

Crowd Simulation Plug-in for Maya

Digital Arts & Science Senior Project: 2004-2005

Team TCRcomplex: *Christopher Dang*
 Jason Fittipaldi
 Josiah Greenewald
 Gregory Kuebler

Project Advisor: Dave Small

Project Coordinator: Kristian Damkjer

Team Website: <http://www.jfitt.com/tcrcomplex>

Table of Contents

1.0 – Introduction

2.0 – The Purpose

- **2.1 – Why We Chose To Do This**
- **2.2 – What Needs We Plan To Meet**
- **2.3 – Our Targeted End-User**
- **2.4 – The Demand For a Crowd Simulation Plug-In**

3.0 – Why We Chose to Develop This Project

- **3.1 – The Qualifications of Our Team**
- **3.2 – Why We Chose To Do This**

4.0 – Related Work

- **4.1 – Similar Software/Technology**
- **4.2 – Differences of Our Project and Prior Work**
- **4.3 – Emulation of Prior Work**
- **4.4 – Limiting Factors**
- **4.5 – Incorporating Related Technology**

5.0 – Project Goals

- **5.1 – Fundamental Project Aspects**
- **5.2 – Development of Project Modules**
- **5.3 – Predicted Project Milestones**
- **5.4 – Development and Design Approach**
- **5.5 – Flocking Behaviors (Steering)**
 - **5.5.1 – Movement Rules**
 - **5.5.2 – Flocking Algorithms**
- **5.6 – Individual Agent Behaviors (AI)**
 - **5.6.1 – How the Brain Works**
 - **5.6.2 – Definitions**
 - **5.6.3 – Behavior Library**

- **5.6.4 – Detection Devices**
- **5.6.5 – Default Agents**
- **5.6.6 – Brain Creation Wizard**
- **5.7 – Animation Control Engine (Locomotion)**
 - **5.7.1 – Driven Key Animation**
 - **5.7.2 – Non-Linear, State-Driven Animation**
 - **5.7.3 – Collision Detection**
 - **5.7.4 – Terrain**
 - **5.7.5 – Animation Adjustments**
- **5.8 – Graphical User Interface (GUI)**
 - **5.8.1 – GUI Structure**
 - **5.8.2 – Brain Creation Wizard**

6.0 – Conclusion

7.0 – References

1.0 – Introduction

The goal of our Senior Project is to create a Maya plug-in that simulates crowds of intelligent agents in a 3D environment. We will also create a narrative animation short to showcase the features of the plug-in. We will use the Maya C++ API and MEL scripting to develop all aspects of our plug-in. The project can be broken down into four distinct modules - Flocking Behaviors, Individual Agent Behaviors (Artificial Intelligence), Animation Control Engine, and Graphical User Interface. Greg Kuebler will be doing the Flocking Behavior section, Josiah Greenwald will be do the Individual Agent Behaviors section, Jason Fittipaldi will be doing the Animation Control Engine section, and Chris Dang will be doing the Graphical User Interface (GUI) section.

2.0 – The Purpose

○ **2.1 – Why We Chose To Do This**

We chose this project because we have a sincere interest in furthering our skills in the areas of artificial intelligence, computer animation, and developing tools that aid in the process of three-dimensional animation.

○ **2.2 – What Needs We Plan To Meet**

Existing crowd simulation tools can cost in excess of ten thousand dollars and this is not an affordable solution on smaller scale animation projects. We hope to provide an affordable and robust alternative to these more expensive solutions.

○ **2.3 – Our Targeted End-User**

Our software is designed to meet the needs of animators who wish to simulate large crowds of autonomous, AI-driven agents. We hope to allow these artists and animators to use our software without the requirement of an extensive programming background. Without the use of a crowd

simulation tool, animators would have to manually key frame the actions of each individual agent.

- **2.4 – The Demand For a Crowd Simulation Plug-In**

There is an increasing need in films (animated *and* live-action) for scenes that require hundreds upon thousands of characters to interact with each other and their environments in an intelligent fashion. Casting thousands of extras to fill complex scenes like this in a live-action film is not a viable solution from a logical or financial viewpoint. Not only would paying such a large number of actors be extremely expensive, but every actor would need to be appropriately trained and outfitted to suit their roles. The creators of AI.implant for Animation made the comment that “the future of the entertainment industry is computer generated ... AI animation will, in the next five to ten years, become the standard to animate characters ... human extras will no longer be used in productions [BioGraphic2004].” AI animation even allows the increased scope of dynamic characters to be used in productions. This is precisely what Weta Digital did for the *Lord of the Rings* series where they designed “Massive” which simulated massive battle scenes between an array of characters such as humans, orcs, and elves [Weta2004]. “Massive” showcased the power of crowd simulation tools to the entertainment industry and has ignited a heavy interest in AI-driven animation.

3.0 – Why We Chose to Develop This Project

- **3.1 – The Qualifications of Our Team**

Our team consists of four senior-level Digital Arts & Science engineers. All four members have demonstrated exceptional artistic *and* engineering abilities in our previous coursework. In *Digital Production Studio 2* there were two team projects that received the highest rating from both their peers in the class and the course instructor Dave Small. Our team is made up of the most talented and dedicated members of those two groups from DPS2.

All the members of our group also left DPS2 with the necessary skills involved with completing a complex and long-term production project. Other areas of our education include artificial intelligence, object-oriented programming principles, and the C++ language.

We are excited about this project and have already invested a lot of time and money into preparing ourselves for the task ahead. We have purchased *numerous* books on the subjects of artificial intelligence, C++, programming Maya, and all the areas involved in creating a 3D-animated short film. Aside from books, we have also purchased an extensive library of high-quality training DVDs that have already proven to be a valuable asset to our production.

- **3.2 – Why We Chose To Do This**

Choosing this topic was easy for us because we have a genuine interest in the areas of 3D animation, programming, and artificial intelligence. As a group, we see ourselves working together after graduation, and we hope to develop our future interests in these areas together. Since we wish to pursue careers in the synergy of these fields, we also hope to use our final project result as a definitive portfolio piece that will showcase our abilities.

4.0 – Related Work

- **4.1 – Similar Software/Technology**

The two pieces of software that draw the most similarities to our project are Massive and AI.Implant. These are both robust programs that assist in creating large crowds of intelligent characters. Massive is a stand-alone application that allows users to import motion-captured data or key-framed sequences to use as motion clips for their characters. AI.Implant provides similar functionality to Massive, but is closer to what we are doing because it is a Maya plug-in.

In the research and education field, there has been a lot of work done involving crowd simulations with intelligent agents. Craig W. Reynolds and Xiaoyuan Tu are the most

noteworthy examples. Reynolds has been doing research in the areas of flocking behaviors that involve a system of *Boids* [Reynolds 1998]. He has featured the success of the Boids model through in a film called “**Stanley and Stella in: Breaking the Ice**” which was presented at the Electronic Theatre at SIGGRAPH in 1987. Xiaoyuan has written a paper called “Artificial Animals for Computer Animation” that involves extensive research involving biomechanics, locomotion, perception, and behavior [Tu 1996]. Reynolds’ and Tu’s research are simulations involving specific models. Their work is not designed as a tool-set but as proofs of the functionality of their system models.

- **4.2 – Differences of Our Project and Prior Work**

The most significant difference between our project and pre-existing work is the cost. Crowd Simulation tools like AI.Implant and Massive cost ten thousand dollars or more. While this may be a feasible solution for a larger studio, we hope to provide a much cheaper alternative for smaller productions or people who just have an interest in animating crowd simulations.

Another area we plan on distinguishing ourselves is the Graphical User Interface (GUI) and easing the interaction between users and the software.

- **4.3 – Emulation of Prior Work**

We certainly plan to emulate some of the proven technologies like Reynolds’ work on varying flocking patterns and the inter-group behaviors such as cohesion, separation, etc. When it comes to emulating existing software like Massive and AI.Implant, we certainly share some similar ideas and plan to incorporate the best of those ideas in our own way.

- **4.4 – Limiting Factors**

There are a number of limiting factors we face, from technological, to financial and time constraints. Technologically, the area of artificial intelligence is a very difficult field and we are not experienced developers in this area. Similarly, we have never developed a Maya plug-in, but we look at this project as a challenge and a learning experience. Financial disadvantages are

obvious – we are a group of students with no monetary backing of any kind. Fortunately, we either own the required software licenses or have them available to us in the university computer labs. Time is a big factor because we have never developed such a large-scale project over such a long period of time. It is hard to gauge how long each task will take us to complete without prior knowledge or personal experience in these areas. We are confident that we will overcome all of these limitations and emerge from this experience with stronger abilities as a result.

○ **4.5 – Incorporating Related Technology**

The only existing technology we are incorporating into our project is Maya. Possible development issues will be interfacing with the Maya C++ API and fully utilizing all of Maya's existing modules (Dynamics, Rendering, etc.). Our main goal is to completely familiarize ourselves with all of the features Maya has to offer and to utilize the best of those within our plug-in.

5.0 – Project Goals

○ **5.1 – Fundamental Project Aspects**

There are four core modules for our project – Flocking Behaviors, Individual Agent Behaviors (Artificial Intelligence), the Animation Control Engine, and the Graphical User Interface (GUI). Each member of the team is responsible for the completion of one module. The individual module responsibilities are as follows:

- Greg Kuebler Flocking Behaviors (Steering)
- Josiah Greenewald Individual Agent Behaviors (AI)
- Jason Fittipaldi Animation Control Engine (Locomotion)
- Christopher Dang Graphical User Interface (GUI)

- **5.2 – Development of Project Modules**

For the plug-in to work properly, all four modules must be completed and fully functional.

However, they can all be developed concurrently without any significant overlaps. Because of the modularity of our plug-in, we will know exactly which functions need to be developed and implemented. This allows each team member to work remotely from the rest of the team.

- **5.3 – Predicted Project Milestones**

At this point in time, it is too difficult to predict our project's milestones. Everything discussed in this proposal is a primary goal to be completed at the end of the year. The following are secondary goals, which we hope to complete if time permits. However, they will not take priority over the primary goals.

Secondary Goals:

- Locomotion over varied terrain types
- Agent variation options – the ability to use one base agent model and spawn numerous, varied offspring agents from that model.
- Advanced collision detection – use of a more accurate bounding box technique that will allow for accurate collisions between agents and obstacles.
- More behaviors included in a robust behavior library.

- **5.4 – Development and Design Approach**

From a design standpoint we are working top-down in order to analyze the bigger picture. This allows us break up more complex tasks into smaller, more specific functions. Once we have fully dissected the project, we will utilize a bottom-up approach, building the plug-in one stage at a time.

- **5.5 – Flocking Behaviors (Steering)**

The flocking behavior aspect of our project controls the movements of the points in space. The individual actions matter within flocking only for the short term contribution to the general crowd characteristics/behavior. The flocking behaviors can be described with movement rules that govern how the individual points interact within the group. The flocking algorithms describe how the user has the ability of directing the group with several algorithms that will create a dynamic control of the flocking group.

- **5.5.1 - Movement Rules**

Separation: This can be described as the kind of personal space for each agent. This space will help determine how far between the other agents the individual agent maintains while in the flocking group. This will help prevent the crowding of the local agents and make the grouping look more natural. This will be difficult task to handle within the flocking system since the plug-in we wish to design will handle different kinds of character geometry. Separation considers proximity queries, which basically mean that that each agent will keep track of its neighbors in accordance to its own space so that body parts do not overlap. This can build the complexity of the system very quickly and we wish to keep the complexity down to a minimum in order simulate the system in real time. From Craig Reynolds's research [Reynolds 2000], he found a lot of interacting particle systems to have the asymptotic complexity of $O(n^2)$. Reynolds has found that a spatial data structure that performs of the task of keeping the agents "pre-sorted" based points in space. This eliminates the needed to examine the entire population of agents or the whole area. Reynolds used a bin-lattice spatial subdivision setup, which is like a box-shaped region of space that is sub-divided into "bins". There are other systems provide a possible better performance such as Popovic [Popo 94] has developed. We will probably try to implement a "pre-sort" proximity queries system

Cohesion: Steer to move toward the average position of local agents [Reynolds 1998].

This is the idea of maintaining a group. Sometimes various agents within the group are separated slightly by obstacles or events and cohesion will help rejoin individual agents to the group when possible.

Alignment: Steer towards the average heading of local agents [Reynolds1998]. This is the normal for each of the agents. Basically this is the direction that that individual agent will be use as its pointer in space. For Example on the model this will like a line coming out perpendicular to the vertical body in the direction the eyes of the agent character. The character will perform such actions of running/walking in the particular direction.

Avoidance: Steer the main group away from static and/or pursuing obstacles (predator).

- *Static obstacles:* This includes such things as walls, trees, or other objects in the environment that will take into account group movement for fleeing. The idea is to allow the user to highlight geometry that will be static obstacles in the scene. This way the agents have the ability to navigate a space with various boundaries.
- *Non-Static Obstacles:* This includes avoiding collisions with other agents, moving objects, and non-autonomous characters. Predators/Prey are considered non-static obstacles; however it will have its own special dynamic definition of controlling intensity and events. The reason for making Predators/Prey have their own specialized options (such as contact collision triggers) is because moving objects and non-autonomous characters will not need all the reactions

and relations of Predator/Prey interaction. This will help reduce unnecessary checks and calculations between various agents and objects.

Sub-Dividing groups:

(*Escape panic* [Saber 2003]) This involves a large herd that can be divided into sub-herds when multiple obstacle factors build up. This has close relationship to avoidance since it uses a calculation of intensity of obstacles build up before a decision is made by sub-leader to split the group for a new escape. For example, if there is a tight space the whole herd cannot effectively pass through at once while maintaining a safe inter-agent distance. If a large herd happens to be divided, there will be various agents that will have sub-leader trait to lead the sub-group. Another example is the predator coming right into a main herd very quickly which will divide the main herd into sub-groups that may scatter left, right, or straight. This will help the scattering effect look more realistic and natural.

- **5.5.2 – Flocking Algorithms**

Goal based – agents move towards a predefined goal. The idea is to allow the user to attach a goal(s) to the scene with a restricted range of values. The agents will go after or avoid these goals depending on the ranking of those values. For example, if a goal has a value between 0 to -1, an agent will avoid the goal, with the worst ranking (-1). This can be attached to the predator agent so the herd/prey agents will recognize the agents that pose a threat. Conversely, the agents themselves can become goals for the predator agent. The goals do not necessary have to be attached to the actual agents.

Path based – Agents that follow a predefined path. There will be intensity values attached to the path that dictate how closely the agents follow the exact path.

Region based – This is a region of 2-D or 3-D space that defines the area in which agents can move. For example, a rectangular fish tank would define the boundaries in which the fish can swim.

○ **5.6 – Individual Agent Behaviors (AI)**

This module deals with the individual behaviors for each type of agent. These behaviors make up the brain of an agent. The brain governs the agent's mind states, actions, and innate qualities. It allows the agents to become autonomous characters when placed in an environment. We will have default agents who already have a brain that can be modified or taken as is, and also a "Brain Creation Wizard" for users who want to create their own agents for the crowd simulation.

NOTE: The user will also be able to set the brain up so that agents follow scene specific actions and are not completely autonomous. An example of this would be if the user wanted a herd of stampeding animals: they could make the agents follow a path and give them only one action...*Run*.

• **5.6.1 – How the Brain Works**

A brain is made up of behaviors. A behavior consists of three parts. The first part of a behavior is called a State. A State is a particular mindset that the agent is in at any given moment. A State can be triggered by either an agent's detection device, or the value of one of the agent's Qualities. When a State is reached, one or more Actions get triggered. Actions make up the second part of a behavior. An Action is what the agent does as a result of a particular State (the animation). Actions take precedence over each other by their priority values. The Action with the higher priority will happen before an action of

lesser priority. When an Action starts, one or more of the agent's Qualities are affected. Quality is the final part that makes up a behavior. Qualities are innate attributes that each agent is born with. These values have a max that can be generated randomly or manually set for each agent. They also change over time according to what action is taking place and they begin changing as soon as that action starts. Qualities can trigger new States, which in turn trigger new actions depending on their values.

Once given a State, an agent will decide its action based on a Fuzzy Decision Tree. This is like a Binary Decision Tree with the "in-betweens" filled in. It will be based on the current priority of the actions. In a Binary Decision Tree there are two decisions, true or false, in this Fuzzy Decision Tree there will be various levels of true and false.

Behavior Example:

An agent detects a predator. The agent's State becomes *Scared* which triggers the Action *Evade Predator*. As the agent runs his *Endurance* Quality decrements and as it gets closer to zero the agent's velocity slows down. Finally when the *Endurance* reaches the value zero the State *Tired* is triggered which causes the agent's Action to become *Rest* (Of course, the priority of *Rest* would have to be higher than the priority of *Evade Predator*). The *Rest* Action causes the *Endurance* Quality to increment. As soon as the *Endurance* reaches a certain value (the priority of *Rest* goes down and the priority of *Evade Predator* grows higher) the State again becomes *Scared* (if the agent still detects the predator). This will keep repeating until another state is reached.

- **5.6.2 – Definitions**

a.) **State** – A mindset that the agent is in at a particular time. States can be triggered by an agent's sensors, or a certain value of a quality the agent has. States can overlap.

- b.) Actions** – These are the actual motion clips that are executed by the agent for a particular state. There can be 0 or more motion clips assigned to each action as explained in the *Animation Control Engine* section.
- c.) Required Abilities** – These are the abilities that agent has to have in order to carry out one or more of the actions.
- d.) Quality** – This is an innate attribute that is affected by the actions taking place. The Quality value determines the priority value of the action.
- e.) Increment Factor** – This is the function that increments a particular quality per period and the means by which it increments. For example this can increment on a slope or by a constant value based on time.
- f.) Decrement Factor** – This is the function that decrements a particular quality per period and the means by which it decrements.
- g.) Priority** – The priority level is the amount of importance the action has. An action with a higher priority will take precedence over one with a lower priority. This can change with the quality that is associated with it. Each priority level is agent specific. (The values given below are placeholders.)

- **5.6.3 – Behavior Library**

This section deals with the various default behaviors that are given in our plug-in. For every behavior and its individual parts there are values that can be dynamically set, modified, or generated randomly. The agents do not have to follow this brain structure. Users can create their own agent brains and also hard code simple actions for short-term, scene specific situations.

NOTE: The following is only a portion of the future behavior library to give the client a more in depth grasp on the brain structure concept.

A.) State: Scared**1.) Action Category: Evade Predator**

- **Actions:**
 - No action-----Priority Level: 0
 - Scared reaction-----Priority Level: Low
 - Alert others-----Priority Level: Low
 - Run away-----Priority Level: Medium-high
 - Hide-----Priority Level: situation spec.
- **Required Abilities:**
 - **Predator Detection** – agent can detect a predator at a certain distance.
 - **Peer Communication** – agent can warn other agents of an oncoming predator.
- **Quality Affected: Endurance**
 - **Increment Factor:** Time; 1 for every time unit; lasts from start of *Rest_Stop* Action to end of *Rest_LayDown* Action
 - **Decrement Factor:** Time; 1 for every time unit; lasts from start of *EvadePredator_Run* Action to start of *Rest_SlowDown*. 0.5 for every time unit; lasts from start of *Rest_SlowDown* to start of *Rest_Stop* Action.
- **Quality Affected: Danger**
 - **Increment Factor:** The proximity of the predator; 1 for every unit of length away while inside of the predator detection radius; starts

when predator is detected and ends when predator is not in radius anymore.

- **Decrement Factor:** Time; 1 for every time unit the predator is outside the predator detection radius; starts when predator leaves the detection radius.
- **Action Priority:**
 - Endurance++ && Danger++ yields Priority++; Endurance-- || Danger-- yields Priority--
 - Max of (Endurance/2)*Danger <=> Highest Priority Level
 - Min of (Endurance/2)*Danger <=> Zero Priority Level
- **Notes:** The agent has a radius of detection and can detect predators within that radius. Notice that this Action has a max priority of Highest therefore it executes over all other actions of a lesser priority. Also sub-actions such as *ScaredReaction* will only execute once for a given situation so not to look unrealistic and repetitious.

B.) Other Agent Attributes:

- **Leadership** – An agent with this attribute can have any number of followers associated with it. The followers basically follow the leader around to the best of their ability. This attribute can be attached to autonomous characters as well as non-autonomous, hand animated characters.
- **Life** – This controls how many hits/attacks an agent can take before it dies.
- **5.6.4 – Detection Devices**

This is the way in which an agent can see its environment. This will be proximity based where there is a radius around an agent and everything inside of that radius sends out a

signal telling the agent what it is. Once the agent detects the presence of something it can react accordingly.

- **5.6.5 – Default Agents**

- **Little Creatures (Biped)** – These beings are going to be the main crowd agents that we will be showcasing. They are small simple-minded characters with simple animations.
- **Bird** – This will be a generic bird that utilizes flocking.
- **Quadruped** – This will be some sort of animal creature that is simple minded and simple to animate.
- **Insect** – This will be a non-flying insect creature that utilizes the flocking attributes.

- **5.6.6 – Brain Creation Wizard**

This is a completely abstract and dynamic process in which the user can easily build new behaviors to create a brain for their own agents. The user will be able to create a behavior to handle almost any situation with two processes.

The first process works by allowing the user to designate one or more target objects (target objects can be static or non-static) and give them a name. Then the user has to give the agents a detection radius of any size for target objects with that name. Now the user can create states that get triggered when the target objects are within the detection radius. Once a state has been defined the user can add actions and their respected motion clips. The user will also have the ability to assign priority ratings to each of the actions and qualities that will affect the priorities' values.

The second process allows users to create states that are triggered by quality values, not target objects. In this method the user creates a state and the actions and qualities

associated with it and then they tell the state to be triggered when one of the qualities reaches a certain value.

- **5.7 – Animation Control Engine (Locomotion)**

This section discusses how we will control the animation within Maya. The flocking system essentially acts upon a particle-based system. The Animation Engine will take those particle and map more complex geometry to them. Typically the complex geometry's center of mass (the central point of the object or skeleton rig) will take over the particle's position.

There are two types of animation techniques we will offer. For simpler simulations animators can use *Driven Key Animation*, but more complex simulations will require *Non-Linear, State-Driven Animation*. Important to both of these animation techniques is how they handle *Collision Detection* and various types of non-flat *Terrain*.

Users will also be able to manually adjust the animation once they decide to halt the simulation, or the simulation completes its scheduled timeframe.

- **5.7.1 – Driven Key Animation**

Driven Key Animation should only be used if the desired outcome is a simple, locomotion-driven animation. Typically, you will want to use Driven Key when your character or object's animation will be *directly* related to one or more of its attributes (XYZ coordinates, speed, acceleration, time, etc.).

An example of Driven Key Animation is a flying bird. This bird will flap its wings based upon how fast it needs to accelerate or decelerate. The bird could also have an occasional head movement based upon time. After a certain period of time the bird will act as though it were looking around at its surroundings.

You can see that this animation technique works well for simple simulations where no real “intelligence” is necessary. However, if animators desire a more interactive, semi-intelligent animation system we have also provided the following *Non-Linear, State-Driven Animation* system.

- **5.7.2 – Non-Linear, State-Driven Animation**

Non-Linear Animation is achieved within Maya by creating motion clips (or one-framed pose) from key-framed animation sequences. Poses can be used as testing prototypes or placeholders for the motion clips. By using Maya’s Trax Editor animators can take a hand-animated character or object and create a clip (or pose) like a walk-cycle, an arm waving hello, and so on. In a style similar to video editing software, these clips can then be positioned at certain frames, scaled, blended together, etc.

As discussed in the *Individual Agent Behaviors* section, a state is the current mentality of an agent. Within each state are possible actions the agent can take once it enters a given state. Animators can determine what actions their agent has and what motion clip (if any) will be associated with that action.

For example, an agent spots a predator and its current state becomes *scared*. Now that the agent is scared, some of its possible actions are *no action*, *scared reaction*, *alert others*, *run away*, and *hide*. Each of these actions has a priority level and the agent will determine the best course of action to take.

The task of the *Non-Linear, State-Driven Animation* system is to correctly string all of these action-triggered animations together. Once a new action is triggered the system

will transition from the current action into the new action. This will be achieved by overlapping the two associated motion clips and blending them together to create realistically animated agents.

The Non-Linear system will be able to determine at which frame to place each clip, how much to scale the clip (start frame to end frame of the clip), if it needs to be looped and how many times, and how to blend between itself and the previous clip.

- **5.7.3 – Collision Detection**

Collision detection is extremely important to prevent agents passing through other agents, static objects in the environment, etc. The flocking behaviors and individual agent behaviors will determine when a collision is imminent. This will then trigger an appropriate state/action for avoiding the object.

If you are using a simple *Driven Key Animation* system, your characters/objects will merely continue what they were doing and move around the obstacles. For the *Non-Linear, State-Driven Animation* system, your characters/objects will transition from their current course of action into their obstacle avoidance reaction.

Animators should ensure that they animate their agents with constraints so that they do not collide with themselves (i.e. limbs intersecting torsos).

- **5.7.4 – Terrain**

Initially, our goal is to achieve movement on flat terrain types. If time permits, we will implement movement over all kinds of terrain types (not just flat surfaces!). By designating terrain geometry in the simulation environment and what part of the agent's geometry will be coming in contact with the ground (if any), we hope to allow agents of any type (biped, quadruped, and beyond) to traverse complex surfaces.

- **5.7.5 – Animation Adjustments**

After each simulation the user will have the ability to adjust the animation in any way they please. They can go back and manually key frame areas, or adjust the layout of the motion clips in the Maya Trax Editor (much like a piece of video editing software).

- **5.8 – Graphical User Interface (GUI)**

The software GUI will be developed for each programming module of the project. The Flocking Behaviors Module, Individual Agent Behaviors Module, Brain Creation Wizard Module, and the Agent Alteration Wizard Module have module-specific settings and attributes that can be dynamically adjusted.

Functionality and effectiveness of the program relies heavily in the way it is presented to the user. Maya's interface is powerful and provides a large set of tools and functions that are not always in use. These tools and functions can be automated and stored as advanced scripts in Maya through its C++ API. Our GUI focuses on increasing the animator's workflow by providing a relevant and custom toolset for each of the project's modules.

- **5.8.1 – GUI Structure**

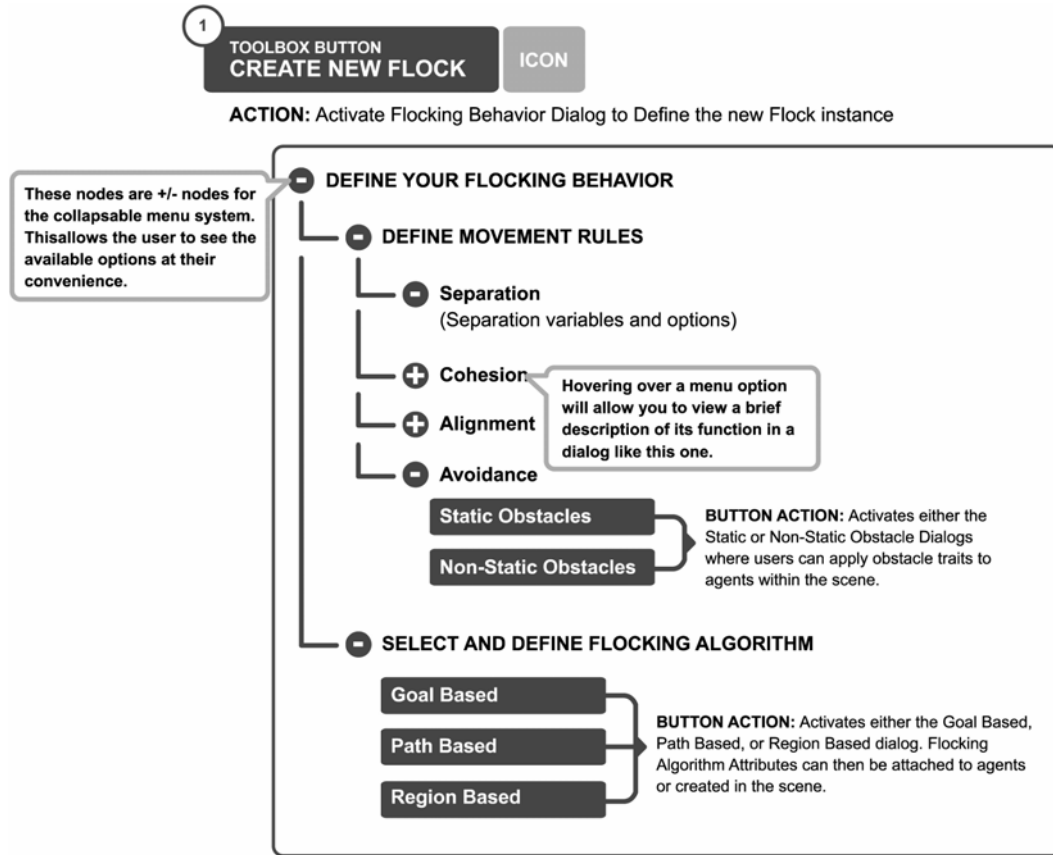
The modules will be controlled through Dialog windows. Each dialog will allow the user to access controls respective to Flocking Behaviors and Individual Agent Behaviors. The GUI will apply changes to the scene in real-time, giving the user instant control over the scene.

The following is an outline of the GUI structure for these two modules:

- Toolbox button that activates the appropriate module dialog
- Collapsible Menu

- Control options and creation tools
 - Customized through Pop-up dialog boxes
 - Buttons
 - Sliders
 - Checkboxes
 - File Browsing Dialog (used to attach motion clips to states and actions)
- 2D simulation (HUD Display)
 - Agents and obstacles represented.
 - Tracks and displays the location of agents in a top-down view.
 - Visualize and control movement easier.

The following is a diagram that illustrates a possible dialog structure for the modules.



• **5.8.2 – Brain Creation Wizard**

Within Individual Agent Behaviors dialog, a Brain Creation Wizard button will be available. This feature allows the user to create customized agents to be used in the crowd simulation. The Brain Creation Wizard has its own popup dialog window. The control interface will be similar to the collapsible menus used for the Flocking Behaviors and Individual Agent Behaviors. For each custom agent, the user will be able to define custom states and actions associated to those states.

6.0 – Conclusion

In closing, we look forward to this project and all of the challenges it may present. We feel we are well qualified and can learn a lot from this experience. The development of a free crowd simulation tool for Maya not only benefits our team, but the 3D animation community as well.

We feel that we have broken our project down into four distinct programming modules. This allows us to develop this tool concurrently throughout each phase of development, while still having the ability to work independently of each other.

7.0 – References

[BioGraphic 2004] BioGraphic Technologies (2004) “AI.implant for Animation: White Paper”

Montreal, Canada; pg 1

http://ai-implant.com/pdf/AI.implant_Games_White_Paper.pdf

[Popo 94] Zoran Popovic (1994) "The Rapid Simulation of Proximal-Interaction Particle Systems"

unpublished manuscript. <http://sulfuric.graphics.cs.cmu.edu/~zoran/actin/pop.ps>

[Reynolds 1998] Craig Reynolds (1987) “Boids ; Background and Update” proceedings of SIGGRAPH

1987, Symbolics Graphics Division and Whitney/Demos Productions, last site update Sept 2001;

<http://www.red3d.com/cwr/boids/>

[Reynolds 2000] Craig Reynolds (2000) “Interaction with Groups of Autonomous Characters”

proceedings of *Game Developers Conference 2000*, San Francisco, California, pages 449-460

published at <http://www.red3d.com/cwr/papers/2000/pip.html>

[Saber 2003] Reza Olfati Saber & Richard M. Murray (2003) “Flocking with Obstacle Avoidance:

Cooperation with Limited Information in Mobile Networks” submitted to IEEE Conference on Decision and Control 2003, California Institute of Technology published at

http://www.cds.caltech.edu/~murray/papers/2003c_om03-cdc.html

[Tu 1996] Xiaoyuan Tu (1996) “Artificial Animals for Computer Animation” Thesis for the Department

of Computer Science at University of Toronto, <http://www.dgp.toronto.edu/people/tu/thesis/thesis.html>

[Weta 2004] Weta Digital (2004); <http://www.massivesoftware.com/>