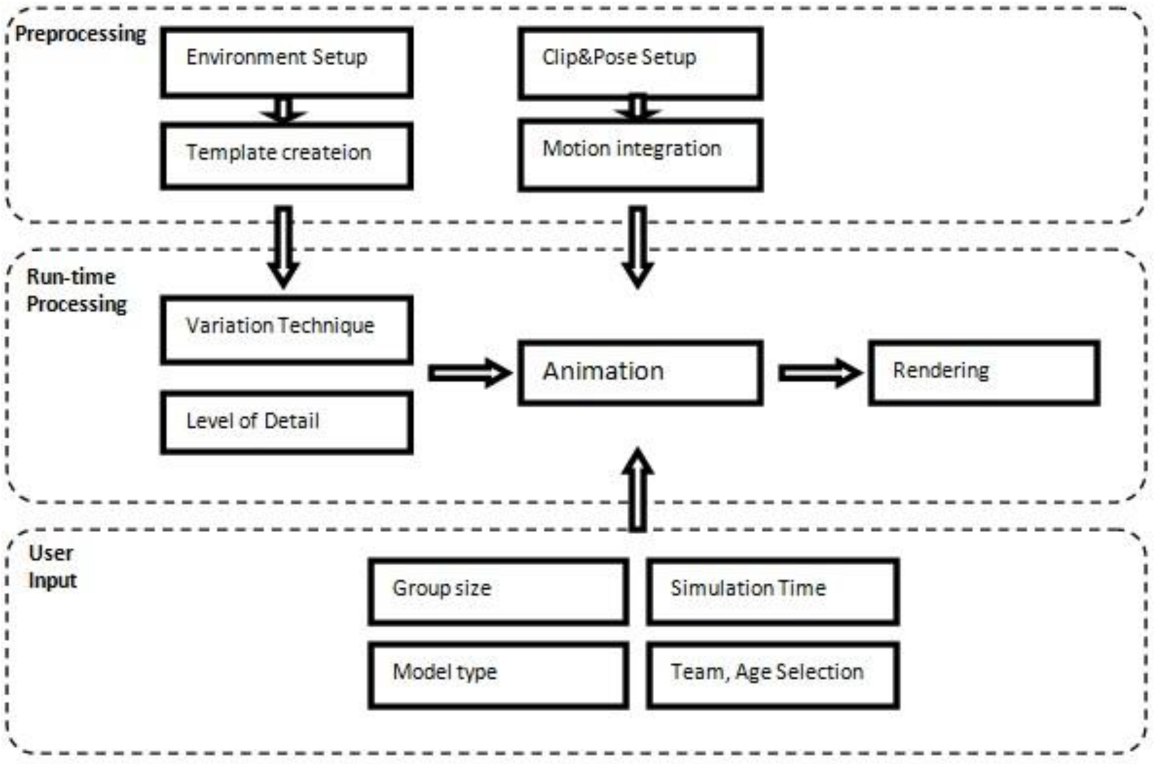


Figure 8. Framework of our work.

Pre-processing: Before the run-time simulation, some data like motion data, human template, material, and environment setup must be prepared in advance [3]. These processes speed up run-time simulation and save a large amount of time. By setting up a motion library, we are able to provide various motions for the template during simulation. For each agent any motion could be easily imported and scheduled by simulation system. Human template also should be modelled first to save the modelling time during simulation. Then the system could pick up any human template according to type demand.

Run-time Processing: During the run-time process, by importing specified models, the system resizes each character based on default rules. Then motion data are imported for creating virtual human behaviours and actions. A behaviour generator is constructed to help system to automatically search for the next motion based on behaviour rules and events until the end of simulation.

User Input: There are only few parameters need to be transferred into the system: The user is able to set his own preference on simulation duration and crowd size; Specified model needs to be displayed in the scene; Setting up the age group of crowd like children, adult or mix. This means user can easily set up a scene with crowds of children participating a sports game in an elementary school or a scene with crowd of adults taking their children joining in an Olympic games in a big stadium; setting up teams of fans there: If there is only one team of fans there, when an event like kicking a goal happens, all the fans will cheer for the goal. They are treated like fans from the same team. However, if two-teams-of-fans option is set up, fans from respective team will support their own team; LOD can be turned on or off by the user according to different situations.

3.1.1 Process of system

The main procedure of our project is described as below:

1. Get input information from user.
2. Initialise the scene.
3. Generate crowd.
4. Resize each character's skeleton.
5. Set up the default pose for each individual which helps import motion sequence later.
6. Use behaviour generator to set the basic information of crowd motion.
7. For each character, execute the chooseMotionClip() procedure, system will automatically pick up various motion from motion library
8. By analyzing external and internal parameters, system searches for next motion automatically
9. Loop until finish scheduling all the motion for each character.

The whole process is shown in figure 9:

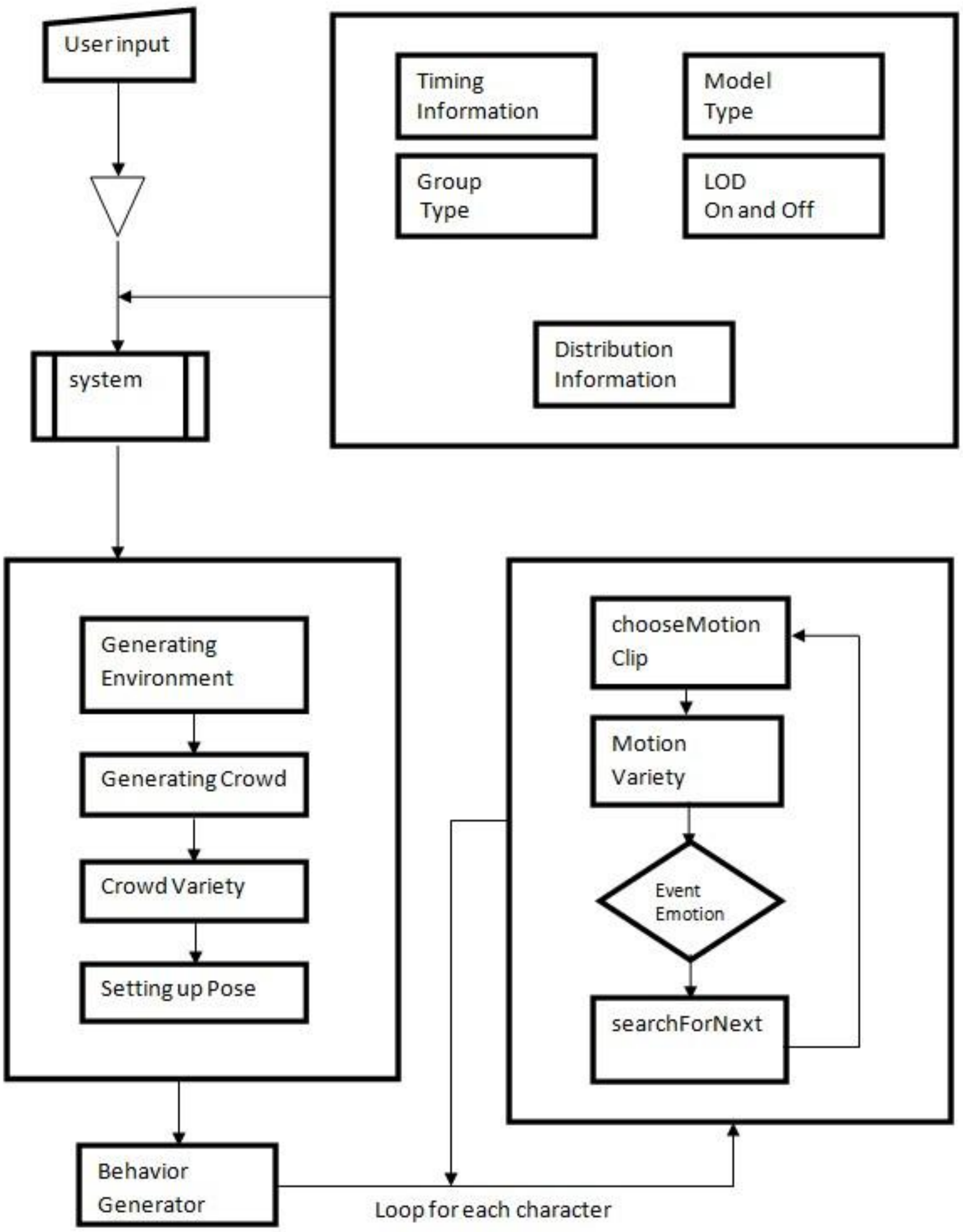
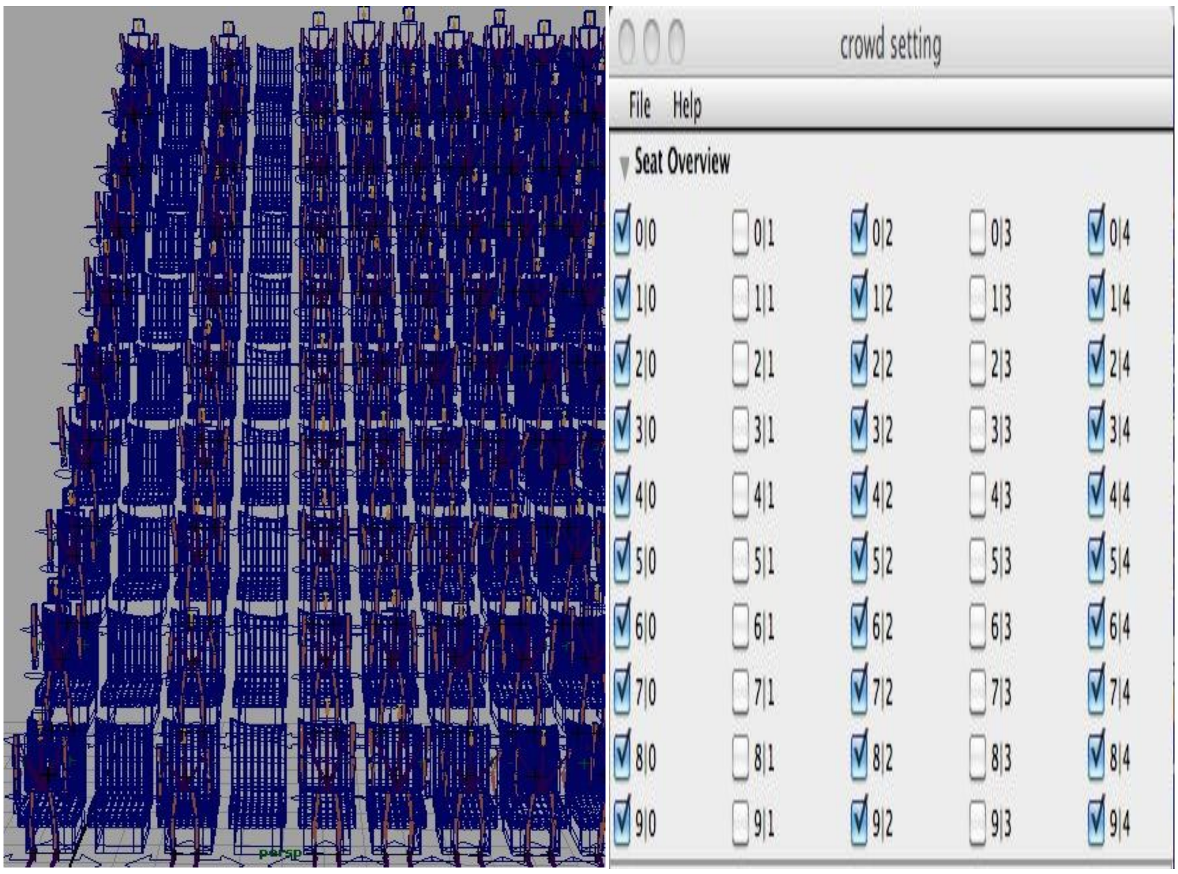


Figure 9. Process of our simulation system.

3.1.2 Distribution of crowd

The first step for creating the crowd is the distribution of the crowd in a scene. The seats in the scene are generated automatically and then user is able to choose the allocation for each audience. In Maya when same objects imported into the scene, each object's name is assigned to a different suffix. The system has to parse the suffix of each seat index to distribute each audience. In figure 10, we can see a crowd is created in the left picture. And in column 1 and column 3, there are no people generated. The right picture shows the corresponding user input. Because the whole simulation is done by laptop, there are some limits to the amount of processing. In our work, due to this factor, we can not load thousands of characters. But hundreds of people are already enough for test. In a word, our system would scale to any number of spectators if given enough processing power. In this scalable system, users can set up from small-size (hundreds) to large-size (thousands) of crowd according to their own preferences.



✓ 0|0	☐ 0|1	✓ 0|2	☐ 0|3	✓ 0|4
✓ 1|0	☐ 1|1	✓ 1|2	☐ 1|3	✓ 1|4
✓ 2|0	☐ 2|1	✓ 2|2	☐ 2|3	✓ 2|4
✓ 3|0	☐ 3|1	✓ 3|2	☐ 3|3	✓ 3|4
✓ 4|0	☐ 4|1	✓ 4|2	☐ 4|3	✓ 4|4
✓ 5|0	☐ 5|1	✓ 5|2	☐ 5|3	✓ 5|4
✓ 6|0	☐ 6|1	✓ 6|2	☐ 6|3	✓ 6|4
✓ 7|0	☐ 7|1	✓ 7|2	☐ 7|3	✓ 7|4
✓ 8|0	☐ 8|1	✓ 8|2	☐ 8|3	✓ 8|4
✓ 9|0	☐ 9|1	✓ 9|2	☐ 9|3	✓ 9|4

Figure 10. The distribution of a crowd.

We should loop for each character. And parse the chair index and then execute the distribution of a crowd:

```
 FOR EACH CHARACTER
{
   ......
   select -r character:root;
   duplicate -un;
   $charIndex = `checkBox -q -label $checkBoxSeatName[$i * $column + $j]`;
   tokenize $charIndex "|" $buffer;
   $tokenFirst = $buffer[0];
   $tokenSecond = $buffer[1];
   ......
}
```

An instance of a model in Maya behaves the exactly same as the original model. This means if an audience is set up as an instance, there must be another one behave the exact same as him. Figure 11 shows that when the original model lifts his right hand, the instance of this model also lifts his right hand. Although this technique could be tolerated in many cases in film and game production, in our case, in order to give the maximum flexibility of control crowd, instance is not used. Instead, when we set up a new human model from human template, we copy all the upstream nodes besides the selected nodes (also including the connections). This means the new model has all the functions of original model. In our project, we don't have effort to focusing on making many delicate motion clips into the motion library. It is animators' work. So there are not enough gestures and actions shown in our project. However, in our project, as long as there were enough motion data created by artists or relative professionals stored in motion library, each agent could have his own distinctive behaviours and actions rather than repetitive motion from same human template.

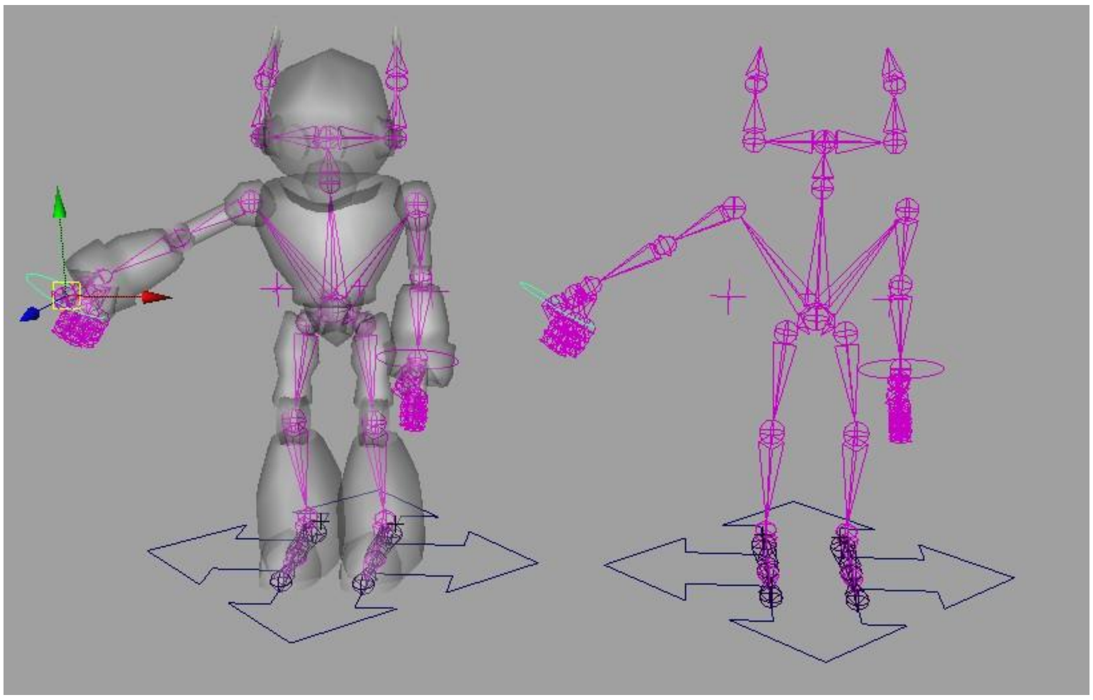


Figure 11. Left is the model template. Right is the instance of model template.

3.1.3 Motion research

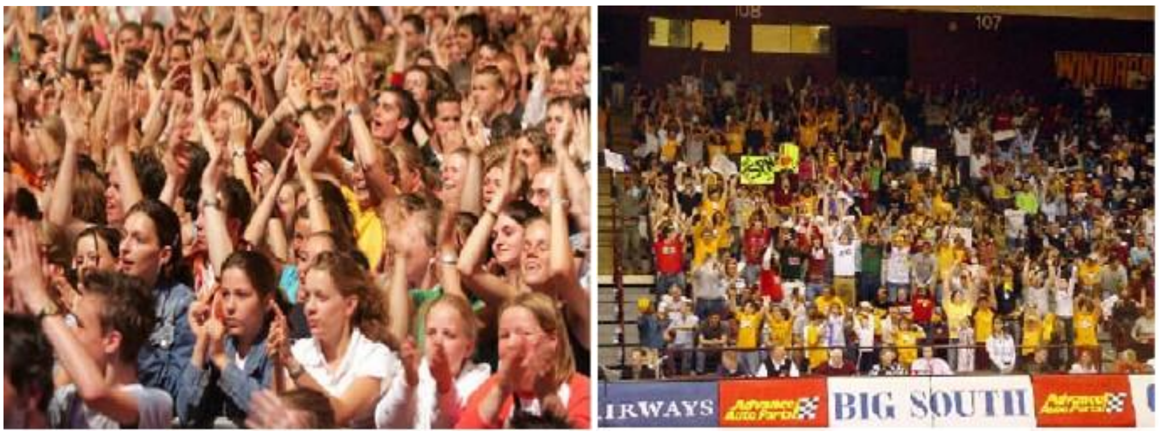


Figure 12. Applause and cheering scene of real crowd

Relative research has been done on the observation of audience-performer interaction. Most of the motion patterns (shown in figure 12) in different types of scenarios have been analyzed from physical and emotional aspects [20]. In our project, in order to get a basic understanding of spectator behaviour representation, we mainly use two methods to study the behaviour representation. First, we study papers which are related behaviour, audience interaction and concert performance analysis. Second, we study how spectators behave in different circumstances by watching various videos from website. In most situations, spectators with fundamental culture knowledge are clear about the timing of cheering, clapping, etc. But some factors like performance type, performance environment might have different crowd responses. Research has shown that government speech has less motion types like applause types than a music live show. In addition, music concerts also have different types of audience motions (behaviours): A rock or R&B performance motion model can not be directly adapted to a classical concert behaviour model.

Some basic rules of audiences are discussed below:

1. It takes less than 1 second from the start of a "cheer" action to its maximum exciting level.
2. During the performance, idle or stand motions of a crowd look more randomly in sports game than other performances like those in concerts.
3. If more than twenty five percent of people in a crowd begin to clap, the whole crowd will quickly clap together.
4. The more excited, the more various of one's movement

Generally, there are mainly 3 common-used spectator behaviours: (1) applause (2) cheer (3) unstructured body movement. From the above research, a classification of the behaviour in some common performances is given in figure 13 [7]:

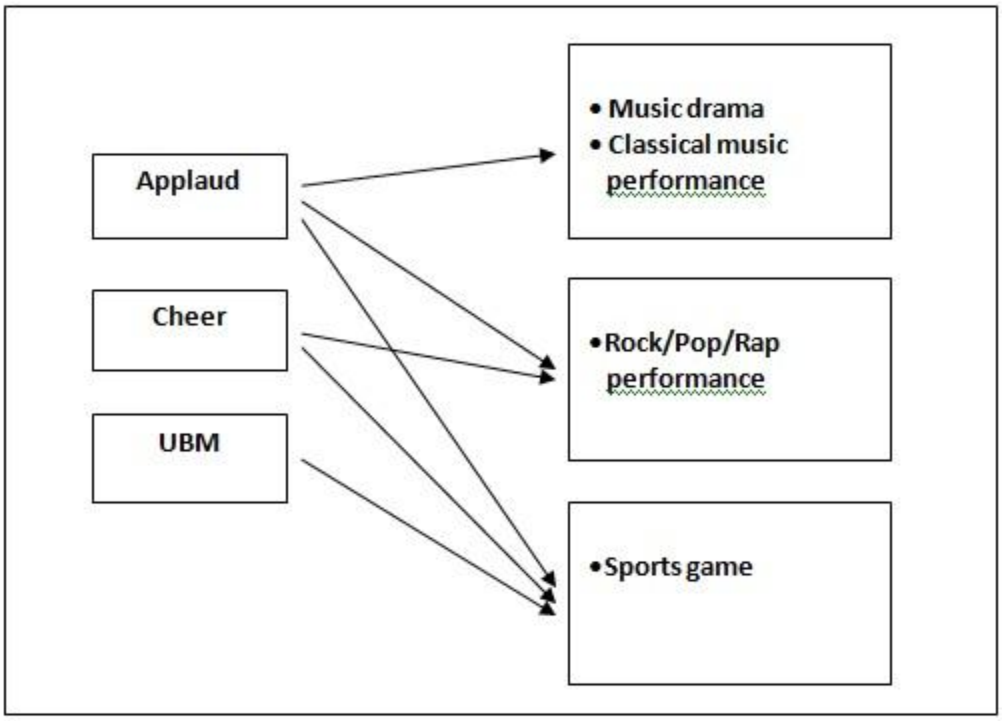


Figure 13. Classification of the behaviour in sport and music performances

3.1.4 Crowd reaction research

In an agent-based simulation environment, agents' behaviours are often triggered by a set of rules. Here the behaviours refer to behaviour models and rules refer to emotion model [11]. Social, psychological factors of audience crowd in public are investigated in our work. Gary Lupyan and Ilya Rifkin provide a general outline of the personal, social, psychological factors of audiences at a chamber music festival [8]. It details the motivation, intention, temperament and habit of audience on both individual level and group level. Many actions and interactions of audiences are described in detail, which provides a detailed reference of natural crowd behaviours to our project. Some actions and events can be directly used into crowd spectator simulation. Soraia Raupp Musse, Branislav Ulicny and Amaury Aubel propose a design model for applause, which can be directly applied to crowd design [10]. The author defines two concepts: benefit and cost. This "benefit" is proportional to the performance and impressiveness of an event. The cost is defined as embarrassment cost and physical cost: Embarrassment cost is an "embarrassment threshold" parameter. It shows the degree of each agent's willing to applause. The physical cost means the applause relevance to time. Generally, as time elapses, the proportion of people who are clapping decreases. The idea here is simple: The benefit (expressiveness) is set as b, and two costs were set as c_1 and c_2. Agent claps when b > $c_1 + c_2$. In opposite, B < $c_1 + c_2$. Figure 14 shows how impressiveness of event is related to duration of applause and proportion clapping. Also two hypotheses [10] are confirmed by analysis of real time recording of audience performance. One indicates that applause often starts very quick but stops slow. Another one is that if one's applause is ignored by others, which means no one follows his applause, then the clapping will stop immediately. In sum, on a social and psychological level, this work gives a very clear view of how to simulate spectators in public.

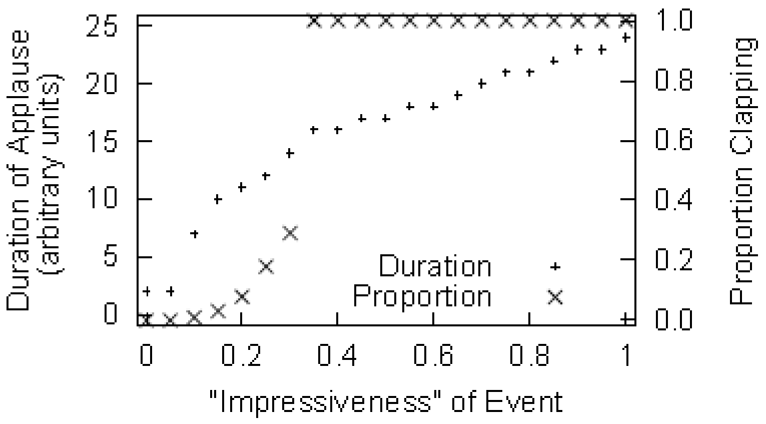


Figure 14. Effect of impressiveness on proportion of audience clapping and duration of applause. Source: Ref. [8].

3.2 Behaviour model

In our project, crowd are formed by groups: It is completely scripted and based on behavioural rules. Crowd includes information which is embedded in each group. Group consists of agents: the size is decided by the information from crowd. It also can be scripted and guided. And they can react to specified events based on different level of expressiveness index during the whole simulation. Each agent just acts according to the parameters passed from group.

As described in Figure 15, the methods for setting audience behaviour are scripted control and rules control respectively. The scripted control is embedded in crowd structure which means if no event happened or no external control existed, crowd will behave under scripted control without any other influence until the simulation ends. The external control includes some events or some other behaviour related rules. If events happened or user gave an external command, some original behaviour or motion of agents will be replaced, and then agents will behave according to external control until the trigger event ends or simulation ends.

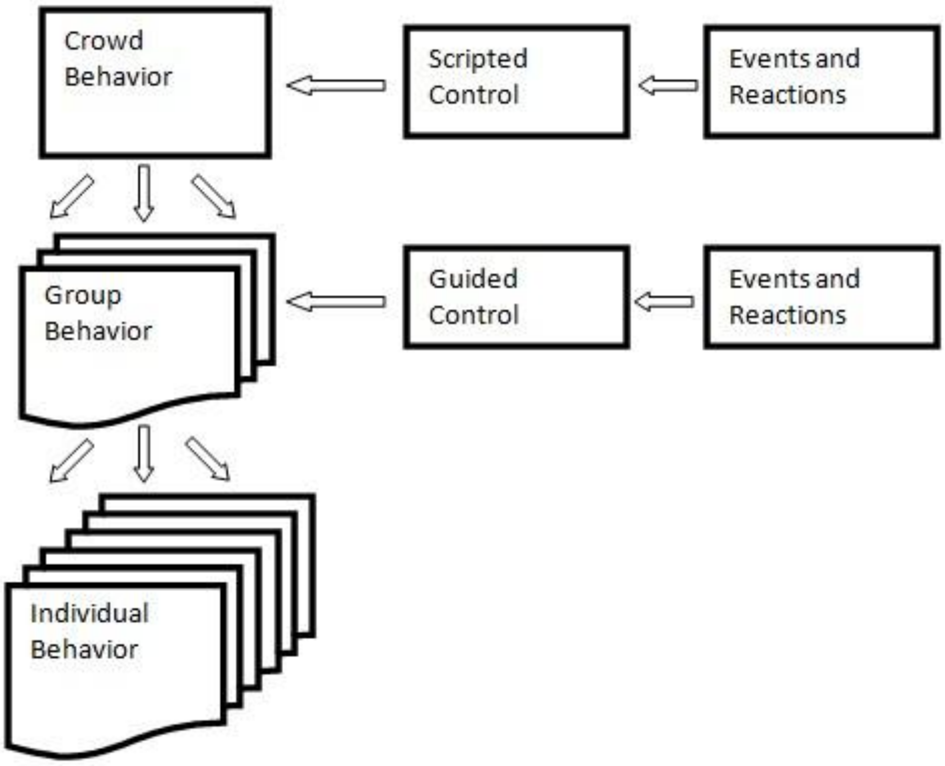


Figure 15. Behaviour model design in our project.

3.2.1 FSM

3.2.1.1 Introduction

FSM, namely finite-state machine, is very efficient on modelling various problems in many research areas such as EDA, communications protocol, biology, linguistics, etc. This behavioural model has already been widely used to design software in crowd simulation. [26] gives a comprehensive illustration of how a structure of FSM is designed in animation production. A FSM system contains a finite number of nodes and connections. Here nodes are the states we will discuss in our project. And connections, namely transitions, are a set of instructions to be implemented once some events are triggered or some conditions are met.

The CFSM (crowd finite state machine) designed in our work is in charge of each individual's motion arrangement. Figure 16 shows part of CFSM in our project: Each circle means a motion state. And each motion state has its own clips which are shown inside the blue circle (in this graph only *cheer* state's clips are shown because of picture space). An arrow means a transition between two actions.

A state in BFSM is represented as a node which contains all its children information (some including itself as child). The whole simulation time is defined by user. Then the system will automatically expand the motion sequences within the specified time period based on external and internal control. This means character's motion sequence is created and extended by continuing finding path in the motion-tree based on behaviour rules until the time is due.

The start state is the default pose which we set up before the run-time processing. Then as the run-time simulation starts, in this graph, the *start* state transits to *sit* state. And then depending on different events or some external controls the state will switch to *turn left*, *turn right* or *stand* state. Generally, each state from our CFSM can be regarded as a high-level motion. And as the *cheer_1* to *cheer_4* clips which are contained in each state are low-level motions. By scheduling a linear sequence of clips according to various conditions and constrains, an agent's behaviour is finally produced from CFSM.

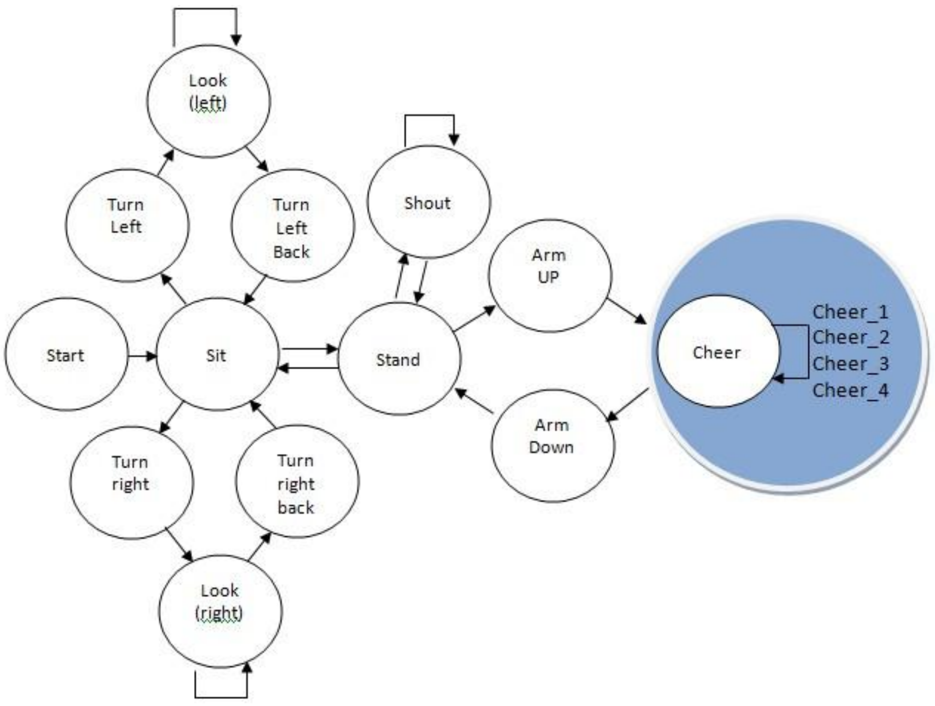


Figure 16. Crowd finite state machine.

3.2.1.2 Transition of FSM

In order to have a more flexible control on these states, each state only has a length of 10 frames. If a state consists of very complex movements, this state will be divided into several parts. This method not only provides more flexibility on BFSM, but also saves the memory space of clip library. In our work, the *cheer* state of a spectator is actually cut into three parts: *Arm_up*, *cheer*, *Arm_down*. For example, if a character wants to perform a cheer action, first the system will schedule an *Arm_up* state to this character and then the *Arm_up* state will automatically searches for *cheer* state and transits to this state. Furthermore this character might want to cheer for certain duration of time. Therefore system allows *cheer* state to last for a specified period of time by controlling its repeating times. At last when this character does not want to cheer, an *Arm_down* state is imported into character to finish the whole motion. This is a cheer motion cycle. After the cheer cycle, the character returns to the *stand* state and starts another state based on behaviour rules.

In order to make the transition of CFSM fluid, each clip must have a default pose at the start and end of its duration. Figure 17 describes a fluid transition of a cheer state. In order to have a fluid cheer motion, character should first transit from *Arm_up* state to default pose. The end frame of *Arm_up* state and the start frame of the *cheer* state share the same default pose. The transition from *cheer* to *Arm_up* is the same. In conclusion, default pose is a good method for fluid transition between different states in CFSM.

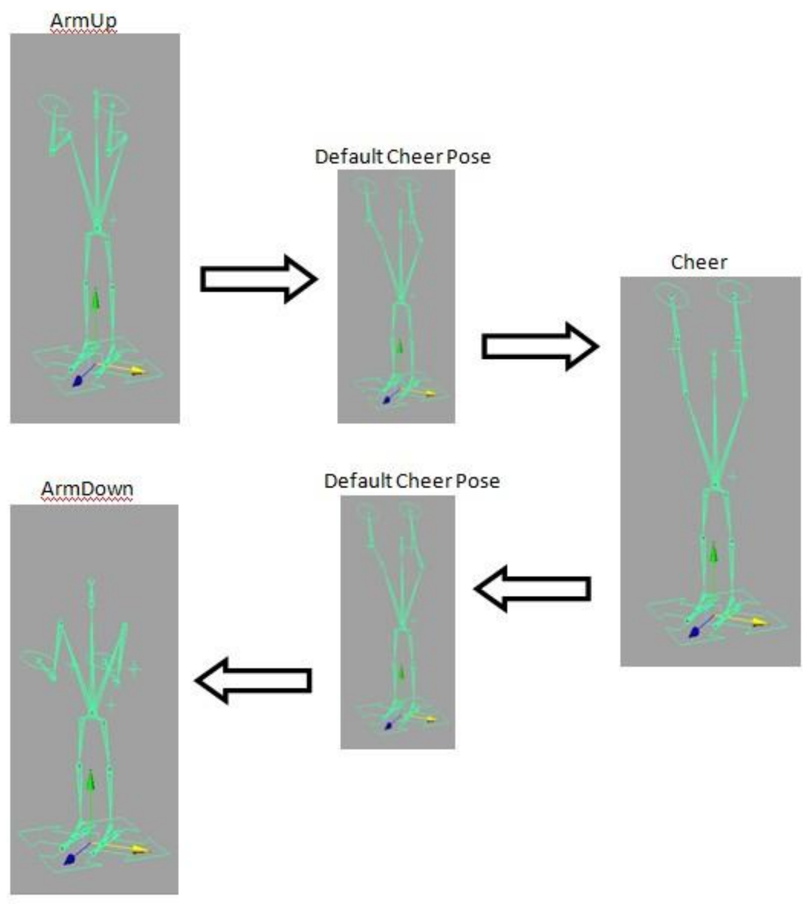


Figure 17. A fluid transition of cheer state.

3.2.1.3 Code implementation

In order to set up each autonomous agent, a self-searching structure is designed for behaviour generation. By passing template type, motion type and timing parameters into this structure, system is able to process each clip's attributes data information and then calculates the start and end frame of each clip on the timeline. Then based on these information, the system is able to arrange the motions. After receiving the timing information of each clip, system starts to search the next motion for each individual from the motion graph. Finally it gets the next motion type and returns this parameter into the nested structure to start another clip scheduling process. This is the basic principle of how the system schedules each state. The code below shows the self-searching structure of our system.

```
global proc int chooseMotionClip(string $charClipSet, string $motionType, int
$startTimeFrame, int $allFrameLength)
{
……
     $calculateStart = `getAttr ($clipName[0] + ".startFrame")`;
     $calculateLength1 = `getAttr ($clipName[0] + ".sourceStart")`;
     $calculateLength2 = `getAttr ($clipName[0] + ".sourceEnd")`;
     $calculateLength = $calculateLength2 - $calculateLength1;
     $endTimeFrame = $calculateLength + $calculateStart;
……
          $motion = searchForNextMotion($charClipSet, $motion);
          chooseMotionClip($charClipSet, $motion, $endTimeFrame, $allFrameLength);
}
```

The function searchForNextMotion() returns a set of motions that audience could choose to perform next. All the motion information comes from CFSM. [25] provides a method that sets weights index to different states: For example, jogging-and-turning(JAT) often has more cost than jogging forward(JF);Jumping requires more effort than JAT and JF; Stop and waiting motion are set to the highest weights. By this method the state which has more weight is less likely to be chosen by the system. Base on weight-based algorithm, our new weighted-based algorithm has been designed. We set weight index for both states and the clips in each state. This method provides a more flexible control on crowd behaviour.

3.2.2 Behaviour variety

Various behaviours are classified into different levels. On the first (low) level, each audience has a list of basic motion clips which can be called as innate behaviours. The basic motion clips are the “inborn” behaviour of each individual. They can idle, look around, talk, etc. When these scripted behaviours are applied, crowd automatically distributes these behaviours among the spectators. There is no need of external control throughout simulation on this level. On the second (high) level, rules and events have been applied. They have higher priority than innate behaviours on the first level. This kind of structure gives a clear priority instruction on agents when they receive various controls at the same time.

In order to make audience not repetitive and emotionless, two possible solutions has been provided by [26]. One method is adding multiple clips into the same state, rather than making one clip for one state. The other method is adding procedural noise to the state to produce variety. Two solutions are both applicable in our project. After comparison, we choose the first method. Because in Maya, the change of each motion will directly reflect on the change of its animation curve. Thus, in order to make more varieties on a motion, we are actually making more varieties on its animation curves. But each motion’s animation curve is encapsulated in each clip. During the whole simulation, there are tens or hundreds of clips needed to be instanced on the timeline for each character set. More importantly, there are hundreds or thousands of character sets in a crowd. Thus it is not very convenient to add procedure noise on the animation curves. The first method discussed below is more space-saving and efficient.

All the characters in this project import motions from the same FSM. Thus in order to add variety into the crowd, in each set (state) system has several similar clips which share the same goal but with subtle differences. Figure 18 shows a part of our clip library. From the picture you can see each motion has been added with four different motion clips. They have their own indexes "one" to "four" which is convenient for searching. The system assigns these motions to each individual randomly or based on some rules. That is one of the method to add variety to the crowd.

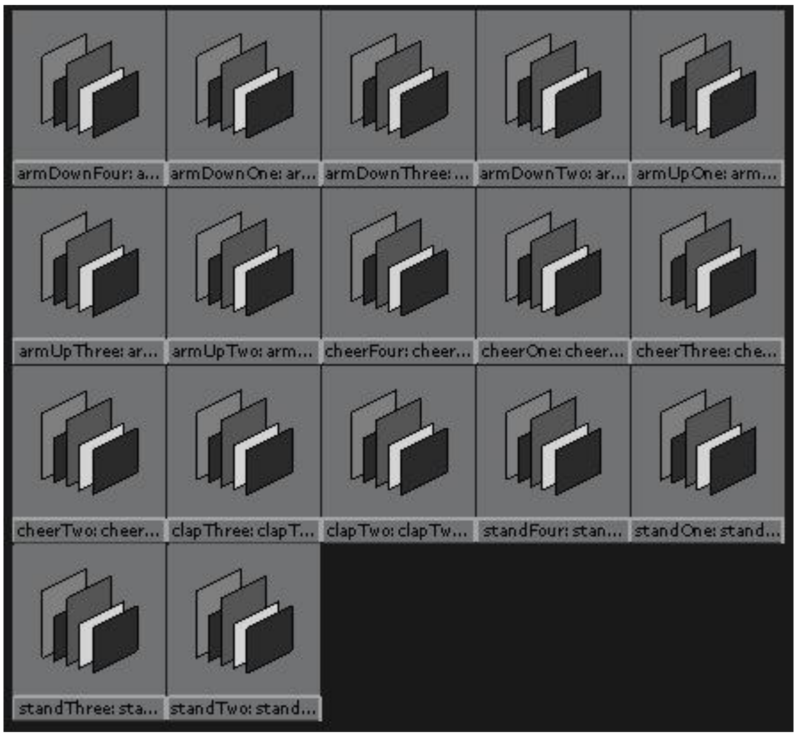
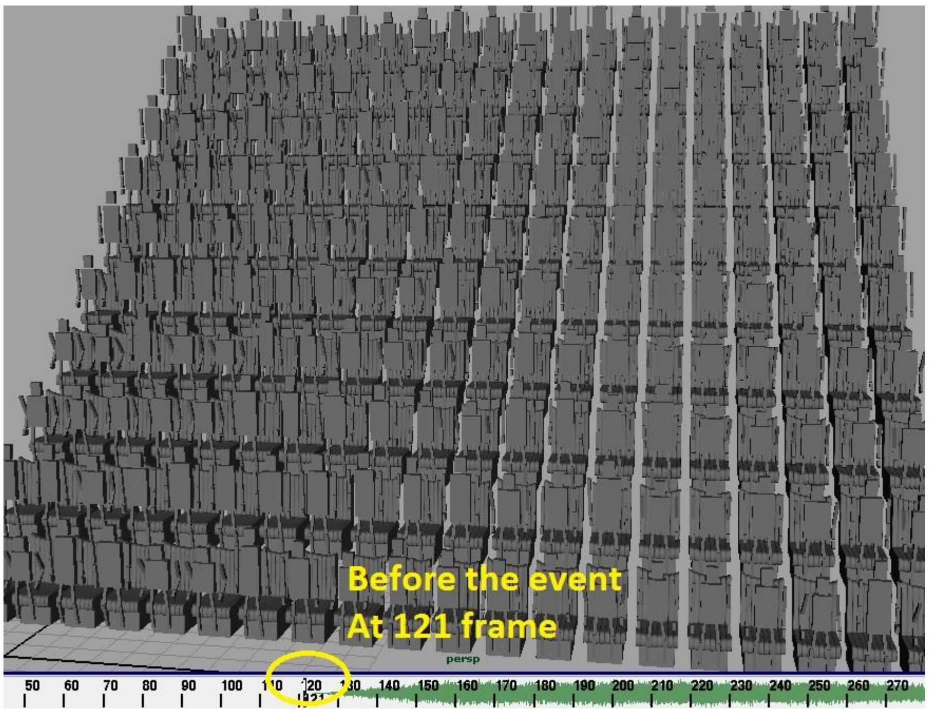


Figure 18. A part of our clip library

3.2.3 Reaction to event

Crowd reaction is controlled by global parameters. A crowd is able to react to any event when relative event signal arrives. Figure 19 shows how the crowd react to the event. This is an event that when the announcer exclaims someone kicks a goal, the whole crowd react to this event immediately to cheer for the goal. This is done by controlling the timing information of each clip. In the left picture, before frame 120, no one would like to cheer, they are just watching games. You can see from the timeline no announcer's voice appears. On the right, we can see after frame 120, when the announcer begins to shout and being excited (you could see from the sound volume), the event triggers cheer motion. This means a cheer signal is sent to each character and then all of them begin to cheer.



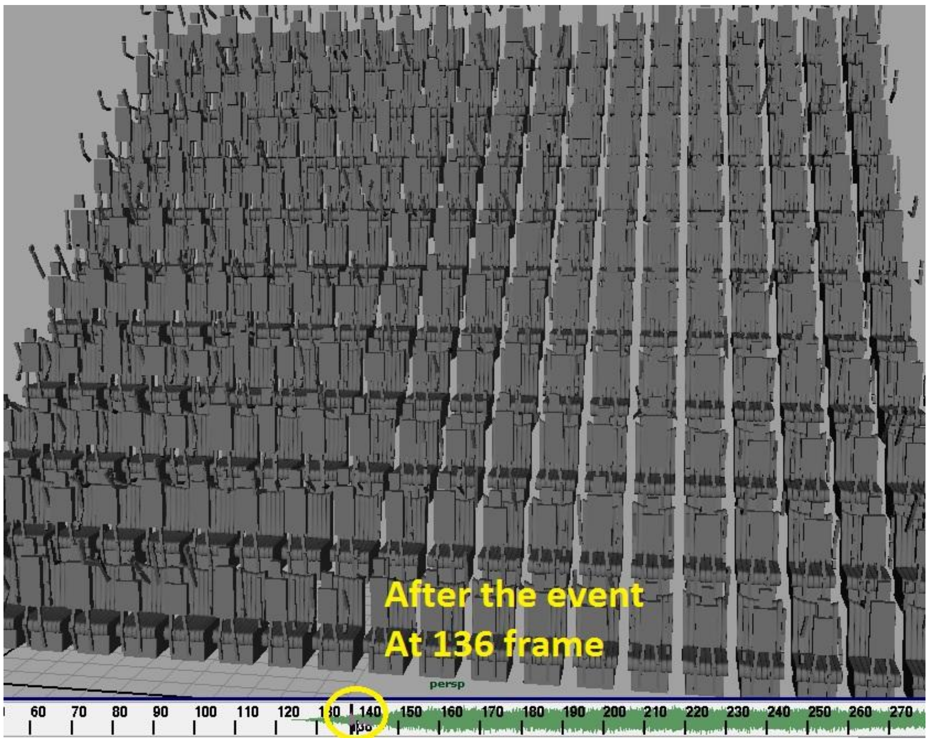


Figure 19. An event is triggered at frame 122.

Before the frame 122 no one cheers. (up)

After the frame 122 everyone cheers. (down)

According to different embarrassment level, people would behave differently. This means different events might have different influences on audiences. Some events with lowest excitement only make few people cheer. Events with middle level of excitement make more people cheer and clap. And those with high level excitement not only trigger agents to cheer, but also make people jump and do more exciting motions. In our project, global parameters are added for emotion and events control. For example, the global embarrassment parameter is able to control clap duration of the crowd. If the embarrassment value is very high, the crowd will not like to clap or clap for only a short period of time. In contrast, lower embarrassment value leads to longer period of time for clapping. The picture (up) in Figure 20 shows at frame 250, crowd are still cheering for the goal. This means, with a lower embarrassment index value, crowd cheers for a longer time. The picture (Down) shows at frame 250, crowd already stop for cheering. This means, with a higher embarrassment index, crowd cheers for a shorter time. Global event parameters are also embedded in the scripted behaviour. Different events have different variables and different weights. When the system searches for the next motion of each agent, it will always check out whether some events happen. Then system can decide what the next motion is for the agent. The sample code below gives a simple demonstration of how the global event and embarrassment parameters control the clips on the timeline.

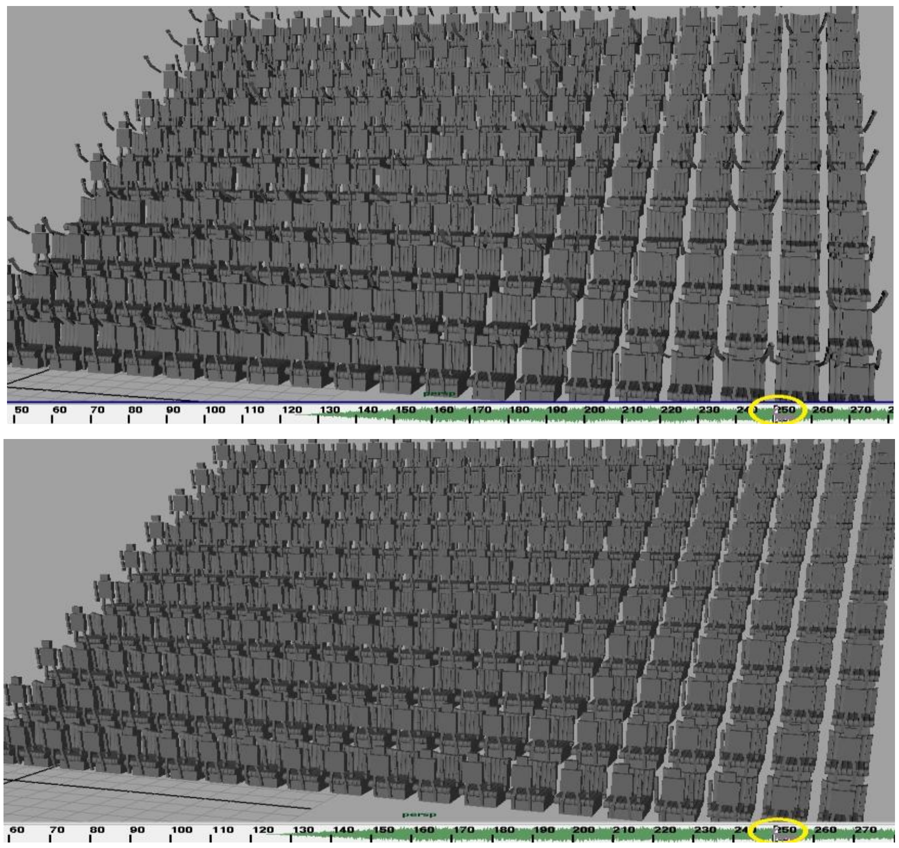


Figure 20. Crowd cheer for a long period.

$embarrasment = 0; (up)

Crowd cheer for a short period.

$embarrasment = 1. (down)

```
global int $sceneEventIndex;
global int $embarrassment;

 ......

 if ( $embarrassment == 0 )
     {
       Set longer time for cheer

     }
   else if ( $embarrassment == 1 )
     {
       Set shorter time for cheer

     }

  ......
```

3.2.4 Interaction

With the development of CG industry, interactive animation technique would gradually replace traditional linear animation technique. However, interactions between agents are more difficult to generate than traditional linear animation. For example, when we want to set up a hug motion for two friends in the crowd, two matched motions from two individuals should be under a precise time control and adaptable to any subtle variation during the simulation. Failing on synchronizing this animation task would lead to agents' arms or heads cross through each other. With the deep research on procedural animation technique, interaction between virtual humans becomes more and more plausible. This kind of close interactive activity is inevitably needed in crowd simulation because people are very likely to engage in social and emotional communication.

Interaction in crowd simulation needs a complex structure of motion library. In our project, our aim is not to make a very complex motion library. Creation of a complex motion library should be done by the corporation between programmer and animator. Hence in our work a simple interaction structure is provided. We regard a small group of people who are interacting with each other as a single entity. They get the same index which means they are in the same group. And then system is able to import synchronized animations into this entity. Thus the spectators in the crowd could be divided into different groups according to this kind of external control. When a team wins, different groups of people behave differently. Figure 21 illustrates when a team wins, one group of people cheer for the winner, while the other groups of people don not cheer for them.

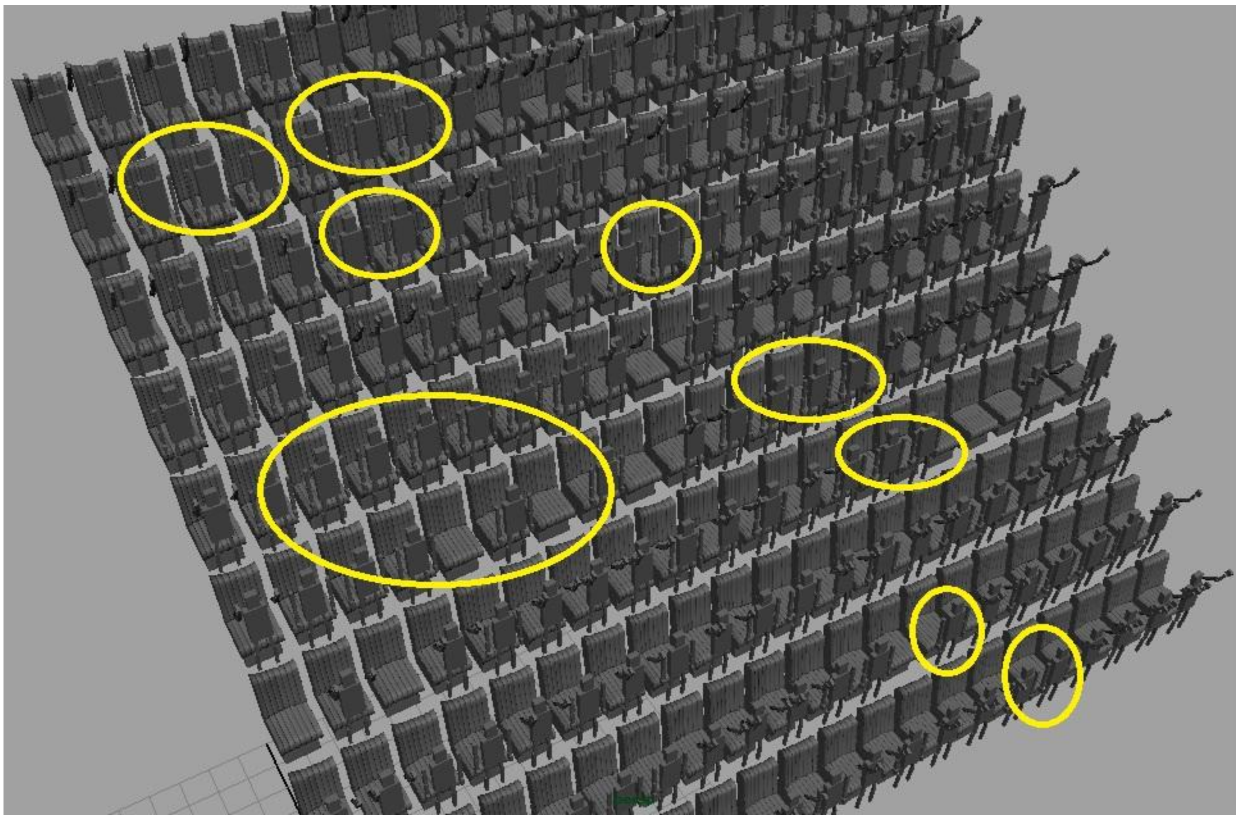


Figure 21. People tagged by yellow circle are some of the fans whose team loses. They don not cheer for the winner.

3.3 Motion arrangement

3.3.1 Motion library

In Maya, Trax editor is a powerful tool for arranging motions for one or more characters. We could just use mouse to drag specified clips into each character’s timeline to make them active. However, when we are facing thousands or ten thousands of crowd, it is impossible to manipulate each character by hand. Thus we need to choose a programming language which could be used to manage the motion library. Mel (Maya embedded language) is chosen because its platform-independent and efficiency feature. In Maya, every object is treated as a node. Figure 22 gives an overview of the nodes we used in our project and next we will discuss the connections between each node and how we utilize these nodes to manipulate crowd motion.

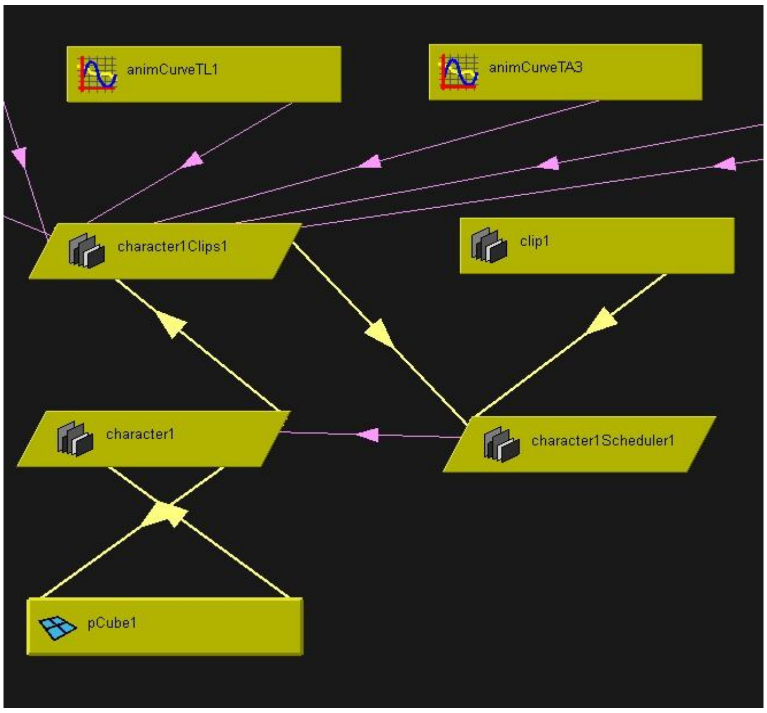


Figure 22. Node used in motion library.

Figures 22 shows how a single character in a crowd connected with a motion library. We can see from this figure that there are seven nodes in this graph.

The two nodes on the top änimCurveTL1, animCurveTA3 represent the two example source motion clips in our project. Their arrows pointing to character1Clips1, the motion library node means these clips are included in the motion library.

Clip1 shown in the graph has an arrow pointing to the character1Scheduler1. It means the instanced clip which is instanced from source motion clip and then scheduled onto the timeline for each character.

The Character1clips1, which represents motion library node, manages source motion data and their motion curves. Each motion clip is represented by an animation curve node which is linked with motion library node, which means we can also change the motion data by adjusting the animation curve. The motion library node can analyse its attached motion data to get their attributes without calculating current time. That is very convenient for our work to schedule motion sequences. The source motion data has their own attributes including Start Frame, Pre Cycle, Post Cycle, Scale, etc.

The character1scheduler1 is the clip scheduler for character1. It manages the placement and blending of character1’s motion clips. From the figure we can see an arrow pointing from character1clips1 to character1scheduler1, which means this clip scheduler has been connected to motion data library. All the motion data in clip library is able to be instanced unlimited times by the clip scheduler. This saves much space for animators to store motion data because some cycle motion like clap, cheer can be repeatedly utilized rather than setting up a new one when needed again and again. Although these instanced data share the same source motion information stored in the motion library, we can change their attributes with the help of clip scheduler when we place each clip.

In a word, motion library manages original motion data and the clip scheduler arranges instanced motion data. This means we should first produce raw motion data in the motion library, and then use clip scheduler to place raw motion data on specified time period. And one individual can only link to one motion data library and one clip scheduler. Figure 23 shows source clips like stand (Index), cheer (Index), etc have all been instanced for many times.

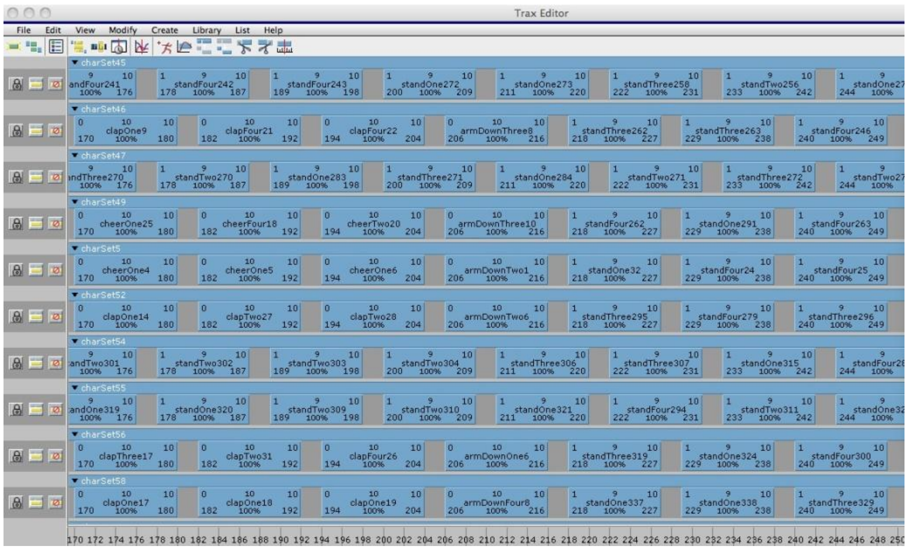


Figure 23. Every source clip is instanced for many times.

3.3.2 Absolute and relative motion sequence

As discussed above, after distributing each audience to each specified position, the system then starts to import motions based on behaviour rules into each agent. Source clips, which are shared by all the agents in a crowd, are made by exporting motions from model template. The model template has its own world and local coordinate information which also have been transferred into each exported clip. When we apply the source clips into each audience, the clips in each audience still preserve the information from original template. Thus an agent whose root position is at (a1, b1, c1) cannot have right actions because source clips are originally produced at the (a0, b0, c0). As a result there will be some exaggerated stretch or bend on the body.

To solve this issue, we need to enable clips applied to each individual using its local coordinate in the scene, not the world coordinate. In another word, the clips instanced from source clips should have a “relative” property rather than an “absolute” property. Figure 24 is a comparison of the result between skeleton applied into relative and absolute properties. The picture (up) shows two characters have the same motion at the same frame. The charSet3 has a “relative” attribute for its motion clips. The other charSet4 is “absolute”. The left figure (down) shows the character behaves well after being set a relative property while the right shows the result of absolute properties. You can clearly see the stretch of the elbow and skeleton scales back to the original human template. This is not the result we want. Therefore during the scheduling process on the timeline, an “all relative” attribute should be added during programming in the instanced clip.

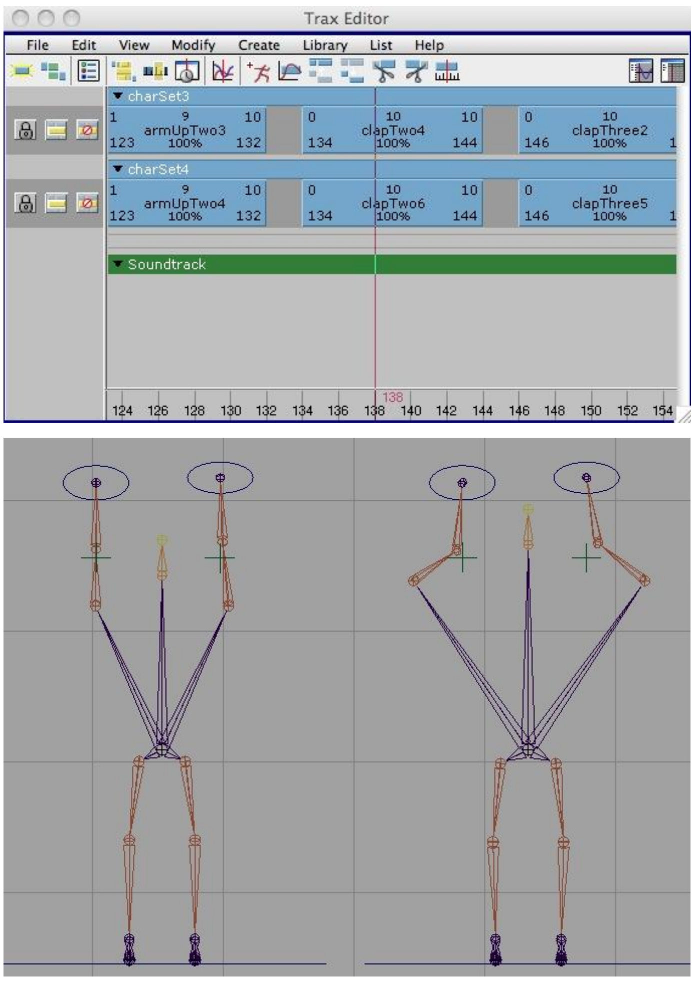


Figure 24. A comparison of relative and absolute properties

```
clip -copy ($motion + $motionIndex + ":" + $motion + $motionIndex + "Source");
        $clipName = `clip -pasteInstance -sc 1 -startTime $startTime -ar $charClipSet`;
```

3.3.3 Transition

A robust, fluent and rapid transition from one state to another state is very important for agents expressing exact emotions during specified events. In another word, when a series of events happen, an individual should have the mechanism, the ability of choosing the correspondent actions which have to commence.

A robust transition needs to have a comprehensive motion graph, a detailed FSM. This can be done by carefully designing the sequence of each state. A rapid transition means when an event happens, the agent will try to finish the current motion cycle as soon as possible and then change to specified motion. Also a direct interpolation of new pose is also possible but it will lead to some motion discontinuity. For example, when an agent clap cycle is still not finished, a new event which has an instruction of making the agent stop clapping happens. A robust transition will first finish the clap cycle and then begin to execute the new motion cycle. While a rapid transition will stop the clap cycle immediately and load the new motion cycle. Although the former can reduce motion discontinuity, it produces some lag to agent's reaction. While the latter also has its disadvantage: although it can react at the exact time, some improper action or gesture will be generated due to the rapid switch. Thus our project tries to strike a balance between the two contradictory factors. After carrying through some tests on motion clips and our FSM model, we put forward two solutions: (1) we cut down each clip's duration to reduce the lag time. (2) Some motions with relative longer duration are broken down into smaller parts. For instance, as mentioned above, the *cheer* state is divided into three parts: Arm_up, cheer, Arm_down. As a result, in our system, the lag is less than 10 frames on average. Figure 23 shows a part of our motion arrangement before the simulation. You can find each clip is cut down into around 10 frames. And big clips have been cut down to satisfy different combination of clips during the simulation.

3.4 Virtual model representation research and LOD

In our work, rendering process is done by the existing render engine in Maya. However, we could also increase rendering time by improving rendering algorithm. This work might be considered as future work of our project. In this section we will discuss about our research on 3 levels of detail used in human model representation.

Human template is the basic representation of a virtual human which consists of skeleton and joints covered with thousands of triangle mesh. It comprises several models for different resolution levels and various appearances to produce different various types of human. Animation sequence information is also included in human template. In short, all the virtual individuals in the crowd are originated from human templates. So a large number of meshes mean a huge volume of information which needs to be processed in each frame. Although, in recent years, graphic card indeed develops rapidly, there are still some constraints. Many researchers still have to consider the trade-off between render cost and quality. Based on Daniel Thalmann's theory, two important sections of model representation are below [2, 4]. Besides the human templates in crowd simulation, rending is also a core part for model visualization. Three main techniques of acceleration technologies for rending are: Culling (visibility, occlusion), Levels of Detail (geometry, animation) and image-based rending (HW layers, imposters). Considering large numbers of polygons in spectator crowd, we often view the crowd as a whole. Image-based rending is more suitable for rending large numbers of people in a virtual environment [19].

3.4.1 Deformable mesh

Deformable mesh, enveloping tens of joints of a human body, has both advantages and disadvantages in crowd simulation [4].

(1) Advantages

1. Flexible animation. Any slight movement like grin or clenching can be animated.
2. The mesh consists of many vertices, which following the joints movements. The changes between each frame can be only referred to those deforming vertices. In another word, this kind of frame information is cheap to store.
3. Procedural animation and blending are feasible by using based on this approach.

(2) Disadvantages

1. Only suitable for tens or hundreds people animation.
2. Used under constrains like camera distance.

3.4.2 Rigid mesh

Rigid mesh and imposter techniques are able to provide faster speed of rending. However, the camera distance needs to be further than that of deformable mesh. In addition, procedure animation is not available here and slight movements can not be achieved.

The apparent difference between rigid mesh and deformable mesh is the pre-processing procedure. In the deformable mesh, the animation sequence information is handled by GPU. However, rigid mesh handles vertex information in CPU rather than in GPU, which increasing speed considerably during the rending process [5].

3.4.3 Imposter

Tecchia F., Loscos C. give a detailed previous works of image-based acceleration technology [19]. Imposter, a less detailed representation, is a dominant image-base acceleration technology for large-scale virtual crowd simulation. Generally the rendering system produces a fake image that rotating to the camera. A virtual human is often divided into different parts based on body part. All the information of imposter is generated in a hidden buffer and then copied to texture memory. Figure 25 gives a comparison of 3D deformable mesh and imposter. From the pictures we can find the clear difference between the two methods. In sum, deformable mesh has a more realistic look than imposter. In my project, it is more preferable to choose the former method.

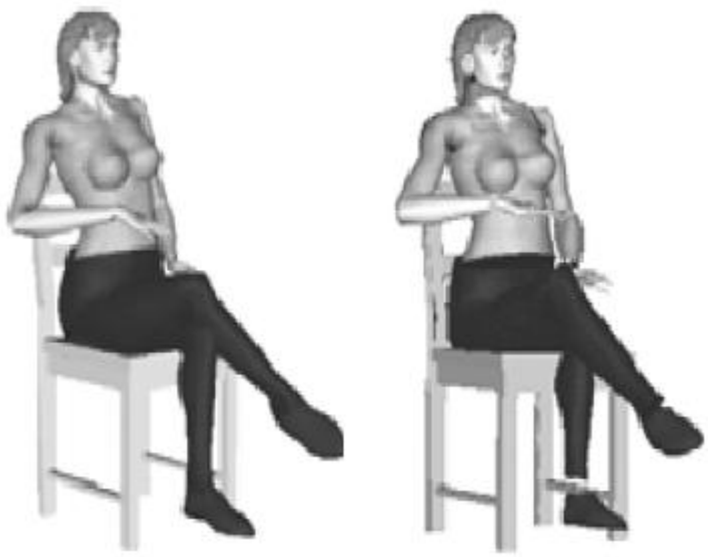


Figure 25. Comparison of 3D deformable mesh (left) and imposter (right)

Source: Ref. [2].

3.4.4 LOD

With the increasing demand on crowd scene in film and game industry, hundreds and thousands of characters in a scene is very common in a crowd simulation task. Thus when facing a large quantity of information which needs to be processed during simulation, simplified and efficient method might be utilized to relieve this situation.

LOD technique is often used in crowd simulation to reduce the complexity of model representation. David Kasik, Andreas Dietrich et al. (2008) give a detailed classification within model visualization field [21]. This article shows LOD has three implementation methods for LOD: discrete, progressive and continuous models. The discrete model and progressive model both have one advantage: they can handle a large object which spanning a very long distance from the viewer or camera. However, these two methods are not efficient enough to handle audiences in crowd simulation. Figure 26 shows examples of discrete and progressive LOD. Jonathan Maïm, Barbara Yersin, Daniel Thalmann also proposed a new method to process crowd visualization [4]. A LOD Selector which is based on the LOD has been designed. They introduced a new parameter ROI (regions of different interests). Based on ROI, each agent can be assigned to different motion planning rules. LOD Selector combines LOD and ROI together to achieve to most efficient-cost visualization.

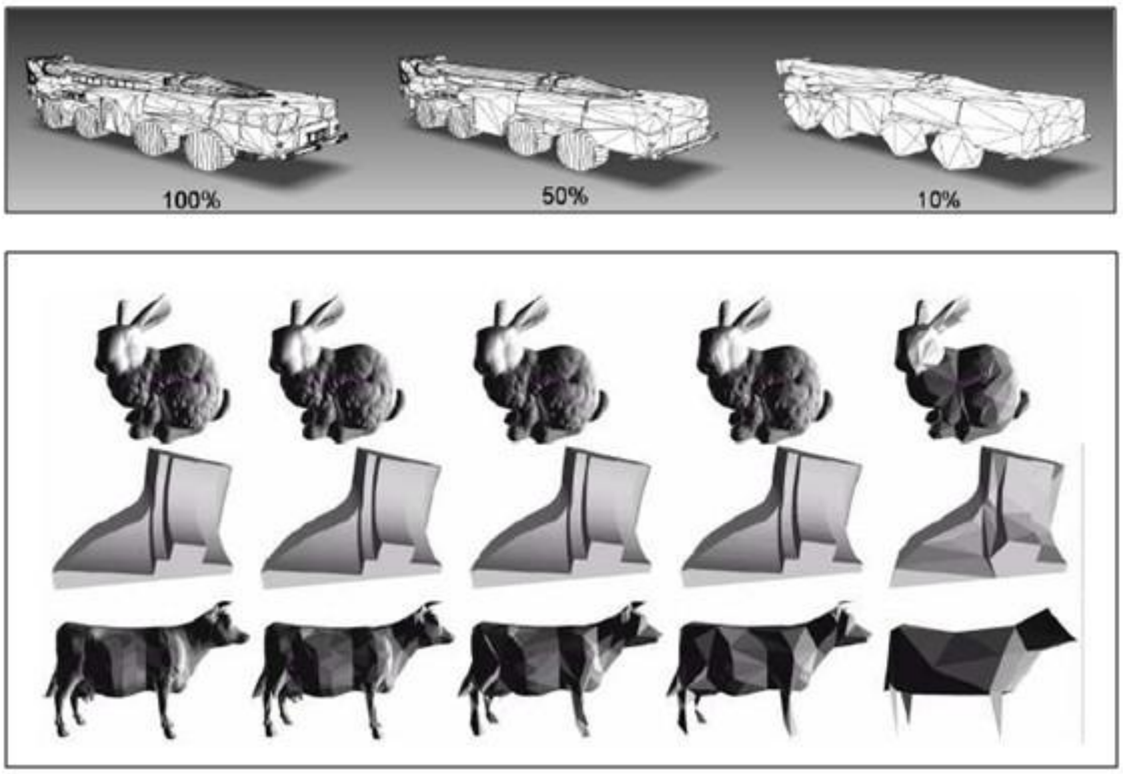


Figure 26. Example of LOD. Source: Ref. [21].

In our project, we adopt the LOD selector technique. After we set up a camera, we can use the camera position and geometry position to calculate the distance between them and then decide which geometry should be rendered during rendering process. In our project, we are not able to spend time on modelling a perfect human model. Thus we get sample geometries of both lower and higher resolution to test our LOD function. During the test, the geometry of each individual switches to lower resolution and higher resolution automatically according to the distance from each person to the camera. By reducing the visual quality without being noticed by the audience, this method helps save rendering time and improves the software performance on populating a large-scale crowd. In figure 27, picture above shows when the distance is less than 10, the system automatically switches to higher resolution. The picture below shows when the distance is further than 10, low resolution geometry is displayed. In order to increase system processing speed, the code is not written in expression editor in the Maya crowd scene. They are all embedded in the original Mel file.

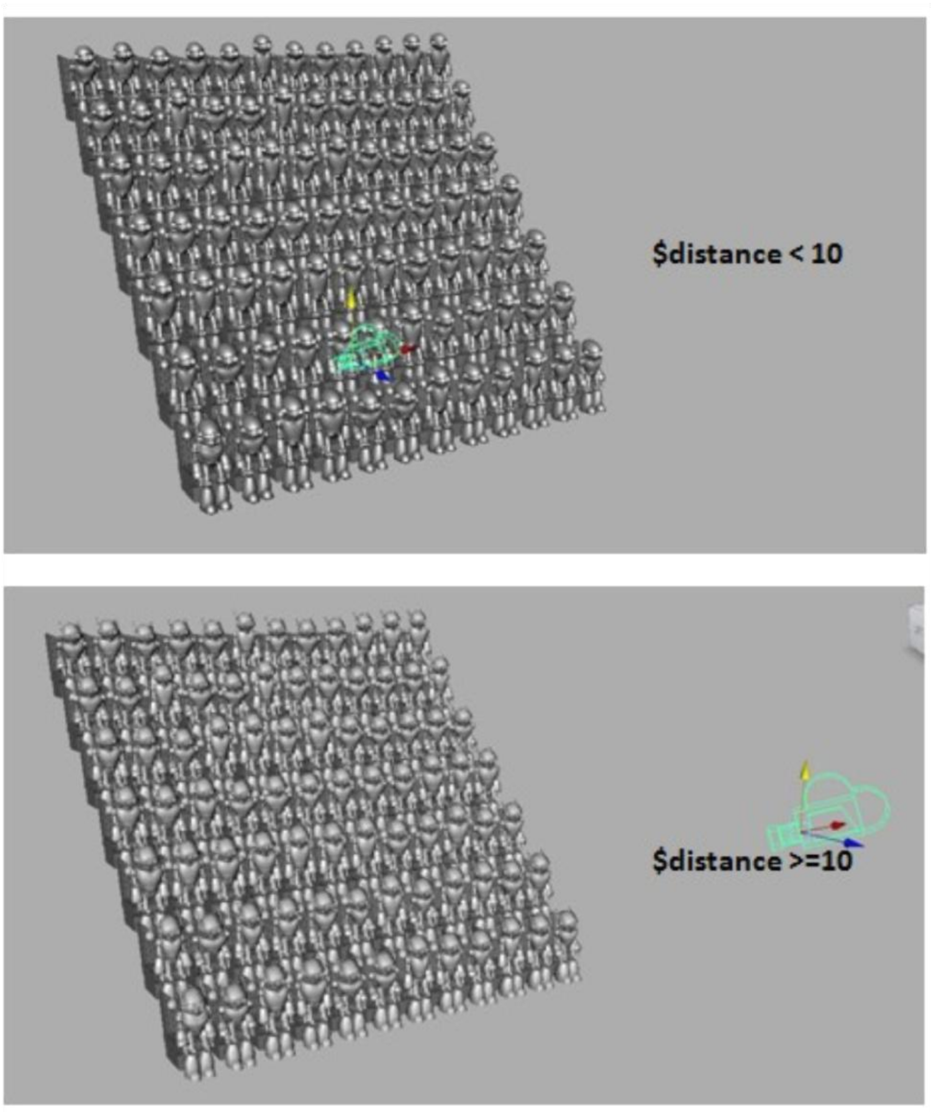


Figure 27. Test of LOD in our system.

High resolution. (up)

Low resolution. (down)

```
$expString = “float $cameraX, $cameraY, $cameraZ, $distance;\n”;
$expString += “$cameraX = (cameraFinal.translateX - robotGeo.translateX) * (cameraFinal.translateX - robotGeo.translateX);\n”;
$expString += “$cameraY = (cameraFinal.translateY - robotGeo.translateY) * (cameraFinal.translateY - robotGeo.translateY);\n”;
$expString += “$cameraZ = (cameraFinal.translateZ - robotGeo.translateZ) * (cameraFinal.translateZ - robotGeo.translateZ);\n”;
$expString += “$distance = sqrt ($cameraX + $cameraY + $cameraZ);\n”;
$expString += “if ($distance < 10 )\n”;
$expString += “{\n”;
$expString += “setAttr "robotGeo.visibility" 1;\n”;
$expString += “setAttr "robotLowGeo.visibility" 0;\n”;
$expString += “}\n”;
$expString += “else\n”;
$expString += “{\n”;
$expString += “setAttr "robotGeo.visibility" 0;\n”;
$expString += “setAttr "robotLowGeo.visibility" 1;\n”;
$expString += “}\n”;
Expression –string $expString –name “testLOD” –alwaysEvaluate true –unitConversion all;
```

3.5 User interface

Our production tool is embedded into Autodesk Maya 2010 environment. By clicking the shelf button shown in Figure 28, the script will automatically compile and run.

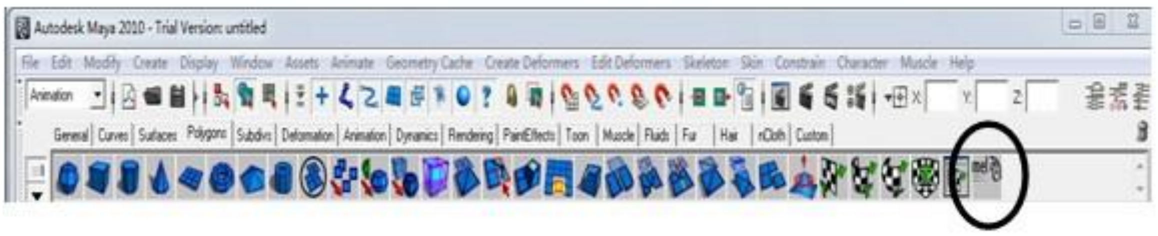
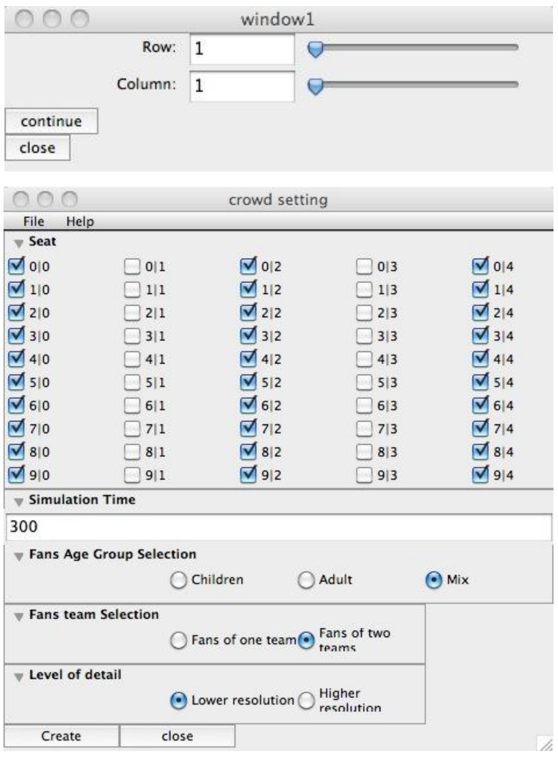

Autodesk Maya 2010 - Trial Version: untitled
File Edit Modify Create Display Window Assets Animate Geometry Cache Create Deformers Edit Deformers Skeleton Skin Constrain Character Muscle Help
Animation
General Curves Surfaces Polygons Subdivs Deformation Animation Dynamics Rendering PaintEffects Toon Muscle Fluids Fur Hair nCloth Custom
mel

Figure 28. The implementation button embedded into Maya

The first window (shown in figure 29) asks user to input the row and column number of the crowd. With the input information, the scene will generate seats for the whole scene. Hundreds of people could be created in my laptop. But this scalable system can create thousands of or more agents if computational power is satisfied.



window1

Row: 1

Column: 1

continue

close

crowd setting

File Help

Seat

✓ 0|0	☐ 0|1	✓ 0|2	☐ 0|3	✓ 0|4
✓ 1|0	☐ 1|1	✓ 1|2	☐ 1|3	✓ 1|4
✓ 2|0	☐ 2|1	✓ 2|2	☐ 2|3	✓ 2|4
✓ 3|0	☐ 3|1	✓ 3|2	☐ 3|3	✓ 3|4
✓ 4|0	☐ 4|1	✓ 4|2	☐ 4|3	✓ 4|4
✓ 5|0	☐ 5|1	✓ 5|2	☐ 5|3	✓ 5|4
✓ 6|0	☐ 6|1	✓ 6|2	☐ 6|3	✓ 6|4
✓ 7|0	☐ 7|1	✓ 7|2	☐ 7|3	✓ 7|4
✓ 8|0	☐ 8|1	✓ 8|2	☐ 8|3	✓ 8|4
✓ 9|0	☐ 9|1	✓ 9|2	☐ 9|3	✓ 9|4

Simulation Time

300

Fans Age Group Selection

Children Adult [✓ Mix]

Fans team Selection

Fans of one team [✓ Fans of two teams]

Level of detail

[✓ Lower resolution] Higher resolution

Create close

Figure 29. User interface

Then the second window (shown in figure 29) pops up. The user could import their desired model from the menu. And finally they need to set up their preference: the distribution of the crowd; the whole simulation time; choosing different age group to have different resizing parameters; fans from one team or two team (further can be extend to more groups of fans); Level of details on and off. Then after user presses the “create” button, the system will automatically generate the desired crowd.

Chapter 4 Visual improvement

In chapter 3, we have implemented the most important part of our simulation: variety of behaviour, variety of actions. In our project, we mainly work on software algorithm and behaviour model structure. Aesthetics work like drawing delicate human skin texture, researching on human body organization is not our concern. These works should be improved by artists. In this section, we will give an overview of crowd variation. With our variation structure and a good understanding of human body proportion and texture editing, an artist can easily make a good visual effect on crowd.

4.1 Colour variety

In crowd simulation, many 'unique' virtual humans are needed to main virtual authenticity. It is impossible to design various templates for each agent and assign colourful, unique texture to each one. In addition, the space for storing such huge information is demanding. Thus it is reasonable and nature to have some techniques to create variety on template, texture and animation for crowd simulation. Generally, in crowd simulation there are three ways to create basic variety for human templates. The first is generating enough human templates for agents. Obviously, it is able to increase variety. However, it is still impossible to model templates for each agent in a large-scale virtual environment where there are hundreds, thousands or even more individuals. It will take huge memory space to store each template. Furthermore, modelling this templates need a lot of time and modelling skills. So based on several templates, the next step is to assign different texture and get different colour for each template. Figure 30 shows the different templates appearances and the result after applying different texture and colours.

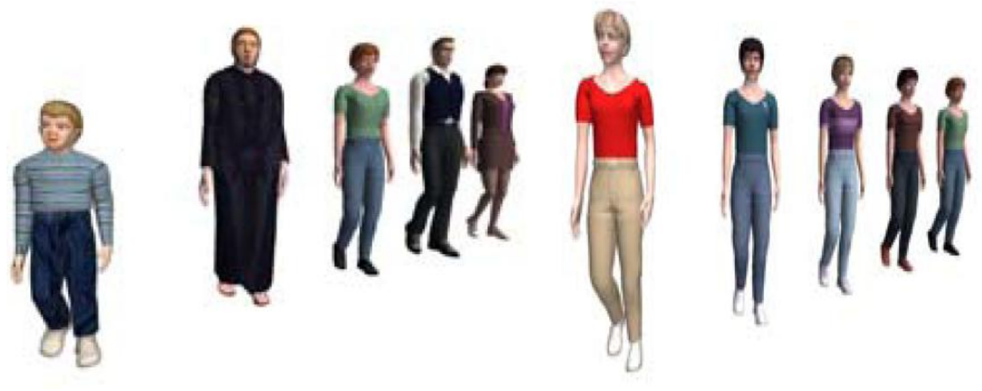


Figure 30. Step 1: making different templates (Left)

Step 2: applying different texture and colours (Right)

Source: Ref. [4].

Based on the RGB system, crowd system has already been able to produce various colours for templates. However, the result is not satisfactory in a very large-scale crowd simulation (thousands or more). Jonathan Maïm, Barbara Yersin and Daniel Thalmann [2008] give a new method which involves HSB system to solve this problem in crowd simulation [4]. HSB has three perceptual variables: hue(H), saturation(S), bright(B). Which is more suitable for controlling constrains to specific parts of templates. It is easier to get skin colour with hue (only yellow and red) and no saturation(S) rather than a combination of R, G and B.

4.2 Body size variety

In the film and game industry, spectator crowd in stadiums or arenas are not strict with individual appearance. While the whole crowd variation is the essential factor to create virtual crowds. By using script to edit different part of human body, it is feasible to create a more vivid crowd with various sizes [3]. This procedure gives a solution of setting up different human models (old, young, adult, boy, girl etc). Moreover, John Kundert-Gibbs, provides a more realistic way to make variation [9]: By giving random number of human part ratio, which concerns about the algorithm with joint setting of only one model, hundreds of different models can be created immediately. In our work, a simple skeleton is set up for test. The user is able to import any type of human skeleton as they like to simulate the scene. A skeleton of one character is shown in figure 31.

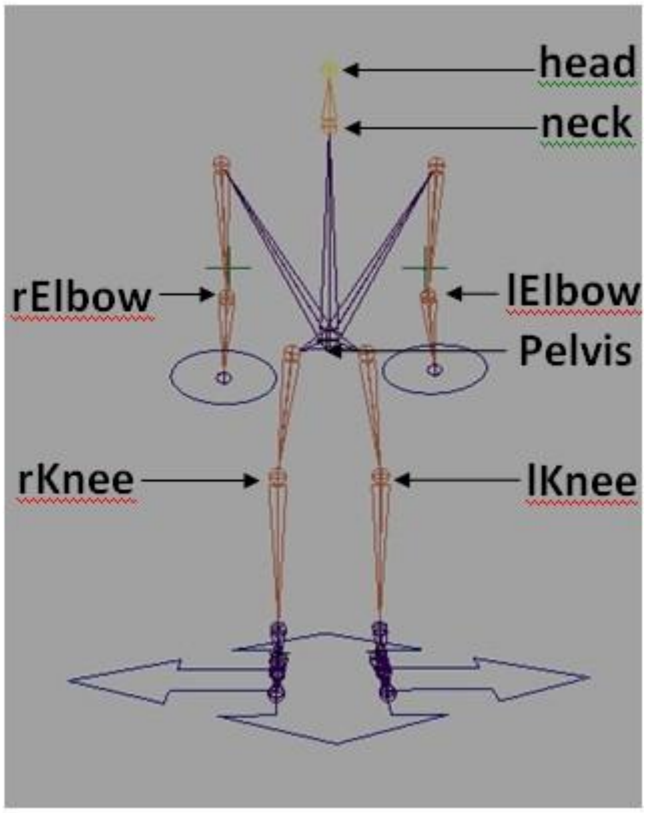


Figure 31. Character’s skeleton used in our work.

In our work, by adjusting the joint of each character, we can set up a crowd scene of various sizes. This procedure is done before importing motion. After each character is imported into the scene, according to user input, system begins to generate resizing parameters which will be used during the scaling process. Then the system loops for each character in the scene and scale its’ joints with the given scaling parameters. By scaling the pelvis joint, we can decide the height character’s upper body. By scaling elbow joint, we can change the length of arm. Figure 32 shows all the controllable joint of one character. We could adjust any joint in this figure to achieve the type of figure we want.

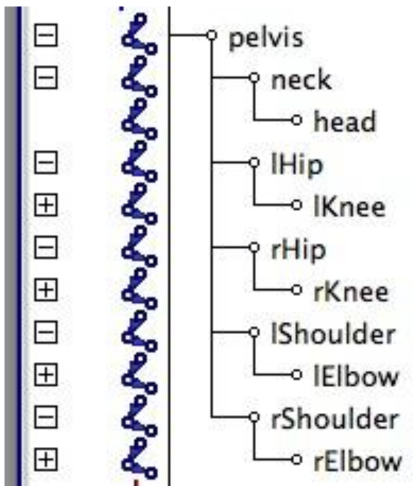


Figure 32. A list of scalable joints

We run into a technical problem during the process of resizing. After we resize each character and then begin to import motion into each character. All the sizes of each character returns to the default size – the original template size.

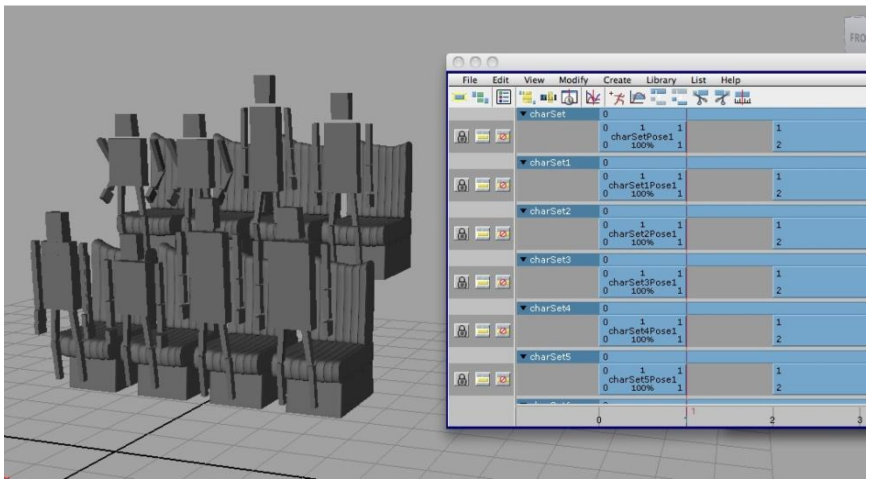


Figure 33. Setting up pose for each character.

This is caused by the stored information in the motion library. Since all the pre-processed motions are made from the original human template. Thus when the motion clips are stored in the motion library, they also store the template's size information including translation, rotating and scaling. Hence although each motion clip is set to "all relative" property as mentioned in section 3.3.2, because of lacking initialization information of each skeleton, The skeleton will changing according to the previous motion clip which is imported into the character. Thus it is a must to store the pose information to preserve the resizing result. Thus after the scaling procedure, we must set up a pose for each character to store the skeleton initialization information. After that, no matter what kind of motion is imported, character will always scale based on the “relative” information. The editor in Figure 33 shows eight poses are added at the 0 frame (before the simulation). After we set poses for each character, eight characters in the scene all have their own different appearances.

```
Setting up the pose for each character:
FOR EACH CHACATER
{
    .......
 pose -name ($charSetName + "Pose") $charSetName;
 $poseName = `pose -q -n $charSetName`;
 clipSchedule -start 0 -ra -in $poseName[0] (`character -q -sc $charSetName`);
   .......
}
```

Chapter 5. Conclusion and evaluation

Crowd simulation has been applied into many areas: film, game, military training, safety science, etc. Hence there are many works about pedestrian evacuation, city crowd simulation and other areas. Our work is a contribution to the film and game industry. We work specifically on spectator simulation in sports games. We design a framework of crowd spectator simulation. An agent's five core features are: identifiable, situated, flexible, goal-directed and autonomous. These features can not be achieved without reasonable behaviour control. Specifically, a behaviour model structure is constructed to provide more flexibility on behaviour control.

Environment setup, distribution of each character and motion library creation is the pre-processing part of the whole work. Whilst our research is limited to the amount of processing I can do on my laptop, our system would scale to any number of spectators given enough processing power. Such a flexible basic structure can be easily embedded into other similar applications. Much code needs to be done here to support the basic structure of our work. Moreover, research on crowd and project model selection (including Flocking system, Particle system, Hybrid system and Agent-based system) has been done during the interim phase. These researches give us a clear view of the whole project.

With many researches on social and scientific factors in crowd simulation before and after the interim report phase, we finally create our behaviour model. Three levels of behaviour are constructed under the scripted control and external control. And audiences' reactions to certain events have also been achieved in this model. The behaviour FSM has a well-structured framework which means more branch clips could be added efficiently and this structure is easily-maintained even by those who have no experience on programming like artists. This means also we don't have a brilliant visual effect on the crowd in our project. But based on our framework, artists are able to make vivid and autonomous crowd more easily. And this structure is portable to be used in other crowd applications besides the crowd spectator simulation. Although in our work behaviour generator is based on FSM and action clips should be manually classified into different sets, this system can be reused in any crowd scenes. Moreover, clips in each state could be easily added, deleted or modified.

FSM is utilized in our model to help sort out behaviour schedule during simulation. Its motion-graph structure is a very powerful tool to be used in our program. We create a self-searching algorithm which allows the system to traverse the motion tree based on behavioural rules. And a multiple-clip state solution is designed for increasing the behaviour variety. Only a number of high-level states are needed to perform a complex crowd. Our behaviour search approach is scalable which means without increasing the FSM's complexity synthesized motions can be added to perform more interesting and human-like behaviours. An agent reaction mechanism has been modelled and implemented. This reaction model allows the crowd to be controlled by only a few events or emotion parameters. It means that it is feasible for animators to manipulate crowd by only adjusting few control parameters from UI.

Many technical details have been solved. We give a detailed description of the solution to these technique problems in our work: Motion scheduling mechanism and how it is represented in

Maya environment has been discussed clearly; some core nodes like motion library, clip scheduler and animcurve have been explained in detail in our project; methodology like relative and absolute properties and motion transition are implemented. Some other implementation like variety on skeletons also has been demonstrated. Our system requires a relatively small space for storing motion data which helps alleviate the resource-limited conditions. Our crowd spectator can perform clap, cheer and idle behaviours with only 200 frames of data in all.

Although in our work rendering process is not our concern, we still give a comprehensive review of it. Because rendering is a parallel research area compared to ours in crowd simulation area. It is a very important visual representation issue. Thus three methods have been discussed which could be an extension of our work. LOD, a technical representation issue, has been solved and illustrated at the end of our project.

In conclusion, our project provides a comprehensive structure of crowd spectator simulation. We have tested our crowd with simple motion data which are keyframed by us. They all work and transit very well. Behaviour model has been evaluated by the reaction and emotion sample discussed in chapter 3. In addition, based on our production tool, it is very convenient for animator to add more motion capture data to make the whole scene more vivid and real. The user interface we developed is very friendly and normally an animator does not have so much learning gap on it. Therefore, we believe that the aims and objects set in our report have been achieved.

Chapter 6. Future development

6.1 Motion capture data

Nowadays, motion capture data has been widely applied in film and game industry. Motion capture data can be indirectly imported into any related model template. In our project, we do not have the chance to get access to motion capture facilities. Thus, on an artistic level, crowd behaviour can not be simulated as same as that of a real human. Although the aim of our project is not to simulate the exact same behaviour from human-beings, it is true that our visual effect could get a better effect from motion capture data.

6.2 Parameterized state

In many pedestrian simulation work, an agent's velocity and direction often changes at run time. So it is also feasible to make crowd spectators' motion clip be re-sampled and redirected. In our project, behaviour like "idle", "clap", "cheer" is still not accurate and completely parameterized. These states only obey the FSM rules and transit according to the pre-defined routine. For future development, it is possible to make all the states contain some parameters. This means based on the basic information of state, system could have a more flexible transition along the whole motion graph. For example, instead of defining "idle" state in the BFSM, it is better to have a definition of "clap toward X direction", "clap oriented to LEFT", etc. This method not only reduces the space for storing motion data, but also improves the flexibility of behaviour planner. Even more, a combination like "clap toward X direction and oriented to LEFT" would provide larger variety of crowd behaviour.

6.3 Crowd dynamics

The crowd pattern and behaviour also might be simulated based on some pure mathematical model. An idea come from thermal dynamics is possible to be applied into crowd simulation. A crowd can be treated as a jar of water boiled on the fire. At first, when the temperature of water is very low, only a few bubbles rise to the surface. This phenomenon is just like at first when the expressiveness index of the game performance is not very high, there are only a few scattered people clapping. After some time (specified time in our project), when the water is boiling, many bubbles rise to the water surface and burst. This is just like the scene that all the audiences stand up and cheer for the excellent goal. This research area might be helpful to the dynamic research of agent-based crowd.

6.4 Path finding and collision avoidance

Some researches on pedestrians have been done. It is possible for us to adapt their algorithm and structure into our future work. For example, in our crowd spectator simulation, it is very likely to have a more complex scene with other types of agents like the securities, referee, and athletes. Most of them need to be designed to have the intelligence to walk, run and manage more complicated actions. The involved work might be related to the path finding, collision avoidance and action point setting, etc.

Reference

[1] Soraia Raupp Musse and Daniel Thalmann, Hierarchical Model for Real Time Simulation of Virtual Human Crowds: IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 7, NO. 2,152-163. (2001)

[2] Daniel Thalmann, Crowd and Group Animation: SIGGRAPH 2004 Course Notes. (2004)

[3] Tianlu Mao, Bo Shu, Wenbin Xu, Shihong Xia, Zhaoqi Wang, CrowdViewer From Simple Script to Large-scale Virtual Crowds: VRST 2007, Newport Beach, California, November 5–7. (2007)

[4] Jonathan Maïm, Barbara Yersin, Daniel Thalmann, Real-Time Crowds: Architecture, Variety, and Motion Planning: Virtual Reality Laboratory, EPFL, Switzerland. (2008)

[5] Barbara Yersin, Jonathan Ma¨ım et al., Crowd Patches: Populating Large-Scale Virtual Environments for Real-Time Applications: I3D 2009, Boston, Massachusetts. (2009)

[6] Suiping Zhou, Dan Chen, Wentong Cai, Linbo Luo, Malcolm Yoke Hean Low, and Feng Tian, Crowd Modeling and Simulation: ACM Transactions on Modelling and Computer Simulation, Vol. 20, No. 4, Article 20. (2010)

[7] Stephanie E.Pitts, Christopher P.Spencer, Loyalty and Longevity in Audience Listening: Investigating experiences of attendance at a chamber music festival: Music & Letters, Vol.89, No. 2. (2007)

[8] Gary Lupyan, Ilya Rifkin, Dynamics of Applause: Modeling Group Phenomena Through Agent Interaction.

[9] John Kundert-Gibbs, Maya Secrets of the Pros: SYBEX Inc., 1151 Marina Village Parkway, Alameda. (2001)

[10] Soraia Raupp Musse, Branislav Ulicny, Amaury Aubel, Groups and Crowd Simulation: EPFL – VRlab. (2004)

[11] Charles M. Macal, Michael J. North, Tutorial on Agent-based Modeling and Simulation: Proceedings of the 2005 Winter Simulation Conference. (2005)

[12] Dean Wright, Bill Westenhofer, Jim Berney, And Scott Farrar, The Visual Effects of The Chronicles of Narnia: The Lion, the Witch and the Wardrobe: ACM Computers in Entertainment, Vol. 4, Number 2. (2006)

[13] Santos, G. and Aguirre, B. E, a Critical Review of Emergency Evacuation Simulation Models: NIST Workshop on Building Occupant Movement during Fire Emergencies, 25–50. (2004)

[14] Chenney, S. 2004. Flow Tiles: Eurographics/ACM SIGGRAPH Symposium on Computer Animation, 233–242. (2004)

[15] Helbing, D., Farkas, I., And Vicsek, T, Simulating Dynamical Features of Escape Panic. Letters to Nature 407, 487–490. (2000)

[16] Eunjung Ju, Myung Geol Choi, Minji Park et al., Morphable Crowds: ACM Transactions on Graphics, Vol. 29, No. 6, Article 140. (2010)

[17] Sipser, M. Introduction to the Theory of Computation 2nd Ed. PWS Publishing Company, Boston, MA. (2005)

[18] Saskia Groenewegen, Improving Crowd Behaviour for Games and Virtual Worlds: FDG 2010, June 19-21, Monterey, CA, USA. (2010)

[19] TECCHIA F., LOSCOS C., CHRYSANTHOU Y., Image-based crowd rendering: IEEE Computer Graphics and Applications 22, 2, 36–43. (2002)

[20] Louise Barkhuus, Tobias Jorgensen, Engaging the Crowd – Studies of Audience-Performer Interaction, 2008

[21] David Kasik, Andreas Dietrich et al., Course Notes Massive Model Visualization Techniques: Morgan & Claypool Publishers LLC. (2008)

[22] Matt Aitken, Greg Butler et al., The Lord of the Rings: The Visual Effects That Brought Middle Earth to the Screen: SIGGRAPH. (2004)

[23] K. Cain, Y.Chrysanthou, F.Silberman, A Case Study of a Virtual Audience in a Reconstruction of an Ancient Roman Odeon in Aphrodisias: The 5th International Symposium on Virtual Reality, Archaeology and Cultural Heritage VAST, pp. 1-9. (2005)

[24] Craig W. Reynolds, Flocks, Herds, and School: A Distributed Behavioral Model: Computer Graphics, Volum 21, Number 4, July. (1987)

[25] Manfred Lau and James J. Kuffner, Behavior Planning for Character Animation: Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, 29-31 July. (2005)

[26] Bill Tomlinson, From Linear to Interactive Animation: How Autonomous Characters Change the Process and Product of Animating: ACM Computers in Entertainment, Vol.3, No.1, January. (2005)

Appendix: Source code

LOD

```
$expString = “float $cameraX, $cameraY, $cameraZ, $distance;\n”;
$expString += “$cameraX = (cameraFinal.translateX - robotGeo.translateX) * (cameraFinal.translateX - robotGeo.translateX);\n”;
$expString += “$cameraY = (cameraFinal.translateY - robotGeo.translateY) * (cameraFinal.translateY - robotGeo.translateY);\n”;
$expString += “$cameraZ = (cameraFinal.translateZ - robotGeo.translateZ) * (cameraFinal.translateZ - robotGeo.translateZ);\n”;
$expString += “$distance = sqrt ($cameraX + $cameraY + $cameraZ);\n”;
$expString += “if ($distance < 10 )\n”;
$expString += “{\n”;
$expString += “setAttr "robotGeo.visibility" 1;\n”;
$expString += “setAttr "robotLowGeo.visibility" 0;\n”;
$expString += “}\n”;
$expString += “else\n”;
$expString += “{\n”;
$expString += “setAttr "robotGeo.visibility" 0;\n”;
$expString += “setAttr "robotLowGeo.visibility" 1;\n”;
$expString += “}\n”;
Expression –string $expString –name “testLOD” –alwaysEvaluate true –unitConversion all;


global string $checkBoxSeatName[];
global int $sceneEventIndex;
global int $embarrassment = 1;


global proc int importCharacter( string $filename, string $fileType )
{
   file -import -namespace "character" $filename;
   return 1;
}
```

SEARCH FOR NEXT MOTION

```
global proc string searchForNextMotion(string $charClipSet, string $originalMotion)
{
     global int $sceneEventIndex;
     int $supportTeam;
     int $randomNum;
     $supportTeam = `getAttr ($charClipSet + ".team")`;
    string $nextMotion;
    if( $originalMotion == "stand" )
       {
         if( $sceneEventIndex == 0 )
           $nextMotion = "stand";
         else if( $sceneEventIndex == 1 && $supportTeam == 1)
           $nextMotion = "armUp";
         else if( $sceneEventIndex == 1 && $supportTeam == 0)
           $nextMotion = "stand";
       }
    else if ( $originalMotion == "armUp" )
       {
            $randomNum = `rand 1 100`;
              if ( $randomNum > 0 && $randomNum <= 50 )
                 $nextMotion = "clap";
              else if ( $randomNum > 50 && $randomNum <= 100 )
                 $nextMotion = "cheer";
       }
    else if ($originalMotion == "cheer" )
       {
         if( $sceneEventIndex == 1 )
           $nextMotion = "cheer";
         else if( $sceneEventIndex == 0)
           $nextMotion = "armDown";
       }
     else if ( $originalMotion == "clap" )
       {
          if( $sceneEventIndex == 1 )
           $nextMotion = "clap";
         else if( $sceneEventIndex == 0)
           $nextMotion = "armDown";
       }
     else if( $originalMotion == "armDown" )
       {
           $nextMotion = "stand";
       }



     ......



  return $nextMotion;
}
```

TIMING AND EVENT CONTROL

```
global proc int chooseMotionClip(string $charClipSet, string $motionType, int $startTimeFrame, int $allFrameLength)
{
        global int $sceneEventIndex;
        global int $embarrassment;
        int $startTime, $endTimeFrame, $motionIndexWeight, $supportTeam;
        int $calculateStart, $calculateLength1, $calculateLength2, $calculateLength;
        string $motionIndex;
        string $clipName[];
        string $motion = $motionType;

        float $scaleNumber;
        $motionIndexWeight = `rand 1 100`;
          if ($motionIndexWeight > 0 && $motionIndexWeight <= 25)
            {
             $motionIndex = "One";
            }
          else if($motionIndexWeight > 25 && $motionIndexWeight <= 50)
            {
             $motionIndex = "Two";
            }
          else if($motionIndexWeight > 50 && $motionIndexWeight <= 75)
            {
             $motionIndex = "Three";
            }
          else if($motionIndexWeight > 75 && $motionIndexWeight <= 100)
            {
             $motionIndex = "Four";
            } // print $motionIndex;
          if( !`objExists ($motion + $motionIndex + ":" + $motion + $motionIndex + "Source")` )
            {
              file -import -type "mayaAscii" -namespace ($motion + $motionIndex) -options "v=0" -pr -loadReferenceDepth "all" ("/Users/kaiguangren/Documents/maya/projects/default/clips/" + $motion + $motionIndex + ".ma");
            }

          $startTime = 1;
          $startTime += $startTimeFrame;
          clip -copy ($motion + $motionIndex + ":" + $motion + $motionIndex + "Source");
          $clipName = `clip -pasteInstance -sc 1 -startTime $startTime -ar $charClipSet`;
          $calculateStart = `getAttr ($clipName[0] + ".startFrame")`;
          $calculateLength1 = `getAttr ($clipName[0] + ".sourceStart")`;
          $calculateLength2 = `getAttr ($clipName[0] + ".sourceEnd")`;
          $calculateLength = $calculateLength2 - $calculateLength1;
      $endTimeFrame = $calculateLength + $calculateStart;
      if ( ($endTimeFrame + 10) < $allFrameLength )
        {
          if ( $embarrassment == 0 )
            {
              if ( ($endTimeFrame > 50) && ($endTimeFrame < 250) )
                { $sceneEventIndex = 1; }
              else
                { $sceneEventIndex = 0; }
            }
          else if ( $embarrassment == 1 )
            {
              if ( ($endTimeFrame > 120) && ($endTimeFrame < 200) )
                { $sceneEventIndex = 1; }
              else
                { $sceneEventIndex = 0; }
            }

......

          $motion = searchForNextMotion($charClipSet, $motion);
          chooseMotionClip($charClipSet, $motion, $endTimeFrame, $allFrameLength);
        }
    return 0;
}
```

Behavior Generator (team option)

```
global proc behaviorGenerator(int $simTime, int $fansTeam)
{
    int $timeLength = $simTime;
    int $teamWeight;
    string $charClipSet;
    string $charClipSetList[] = `ls -sets "charSet*"`;
      for($charClipSet in $charClipSetList)
         {
            if( $fansTeam == 1)
             {
              setAttr($charClipSet + ".team") 1;
             }
            else if( $fansTeam == 2)
             {
              $teamWeight = `rand 1 100`;
              if ( $teamWeight > 0 && $teamWeight <= 50 )
                setAttr ($charClipSet + ".team") 0;
              else if ( $teamWeight > 50 && $teamWeight <= 100 )
                setAttr ($charClipSet + ".team") 1;
             }
            chooseMotionClip($charClipSet, "stand", 0, $timeLength);
         }
}
```

Person Generation

```
global proc generateSpecifiedPerson(int $gspRow, int $gspColumn, int $gspGroupIndex)
{
    global string $checkBoxSeatName[];
    int $row = $gspRow;
    int $column = $gspColumn;
    int $groupIndex = $gspGroupIndex;
    int $i, $j;
    int $choice;
    int $tokenFirst, $tokenSecond;
    string $buffer[], $poseName[], $charIndex;
    float $transX, $transY, $transZ;
    float $scaleJointX, $scaleJointY, $scaleJointZ;


    for( $i = 0; $i < $row; $i++)
     for( $j = 0; $j < $column; $j++)
        {
          $choice = `checkBox -q -value $checkBoxSeatName[$i * $column+ $j]`;
          if( $choice == 1)
              {
                select -r character:root;

                duplicate -un;
                $charIndex = `checkBox -q -label $checkBoxSeatName[$i * $column + $j]`;
                tokenize $charIndex "|" $buffer;
                $tokenFirst = $buffer[0];
                $tokenSecond = $buffer[1];
                $transX = $tokenSecond * (1.5);
                $transY = $tokenFirst * (2.0);
                $transZ = $tokenFirst * (-2.0);
                $rootList = `ls -tr "root*"`;
                $rootNb = size($rootList);
                $rootName = $rootList[($rootNb - 1)];
                $charSetList = `ls -sets "charSet*"`;
                $charSetNb = size($charSetList);
                $charSetName = $charSetList[($charSetNb - 1)];
                move -r $transX $transY $transZ $rootName;
                if ($groupIndex == 1)
                  {
                   scaleValueSetup(1);

                  }

                 else if ($groupIndex == 2)
                   {

                  scaleValueSetup(2);

                   }

                 else if ($groupIndex == 3)
                   {

                  scaleValueSetup(3);
                   }
                  else
                    error "You should choose 1 -3 for child adult mix";
                scale -r $scaleJointX $scaleJointY $scaleJointZ ($rootName + "|" + "pelvis");
                pose -name ($charSetName + "Pose") $charSetName;  // pose -name ( "charSet1"+ "Pose") "charSet1";
                $poseName = `pose -q -n $charSetName`;   // $poseName = `pose -q -n "character:charSet"`;
                clipSchedule -start 0 -ra -in $poseName[0] (`character -q -sc $charSetName`); // clipSchedule -start 0 -ra -in fighting1 ("charSet1" + "Scheduler1");


.......



              }
         }
}
```