Master Project

ZHUO YAO LU

MSC Computer Animation

Media School

Bournemouth University

NCCA 2005

Contents

Part 1

Flocking System in Maya

Mel Script

ZHUO YAO LU

Media School, Bournemouth University, UK
September 2005

CHAPTER 1

Introduction

Simulated group behaviors are always a difficult and challenging topic computer graphics (CG), but it is still wildly used in the industry today. Successful examples include *Finding Nemo* and *The Lord of Ring* trilogy. There are some existing production tools which can be used to simulate group behaviors in 3D, such as MASSIVE and AI.IMPLANT. They are very powerful, and able to simulate most of group behaviors, however they are independent software packages and complex to use. Maya is the most wildly used 3D package. Our flocking system is created using Maya Mel script language, as it can create and simulate all main flocking behaviors conveniently and effectively.

CHAPTER 2

Flocking System

2.1 What is the flocking system?

A “flock” is defined as a group of sheep, birds, fish or people in dictionary. In the real world, a group of creature should have several certain behaviors, for example: random move individually, stay close each other as a group but not too close, move avoid obstacles in the environments. Flocking system is a system that can create a group of objects and simulate their flocking behaviors automatically in 2D or 3D. Let we find out how those real world behaviors are explained in 3D.

2.2 The flocking behaviors in 3D

Flocking can be characterized as having a moderate number of members (relative to particle system and autonomous behavior), each of which is controlled by a relatively simple set of rules that operate locally. The members exhibit limited intelligence and are governed by relatively simply physics.

------ Rick Parent [1. p246]

There are two main forces work in keeping a group of objects behaving like a flocking: *collision avoidance* and *flock centering* [1]. Collision avoidance is relative to other member of flock and the other elements in the scene like obstacles. Flock centering has to do with each flock member, it try to force all members stay in a certain area and move in the same direction. Each flock

member has two controls, a local control and a global control [1]. Local control is controlling the member avoid collide with the other members and its velocity match its immediate neighbors. There are three processes might need to be modeled in the local control: physic, perception, reasoning and reaction [1]. These three processes work similar as they work in the particle system. Global control likes its name, it influence all the members and controls their moving direction.

CHAPTER 3

Related Work of Flocking

3.1 Particle system

The first way of controlling a large number of objects in 3D is Particle System, which is introduced by William Reeves in 1982 [2]; he used to create a sequence of images for the movie *Star Trek II: The Wrath of Khan*. Each particle in this system presents as a single point in space; they could not be complex models. This technique is widely used to simulate fuzz objects today.

3.2 Early flocking system

In 1987, Craig Reynolds created another particle system from Reeves original idea. The particle system he created replaces the points as animated objects. He called each those animated objects “Bird-oids” or “Boids” [3]. The first version of his system, boids consist three simples’ behaviors, *Collision Avoidance, Velocity Matching* and *Flocking Centering [8]*. *Collision Avoidance* is a behavior of the boid avoiding collision with nearby flock members. *Velocity Matching* is a behavior of boid attempting to match velocity with nearby flock members. *Flocking Centering* is a behavior of boid attempting to stay close to nearby flock members.

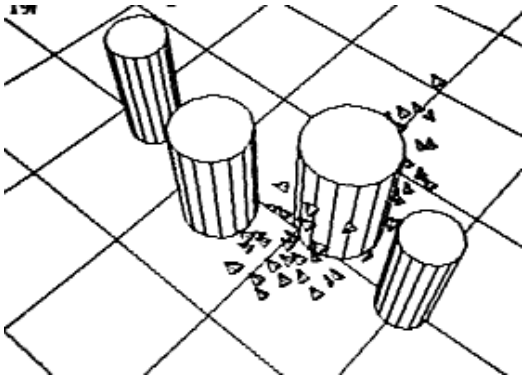
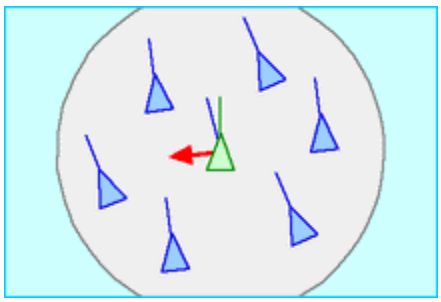


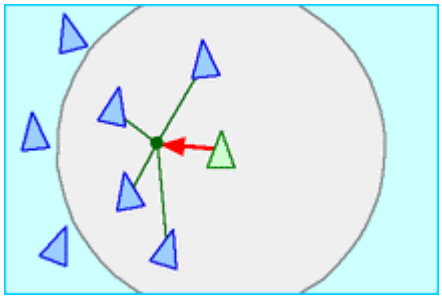
Figure 2: simulated boid flock avoiding cylindrical obstacles (1986)

From *http://www.red3d.com/cwr/boids/ [3]*

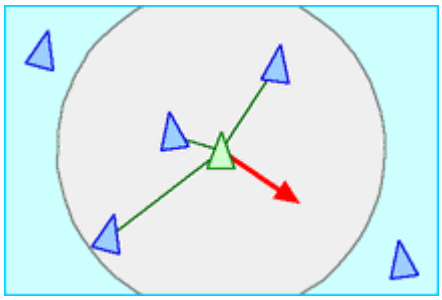
Later, he modified these three behaviors and renamed them as “steering behaviors”, *Separation*, *Alignment* and *Cohesion [3]*. They describe an individual boid maneuvers on positions and velocities its nearby flock members.



Alignment: steer towards the average heading of local flock members



Cohesion: steer to move toward the average position of local flock members



Separation: steer to avoid crowding local flock members

Figure 1: from http://www.red3d.com/cwr/boids/ [3]

Here, the boid reacts only the flock members within its certain small neighborhood around itself. The neighborhood is defined by a distance and an angle. The distance is calculated from the boid center and the angle is measured from the boid's direction of fight.

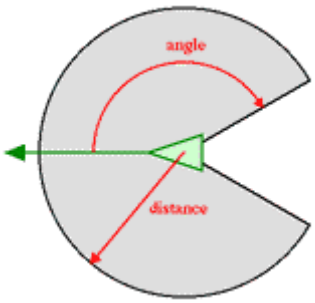


Figure 3: a boid's neighborhood

from http://www.red3d.com/cwr/boids/ [3]

3.3 Ways of simulating flocking behaviors

We mention before, collision avoidance and flock centering are the main behaviors in flocking system. There are several ways can be used to simulate avoid collision. One way is from Reynolds's paper [4], model the flock member's field of view and visual processing, a balance must be decided between the complexity computation and the realistic effect.

Another simple way to do it is creating a repelling force field around each object [1], which is as long as the safe distance of the object. When the flock member gets close to the object, this field start gently push the member away

from the object. As the member gets closer, the force grows stronger. Because this repelling force field has effect distance and it grows weaker from the object surface to the outside. When flock member get closer enough to the object, the force always strong enough repel it away, but when the member away the object, the force get weaker, the member will head back toward the object. This cycle of veering will be repeated again and again. Showed as below.

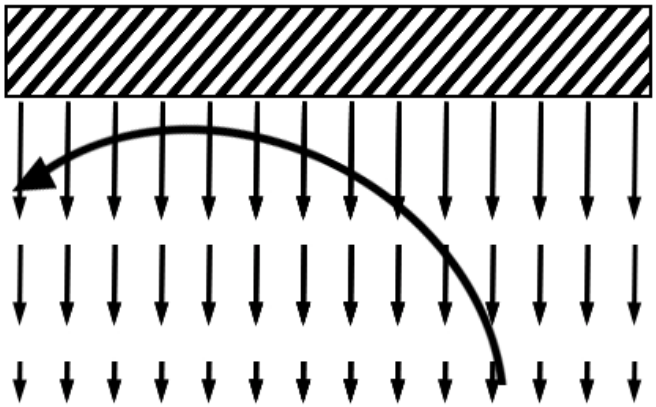


Figure 4: Force field collision avoidance [1]

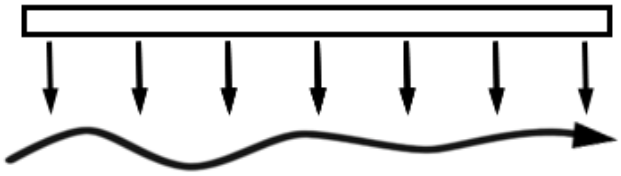


Figure 5: attempt at parallel movement [1]

Another problem of using repelling force field, when the flock member moves directly to the object surface, means its direction parallel the force direction,

the member have an unnatural movement, it stops and back up. Showed as below.

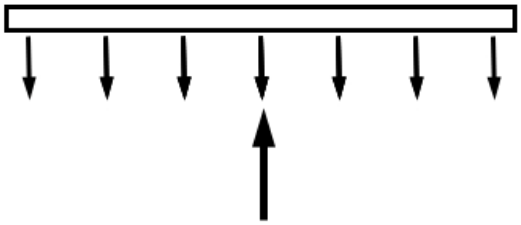


Figure 6: attempt to move directly toward a surface [1]

Also when there is a gap or a hole in the object, because the force is created surround all the surface of the object, if the gap or the hole is not big enough, the average of force in it still can repel the member, that is mean the members are not able to go though the gap or the hole. Showed as below.

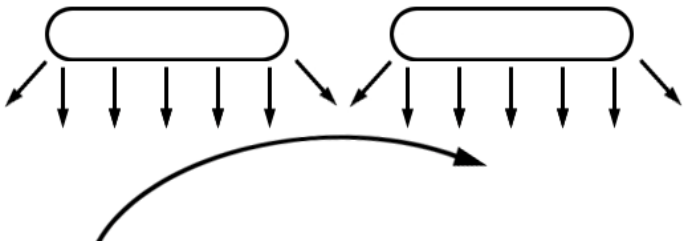


Figure 7: attempt at finding a passageway [1]

For solving those problems, we can use a boundary sphere around the object to divert the member moving path around the object. Once the member is inside the influence area of the boundary sphere, its direction vector will be

tested to see if indicates a potential intersection with the bounding sphere of the object. The way of calculation is the same as the ray tracing. For more detail see Rick Parent's book [1].

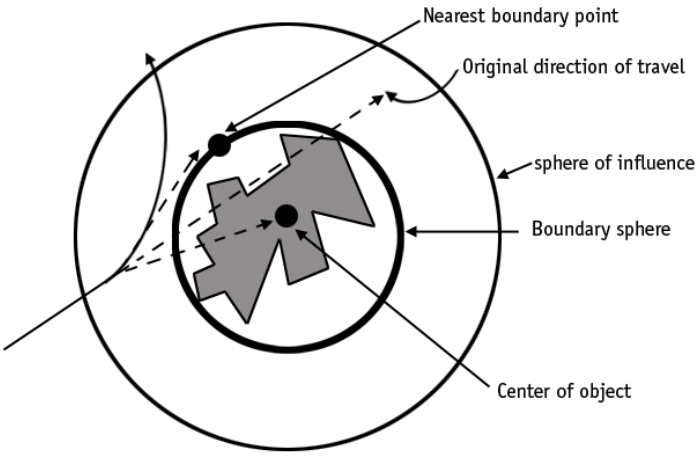


Figure 8: avoid a bounding sphere [1]

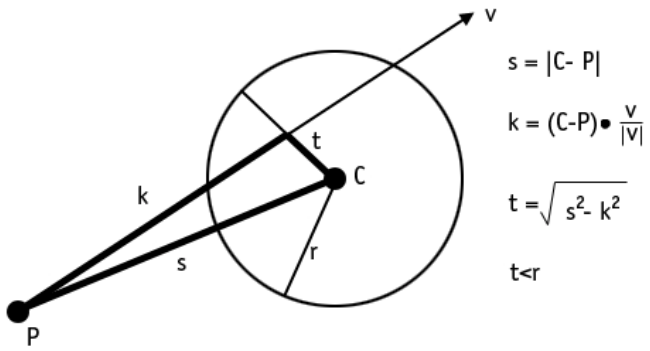


Figure 9: Testing for potential collision with a bounding sphere [1]

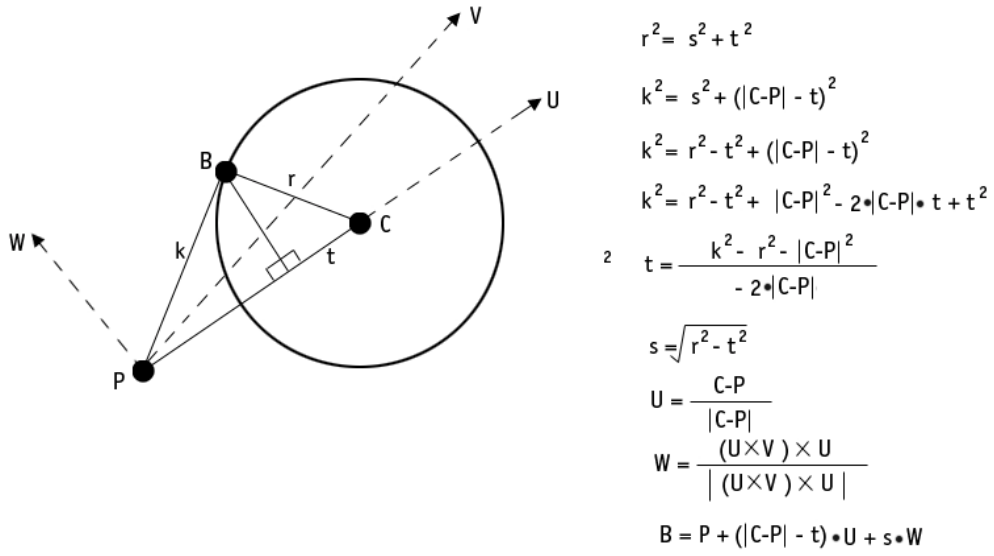


Figure 10: Calculation of point B on the boundary of a sphere [1]

3.4 Software of simulate group behaviors

In the last few years, there are many relative works has been done to simulate groups and flocking behaviors in 3D. Some of the soft wares are developed, for example MASSIVE and AI.IMPLANT.

AI.implant is a real-time interactive artificial intelligence software, which enables digital content creators to create incredibly rich character interactions, resulting in enhanced realism and immersive ness in games, animation and simulation. It is used as a plug-in in Maya and 3d-max those 3D packages. [6]

Massive is a soft ware for creating character crowd simulation. Each character

is controlled by its agent brain; Massive system is setting and running these agents. Massive is a totally independence package. [7]

Those soft wares are very powerful and able to simulate realistic behaviors. The object is created by them has high artificial intelligence and can evolve behaviors.

CHAPTER 4

A simple crowd system using Mel script

Maya has a great particle system, like the particle system that created by Reeves, the particle represents as a very simply object in space, like points and sphere. It is usually used to simulate fuzz objects. This particle system also allows the user animate a large number of objects by using instance function, which use geometry replaces the points in the particle. User can created expression and animate the particle for simulating group and flock behaviors.

Actually, there is another way to simulate flocking behaviors in Maya ---- using Mel script. Mel script language is the heart of Maya. Maya interface is created using Mel. What we can do in the Maya interface also can be done using Mel script. There are many ways using Mel script to simulate flocking behaviors. At first, we look at a simply crowd system in *Mel Scripting for Maya Animators* this book, which is written by Mark R.Wilkins and Chris Kazmier [5].

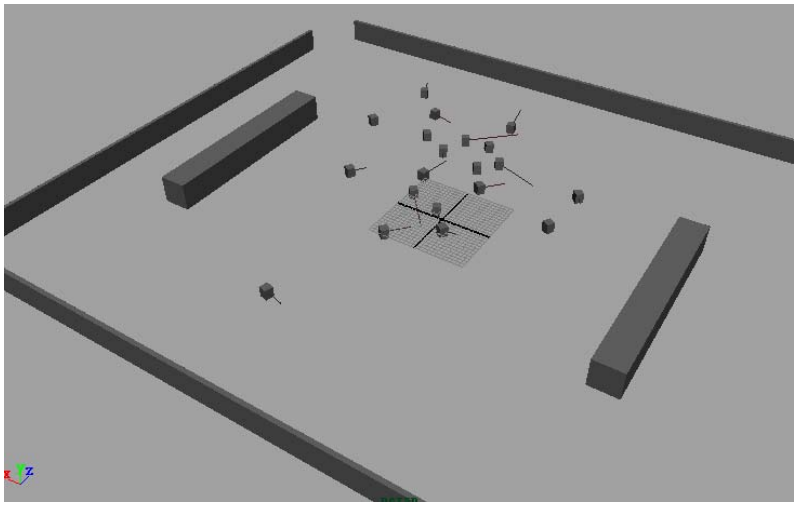


Figure 11: a simple crowd system using mel script in Maya [5]

The flock member here are called “Vehicle”, they have two basic group behaviors, random moving and collision avoidance. Vehicles move around avoiding each other and the obstacle objects in the scene. To achieve the collision avoidance behavior, all the vehicles are created as active rigid bodies and all obstacle objects are created as passive rigid bodies. Each vehicle object has its own radial field, it is used to push another vehicle away when they get close. Obstacle objects do not have radial field, but because they are passive rigid bodies, they also can reacts with vehicles. For the random moving behavior, each vehicle connects with an expression node that defines its movement and calculates its moving direction. All the vehicles here only move in the XZ plane. The vehicles model are atomically created by the script, user do not have much control.

CHAPTER 5

Flocking system in Maya

5. 1 Introduction

We tried to use Maya Mel script language to create a system, which can simulate flocking behaviors easily and convenience. Also, it has better realistic result and less computation than using particle system. Why choose Mel script? First, Maya is the most popular 3D package in the world and also is the main 3D package to be used in the industry today. Second, Maya is a very open software, it allows the user modify or create their own features with its language. There are two ways to create tools in Maya, using Mel (Maya Embedded Language) or using API (Application Programmer Interface) languages. Mel is a powerful and easy to learn scripting language. API provides better performance than Mel, but it is also much complex. Third, Maya already provides fantastic dynamic system, it is able to help user simulate most realistic natural effect. We can use this dynamic system to achieve what we want easily in Maya.

5.2 How this system work

Our system uses rigid body collision and dynamic field to simulate flocking behaviors. We are going to describe how those options work behind the interface here. For more detail about how to use the interface please see the *Flocking System User Guide*.

After you source the script in Maya, type “flockMain;” in the script editor and run the script, a flocking system window is opened. Showed as below. (For more detail see the *Flocking System User Guide --- source the script*)

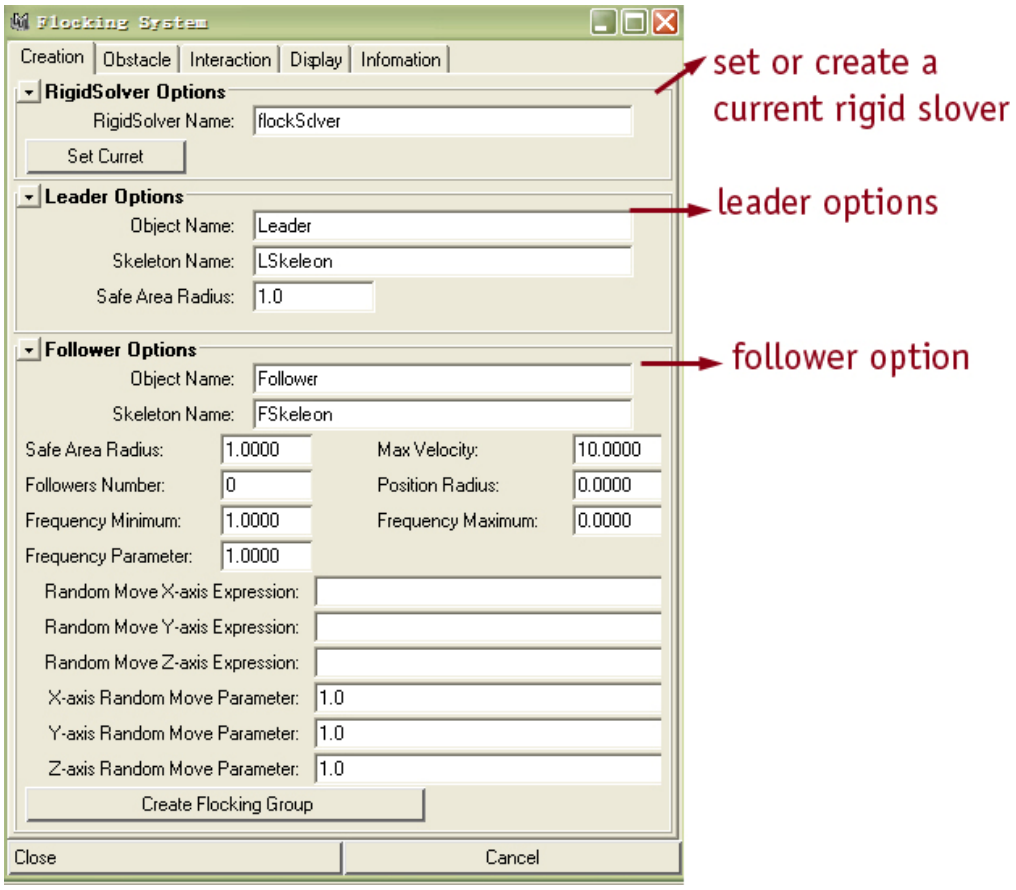


Figure 12: flocking system interface 1

Here, “leader” is the object you created for leading the flock members. The flock members are called “followers”, and the objects that the followers moving avoid is called “obstacle” in this system. The leader can be any object you like, an animated object (for example a fish, a butterfly) or simply just a locator. The follower can be any animated object too, but it must have skeleton in this version. Let we look closer to each option.

5.2.1 Rigid Solver

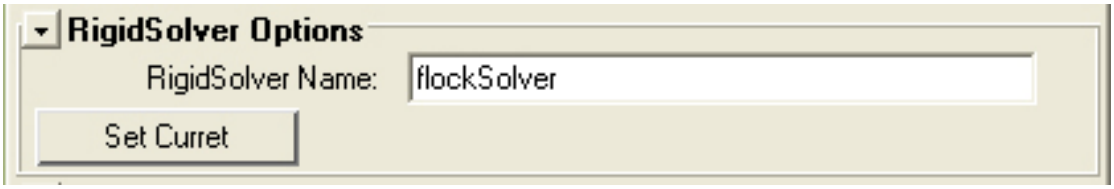


Figure 13: flocking system interface 2

Rigid solver controls the accuracy of the rigid body solution; all the rigid bodies connected with it, rigid bodies can react each other. This option allows the user specify the rigid solver before create the flocking group. If the user set the rigid solver that already exists in the scene, the flocking group can react with all the rigid bodies using this solver. This is very useful for combine the flocking system with another work. Also, user can create another new rigid solver for the flocking system using this option.

Figure 14: flocking system combine with another work

See example FSExample_11.avi and FSExample_11.mb for flocking system work with another object in the scene.

5.2.2 Leader

Figure 15: flocking system interface 3

The leader can be any object, an animated object or simply just a locator. When the leader is created, the original leader will be placed as a child under a locator, which is named *LeaderNameLocator*. For example, you create a leader called “Leader”, it will be placed under a locator called “LeaderLocator”. Showed as below.

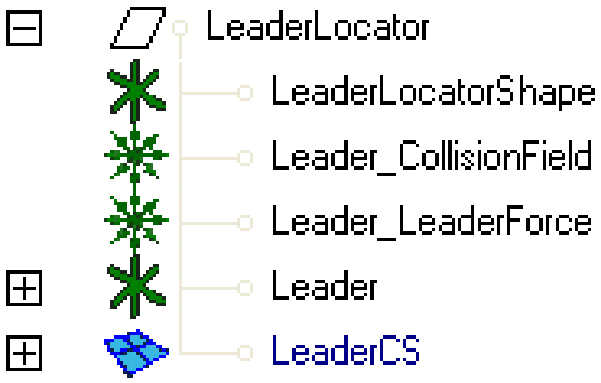


Figure 16: LeaderNameLocator

From the last diagram, you can see there is a polygon sphere and two fields were created at the same time. The polygon sphere is created as a rigid body and named *LeaderNameCS*. In this example, it is “LeaderCS”, the radius of this sphere is depending the value of the *Safe Area Radius* option. This polygon sphere is used to calculate the collision between the leader and all

followers, also it is connected with two fields. One is the *LeaderName_CollisionField* , which is used to push away the followers when they get close to it. In this example, it is called “Leader_CollisionField”. Another field is the *LeaderName_LeaderForce*, which is used to force all followers move follow the leader. It is called “Leader_LeaderForce” in this example. After you create *LeaderNameLocator*, select it and open the channel box, you can see a *Following* attribute. User can key frame it, it controls whether followers move follow the leader or not. Showed as below.

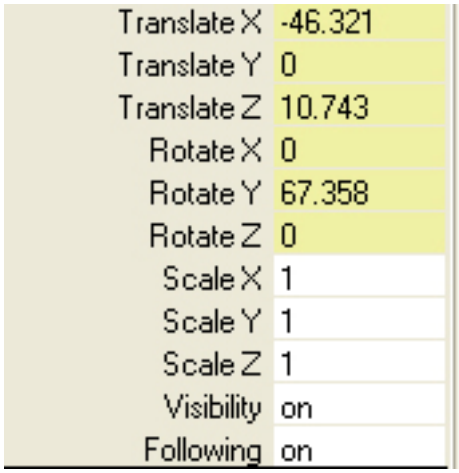


Figure 17: LeaderNameLocator Following Attribute in Channel Box

See example FSExample_6.avi and FSExample_6.bm for key frame the *Following* attribute of leader.

See example FSExample_7.avi and FSExample_7.mb for using an animated object as leader.

5.2.3 Follower

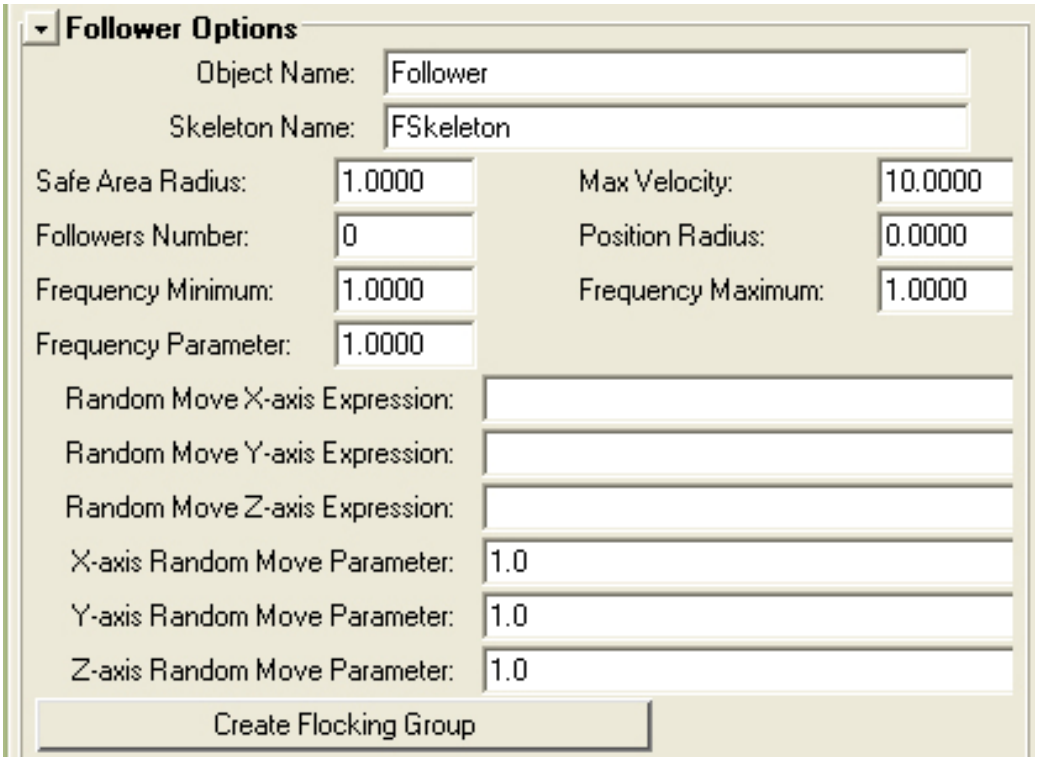


Figure 18: flocking system interface 4

The follower can be any animated object too, but it is required skeleton in this version. When followers are created, each follower is contained in a polygon sphere. Showed as below.

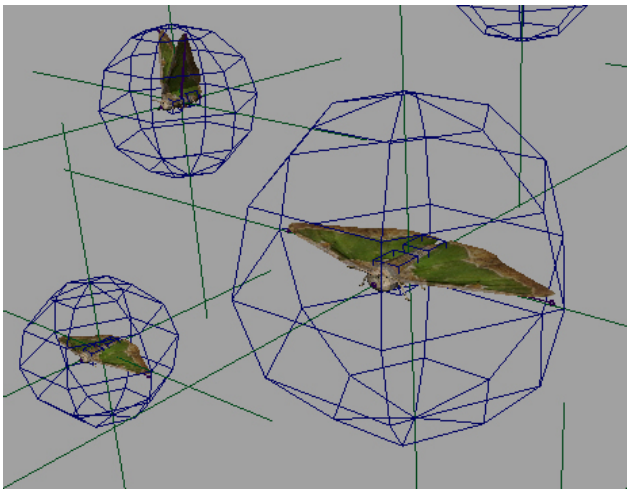


Figure 19: followers

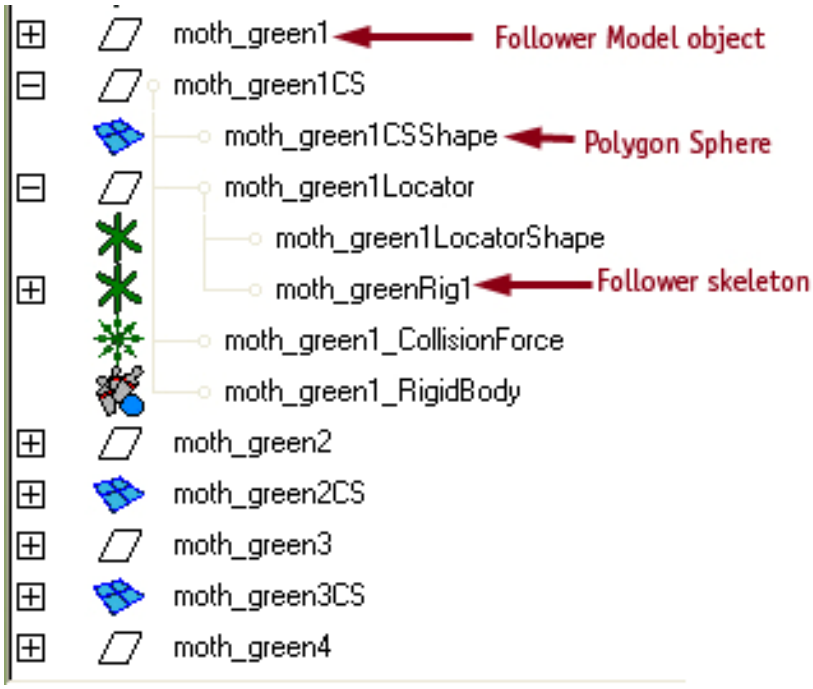


Figure 20: Follower in the outliner

Open the outliner, we can see the followers are named a digit number with its original name. For example, your follower model name is “moth_green”, and its skeleton name is “moth_greenRig”, if you create ten followers, their models will be named from “moth_green1” to “moth_green10”, and their skeletons will be named from “moth_greenRig1” to “moth_greenRig10”. Let we look at a follower “moth_green1”, its relative follower skeleton “moth_greenRig1” is placed under a locator that called “moth_green1Locator”, and they are the children of the “moth_green1CS”. “moth_green1CS” is the polygon sphere, which is created at the same time with the follower. It is used to calculate the collision of the follower, it can not be rendered. If you like to replace the follower, you can select this polygon sphere and move them. The radius of this sphere depend the value you entered in the *Safe Area Radius* option. Usually it should be set

bigger than the follower model. Each polygon spheres is created as rigid body and connected with a radial field, which call *FollowerName_CollisionForce.* In this example, it is called “moth_green1_CollisionForce”. This collision field is used to push away the other followers when they get close to it. The maximum effect radius of this field is defined by the value in the *Safe Area Radius* option too. The relation between them in this system is:

Maximum effect radius = Safe Area Radius * 1.2

Followers are not only controlled by its leader, they are also controlled by its own expression. There are two main functions in follower’s expression. One function controls the follower model rotation in a correct direction. Another function finds out and defines how the follower moves. If the *Following* attribute of the *LeaderLocator* is on, the follower moves follow the leader; otherwise it is controlled by its expression. Let we talk more detail about these two function.

The flocking system requires the model used as followers faces to the z-axis positive direction. Showed as below.

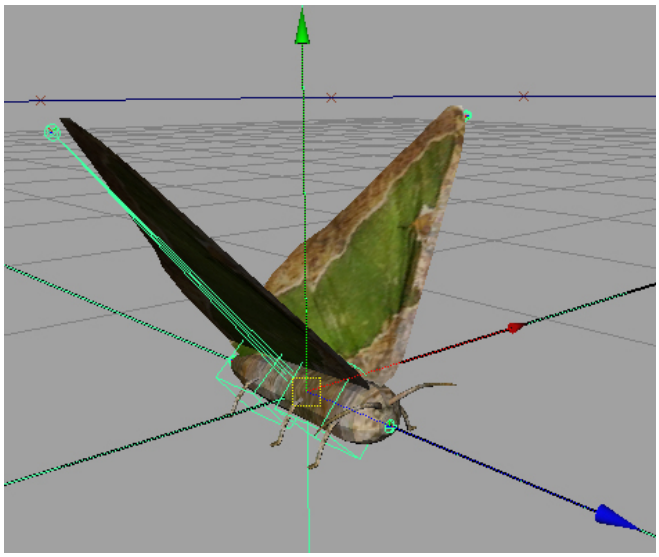


Figure 21: follower model faces z-axis direction

After you create a flocking group and play the animation, at first the follower expression have to measure a vector between its object center and the leader. Showed as below.

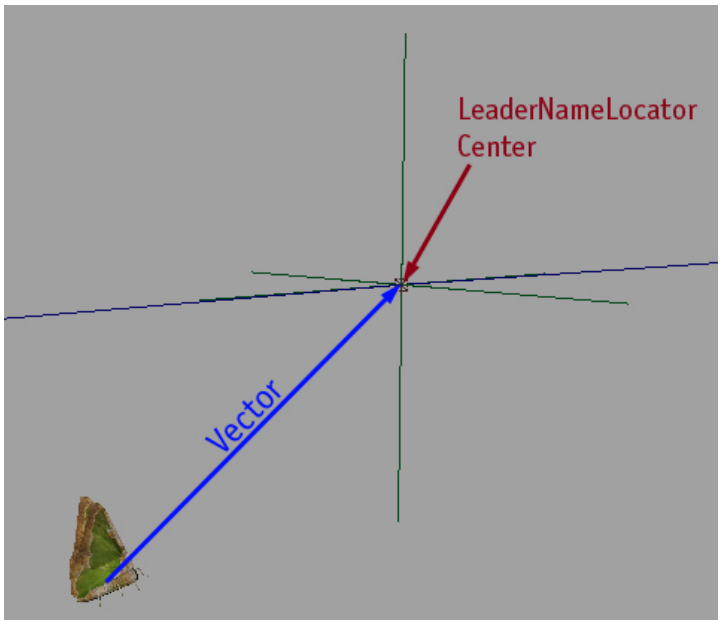


Figure 22: the vector form leader center to the follower center

Second, project this vector on the ZX plane and find out the angle between this projection vector and the z-axis, then rotate the follower in y-axis in this angle.

Third, the expression calculates the angle between the vector and the z-axis this time and rotates the follower in x-axis. Because the expression is run every frame, the calculation is done very frame, the follower always faces to the leader.

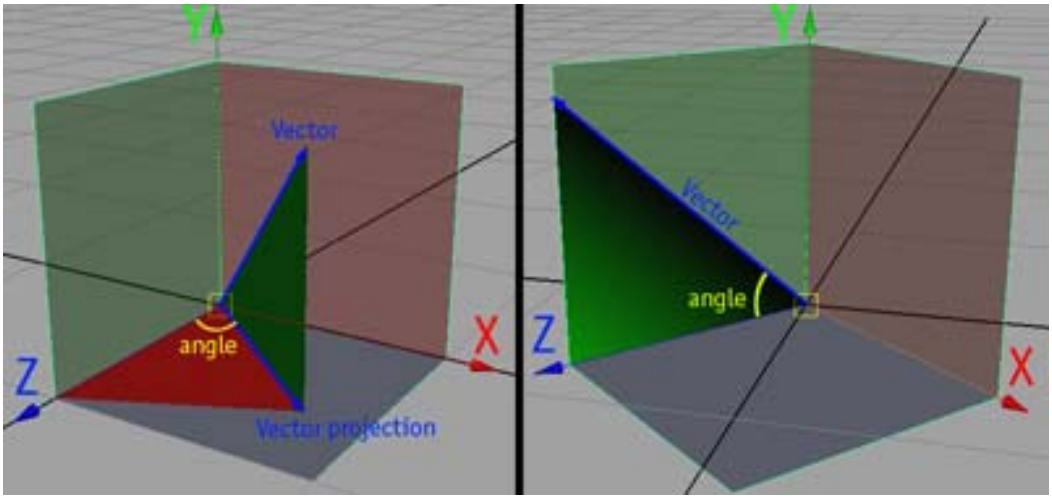


Figure 23: follower rotation

The second function of the expression finds out which situation the follower is. When the *Following* attribute of the leader is on, the follower moves follow the leader. When it is off, the expression defines the movement of the follower. Actually, user can defines this part of the expression using the last six options in the *Follower Options* section. Showed as below.

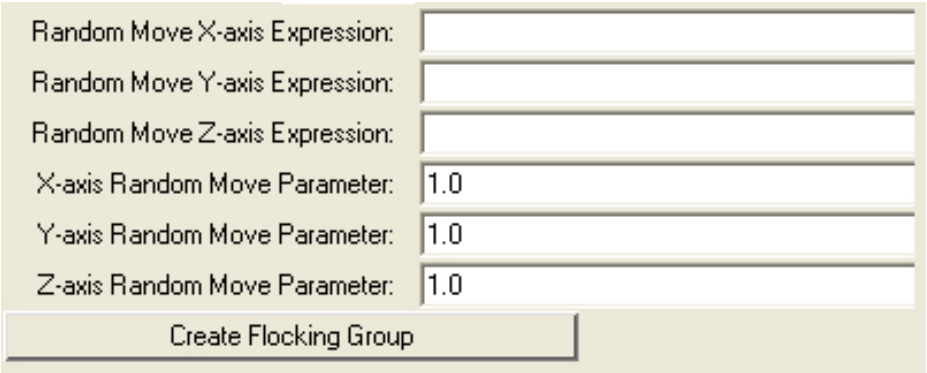


Figure 24: flocking system interface 5

If you leave those options as space, you can see the followers stop move when you turn off the *Following* attribute. What you enter in those option, they will be connected with the follower rigid body impulse attribute. For example, if you enter “noise(time)” in the *Random Move X-axis Expression* option and enter “5” in the *X-axis Random Move Parameter* option. The x attribute of the relative follower rigid body impulse will be “noise(time)*0.45678”. The number here is calculated from “rand(-5,5)”. The relation can simply describe like:

Follower rigid body impulse x = Random Move X-axis Expression* X-axis Random Move Parameter

See example FSExample_1.avi and FSExample_1.mb, the following attribute off on the frame 25 and did not use the *Random Move X-axis Expression, Random Move Y-axis Expression* and *Random Move Z-axis Expression* options.

See example FSExample_14.avi and FSExample_14.mb for set *X-axis Random Move Parameter, Y-axis Random Move Parameter* and *Z-axis Random Move Parameter* options as 0.

When you create a flocking group, you do not want all followers have the same animation cycle. The *Frequency Minimum* and *Frequency Maximum* are created for this reason. All followers animation cycle will be scale between these two values when the followers are created.

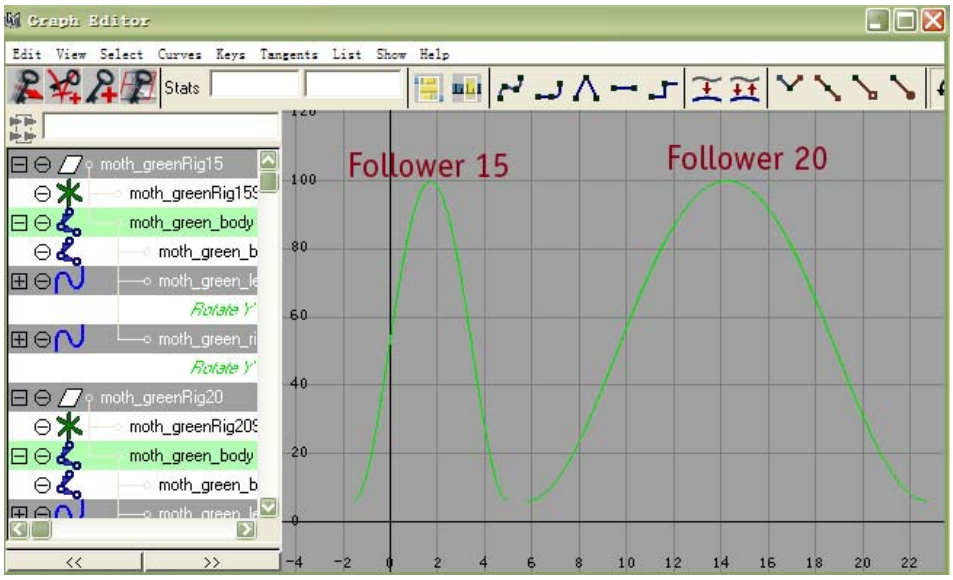


Figure 25: follower animation cycle

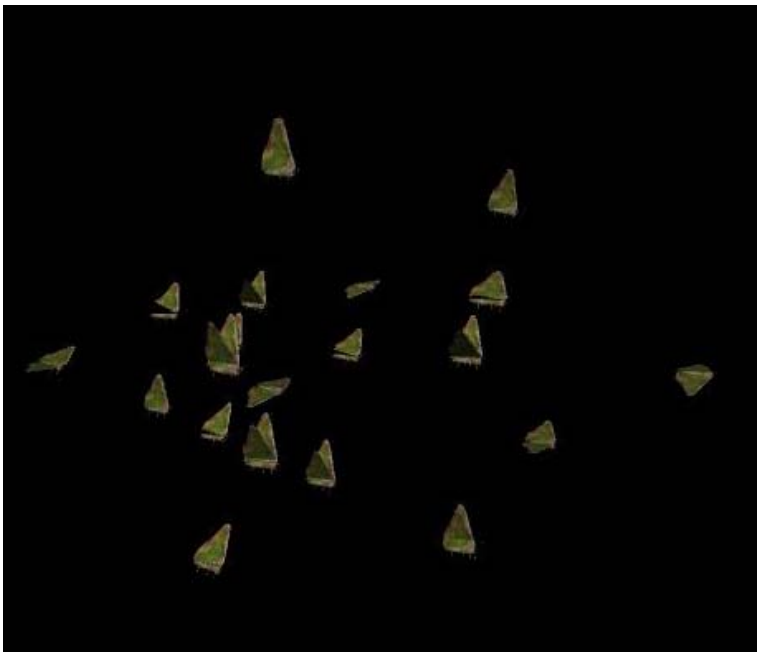


Figure 26: different scale animation cycle

After the follower animation cycle is scaled, its speed of following the leader should be changed too. For example, a fly cycle of the moth is longer than another moth, its speed should be slower than another moth. How much different of them depend the value of *Frequency Parameter* option. The

relation of them in this system is:

Speed = 1/ (animation cycle * Frequency Parameter)

Max velocity option is control the maximum velocity of the follower when it moves follow the leader. If you set this value always smaller than the leader speed, the leader will be always in front of all followers. This value is connecting with the *Damping* attribute of the follower rigid body, if the speed of the follower larger than the maximum velocity, the value of the damp will be increased, and then the speed of the follower will be slow down, otherwise the value of the damp will be set to 0.

See example FSExample_2.avi and FSExample_2.mb for followers speed slower than the leader. See example FSExample_3.avi and FSExample_3.mb for followers speed faster than the leader.

5.2.4 Obstacle

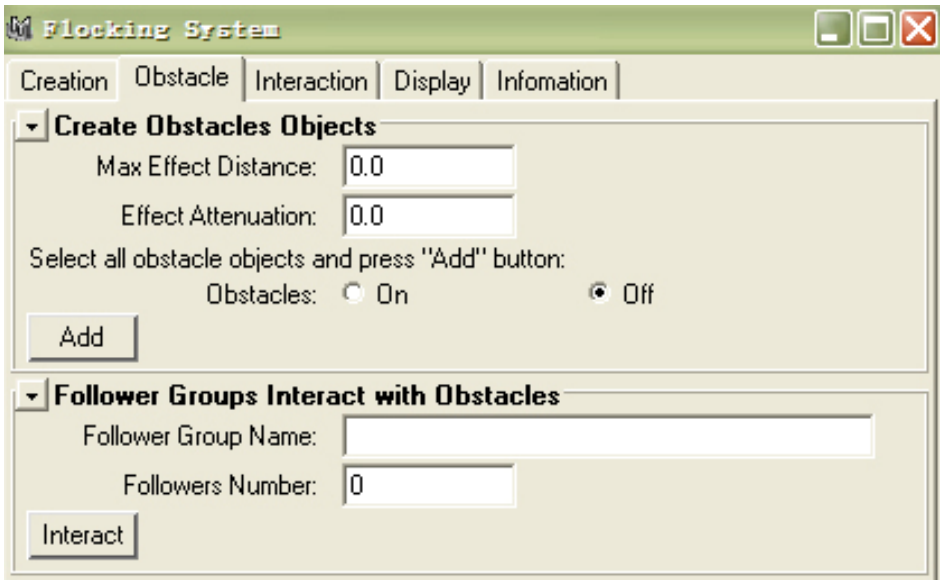


Figure 27: flocking system interface 6

This system allows you create all obstacles at one times using the same setting, but the objects are used must be polygon or NURB objects. After you create obstacles, you can find they are changed as rigid bodies and each is connected with its own radial field. The radial field does not affect the followers until you use the *Follower Groups Interact with Obstacle* options. Basically, the options here just connect the radial fields of the obstacles with each follower. Before you use these options, because the obstacles are created using the same rigid solver with the followers, they react as rigid body collision. After you use this option, the radial field is connected with each follower, when followers get close to the obstacle, they will be pushed away. How close the follower can get depend the value you enter in the *Max Effect Distance* option.

See example FSExample_4.avi and FSExample_4.mb for without using *Follower Groups Interact with Obstacle* options.

See example FSExample_5.avi and FSExample_5.mb for using *Follower Groups Interact with Obstacle* options.

5.2.5 Follower group interactive

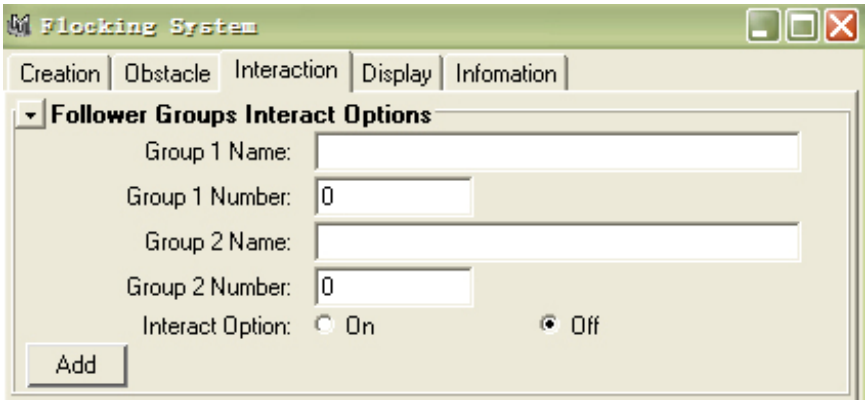


Figure 28: flocking system interface 7

Our system allows user create as many flocking groups as they want. You can create different follower group with the same leader, or you can create them with different leader. If you create different group of follower using the same rigid solver, they react each other as rigid body collision, the collision field of the follower has no influence on another group followers. In most of case, they work well without this influence, but if you like to create more interesting effect, you may want this influence. The options here are created for this reason.

They can connect the collision field between two different group followers.

See examples FSExample_8.avi and FSExample_8.mb for without using *Follower Group Interactive* options, FSExample_9.avi and FSExample_9.mb for using these options.

See example FSExample_10.avi and FSExample_10.mb for different follower group using the same leader.

5.2.6 Display

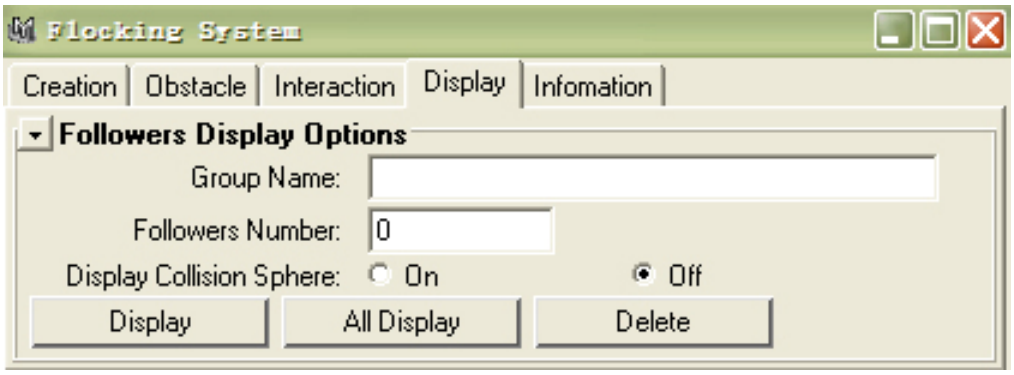


Figure 29: flocking system interface 8

The options here are created for convenient the user. After you create the flocking group, you can see the polygon spheres of the followers display as wire frame in the scene, they can not be rendered. But maybe you want to remove them from the scene. If you do, simply just open the Display Collision

Sphere option, then press All Display button. Also you can specify which group of follower spheres you want to non display using these options and *Display* button. How does it work behind the interface? Our system is sensitive to name, all the polygon sphere used as rigid body in this system are named with “CS”. When you press the button, the system find out all spheres named with “CS”, and then turn off their *visibility* attribute.

Each follower contains a rigid body, you can not directly delete it after you created them, if you do you will get an error message on it. You can delete the whole group of follower by using *Delete* options here. The system will break all the connection of the followers at first, and then delete them, disconnection all connections of the follower can prevent errors and warning.

5.3 Some theories of flocking system

5.3.1 Using rigid body

For simulating collision avoidance behavior, we need to create collision between each object and member in the flocking group. We can achieve this by using rigid body in Maya. Maya provides two kinds of rigid bodies, active and passive. An active rigid body reacts to dynamics, a passive rigid body can react with active rigid body by calculating collision, but dynamics have no effect to passive rigid body. Another important thing is we can not key frame the active rigid body in Maya, but we can key frame the passive rigid body. Our system creates all the followers as active rigid bodies, and all the leaders and obstacles as passive rigid bodies, this allows the user key frame the leader and move around the obstacle.

5.3.2 Using polygon sphere

May be you already notice, each follower and leader object created in our system are contained in a polygon sphere. There are four reasons for using polygon spheres. First, like we mention before, all followers and leaders work as rigid bodies in our system, but Maya only can create rigid body with polygon object or NURBS object. For making our system more flexible and useful, we

create a polygon sphere contain the original follower and leader object, and use this polygon sphere as rigid body without change the original object, that means user can use any kind of object as follower and leader, they will work exactly as using polygon object.

Second, using polygon sphere can reduce the calculation. After you create the flocking group and play the animation, the collision has to be calculated between each object in every frame, which is a huge computation. Another, Maya calculates collision between object and object by calculating each face collision. If user uses a complex model, the collision calculation will be increased. Using polygon sphere instead the original object can prevent this problem. Whatever the object user uses, the collision calculation will be the same.

Third, our system creates a repelling field for each follower and leader. In Maya, we choose using radial field as repelling field. The polygon sphere is connected with this repelling field, it works as a bounding sphere here. Like we discussed in chapter three, using sphere is the best way for divert the follower moving path and simulate the natural movement.

Forth, our system allows the user define the radius of the polygon sphere. That

mean user can define the space between follower and follower, follower and leader. It help user make more interesting effect in the animation.

5.3.3 Using repelling Field

All the objects created by our system are connected with repelling fields, include obstacles. The repelling field of the follower and leader can push another follower away gently, it can smooth the follower movement. Using the repelling field on the obstacle can simulate the view field of the follower, when the follower get close to the obstacle, it will be push away before they hit on the surface of the obstacle, this look like the follower can “see” the obstacle and move away from it.

Using repelling field not only for simulate more natural effect, it also can reduce error in Maya. There is only one side of a rigid body’s surface can collide with another rigid body’s surface in Maya, when two or more rigid bodies pass through each other, a warning message will be printed out. This situation happens a lot when the active rigid body has high velocity in the animation. Using repelling field can repel another rigid body away before they hit together, prevent they pass through each other.

5.3.4 Local control and Global control

Our system allows the user keys frame and controls the leader. But the followers are controlled by its local control and global control. Global control is the influence of the leader. It control the followers move direction and situation. It is achieved by the *Leader Field* and the *Following* attribute of the leader. Local control controls the follower rotation, velocity and random movement. Local control is achieved by the follower expression.

CHAPTER 6

Further work and conclusion

Our system succeeds at create flocking groups and simulate their behaviors easily and conveniently in Maya without using another software. It is also flexible and can be used on any kind of object.

In the further, we like to improve and add more functions in our system. For example, our system create polygon sphere as the collision object for the followers automatically. If the object of the follower has long shape, using sphere make too much space between the followers.

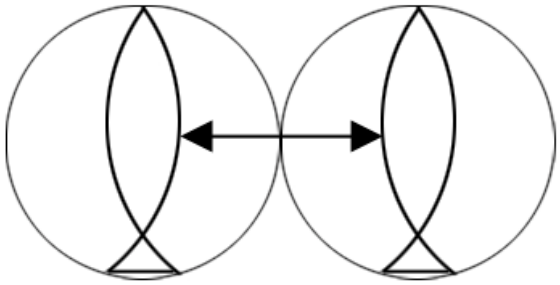


Figure 30: space between two long shape followers

In the further work, we like to provide more chooses in the system, which allows the user decide the shape and the size of the collision object. Another, our system requires the follower has skeleton in this version. We are going to make it as options that allow the user choose. Furthermore, we are going to do more tests for our system and make it more powerful and flexible.

Reference:

1. Rick Parent, *Computer Animation Algorithms and Techniques*. 2002 by Elsevier Science (USA). ISBN: 1-55860-579-7.
2. Willian Reeves, *Particle Systems ----- Technique for Modeling a Class of Fuzzy Objects,* 1983. ACM Special Interest Group on Computer Graphics and Interactive Techniques, 2005.
3. Craig Reynolds, 1985, *Boids* [online]. Poole, Bournemouth University. Available from: http://www.red3d.com/cwr/boids/, [Accessed, 10 August 2005].
4. Craig Reynolds, *Physically Based Modeling Course Notes*, SIGRAPH 88, August 1988, Atlanta, Georgia.
5. Mark R. Wilkins and Chris Kazmier, *Mel Scripting for Maya Animators*, 2003 by Elsevier Science (USA), ISBN: 1-55860-841-9.
6. AI.implant [online]. Poole Bournemouth University. Available from: http://www.biographictech.com/, [Accessed, 5 August 2005].
7. Massive [online], Poole Bournemouth University. Available from: http://www.massivesoftware.com/, [Accessed, 5 August 2005].
8. Jonathan Macey, *Msc Programming for Graphic Course Note*, 2005 [online].http://dec.bournemouth.ac.uk/staff/jmacey/proggraph/proggraph.html, [Access 2005].

Bibliography:

1. Craig W. Reynolds. *Flocks, Herds and Schools: A Distributed Behavioral Model*. 1987.
2. Matt Anderson, Eric McDaniel and Stephen Chenney. *Constrained Animation of Flocks*. 2003.
3. Manfred Lau and James J. Kuffner. *Behavior Planning for Character Animation*. 2005.
4. Alfredo Pina, Francisco Seron and Diego Gutierrez. The ALVW system: an interface for smart behavior-based 3D Computer Animation.
5. Dick Tsur, Hitachi Corp and Jeffrey D. Ullman. *Query Flocks: A Generalization of Association-Rule Mining*. 1998.
6. Martin S Olivier. *Flocks: Distributed Proxies for Browsing Privacy*. 2004.
7. Ali Raza Butt, Rongmei Zhang and Y. Charlie Hu. *A Self-Organizing Flock of Condors*. 2003.

Part 2

Breaking Tool

ZHUO YAO LU
Media School, Bournemouth University, UK
September 2005

CHAPTER 1

Introduction

Breaking happens a lot in our true life. But this everyday effect is difficult simulated in 3D. Our tool is created using Maya Mel scripting language. It can simulate simple breaking effect in 3D easily and realistic.

CHAPTER 2

How does breaking tool works

After you source the script, type “breakMain;” in the script editor. The breaking tool window is opened. Showed as below.

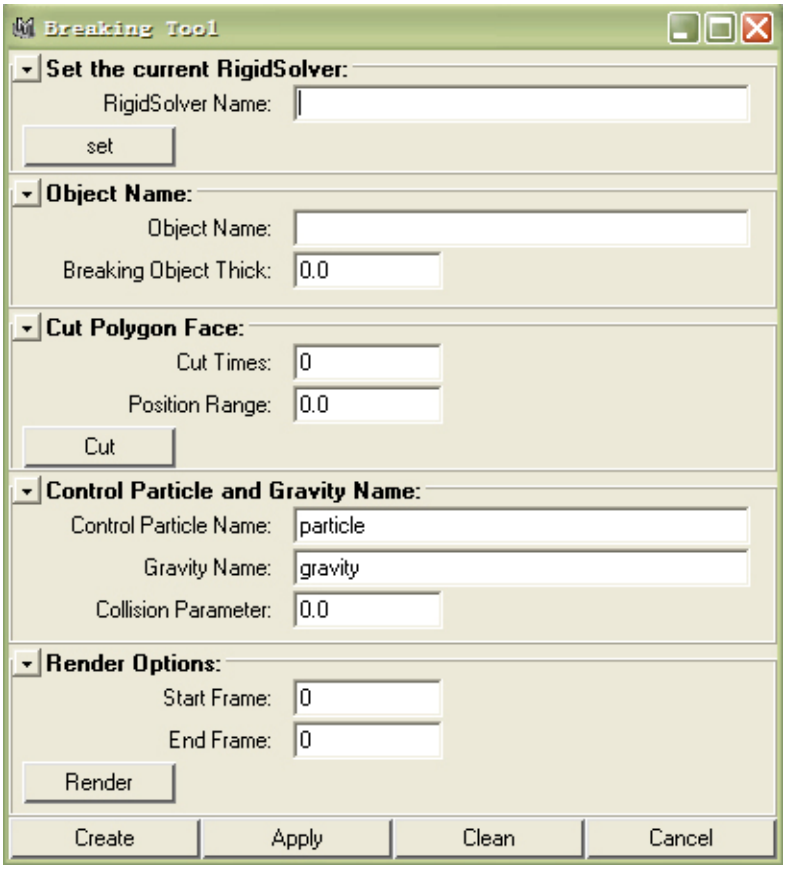


Figure 1: breaking tool interface 1

Before generate the breaking tool, there should be a particle, a gravity field and the polygon object prepared for breaking in the scene. And they must not be connected.

The breaking tool uses a particle as an engine to generate the breaking action, and uses rigid bodies to simulate the breaking effect. This tool only can use with polygon object. The shape of the breaking pieces depend the faces of the

polygon object. At first, the system will separate all the faces of the polygon object into individual piece object. Second, extrude the face of separated pieces. Third, change the breaking piece to an active rigid body, and connect it with the particle collision. Forth, create a radial field for each separated piece and replace it to each piece central position, but do not connect them. Fifth, create an expression for each separated piece. It controls the collision attribute of the active rigid body, and also controls the connection of the radial field and the gravity field with the breaking piece rigid body.

How the expression works? After you generate the breaking tool, the original polygon object has been chip off and separated into individual pieces. Each breaking piece is an active rigid body. Before the particle hit on the breaking pieces, those breaking pieces stick and cross each other. When two or more rigid bodies cross each other in Maya, a warning message will be printed out. The animation will be extremely slowed down. For prevent this problem, we need to turn off the collision attribute of the active rigid body. But after we play the animation, when the breaking pieces are pushed away their original place, they no longer stick and cross each other, the collision between each breaking pieces must be calculated, the collision attribute need to be turned on. The time of turning on the collision attribute is according the value in the *Collision Parameter* option. This value is connected with the condition in the expression

of the breaking pieces, when the piece fly away its original position more than this value, the collision attribute is on.

The same reason for control the connection of the rigid body with the radial field. The radial field of each breaking piece is used to repel another piece away. We mention before, when two or more rigid bodies cross each other, the warning message will be print out. This situation happens a lot in the animation, especial the rigid body has high velocity. Using radial field can reduce this problem. The radial field will repel the other piece away before they hit and cross each other. But the radial field should not work at the beginning of the animation, when the breaking pieces still stick together. In the expression, the connection between the radial field and the breaking piece rigid body is controlled by the same condition as the collision attribute.

For the connection of the gravity field, the expression uses another condition – calculating the velocity of the breaking piece rigid body. At the beginning of the animation, when the breaking pieces still stick together, their velocity equal to zero. There is no connection between the gravity field and the breaking piece rigid body. When the animation is playing, the particle hit on the breaking piece, the breaking piece fly away and its velocity does not equal to zero anymore. This time, the connection is created.

Because the animation have to replay correctly, the connection between fields and the rigid bodies should be broken in the beginning of the animation, and also the collision attribute of the rigid body should be turned off again. Those actions are controlled by the expression too.

Now let we look closer to each option and see how they work behind the interface. For more detail about how to use the interface, please check the *Breaking Tool User Guide*.

CHAPTER 3

How interface works

3.1 Rigid solver

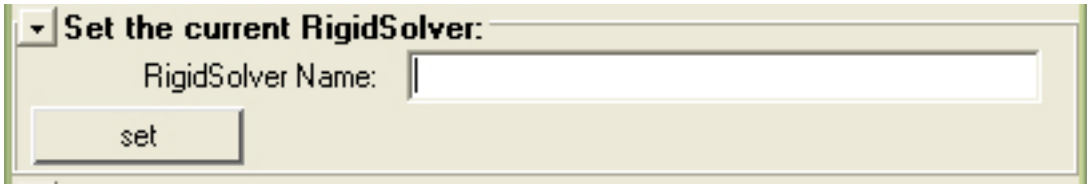


Figure 2: breaking tool interface 2

Rigid solver controls the accuracy of the rigid body solution; the rigid bodies connected with it, which can react each other. The option here can specify a rigid solver, if it already exists in the scene, it will be set as the current rigid solver. Otherwise, the system creates a new rigid solver with the name user entered. This can help user combine it with another work.

3.2 Object

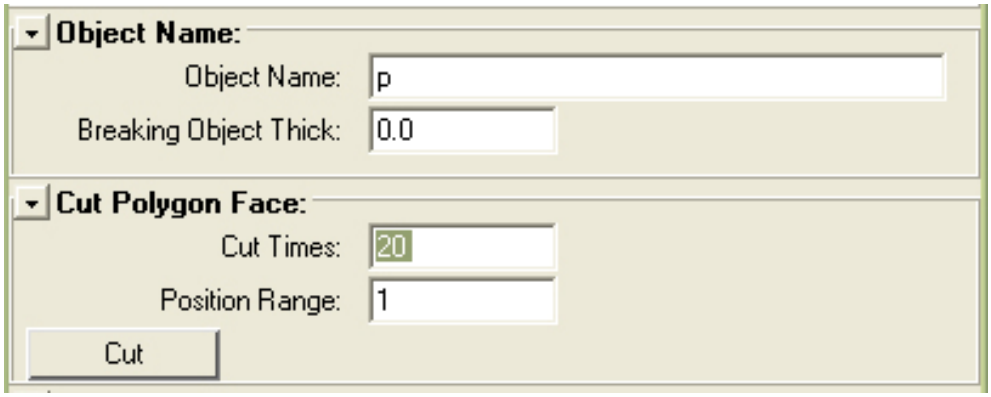


Figure 3: breaking tool interface 3

The value of the *Breaking Object Thick* option defines the thick of the breaking pieces. It is connected with the extrude value in the script.

Because the shape of the breaking pieces depend the shape of original polygon object faces. For simulated more natural and interesting effect, we can use *Cut Polygon Face* options to random cut the polygon object. It can create different shape of the face. Showed as below.

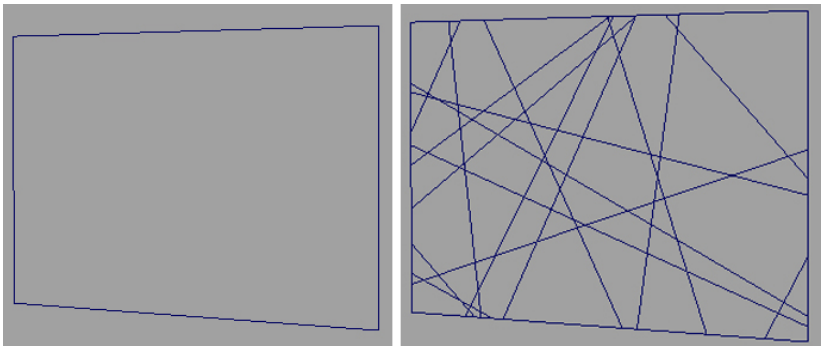


Figure 4: use Cut Polygon Face option

3.3 Particle and gravity

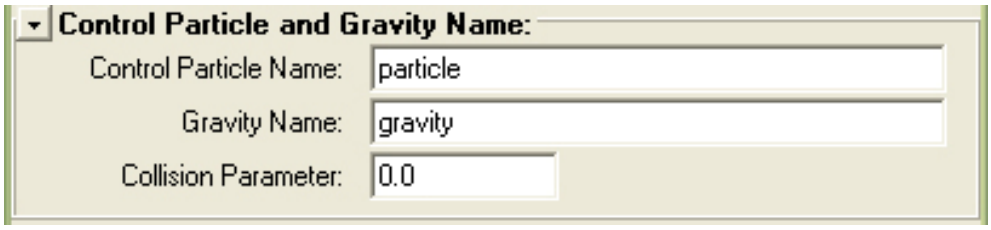


Figure 5: breaking tool interface 4

The options here specify the particle and the gravity field for the breaking. The *Collision Parameter* value is connected with the condition in the expression. Because the breaking tool use a particle as an engine to generate the breaking action. After generate the breaking tool, user can edit the particle and gravity

attributes to get different breaking effect. For example, change the particle emitter speed can change the speed of breaking pieces.

See examples in the *BreakingToolExample* file.

3.4 Render

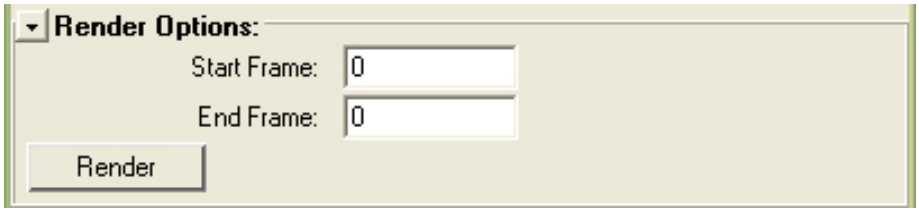


Figure 6: breaking tool interface 5

For some reasons, the breaking animation sometimes can not get the correct result by using batch render in Maya 6.0. This problem seems does not exist in Maya 6.5. In this case, we create render options here. User can set up the render setting in the Render Global Setting window, then uses these options to render.

CHAPTER 4

Further work and conclusion

The breaking tool succeed at simulate simply breaking effect in Maya. But it still has some problem waiting to be improved. For example, it is difficult to use with a complex object, which has lots of face, the breaking animation will be extremely slow. The radial field of each breaking piece is created by the scripting automatically, the way of setting the maximum distance for radial field is not enough precise, it need to be improved. We are going to do more test and solve out those problem in the further work.

Bibliography:

1. Rick Parent, *Computer Animation Algorithms and Techniques*. 2002 by Elsevier Science (USA). ISBN: 1-55860-579-7.
2. Mark R. Wilkins and Chris Kazmier, *Mel Scripting for Maya Animators*, 2003 by Elsevier Science (USA), ISBN: 1-55860-841-9.
3. Jonathan Macey, *Msc Programming for Graphic Course Note*, 2005 [online].http://dec.bournemouth.ac.uk/staff/jmacey/proggraph/proggraph.html, [Access 2005].

Part 3

Animation Report

1. introduction

This animation used the flocking system and breaking tool techniques to simulate a large number of moths flying in 3D.

2. Background of the animation

Figure 1: animation image ----- tower

We chose a Chinese traditional tower as the only building in this animation. The style of the tower is from hundreds years ago the south of China. The reference picture showed as below.

Figure 2: reference picture of the tower

We chose the night time for the animation. The sky and fog are created using fluid dynamic. There are about 400 moths in the third shot and the fifth shot. They are flying tracing the magic ball and avoiding the tower building in the animation. We used three different moth models in the animation.

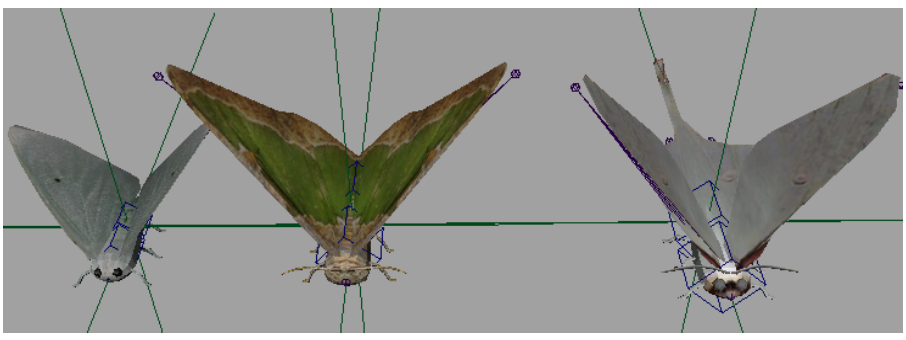


Figure 3: moth models

3. Techniques

We used flocking system technique to create the moths and their behaviors. In the flocking system, all the moths are created as followers, the tower building is an obstacle object, and the magic ball is the only leader in this animation. After we created all the moths, we animated the leader. The moths are flying tracing it, and at the same time they have to fly avoid the tower and the magic ball. In the first shot, you can see clearly this effect, the moths are tracing the magic ball, but can not get too close to it. A bit trick to animate the leader here, after you created lager number of moths in the scene and try to animate the leader of them, the computer run extremely slow. For solve this problem, we animated a locator instead the leader at first, and then paste all its key frames to the leader.

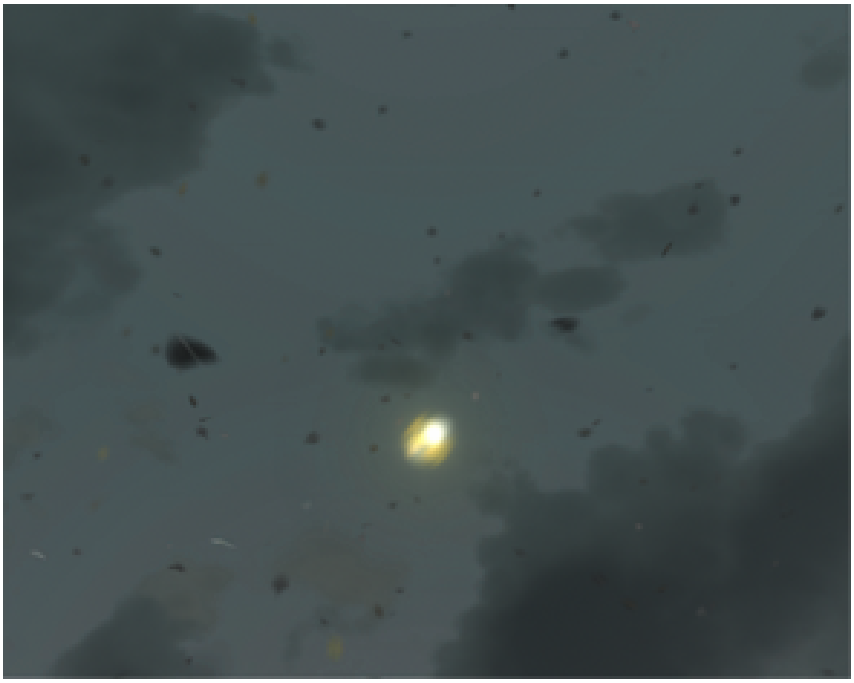


Figure 4: the first shot

In the third shot of the animation, we used the breaking tool and the flocking system at the same time. The breaking tool generated the breaking class. The moths are flying avoid the breaking class in the shot.

Figure 5: the third shot image

Because the tower builder is a complex model, for reduce the collision computation. We created another simple object as obstacle instead the tower. For example, we used cylinder instead the tower in the second, forth and fifth shots, and we used polygon box instead the tower in the third shot.

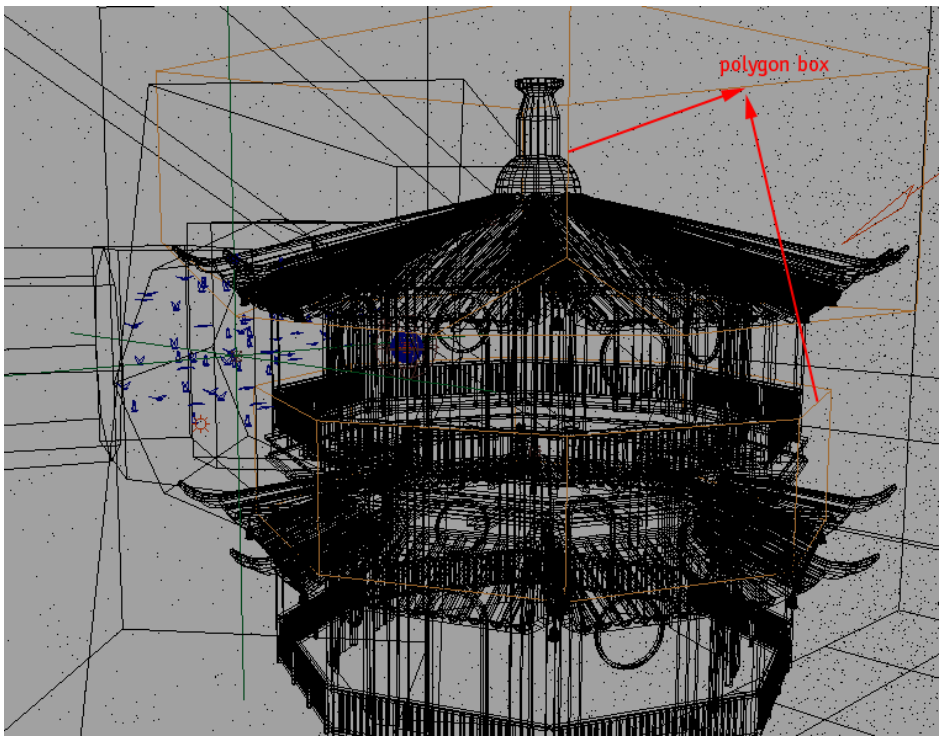


Figure 6: polygon box instead tower for calculating collision

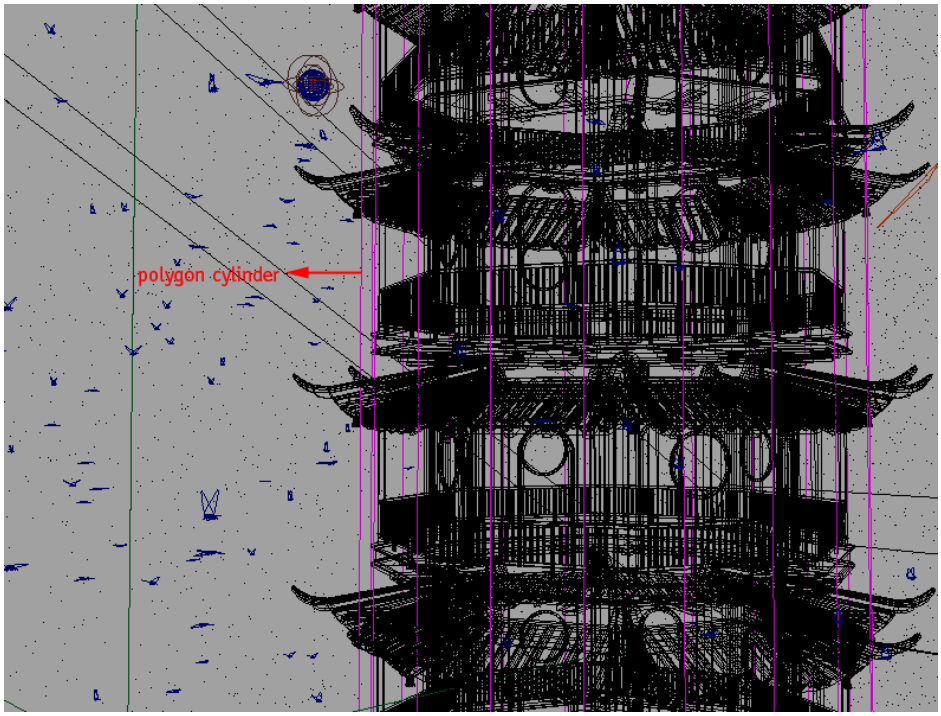


Figure 7: polygon cylinder instead tower for calculating collision

4. Conclusion

The flocking system and breaking tool worked very well in this animation. But controlling large number of objects still makes the computer run very slowly. If you use flocking system to create large number of objects, you are not able to preview the animation in the interface. Also you need to take a long time to do the render.

Part 4

The Flocking System

User Guide

Source the script:

1. First, store the script in Maya/scripts directory or another directory you like.

2. Run Maya program, open the script editor, type:

   source "maya/scripts/flockSystem"; (or another full path of the script)

3. After you source the mel script, type:

   flockMain;

   Run the script.

4. For you convenience, you can type:

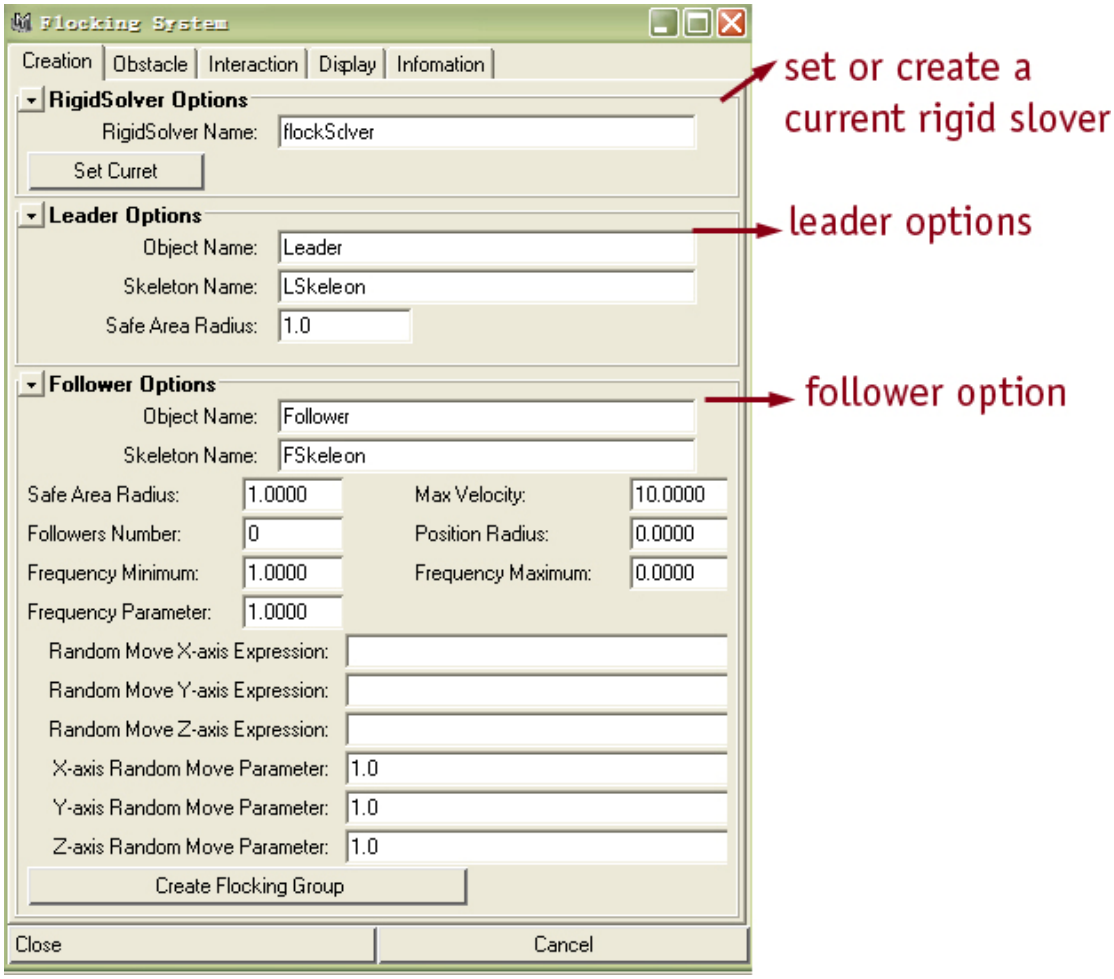
   source "maya/scripts/flockSystem";
   flockMain;

   in the script editor, then select all of them, go to the script editor menu, *File→Save Selected to Shelf*, enter a name in the pop up window. Now, you created an icon on the shelf, when you want to run the script ,press the icon and without type anything in the script editor.

The flocking system interface options:

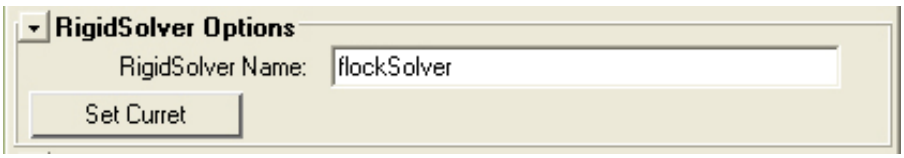
1. After you run the script, the flocking system window is opened. It contains five different tabs, *Creation, Obstacle, Interaction, Display* and *Information*.

2. Creation tab.



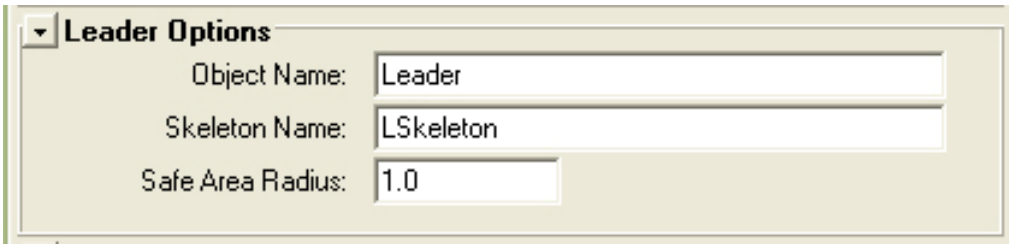
There are three sections in this tab.

--RigidSolver Options:



Enter a rigid solver name, press *Set Current* button. If it already exists, set it as a current rigid solver; otherwise create a new rigid solver with this name.

--Leader Options:



Object Name ---- leader model object name

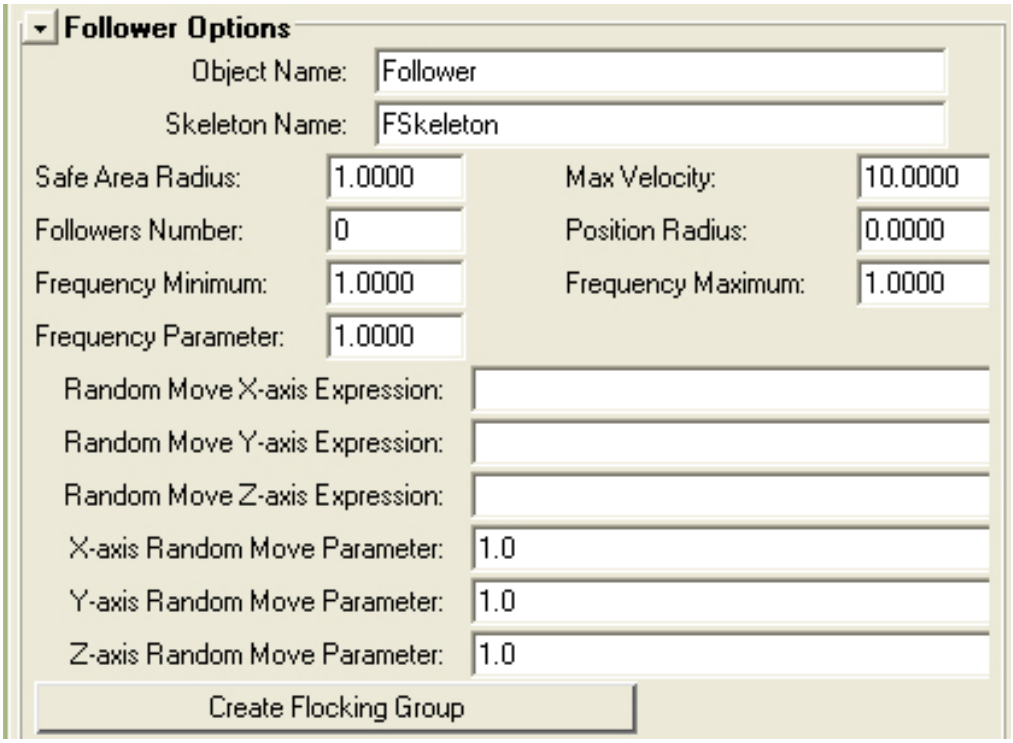
Skeleton Name ---- if the leader has skeleton, enter a name, if it does not, leave this option as space.

Safe Area Radius ---- the safe area of the leader.

The leader can be any object you like, an animated object (for example a fish, a butterfly) or simply just a locator.

See example FSExample_7.avi and FSExample_7.mb for using an animated object as leader.

--Follower Options:



Follower can be any animated object too, but it must have skeleton in this version.

Object Name --- the follower model object name

Skeleton Name ---- follower skeleton name, follower is required a skeleton.

Safe Area Radius --- the safe area of the follower

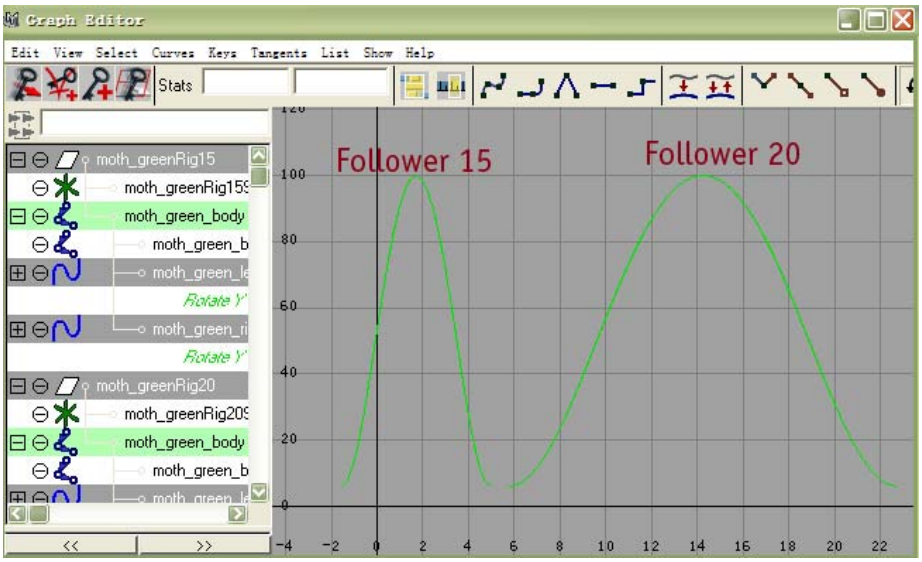
Max Velocity --- the maximum velocity of the follower, when they are following the leader.

Followers Number --- a number of followers

Position Radius --- all followers place around the leader in this radius range area.

Frequency Minimum --- a minimum animation cycle frequency scale

Frequency Maximum --- a maximum animation cycle frequency scale, all followers animation cycle frequency will be scaled between minimum and maximum values. Showed as below.



Frequency Parameter --- a parameter value for controlling the relationship between the frequency of follower and its following speed, when they move follow the leader.

See example FSExample_12.avi and FSExample_12.mb for set *Frequency Minimum, Frequency Maximum* and *Frequency Parameter* options 1.

See example FSExample_13.avi and FSExample_13.mb for set :

Frequency Minimum = 0.12, Frequency Maximum = 2.8, Frequency Parameter = 4.

Random Move X-axis Expression, Random Move Y-axis Expression and *Random Move Z-axis Expression* these three options are used to define

followers' movement when they do not follow the leader. If you leave these options as space, when you turn on the Following attribute of the leader, the followers will stop move.

X-axis Random Move Parameter, Y-axis Random Move Parameter and *Z-axis Random Move Parameter* options are used to create a different numbers using with last three expression option. If you set all these options as 0, the follower will stop move when you turn on the *Following* attribute.

See example FSExample_1.avi and FSExample_1.mb, the following attribute off on the frame 25 and did not use the *Random Move X-axis Expression, Random Move Y-axis Expression* and *Random Move Z-axis Expression* options.

See example FSExample_14.avi and FSExample_14.mb for set *X-axis Random Move Parameter, Y-axis Random Move Parameter* and *Z-axis Random Move Parameter* options as 0.

After you finish all the options in this tab, press *Create Flocking Group* button. Now you should see all followers are placed around the leader, wherever the leader is in the scene. Open the outliner, you can find the leader has been place as a child in a locator, which is named *LeaderNameLocator*, you can animate or paste key frames to this locator. Also when you select this locator, you can see an attribute called *Following*

in the channel box. When this attribute is on, all the followers move follow the leader, when it is off, followers move according its expression. User can key frame this attribute.

See example FSExample_6.avi and FSExample_6.bm for key frame the *Following* attribute of the leader.

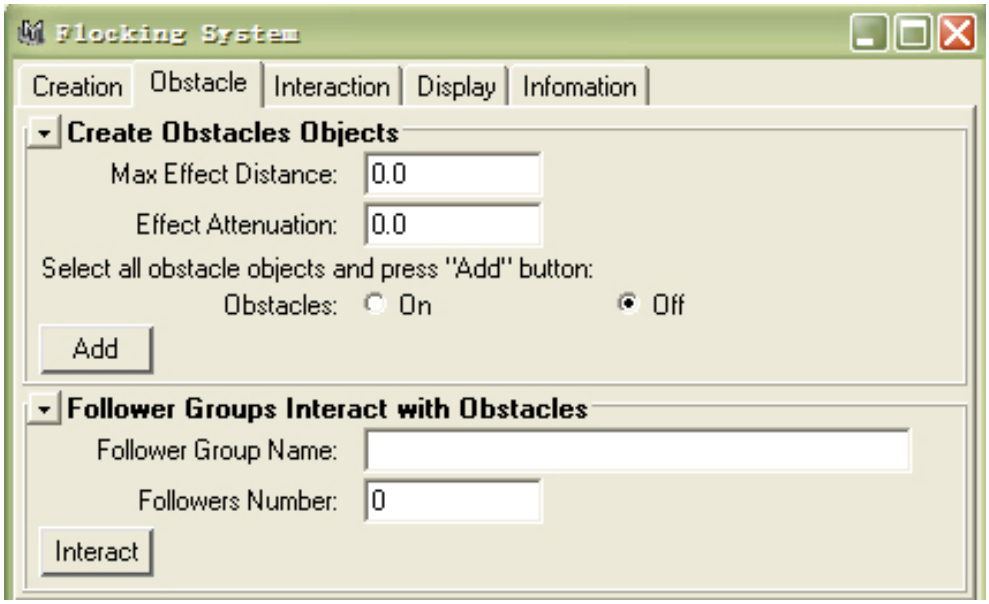
You can change all the setting and press *Create Flocking Group* button to create another group followers and leader. Also this tool allows you create different follower groups using the same leader. If you like to do this, simply just change the setting in the *Follower Options* section, and leave the *Leader Options* the same, then press *Create Flocking Group* button again.

See example FSExample_10.avi and FSExample_10.mb for using the same leader.See example FSExample_9.avi and FSExmple_9.mb for using different leader.

TIP:

A little tip of key frame the *LeaderNameLocator*. After you create the flocking group, if you key frame the *LeaderNameLocator,* the computer will be slowed down, especial when you have lots followers. What you can do is you can add small number of followers at first, then key frame the *LeaderNameLocator.* After you finish key frame it, you can add as many followers as you like to the leader. Another way to do this is you can create a locator and key frame it, and then paste all its keys to the *LeaderNameLocator*.

3. Obstacle Tab:

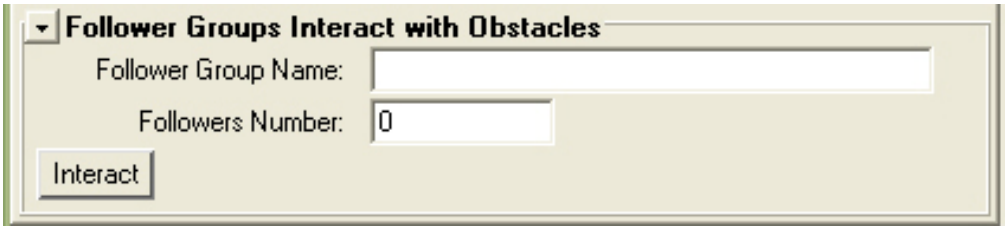


The options in this tab are used to create obstacle objects.

Max Effect Distance --- a maximum effect distance, it calculate from the centerof the obstacle object.

Effect Attenuation --- an attenuation value of obstacle object effect

After you fit these two options and turn on the *Obstacle* option, select all obstacle objects in the scene, then press *Add* button. All object you selected are changed as rigid bodies. Play the animation, you can see followers move avoid the obstacle objects. If you want to create more natural effect, go to the *Follower Groups Interactive with Obstacles* section.



Follower Group Name --- enter the follower name, which is the same as you entered in the *Object Name* in the *Follower* Options.

Followers Number --- enter how many followers in this group.

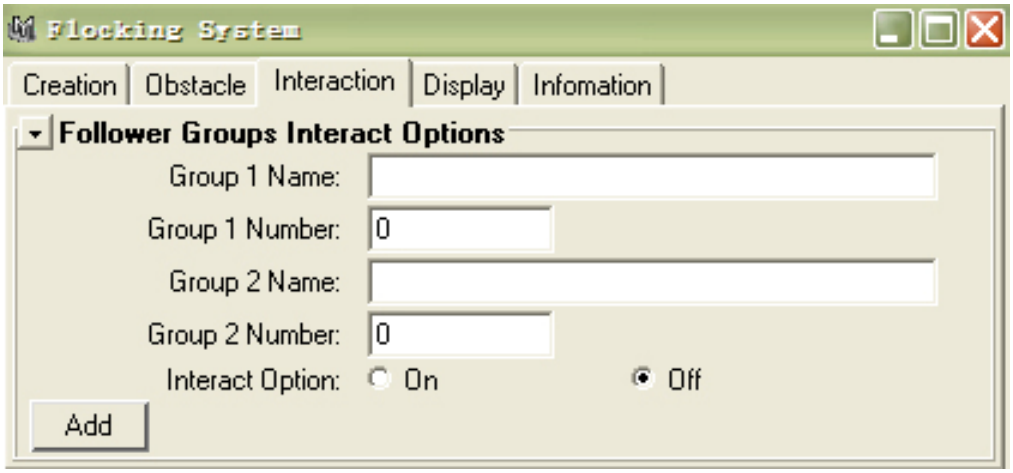
After you fit in these two options, select all the obstacle objects and press *Interact* button, now play the animation again, you can see the difference.

See example FSExample_4.avi and FSExample_4.mb for without using *Follower Groups Interact with Obstacle* options. See example FSExample_5.avi and FSExample_5.mb for using *Follower Groups Interact with Obstacle* options.

TIP:

The flocking system is sensitive to name and your selection, when you create obstacle objects, if you get error or warning message on it, you can try press “z”, undo what you just did, then reselect the objects, make sure the objects you selected are only polygon or NURB objects, then try the options again. Also you can try to rename the objects and try recreate again.

4. Interaction Tab:

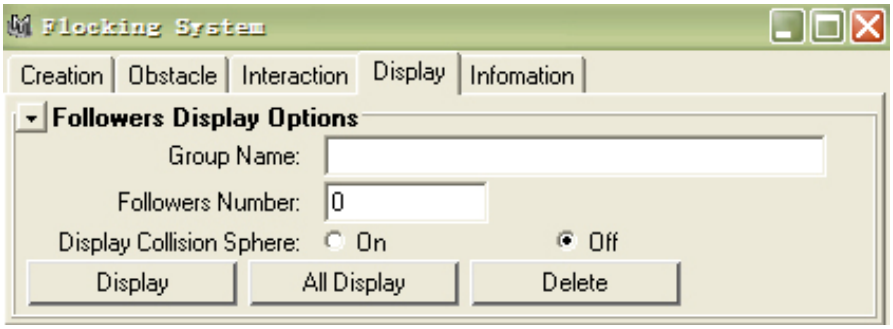


You can use this system to create as many follower groups as you like. If they are created using the same rigid solver, they interactive each other as rigid body collision. If you like to make them interact more natural, the options in this tab may help you. Enter follower group name and number in the relative option, turn on the *Interact Option*, then press *Add* button,

create a connection with these two groups.

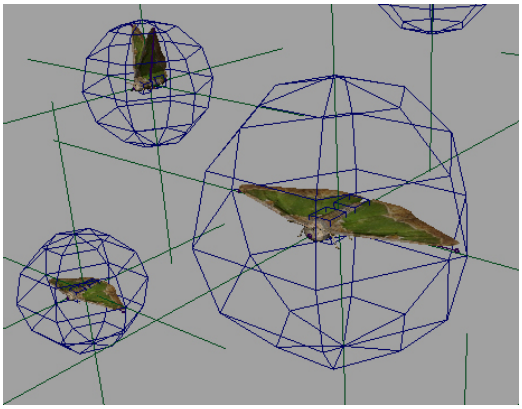
> See examples FSExample_8.avi and FSExample_8.mb for without using *Follower Group Interactive* options, FSExample_9.avi and FSExample_9.mb for using these options.

5. Display Tab:



Group Name --- the follower group name

Followers Number --- followers number in this group



When you create followers, you can see each follower is contained in a sphere. Those sphere can not be render, they display as wire frame in the scene. If you do not want they display in the scene. Turn off the *Display Collision Sphere* option, then press *All Display* button. If you want they

display again, turn the option on and press button again. Also, you can specify which spheres of follower group display or not display, simply enter the follower group name and the followers' number in the option boxes, and then press *Display* button, if you press *Delete* button, the follower group you specified will be deleted.

6. Information Tab

   It contains some information about the script and author.

Part 5

The Breaking Tool User Guide

Source the script:

5. First, store the script in Maya/scripts directory or another directory you like.

6. Run Maya program, open the script editor, type:

   source "maya/scripts/breaking"; (or another full path of the script)

3. After you source the mel script, type:

   breakMain;

   Run the script.

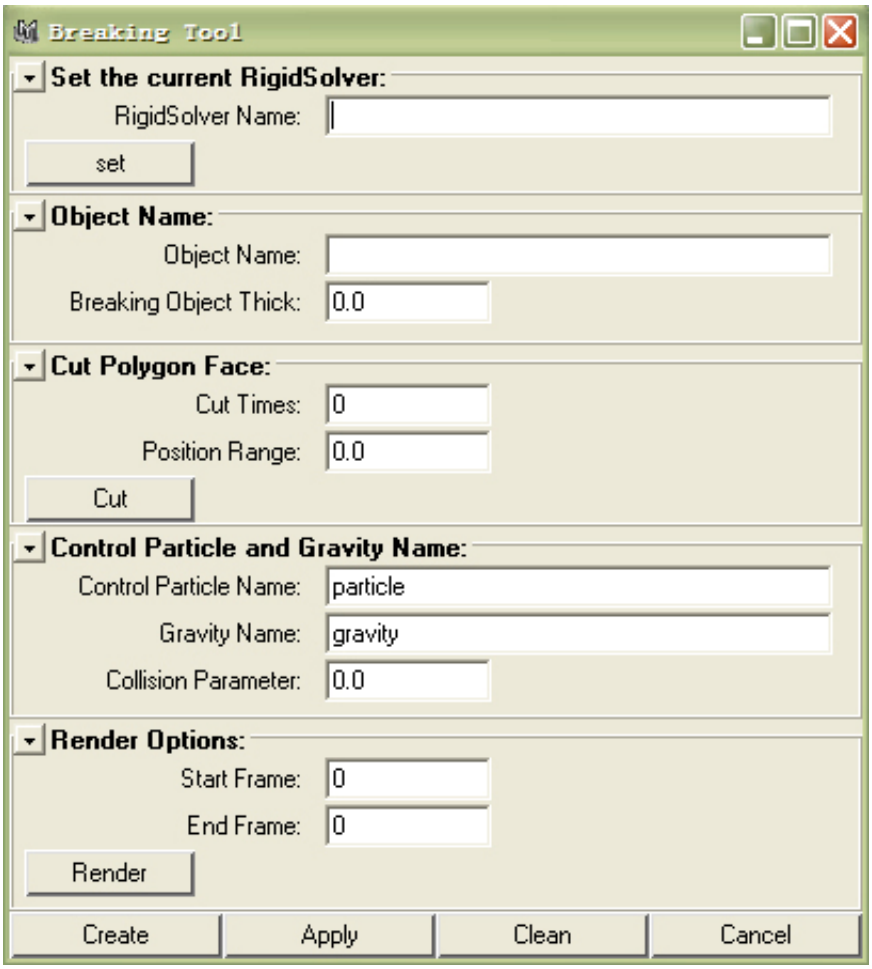
4. For you convenience, you can type:

   source "maya/scripts/breaking";
   breakMain;

   in the script editor, then select all of them, go to the script editor menu, *File→Save Selected to Shelf*, enter a name in the pop up window. Now, you created an icon on the shelf, when you want to run the script ,press the icon and without type anything in the script editor.

The breaking tool interface options:

1. After you run the script, the breaking tool window is opened. Showed as below.

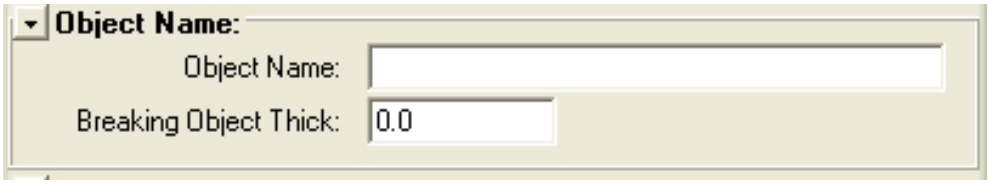


2. Set rigid solver

Specify a rigid solver. Enter the name of the rigid solver, if it already exists, set

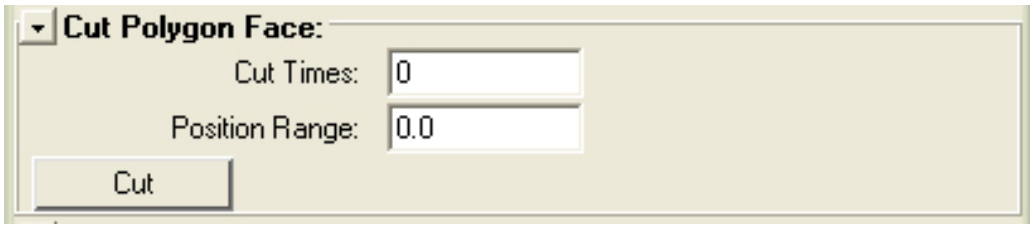
it is the current rigid solver, if it does not exist, create a new rigid solver with this name.

3. Object



Enter the name of the object you want to break. And specify how thick of the breaking pieces.

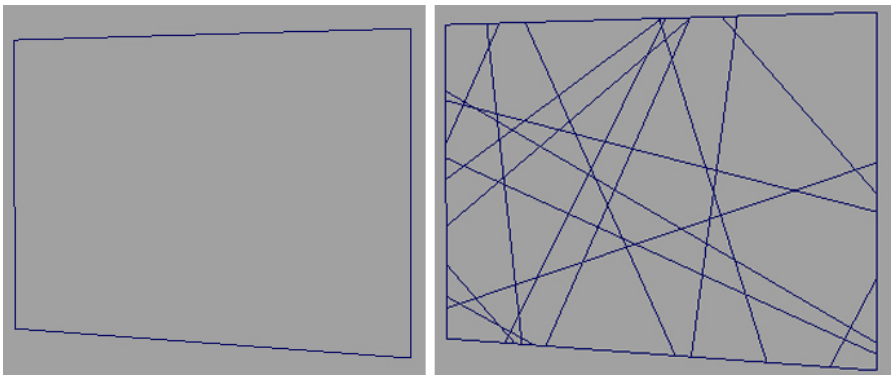
4. Cut polygon face



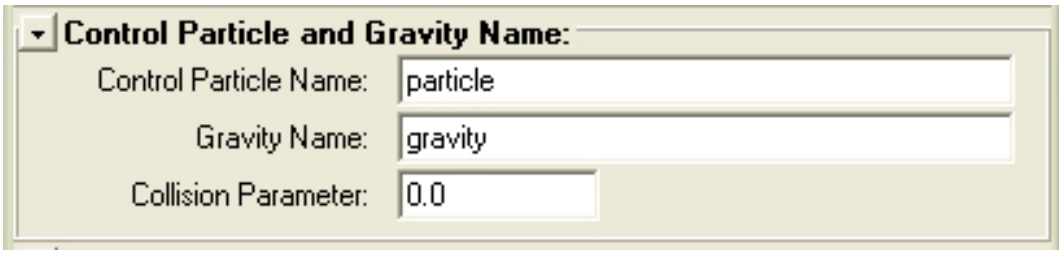
Cut Times ----- specify how many time of cutting you like to do.

Position Range ----- specify the cut plane center position area.

For simulate more interesting effect, user can use this function to cut the polygon object automatically.



5. Particle and gravity



Control Particle Name ---- specify a particle using in this tool

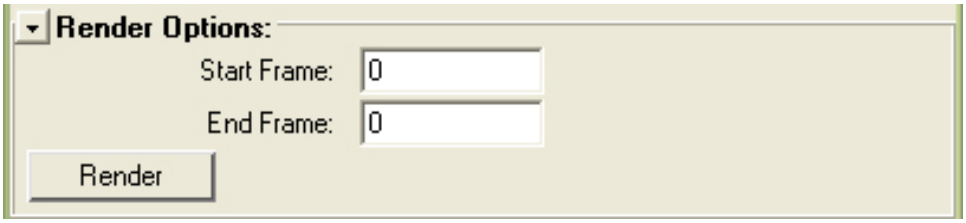
Gravity Name ----- specify gravity flied

Collision Parameter ---- a parameter for specify the collision time

Before you generate the breaking tool, there should be a particle, a gravity field and a polygon object in the scene. And they must not be connected.

The particle is used as an engine to generate the breaking action. Gravity field is used to simulate the breaking pieces falling on the floor.

6. Render



Sometimes, you can not get the correct simulation result by using the batch render. In this case, you can use options here.

7. Edit the particle and gravity attribute.

Because the breaking tool uses a particle as an engine to generate the breaking action. After we generate the breaking tool, we can edit the particle attributes and the gravity attributes to get different breaking effect.

See examples in the *BreakingToolExamples file*.

CD Contents

BreakingToolExamples ------- all examples created by breaking tool (avi & mb).

FlockingSystemExamples --- all examples created by flocking system

(avi&mb).

ForWindows ------- breaking tool version for Windows system.

Flocking system version for Window system.

ForLinux -------- breaking tool version for Linux system

flocking system version for Linux system

AnimationMayaFiles ------ all Maya mb files of animation

Animation ---- mpeg file

MasterProject ---- PDF

Notice:

Some Maya files should use Maya 6.5 in the CD.