Plasma 2.1 3DS Max Plugin
DOCUMENTATION

VERSION 1.0

TABLE OF CONTENTS

PLASMA PLUGIN

OVERVIEW

The Plasma plugin is a 3ds Max plugin that was developed by Cyan specifically for the Plasma engine. It converts the 3ds Max internal data format into Plasma exported objects that are read directly into the Plasma engine.

There are components that are attached to scene objects that tell the engine which features to use and parameters as to how these features should be rendered.

Also, game logic is tagged with this plugin.

GETTING STARTED: BASIC MAX SCENE

This tutorial takes you through the steps of creating a very basic scene in Max, which will run in the Plasma engine. There are numerous links to other related topics that, once you have worked through it all, you should have a good idea of the Max-to-Plasma process we use at Cyan.

1. Set up your local file system.

2. In Max, configure destination paths for your exports, textures, etc.

   Open the Max Customize Menu –> Configure Paths dialog.
   Set these paths:

   · **Export** path to your Plasma2 root directory. The Plasma installer default for this is C:\Plasma2release.

   · On the **Bitmap tab**, include your local machine locations where Max should look for texture files.

   · **Scenes** path to your Plasma2 root\Max directory.

3. and 4. deleted because they don't apply anymore

5. In Max, start a new Max file.

6. Draw a floor in the Top viewport. (this should be a thin box object, rather than a plane).
   Attach a Physical Terrain component (in the Physics group in Component Manager). how? (This component is required to make the floor 'solid' so the avatar doesn't fall through at runtime.)
   Leave the Component Manager open.

7. Draw a dummy in the Top viewport and position it slightly above the floor plane. (The dummy is a place holder for the avatar, which is seen during runtime only.) It cannot extend below the floor plane or the avatar will drop through at runtime. It must be drawn in the top viewport (but repositioned in one of the side views) so the avatar stands upright, not lying on his side.

Attach a Starting Point component. (in the Type group in Component Manager) Every scene requires a Starting Point.

8. Draw an object, like a sphere. This isn't really required, but it'll give you something to look at during runtime. Apply a Plasma Standard material. how?

9. Include a Max omni light. Reposition it to a place above the ground plane.
   Plasma runtime lights are also available from the Create Lights panel. You don't need them for this tutorial, however, you will use them extensively. See these links for information: Plasma Lighting, Runtime Lights

10. Open the Age Description Manager (Plamsa menu) and click the <New> button next to the Age field. Type a name for your training age.

11. Click the <New> button next to the Page field. Type something for a page name, like 'test' so you can see how it's used. Click <OK>. This creates a data file that identifies your age, which is required by Plasma. details on the Age Description file and tool

12. Select all scene objects and attach a Page Info component in the Misc. group in Component Manager) to them. In the Page Info component parameter UI panel, select the Page you designated in step 11. If you miss attaching the Page Info component to an object, a dialog comes up during export asking if you want to attach it then.

13. Export the scene. (File –>Plasma Export). Note: You can also use the plain 'Export' but without the faster exporting options offered by Plasma Export.

14. **Note**: You might have a debug version of both the plClientSetup and plClient executables in your root directory (plClientSetup_dbg and plClient_dbg). They run your scene the same as the non-debug version but give more information in the case of an engine problem. You can use either version, but the non-debug versions use fewer machine resources. The debug version are generally used only by the programmers.

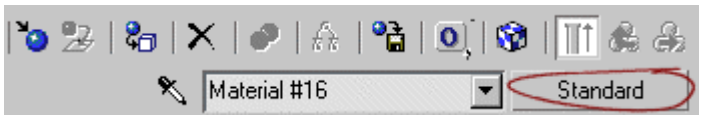
Attaching a Plasma 2 component to an object

1. Select the object to which you want to attach a component.
2. From the Max - Plasma menu, open Component Manager. Component Manager details
3. From the New Component dropdown list, select the component you want.
4. Click the <Attach to Selected Object> button.

Applying a Plasma Material

Cyan has added several Plasma-specific materials to the Max Material Editor and these are used exclusively in our ages. They are listed in the Max Material/Map Browser.

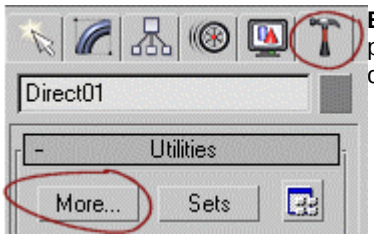
1. Select the object to which you want to apply a material.
2. Open the Material Editor.  Materials overview/details
3. Select an unused sample slot, then click the material selection button.



4. From the list of material types, select **Plasma Standard** and click OK. The Plasma Standard material UI is displayed.
5. In the **Layer Parameters** panel, click the button next to Base Layer. The Texture Parameters panel is displayed. Plasma Standard material details.
6. On the **Texture Parameters** panel, click the button next to **Texture** to display a Windows-type browser.
7. From the browser, locate and select a texture file to apply. Select the texture file and click <Open>. The file's name should now show on the button next to **Texture** and the the sample slot should show a sample of the texture.
8. Now apply the texture by either dragging it onto the object or clicking the Max apply button  .

Component Properties

Access the Component utility panels from the Max Utility tab. From the list accessed through the <More...> button, select 'Component Util'.

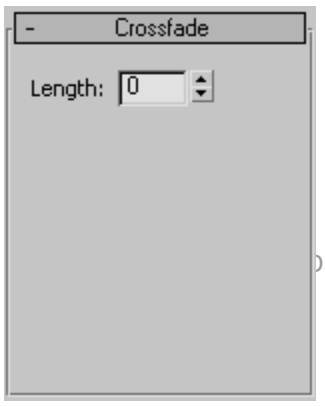


Efficiency tip: Place a button for the Component utility on your Utilities panel by clicking the Configure Button Sets button and using the organizational tool.

Now, when you select a scene object, its component parameters are displayed in the Component utility, where you can adjust them. Component Manager details

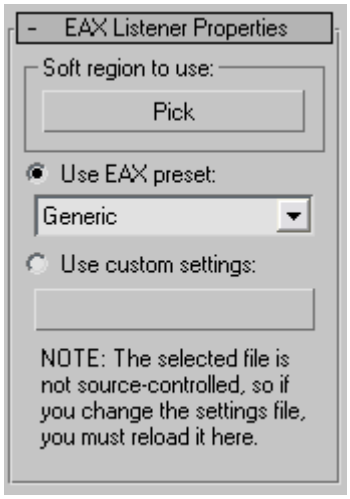
AUDIO

CROSSFADE



Sets the DirectMusic crossfade.

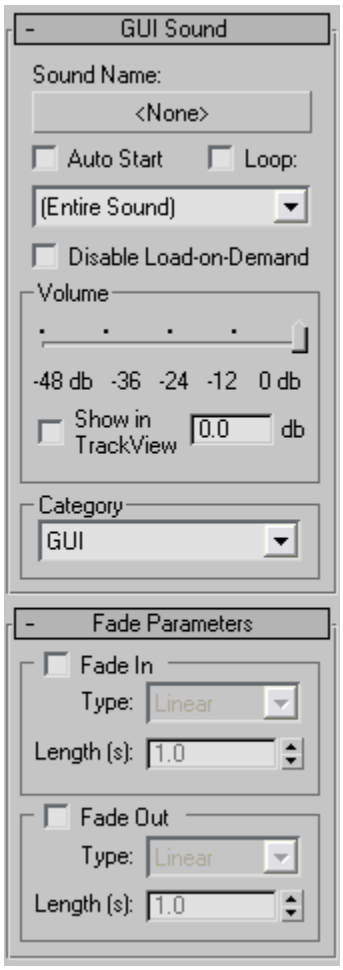
EAX LISTENER



This component sets the soft region and the properties to the EAX listener.

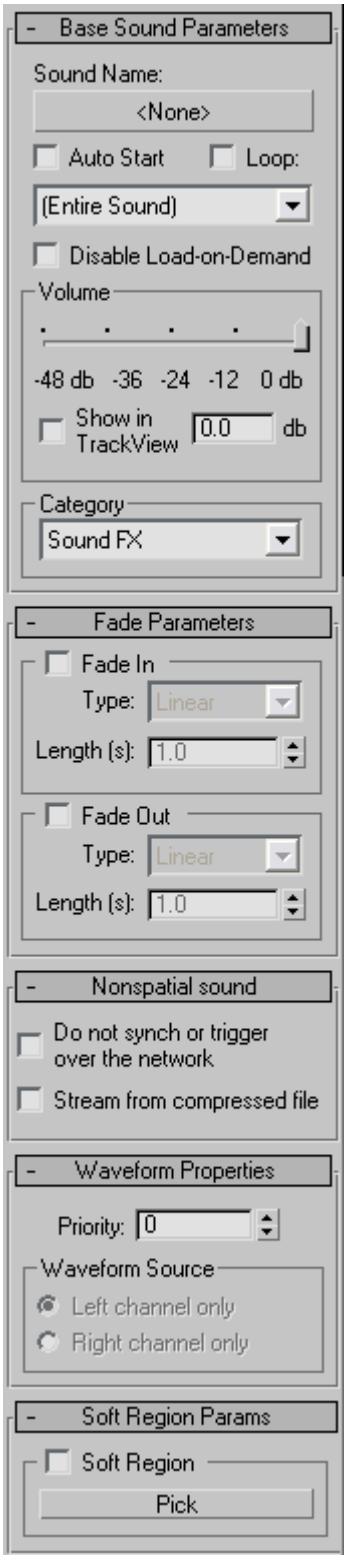
Presets can be:
Generic, Padded Cell, Room, Bathroom, Living Room, Stone Room, Auditorium, Concert Hall, Cave, Arena, Hanger, Carpeted Hallway, Hallway, Stone Corridor, Alley, City, Forest, Mountains, Quarry, Plain, Parking Lot, Sewer Pipe, Underwater, Drugged, Dizzy, Psychotic.

GUI Sound



This creates a sound and properties to be used in a GUI dialog.

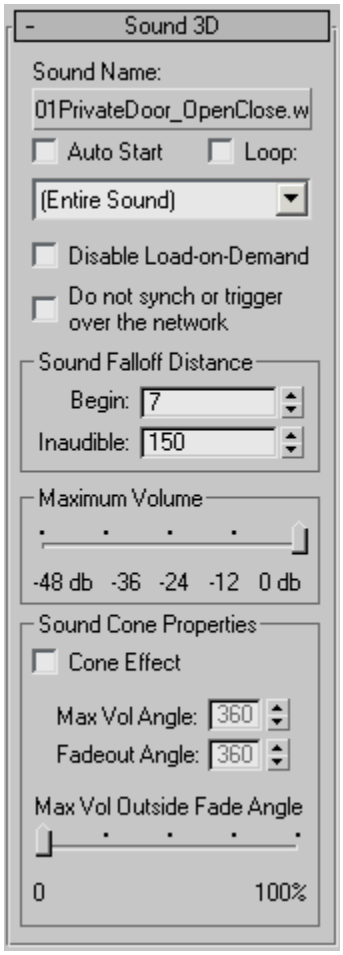
NONSPATIAL SOUND



PHYSICS SOUND GROUP

RANDOM SOUND

Component Parameters



Sound Name - Click to browse AssetMan for the sound you want. AssetMan embeds a media player for listening. You can select stereo files as sources, and when you do, you'll get the option of selecting which channel you want to use as a source (left or right). This allows you to, for example, use a single stereo sound to store two mono tracks that will then be separated into two mono 3D emitters.

Auto Start - Check this to start the sound upon game start. Unless you also check 'Loop' the player might never hear the sound, since in the Parable world, they might not enter this location when you think they will. It's better to trigger the sound.

Loop - Check to activate looping through the sound file. Looping begins after one time through the whole file. From the dropdown list, select the Notes Track segment name, if applicable. Otherwise, the whole sound file is played and looped through.
Note: you can start looping during runtime with Python code.

Disable Load-on-Demand - When checked, this property ensures that this sound file is loaded when the age loads at runtime, regardless of the LOD setting in the Parable setup dialog. (The

Do not synch or trigger over the network -

Sound Falloff - Set the distance at which the sound's volume begins to diminish and at which it is not longer audible.

Begin: Plasma 2 starts reducing the sound's volume when the player reaches this distance (in feet).

Inaudible: The sound's volume is at zero when the player reaches this distance.

Maximum Volume - You may choose to reduce the maximum possible volume of the sound file by using this slider. For example, if you want the sound to play at 50% of the file's normal volume, move the slider to the 50% mark.

Sound Cone Properties - (see the detailed explanation)
The sound cone direction is the negative of the object's local Z-axis in Max.

Cone Effect:

Max Vol Angle: The volume before attenuation.

Fadeout Angle: The size of the inside portion of the sound cone. (See below for details.)

Max Vol Outside Fade Angle:

Minimum & Maximum Distances

There's been some misconception (on my part, maybe yours too) of the way that the minimum and maximum distance settings work with the sound system. Here's a handy guide that explains it in great detail. Please note that in next week's release of the game and the plugins I have fixed the ranges in the component to be from 1 to 1000000000...

Minimum and Maximum Distances
As a listener approaches a sound source, the sound gets louder; the volume doubles when the distance is halved. Past a certain point, however, it is not reasonable for the volume to continue to increase. This is the minimum distance for the sound source.

The minimum distance is especially useful when an application must compensate for the difference in absolute volume levels of different sounds. Although a jet engine is much louder than a bee, for practical reasons these sounds must be recorded at similar absolute volumes. An application might use a minimum distance of 100 meters for the jet engine and 2 centimeters for the bee. With these settings, the jet engine would be at half volume when the listener was 200 meters away, but the bee would be at half volume when the listener was 4 centimeters away.

The default minimum distance for a sound buffer, DS3D_DEFAULTMINDISTANCE, is defined as 1 unit, or 1 meter at the default distance factor. Unless you change this value, the sound is at full volume when it is 1 meter away from the listener, half as loud at 2 meters, a quarter as loud at 4 meters, and so on. For most sounds you will probably want to set a larger minimum distance so that the sound does not fade so rapidly as it moves away.

The maximum distance for a sound source is the distance beyond which the sound does not get any quieter. The default maximum distance for a DirectSound 3-D buffer (DS3D_DEFAULTMAXDISTANCE) is 1 billion, meaning that in most cases the attenuation will continue to be calculated long after the sound has moved out of hearing range. To avoid unnecessary processing, applications should set a reasonable maximum distance and include the DSBCAPS_MUTE3DATMAXDISTANCE flag when creating the buffer. This flag is automatically set on standard 3-D buffers created as part of an audiopath; see Standard Audiopaths.

The maximum distance can also be used to prevent a sound from becoming inaudible. For example, if you have set the minimum distance for a sound at 100 meters, that sound might become effectively inaudible at 1,000 meters or less. By setting the maximum distance at 800 meters, you ensure that the sound always has at least one-eighth of its maximum volume regardless of the distance. In this case, of course, you would not set the DSBCAPS_MUTE3DATMAXDISTANCE flag.

The following illustration shows how minimum and maximum distance affect the loudness of a jet and a bee at increasing distances.

Sound Cone

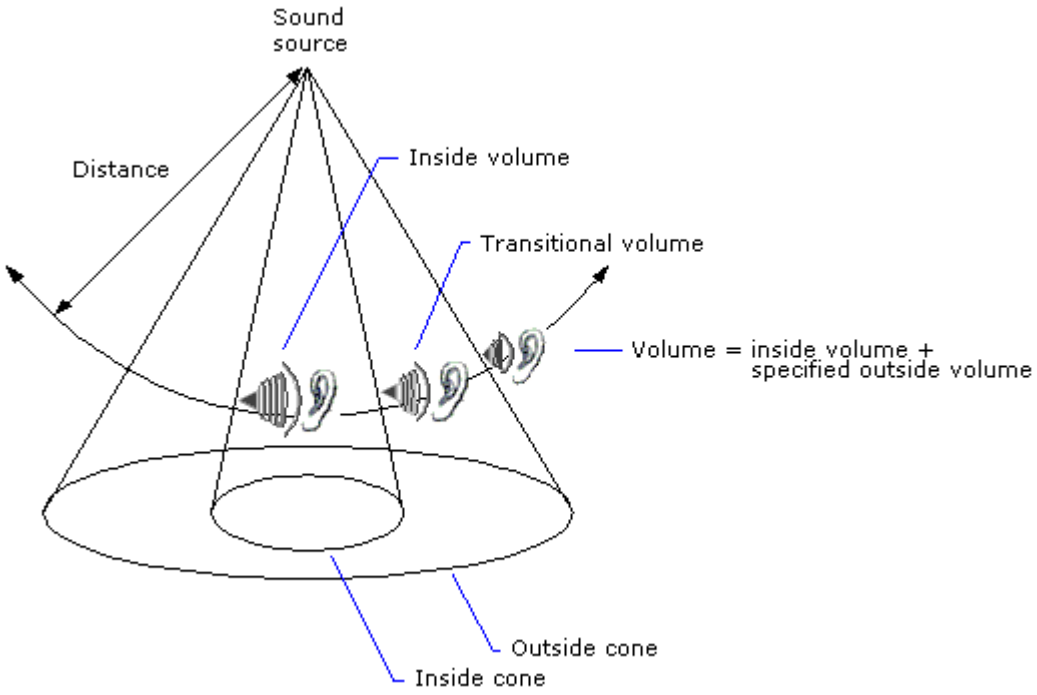
A sound with no orientation has the same amplitude at a given distance in all directions. A sound with an orientation is loudest in the direction of orientation. The model that describes the loudness of the oriented sound is called a sound cone. Sound cones are made up of an inside (or inner) cone and an outside (or outer) cone.

At any angle within the inner cone, the volume of the sound is just what it would be if there were no cone, after taking into account the basic volume of the buffer, the distance from the listener, the listener's orientation, and so on.

At any angle outside the outer cone, the normal volume is attenuated by a factor set by the application. The outside cone volume is expressed in hundredths of decibels and is a negative value, because it represents attenuation from the default volume of 0.

Between the inner and outer cones is a zone of transition from the inside volume to the outside volume. The volume decreases as the angle increases.

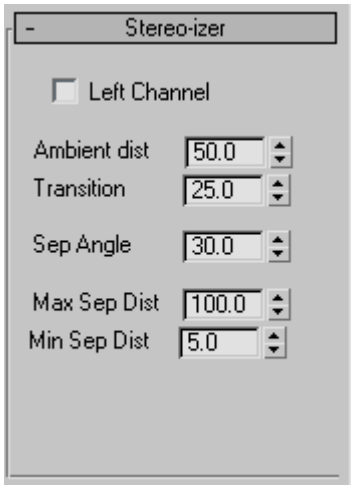
The following illustration shows the concept of sound cones.



Every 3-D sound buffer has a sound cone, but by default a buffer behaves like an omnidirectional sound source, because the outside volume is not attenuated, and the inside and outside cone angles are 360 degrees. Unless the application changes these values, the sound does not have any apparent orientation.

Designing sound cones properly can add dramatic effects to your application. For example, you could position a sound source in the center of a room, setting its orientation toward an open door in a hallway. Then set the angle of the inside cone so that it extends to the width of the doorway, make the outside cone a bit wider, and set the outside cone volume to inaudible. A listener moving along the hallway will begin to hear the sound only when near the doorway, and the sound will be loudest as the listener passes in front of the open door.

STEREO-IZED



CAMERA

OVERVIEW

Plasma includes a growing list of camera types, each of which provides a specific type of camera behavior. Most of these camera types take parameter settings for precise adjustment. The programmers refer to the camera types as "brains" since in reality the component gives a plain camera smarts about how to follow a target or when to be active.

Plasma manages the scene's cameras with a single list. The currently active camera is the first one on the list. When another camera is triggered, it's placed in the top spot on the list and Plasma manages the transition between the two. The list always has at least one camera on it, which is called the default camera. The default camera is always the last camera in the list and cannot be removed. This is the camera view the player will have upon linking in to an age.

You can designate any camera as the default camera by attaching a Make Default Camera component to the camera of your choice. But if the link-in spot in a scene does not contain a default camera by design, Plasma will place one there for you.

CAMERA TYPES

Fixed Camera — A simple camera that stays where you place it in the scene.

Animated Camera — A fixed camera that has been animated and has an Animation component attached to it.

Auto Camera — A camera that moves relative to the player's position and is frequently called a **follow cam**. Offset parameters are set in the component rollout.

Rail Camera — A camera that follows a predetermined path, keeping the player in view.

Circle Camera —

First Person Camera —

Allow Camera Panning —

Allow Camera Zoom —

Avatar POA —

Object POA —

Camera Region —

Override Transition —

POINT OF ATTENTION

Usually a camera's point of attention (POA) is the avatar, but there are some exceptions:

- A specific location or object (object with an Object POA component attached) may be the POA.
- Objects tagged as "interesting" (object with a Player Attention component) cause a camera to adjust itself to include both the object and the avater in the player's view.
- When interacting with other avatars, the camera will try to frame both the player's avatar and other nearby avatars.

CAMERA BRAINS

The camera system supports several modes. Each mode is referred to as a camera brain. The brain represents the code and a set of data parameters. In general, the parameters for a given brain are fixed. If you want to change the behavior of the camera you will switch to a new brain that has new data.

Brains are created by placing a camera in the Max Scene. Components attached to the camera will specify any additional parameters. Depending on the type of brain, the in-scene camera's parameters are treated differently.

BRAIN MANAGER

The brain manager is just a list of brains. The brain at the head of the list is the active camera. New cameras can be pushed onto the front of the list, popped off, or removed from any point in the list. Whenever the camera changes, the Brain Manager figures out what transition must take place and manages the transition.

DEFAULT CAMERA

There is always a default camera, and its parameters are set by the engine. This is set as the bottom of the Brain Manager's stack upon link-in to an age and cannot be popped off. If this is not the camera which is desired at the link-in point then the correct camera should be triggered or setup with a Camera Region. The default camera for the age can be overridden using the Set Default Brain function at runtime. This takes a camera from the scene at runtime and makes it the new default camera for this age.

Console commands that manipulate the camera are for testing and production uses only. No Age should rely on them.

SIMPLE BRAINS

A simple brain is essentially one where the artists specify the location of the camera origin.

FIXED CAM

The fixed cam just a camera that is placed at a particular location in Max. The in game camera will respect the location, direction, and fov of the max camera. The Fixed Cam allows for you to specify a specific Point of Attention (POA) for the camera. This way the camera, although "fixed" can be set to follow the player.

ANIMATED CAM

An animated cam is like a fixed cam but it is animated in max. Location, direction and fov can all be animated. This kind of camera is created by simply placing an animation component on a fixed camera. The animation of the camera can be controlled/triggered just like any other animation.

AUTO CAM BRAIN

This is a camera brain whose origin point movement is controlled by the engine. The camera moves relative to the player's position and has several options. I am sure this list of options will increase over time but the first version will support those listed here.

OFFSET

The Auto Cam will attempt to keep itself at a particular offset from the Point Of Attention. This offset will be different for different instances of the Auto Cam (closer in tight spaces etc.).

There are two ways to specify the offset:
World Relative - The offset is specified in World Space (Max file space) and ignores any rotation.
POA Relative - The offset is computed in the coordinate system of the object that is the POA.

This is best illustrated when in the case where the POA is the player. World Relative produces a more cinematic camera where the player can rotate around while the camera stays in one place. POA relative would be used to create a "behind-the-back" cam.

COLLISION

The Auto Cam will support collision with the environment. The collision parameters will be built into the engine. Ie. There will be no physics component on the camera. There will be a set of console commands for setting these values but only for experimentation/debugging. These will be the only way to turn off collision for the Auto Cam.

RAILS

The auto cam can be constrained to stay on a rail. There is currently a line follow component in plasma 2, which includes some of this functionality. The camera rail will likely need more parameters as it develops so it should probably be created as a derived class of the line follow modifier.

LOS

This option indicates that the camera will adjust its location to try to insure that the POA is visible. When checked there will be a ray cast from the POA back to the camera. If the LOS is not clear then the camera will move in front of the object that is blocking the view.

AUTO CAM BRAIN COMPONENT

This component if placed on a camera object in max, makes this camera into an auto cam brain.
LOS [] - Checkbox to enable los checking for POA
Offset Type - Radio Button for Player Relative or World Relative
Offset X,Y,Z - Scalar offset values in Feet.
LAG Scale - Scale factor to control the camera lag

RAIL COMPONENT

The rail component can be placed on a camera object that will have an Auto Cam brain. Placing this on other cameras is an error. Only one rail component can be added per camera.

CAMERA POINT OF ATTENTION

In most case the camera target, point of Attention (POA), lookat etc. is the player. There some times when this isn't true:

1. Cameras may specify a specific location or object (potentially moving) as their POA.
2. Objects Tagged as interesting. When the player gets near an object that is tagged as interesting the camera will adjust the view to attempt to show both the player and the interesting object in frame.
3. Other Avatars you are interacting with. When you are in conversation with other avatars the camera will attempt to frame you in the scene. Basically the camera system declares that other avatars you talk to are interesting objects.

Note that you can specify a object for the POA and also specify an offset for the POA relative to that object. For example. You can make the camera focus on a point 10 feet in front of the player.

LAG/ CURRENT AND DESIRED POSITION/TARGET

When a camera has a dynamically changing position or POA, it will determine its desired state based on the current settings and scene information. It will then move towards the desired position. If we simply set the camera to this position the effect is often too jerky. So we would like to incorporate some lag. The current system allows for the specification of speed and acceleration for the camera's position and orientation. This is tricky since the POA's velocity can vary widely and it becomes possible for the camera to "lose" the player. We need the speed to be controlled in a different way. We will have to experiment but one thing to try would be this: The velocity should be a function of how far the camera is from its Desired position. There will be separate curves for position and target. Remember only the AutoCam supports a dynamically controlled camera position. The basic curves will have to be established via experimentation. The camera will behave this way in each age and the basic curve will not change. However, the system will support a camera brain parameter that scales the curves.

The camera will also support an instantaneous mode where there will be no lag. Whether this will be an option for users has not yet been decided.

POINT OF ATTENTION COMPONENT

Specify What does this camera look at. If there is no such component then the Default is the player and the interesting objects.
Checkboxes: Ignore Interesting Objects [] - Don't worry about objects tagged as "interesting" when processing this brain
Tracking Speed Scale - Scalar - Scale factor for speed of camera

POINT OF ATTENTION (RADIO BUTTONS)

Player - Your player.
Scene Object - A particular object in the scene.
Camera Default - Ie what the camera is looking at in max. (fixed cam)
POA Offset - 3 Scalars - X,Y,Z offset of true POA from POA target. This is relative to the POA's coordinate system

OBJECT OF INTEREST COMPONENT (FORMERLY PLAYER ATTENTION COMPONENT)

Used to tag objects in the scene as interesting. The component supports a radius which the player must be in for the object to be "interesting" and a scalar to determine how interesting.

TRANSITION

When you switch from one brain to another there is a transition. Transitions can be cut or track. If the POA is the same for the two brains then it should remain in view throughout the transition.

CAMERA SPECIFIC TRANSITIONS

The system will include the ability to specify the transition type for each camera to camera transition. Ie. Camera A to Camera B is different than Camera C to Camera B and Camera B to Region A. What this means is that each region will need to have a transition list which determines what happens when the avatar moves into that region. Each item in the list will consist of the name of the region the avatar is coming from and the kind of transition that will take place. This specification will only be necessary then the default behavior is overridden. There will be a default transition specification for changing between each time of camera brain. These defaults will be set by the engine and won't be part of any per age information.

If the camera needs to switch again before a transition completes, then the current in transition camera state is taken as the current cam and the transition occurs between that state and the new camera. The transistiion which occurs will be the one which is determined by the second transisiton.

For Example:
If A is in Transition to B and during this transition a switch to C occurs. The particular transition which occurs will be the B -> C transition not the A->C transition.

TRANSITION COMPONENT

This component specifies the kind of transition that occurs when you transition to this camera. You can have several of these components on a camera to select how transition into this should behave. This is used to override the default kind of transition that the engine normally uses for switches between these kinds of brains. When the transition is between two simple brains (fixed, animated) the default behavior is to cut. In all other cases the default behavior is to track.
Parameters:

From Camera : Max Camera Object : Name of a camera which can transition into this one

Track Option [] - Checkbox to make this transition a track. Otherwise it will cut.
Track Parameters Rollout: Optional rollout to set specific tracking parameters.

Parameters:
Target Track Time : Scalar (0 to 20) : (secs) for Camera Target to transition. 0 is cut

Position Track Time : Scalar (0 to 20) : (secs) for Camera Position to transition. 0 is cut.

FOV Track Time: Scalar (0 to 20) : (secs) for FOV to transition 0 is cut.

CAMERA REGIONS

Camera regions are prepackaged triggering for cameras. They are detector and responder all in one. Each region is associated with a particular camera brain. When the player enters the region the camera brain associated with that region is pushed onto the front of the stack. Note: the behavior of these regions is really not a camera issue per se. The same rules for inclusion/exclusion for these regions apply whether they are reverb regions, camera regions or whatever. Information is just included here as a reference.

We currently support only convex regions. These are simple to define and evaluate. They should be used whenever possible for camera control, as they are the simplest mechanism for this. The boundary of the regions will have some histerisis s. Ie. It won't swap back and forth quickly between two regions. If I go from A to B and then Back to A the camera won't switch to A unless enough time has elapsed. This time will be settable from the console but will default to 1 second.

The region will also have an option to indicate that this is a one-time trigger. It will only occur the first time an avatar enters it. The region can be re-enabled via a message or responder command but otherwise it will only happen once, say the first time the player enters a room.

CAMERA SCRIPTING

In addition to the camera regions, Cameras can be set via scripting or external modifiers.

RESPONDER

The responder will have two new commands CameraPush, CameraPop for setting Camera Brains.

PYTHON

Python will also have commands to change the current camera.

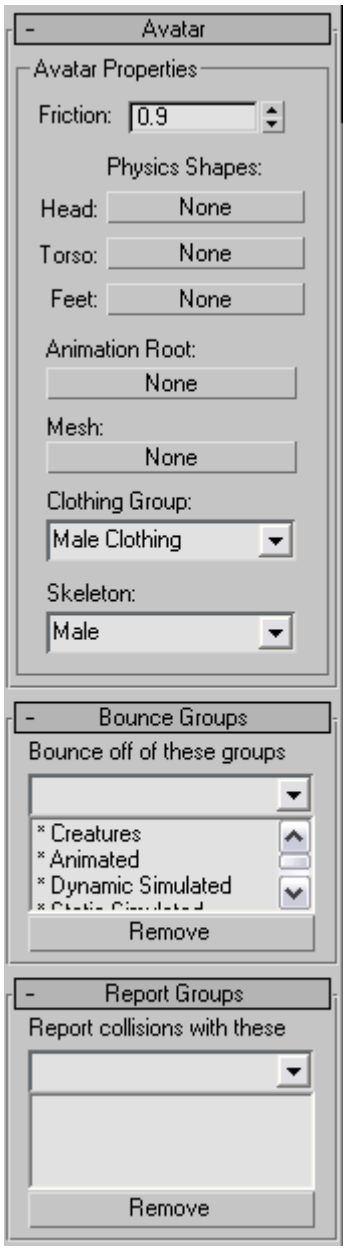
C++

External C++ modifiers will be able to send messages to the BrainManager to switch brains

CHANGING BRAIN PARAMETERS

If necessary we will provide a separate API for changing parameters of the brains at runtime. This won't be in the first version.

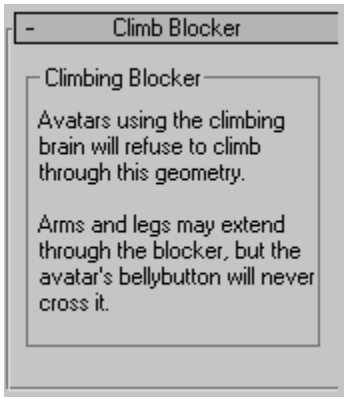
AVATAR


Avatar
Avatar Properties
Friction: 0.9
Physics Shapes:
Head: None
Torso: None
Feet: None
Animation Root:
None
Mesh:
None
Clothing Group:
Male Clothing
Skeleton:
Male
Bounce Groups
Bounce off of these groups
* Creatures
* Animated
* Dynamic Simulated
Remove
Report Groups
Report collisions with these
Remove

AVATAR CLIMB TRIGGER

AVATAR FOOTSTEP SOUND

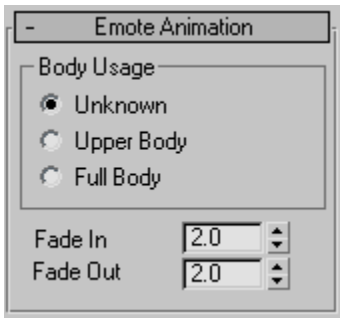
CLIMB BLOCKER



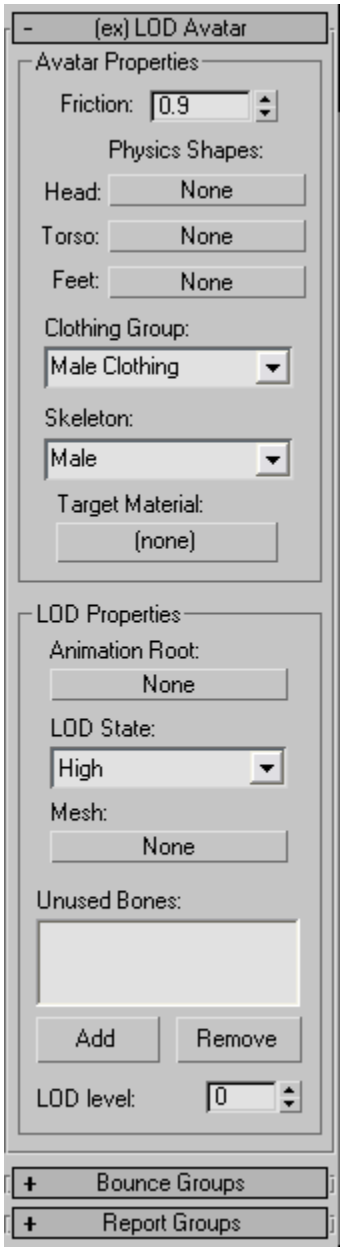
COMPOUND ANIMATION

COMPOUND CONTROLLER

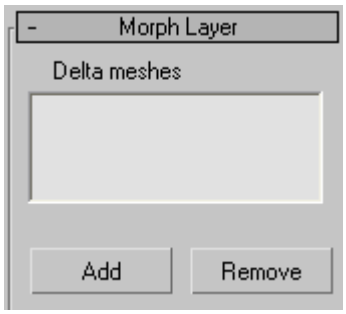
EMOTE ANIMATION



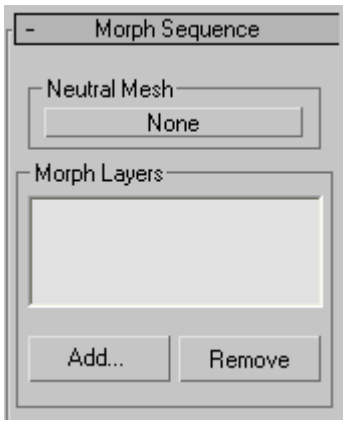
LOD AVATAR



MORPH LAYER

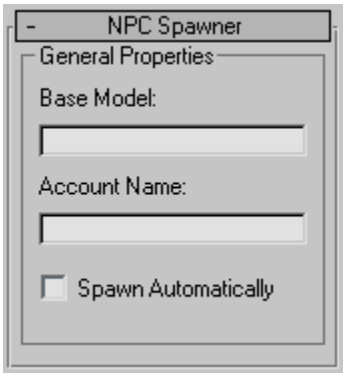


MORPH SEQUENCE



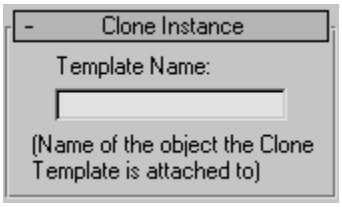
MULTISTAGE BEHAVIOR

NPC SPAWNER



CLONE

INSTANCE



TEMPLATE

DISTRIBUTOR

Introduction

The Distributor is a system for the creation, maintenance, and processing of complex aggregates. The power of the Distributor is in things that aren't important as individual objects, but only as a collection. The Distributor offers several benefits.

- It allows you to work at a higher level of abstraction. If you want to make a forest, you can work on it as a forest, rather than as 100 trees.
- It automatically does many things that you would do, but wouldn't enjoy doing. It attempts to take on as much of the grunt work as possible, freeing you to concentrate on the art work.
- It makes for assets more robust in the face of change. Major restructuring of the underlying terrain will generally require no reworking of the ground cover under control of the Distributor system.
- It allows for higher optimizations in the rendering. Because the system as a whole has more information about what is being attempted and how it is constructed, it can perform optimizations that couldn't be applied to generic scene geometry.
- It insulates the assets from changes in system requirements. The generated geometry is currently generated offline and exported as normal scene data. But on a different platform (e.g. Xbox), or over a slow bandwidth network, it would probably prove more efficient to generate the data at runtime on the client machine.

The Distributor allows technical restructuring as needed without having to rework assets.

Terms

- Cluster – the master component, runs the Distributors
- Distributor – actually distributes replicant geometry over scene surfaces according to user parameters.
- Replicant – an individual mesh to be replicated and distributed over one or more surfaces.
- Surface – the mesh or meshes over which you will be distributing replicants.

General Tips

- Limit the number of materials you use, preferably to one per distributor. Tile your plant bitmaps into a single texture, and use your UV mapping to select which plant each replicant gets.
- Don’t hand edit the output of the Cluster. Adjust with the Distributor parameters. Any hand editing of the output will be lost the next time the scene is generated.
- Reuse your Fade Out (and Fade In) parameters. Setting one distributor to fade out at 60 feet and another to fade out at 65 feet will not give much of a perceptual difference, but it will prevent the system from batching the output of the two Distributors together.
- No you can't use View Facing on distributed objects as the cost would be prohibitive. The system will ignore you if you try.
- No you can't add shadows to distributed objects yet…maybe later.
- You can use as many distribution probability maps as you need. Different distributors can use specific maps or share maps with other distributors depending on what replicants you want where.
- The wind can't move mesh distributions in waves. You can randomize the motions somewhat with the flexibility component and by making secondary distributions attached to secondary child Wind Bones with different values.
- When using 'Surf Offset Min/Max' [Conformity to Surface] to make floating objects, remember that the distributor will be filling a volume rather than covering a surface. You can fit *many* more objects into a volume than on a surface with any given set of density and spacing values.

Getting Started

The most basic usage is straightforward and simple. Start off just seeing your replicants getting spread about, and then work into more control with the various parameters.

To start, you will need at least one surface (e.g. the terrain mesh(s)) over which you want to distribute stuff.

Second, you will need at least one object to distribute.

Once you have these two Max objects, you need at least two components.

For each distinct distribution of replicants, you need a Distributor Component. The Distributor Component is attached to the mesh or meshes over which you will be distributing replicants (the Surface). One Distributor can distribute any number of replicants, but they will all be distributed with the same pattern. One surface may have any number of distributions, and hence Distributors, over it. Surfaces may also share Distributors, one Distributor may be applied to a group of surface meshes, and the distribution will logically span all those surfaces.

The second component is the Cluster Component. The Cluster Component manages the multiple Distributors, and optimizes the output. It should be attached to all the meshes which have Distributors it will manage.

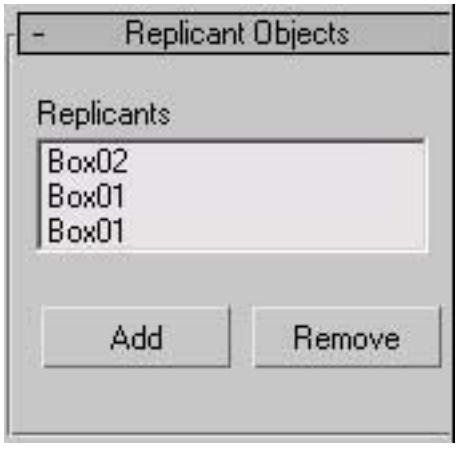
Apply the Distributor and the Cluster Component to your surface mesh(s). Select the Distributor in the Component Util. Go to the Replicant Objects rollup (the first one and only one open). Click on Add. Now select your replicant object. Okay, you're ready to go. Select the Cluster Component in the Component Util. Hit Go. The rest is just fine tuning the parameters, as described below.

If you are trying to adjust the **Distribution**, you want one of these rollups:
Spacing
Orientation Dependence
Altitude Dependence
Probability Bitmap

If you are trying to adjust **Appearance**, you want one of these rollups:
Replicants
Scaling
Replicant Orientation
Conformity to Surface
LOD Fade
Wind Effect

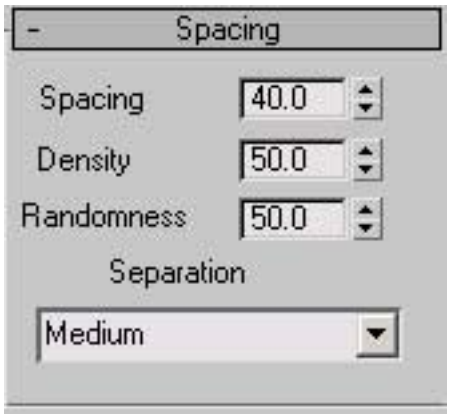
Distributor Roll outs – Attach to Base Surface Mesh

Replicants



The list of objects you want distributed over the surface. To add to the list, click on the Add button, then select the object you want to replicate, either by mouse picking in the scene or through select by name. To remove an object from the list, click on the object's name in the list box, and click the Remove button.
A Distributor can have any number of objects to replicate. For each position it decides to drop a replicant into the scene, it will randomly select one replicant from the list. Adding an object into the list twice will double its chances of getting picked, and therefore double its frequency of occurrence in the scene. For example, if the replicant list consists of ObjectA once, and ObjectB four times, the output will be a mix consisting of 1/5 ObjectA and 4/5 ObjectB.
All replicants within a Distributor should share the same material. The Scale of all replicant's transforms should be reset before adding them to a distributor. The orientation of the replicant can optionally affect its placement in the world. See Replicant Orientation below.
The placement of the pivot point on the replicants is key. The pivot point should be at the ground level of the replicant (the base), with the Z axis pointing toward the end protruding out of the surface.

Spacing



Spacing

The nominal spacing between replicants distributed over the surface. In feet.

Density

The overall probability that a spot on the spacing grid will receive a replicant. A lower density reduces the number of replicants, a higher density will produce a more even coverage. In Percent.
So if Density is 100.0 you get as many replicants as will fit Spacing ft apart. If you set Density to 0 you get no replicants (don't do that).

Randomness

Amount to randomly perturb the replicant positions from a hypothetical grid. A randomness of 0% will put the replicants on a somewhat regular

grid (with distance between grid points of Spacing), a randomness of 100% will produce a totally random coverage.

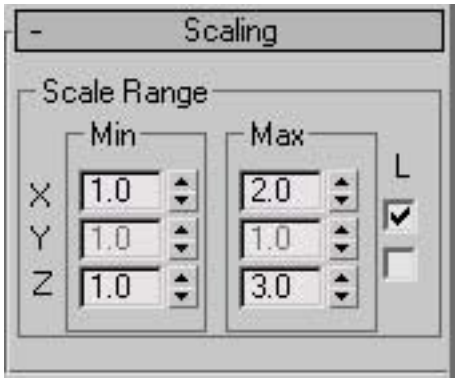
Separation

Amount to prevent interpenetration between replicants. In abstract, the system will place a protected zone around each replicant. The system won't place the replicant anywhere that would make its protected zone intersect any other protected zone. Once the replicant is placed in the scene, no other replicant will be added in a way that would make its protected zone intersect the first's. The options relate to the size of the protected zone, and are:
None – Allow replicants to be placed directly on each other.
Low – Create a one foot cube protected zone around each replicant.
Medium – Protected zone is the replicant's bounding box.
High – Protected zone is a box of size Spacing cubed.

Tip on Spacing

You can decrease the number of replicants distributed over your surface either by increasing the Spacing or decreasing the Density. Note that the distribution process will go faster if you increase the Spacing versus decreasing the Density. Try to use the Spacing first to get about the right density, and then tweak with the Density setting.

Scaling



Scale Range Min/Max

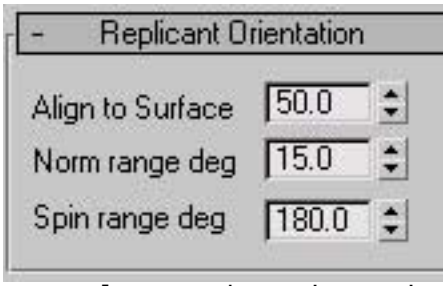
Amount to randomly scale replicants as they are added to the scene. You may supply a minimum and a maximum amount to scale in each dimension. The system will pick a random scale amount in that range for each replicant. The dimensions are in the replicant's native space based on its pivot point. A scale value of 2.0 will double the size of the replicant.

Lock

By default, this is a non-uniform scale. For example, if you have minimum and maximum scales of 1.0 and 2.0 respectively for all three

dimensions, the system might select a scale of 1.0 in X, 1.5 in Y and 2.0 in Z.
There may be times that this isn't what you want. There are two check boxes to lock dimensions of the scale (on the right). The first check box locks the X and Y scaling. So from top down, replicants will remain constant in their proportions, but their height will vary independently. When the XY lock is selected, the Y scale values will be greyed out and ignored, as X and Y will both be scaled by the same amount, as determined from the X scale range.
The second scale lock check box will tie all three scale dimension. When selected, the Y and Z scale ranges will be greyed out and ignored, as X, Y, and Z will be scaled by the amount determined from the X scale range. When selected, replicants will vary in size, but remain constant in their proportions in all three dimensions.

Replicant Orientation



Control over the orientation of replicants distributed over the surface.

Align to Surface

Amount to base the "up" direction of the replicant on the surface normal at the position the replicant is being placed, versus the "up" direction of the original replicant object. (The "up" direction of the replicant is the direction the Z (blue) axis of its pivot point points.)With a value of 100%, the "up" direction of each replicant will point directly out of the surface. With a value of 0%, the "up" direction of each replicant will be the same as the original. So to make vines that hang straight down, turn your original vine upside down and set Align to Surface to 0%.
Note that in no case will the system allow a replicant to poke down into the surface over which it is being distributed. If you applied the hanging vines to a surface which faces up, the vines will lay flat on the ground pointing down as much as they can. Sometimes this is what you want, for other cases see the Orientation Dependence section.

Normal Range deg

Amount in degrees to randomize the "up" direction. The up of the replicants in the scene will fill a cone of this angle. This randomness is applied after the check for meeting Orientation Dependence requirements (see Orientation Dependence below).

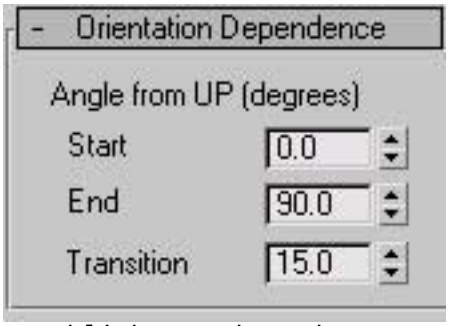
Spin Range deg

Amount to randomize the spin of the object about its “up” direction. With a spin range of zero, replicants will try to keep the Y axis of their pivot points aligned with the source replicant’s. 180 degrees will apply a random spin to each replicant placed in the scene.

Use Surface Face Normals Checkbox

Rather than interpolate the vertex normals to the position on the face at which the replicant is being dropped, use the flat face normals. This has proven useful for objects which mate with the surface over an extended area (e.g. crater decals), particularly over rough surfaces. Note that this also affects the Orientation Dependence described below.

Orientation Dependence



Establishes a dependence on the surface normal of whether to place a replicant at that position. The angles are based on the angle between the surface normal and the UP direction. So 0 is up, 180 is down, 90 is in the XY plane.
Note that the angle is based on the surface normal, not the up direction of the output replicant. So even if you force all your replicants to point straight up (Align to Surface = 0), they will still fail the Orientation Dependence test if the surface they are applied to fails.
Note also that the surface normal used IS affected by the Use Surface Face Normals checkbox in Replicant Orientation (described above).

Start

Minimum angle from up to allow replicants.

End

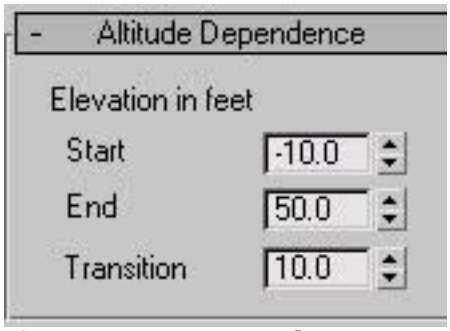
Maximum angle from up to allow replicants

Transition

Number of degrees over which to transition from not allowing replicants to allowing full coverage.
For example, say the Start angle is 45 degrees, the End angle is 90 degrees, and the Transition is 5 degrees. Then where the surface normal is within 45 degrees of pointing up there will be no replicants placed. At 45 degrees there will be very few, ramping up to full coverage at 50 degrees (5 degree transition). There will be full coverage from 50

degrees to 85 degrees. Coverage will ramp down from 85 to 90 degrees. There will be no replicants where the surface normal is more than 90 degrees from up (i.e. where the surface is vertical or at all downward facing).
The angles 0 and 180 are treated slightly differently. If 0 or 180 is specified as Start or End, then there is no transition to that angle, full coverage is maintained all the way to straight up (0) or straight down (180). For example, setting Start to 1, End to 90, and transition to 10 would cause no coverage between 0 and 1 degrees from up, ramping up to full coverage at 11 degrees, full coverage from 11 to 80 degrees, ramping down to no coverage at 90 degrees. But setting Start to 0, End to 90 and transition to 10 would cause full coverage from 0 to 80 degrees, ramping down to no coverage at 90 degrees.
Set Start and End to zero (0) to disable.

Altitude Dependence



These are very analogous to the Orientation Dependence. Altitude Dependence can be used, for example, to limit a type of fern to only appearing close to sea level. Set Start and End to zero (0) to disable. All units are in feet in Max’s world coordinates.

Start

The minimum height at which to allow replicants.

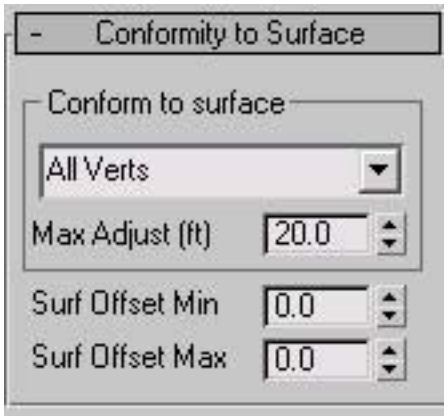
End

The maximum height at which to allow replicants.

Transition

The distance over which to transition from not allowing replicants to allow full coverage.

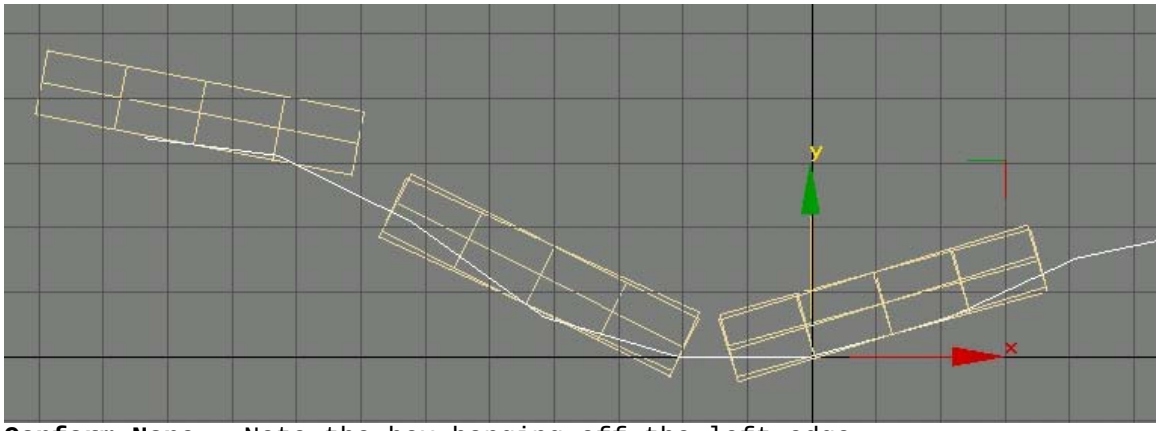
Conformity to Surface



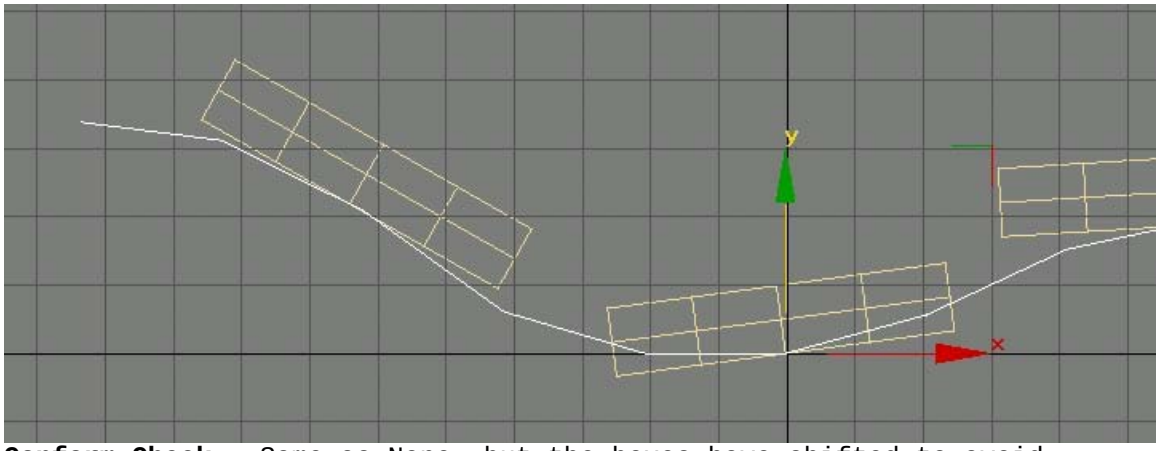
Conform to surface (type)

Options on how to conform to underlying surface. The choices are:
None – Just make sure the pivot of the source replicant is on the surface.
Check – Make sure all vertices are over the surface. Prevents extended replicants from hanging over edges. See Max Adjust below.
All Verts – Distort the replicant mesh copy so that each vertex keeps its original height (relative to its pivot point) over the surface onto which it is being distributed. See Max Adjust below.
Bottom – Like All Verts option, but distorts based on height (relative to pivot point). Verts at the base of the replicant conform exactly to the surface, verts at the top of the replicant don't conform at all, and verts in the middle conform half way. See Max Adjust below.
Base – Like Bottom option, but only vertices at the very base (within 6 inches) are adjusted. See Max Adjust below.

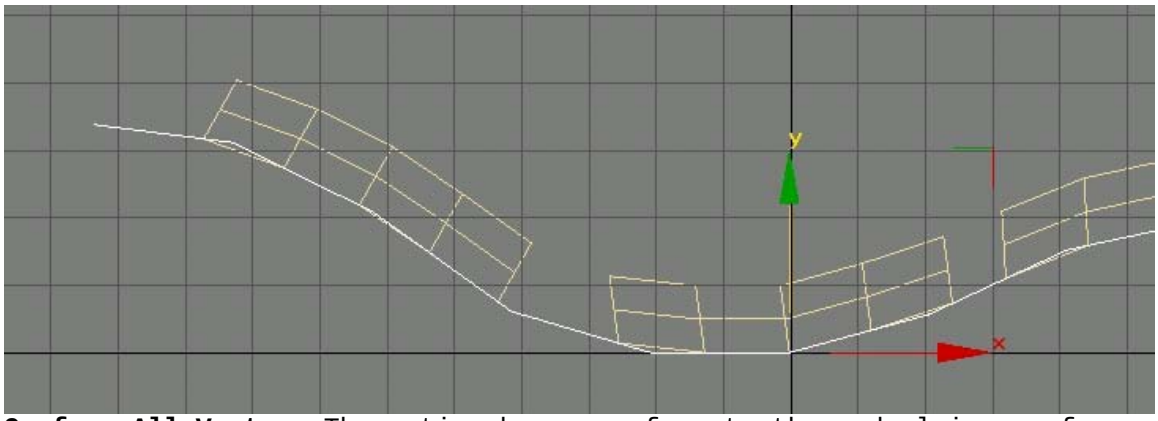
The following illustrate the different conform type behaviors.



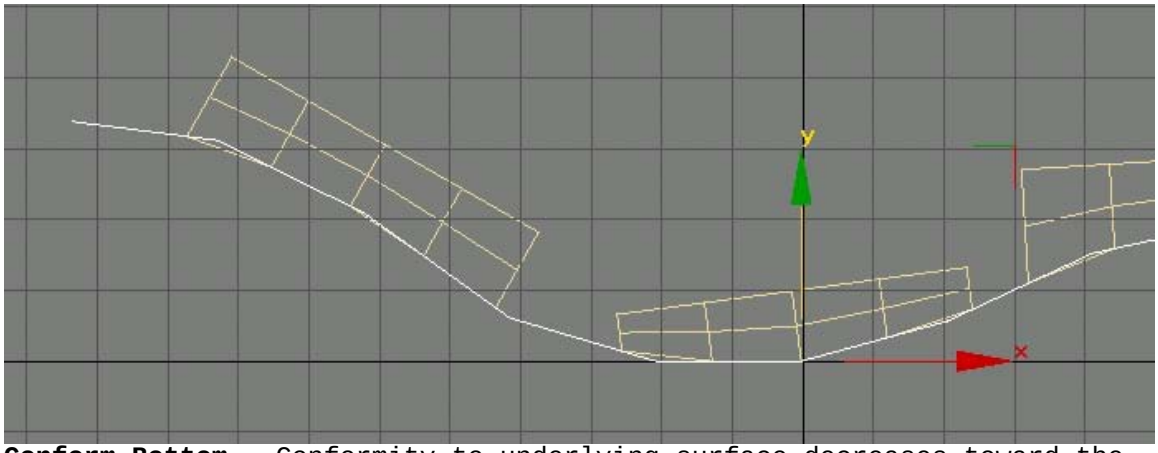
Conform None – Note the box hanging off the left edge.



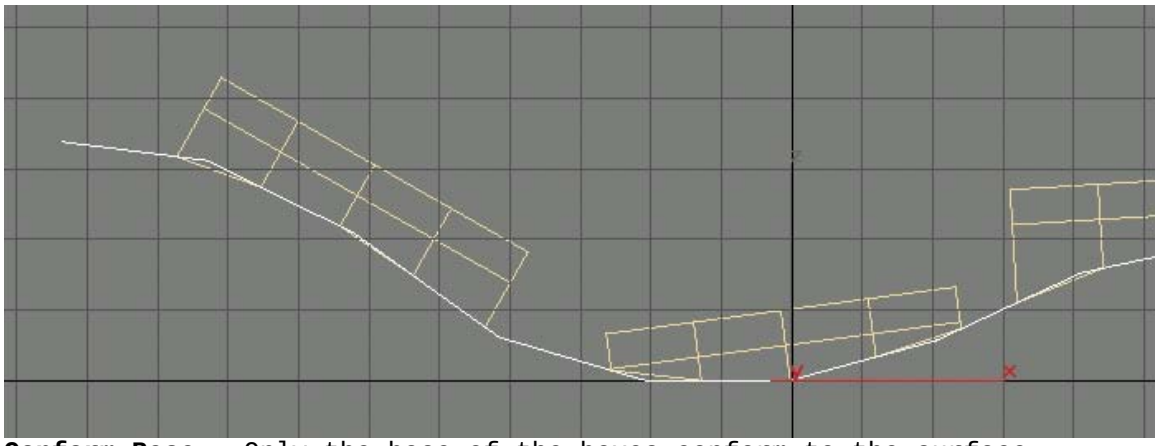
Conform Check – Same as None, but the boxes have shifted to avoid hanging off the edge.



Conform All Verts – The entire boxes conform to the underlying surface.



Conform Bottom – Conformity to underlying surface decreases toward the top of the boxes.



Conform Base – Only the base of the boxes conform to the surface.

Max Adjust (ft)

The maximum distance to allow the surface being distributed over to drop from under a replicant or rise up into a replicant.

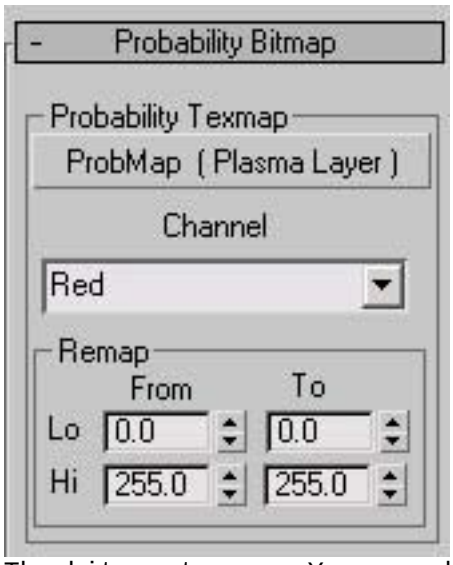
Surf Offset Min/Max

Specifies a range to offset replicants over the surface onto which they are being distributed. For example, Offset Min and Max of 10.0 and 15.0 respectively would distribute the replicants in a layer five feet thick starting ten feet off the ground.

Probability Bitmap

The Probability Bitmap allows distributions to be hand painted onto a surface. The bitmap won't be exported, so it can be any size convenient. Unless you already happen to have a convenient UV mapping, you should apply the mapping to one of the upper channels which Plasma otherwise ignores.

Probability Texmap



The bitmap to use. You can drag and drop from the Material Editor here.

Channel

Which color or combination of colors from the bitmap to base the distribution on. This allows the same bitmap to convey different distributions.

Remap

Allows remapping of bitmap values into distribution probabilities. For example, to invert a bitmap distribution, leave From Lo and Hi at 0 and 255 respectively, but swap To Lo and Hi to 255 and 0 respectively.

LOD Fade

The Distributor generally distributes more polygons into your scene than you can afford to render all the time. You can use the LOD Fade parameters to limit how far away you can see your replicants, and how quickly they blend in. All parameters are in feet.

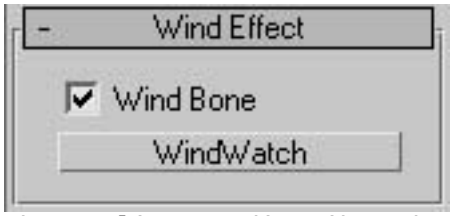
Fade Out

Start/End – Replicants will be fully visible up to Start distance from the camera, and then fade out gradually becoming fully invisible at End distance from the camera. Set Start and End to zero to disable.

Fade In

Checkbox – Check this check box to enable the Fade In.
Start/End – Replicants will be fully invisible closer than Start distance from the camera. They will gradually fade in becoming fully visible at End distance from the camera.

Wind Effect



The replicants distributed can optionally be affected by wind through hardware skinning. The Wind Bone can either be a manually animated object or, preferably, animated via the Wind Component described below. See also the Flexibility Component description below for controlling the effect of the wind on individual replicants.

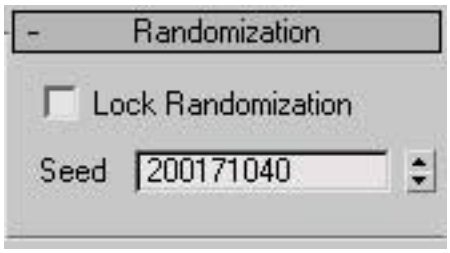
Wind Bone Checkbox

Check this check box to enable selection of a wind bone. If unchecked, any selected Wind Bone is ignored.

Wind Bone Select Box

Click on the select box and select the object to use as a Wind Bone by mouse picking in the scene or through the select by name dialog.

Randomization



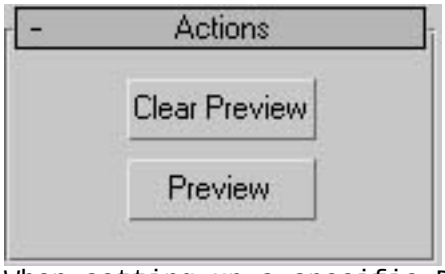
Lock Randomization Checkbox

Once randomization is locked, the system will attempt to generate the same set of replicants each time they are distributed. Keep in mind that changing the distributor parameters or large changes to the underlying geometry will prevent the same set of replicants from being generated.

Seed

The random number seed currently used.

Actions



When setting up a specific Distributor, you can use the Preview function to get a quick rough idea of what your current settings will generate. The output of the Preview should be considered a rough draft and not exported. Keep in mind that interactions between different Distributors (e.g. through the Spacing/Separation) may cause somewhat different output than generated through Preview.

Clear Preview

Clears anything generated from Preview.

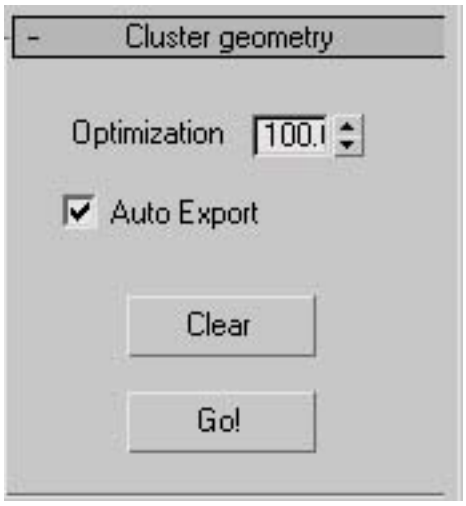
Preview

Runs the Preview generation.

CLUSTER

Cluster – Attach to Base Surface Mesh

The Cluster Component runs the Distributors, collects their output, and optimizes it. There can be only one Cluster Component on an object, but the same Cluster Component can be on many surfaces. It is attached to the base mesh(s) of the surface(s) over which you will be distributing replicants.



Optimization

Level controls how much optimization is performed. It should generally be left at 100%.

Auto Export

The output of the Distributor system can be large and unwieldy. You generally don't want the output cluttering up your Max file and slowing things down. If this box is checked, then at export the Cluster Component will check to see if it has been cleared. If so, it will automatically run all its Distributors before the export process and clear them back out again after the export is done.

Clear

Clears all the distributed replicants generated by this Cluster Component out of the Max File.

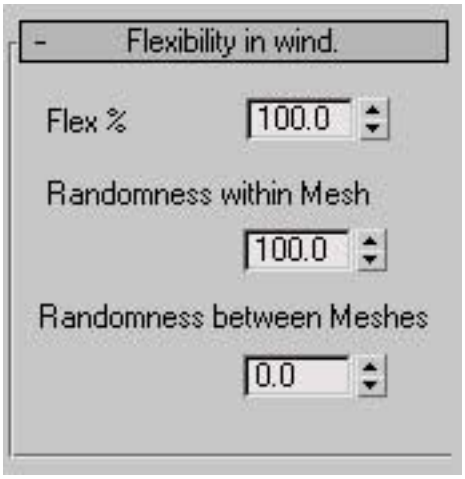
Go

Finds all Distributors attached to all Max objects this Cluster Component is attached to, runs them, and optimizes the output. The optimized output will appear as objects prefixed with the Cluster Components name, and be available as a selection set of the same name as the generating Cluster Component.

DISTRIBUTOR

FLEXIBILITY

Flexibility – Attach to Replicant



Flex %

The overall flexibility of this mesh. The Flexibility Component is typically attached to the replicant meshes that a Distributor with a Wind Bone will be distributing. A Flexibility Component currently has no effect without a Wind Bone specified on the Distributor.
A flexibility of 100% will have the top of the mesh move in unison with its Wind Bone, while the base of the mesh remains planted. This is generally more than you want. Values are typically around 30%. A very small value will keep the mesh from swaying at all, base and top remaining fixed.
A value of zero (0) is treated differently. For a flexibility of zero, the mesh will no longer bend, but the base will no longer remain fixed. The entire mesh will move as a unit with the Wind Bone. So to make a rigid plant, use a flexibility of something like 1%.

Randomness within Mesh

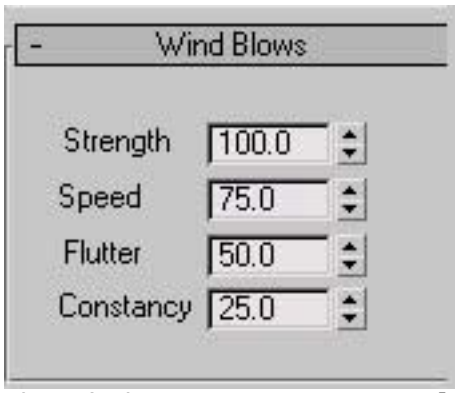
This governs how much variation within a mesh there is in reaction to wind. With a value of 0%, the replicant will bend uniformly with the wind. With a value of 100%, different parts of the replicant will bend different amounts.

Randomness between Meshes

The amount of variation in bend amount between different replicants. With a value of 0% (and 0% Randomness within Mesh), all replicants will bend in reaction to the wind in unison. With a value of 100%, each replicant will bend a slightly different amount.

WIND BONE

Wind Bone – Attach to Dummy Wind Representative



The Wind Bone Component applies a procedural wind effect to an object, making it suitable for use as a Wind Bone in a Distributor. Wind Bones may be cascaded. Typically, you would set up a hierarchy of bones, with Parents having more Strength and Children having more flutter.

Strength

Overall strength of the wind effect, as a percentage from 0% to 100%.

Speed

The general rate of smooth oscillations, with lower values oscillating slower, and higher values oscillating faster.

Flutter

Flutter or twitchiness of the wind. Higher values will put more flutter into the wind.

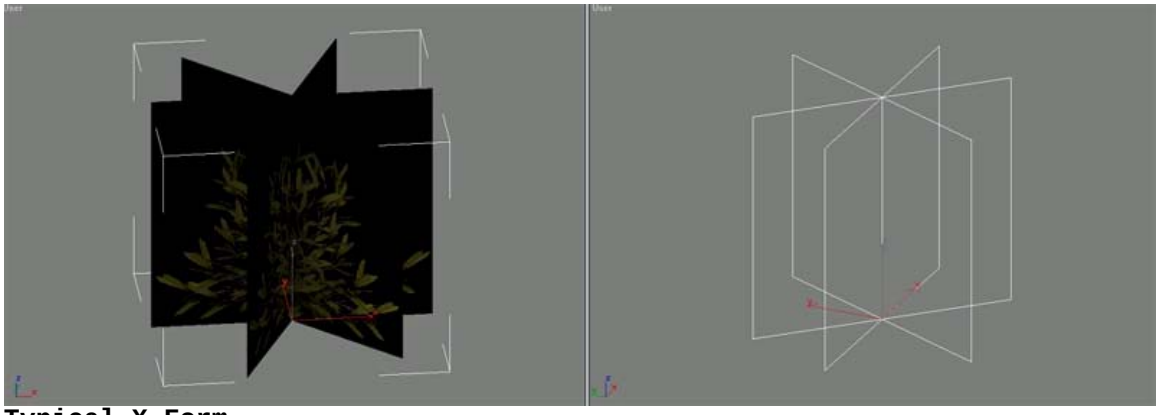
Constancy

With a constancy of 0%, the wind will blow for a while, then completely die, then blow some more. With a constancy of 100%, the wind will always be blowing (although it will blow with varying amounts).

X-FORM

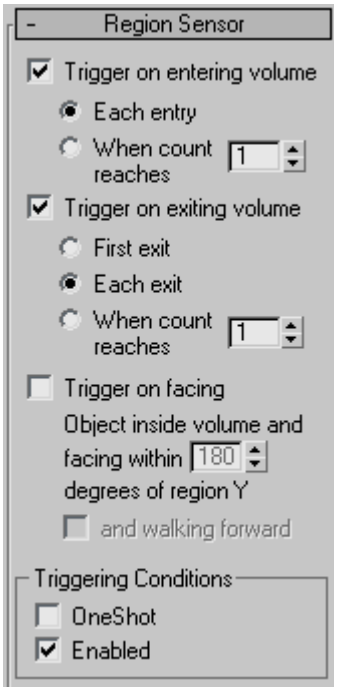
X-Form Component – Attach to Replicant

The X-Form Component is attached to replicants to let the system know that that’s what they are. This lets the system shade them differently, preventing shading artifacts from giving away the trick. The X-Form Component currently has no options.



Typical X-Form

REGION SENSOR



✓To each region, attach a **Region Sensor component** and be sure it triggers on entering and on exiting the region.

***NOTE**: The avatar is considered "inside" the region if its collision hull intersects the region in any way. So the timing of enter/exit triggers depends on this. If you have two regions right next to each other, it is possible for the avatar to be in both (In this case the footstep code will assume the proper sound surface is the region you entered last.)*

RESPONDER

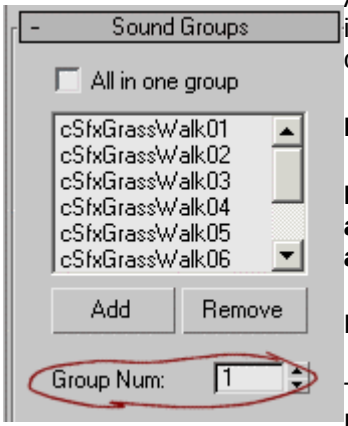
✓Also to each region, attach a **Responder** that specifies the Region Sensor as the Detector.

✓Click the Add... button in the State property and choose the Footstep Surface state.

Now when the avatar enters the region, it will switch to sounds for that surface. When you
leave a region, the avatar goes back to whichever surface it had previously. This way you can
nest regions inside others, like a small puddle inside a large room with a stone floor.

- All footstep sound is setup on the Global Male avatar and its animations, so the scene must export a male avatar. No one but the sound designer or avatar artist needs to make adjustments to footstep sounds.
- Production's part in footstep management is on the surfaces where the avatar will walk and the look of the prints.
  At runtime, players see footsteps generated only while they are present. No footsteps made prior to linking in will not be seen.

Footsteps, the sounds



All the footstep sound work is done for you on the Global Male and its animations. This gray portion of the doc is for your information only.

Footstep Sounds: the avatar footstep object

NOTE: All other avatar types get the footstep data from this avatar. This means you will not hear footstep sounds on other avatars unless you export the male avatar.

FootstepSoundObject

The male avatar MAX file includes a single dummy object named FootstepSoundObject. Attached to that object are all of the footstep sound files (each is a **3D Sound component**) plus the **AvatarFootstepSound component**. *As more footstep sounds are needed, they will be added to this object by to the sound designer or the avatar artist*.

Random Sound Emitter Component

Also on the *FootstepSoundObject* are Random Sound Emitter components, one for each group of footstep sounds. The purpose of these components is to randomize the sounds in each group so the stepping sounds are more realistic.

The groups are formed by adding individual sound files to the Sound Group rollout that's associated with the Random Sound Emitter component. The Random Sound Emitter is attached to the *FootstepSoundObject*.

The footstep code associates footstep animations with Sound Groups according to this list:

Group Num 1: walking
Group Num 2: running
Group Num 3: turning (in place)
Group Num 4: landing
Group Num 5: swimming

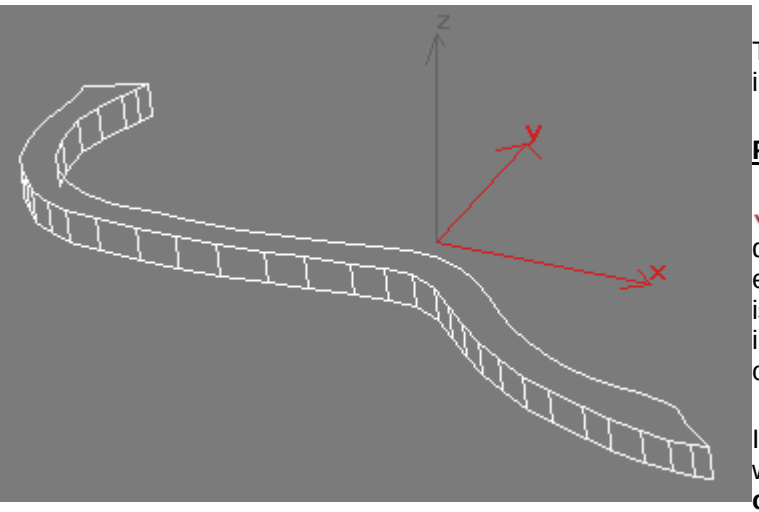
So, when a walk animation is in progress, the sounds listed in group 1 are randomly played.

Footstep Sounds: walk/run/etc animation files

On each of the global male avatar animation files, the notetrack in TrackView specifies the start and stop positions of the animation. For example, in the bas.animations.male.walk.max file, the notetrack for the walking animation includes "SndLeftFootDown@marker" and "SndRightFootDown@marker" on the spots in the animation where a footstep sound should occur. (The landing animation can just use one marker or the other.)

At runtime, whenever the avatar hits these points in the animation, it will play a random sound from the appropriate group.

Footsteps, the scene



The next step is to specify the surface sounds to play in the different locations throughout the scene.

Footstep Sounds: region sensor & responder

✓In the scene file, where the avatar will be walking, draw simple geometry to define a region volume for each footstep type. For example, in the Garden there is a region for the sandy path that leads from the link-in point, another region for the stone walkway and others for the grassy areas.

In the Garden, the region shown at left defines a dirt walkway. Attached to this object is a **Region Sensor component** (see below) that is set to trigger upon entering and exiting the volume.

Also attached is a **Responder component** (see below) that is set to respond to the trigger with a Foot Surface(Dirt) state.

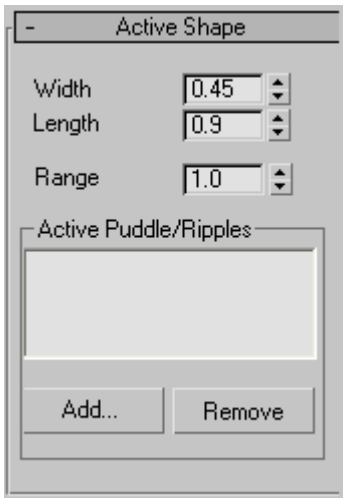
Footsteps, the prints

The final step is preparing the geometry on which the prints will show and setting up the actual print textures. This is where you'll use the FootPrint components. There are five (at this writing). The component code makes the prints conform to the contours of the surface on which they are placed – without any effort on your part.

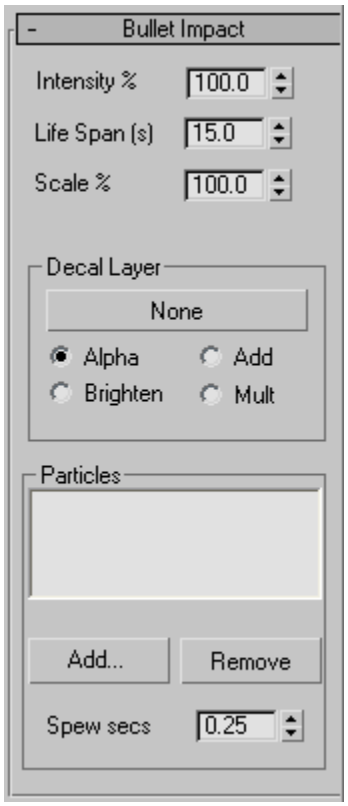
Preparing the terrain

✓To the path where you want the prints to show, attach one of the following footprint components (Active Shape, Dirty/Wet, Foot Print, Print Shape, Puddle, Ripple):

ACTIVE SHAPE



BULLET



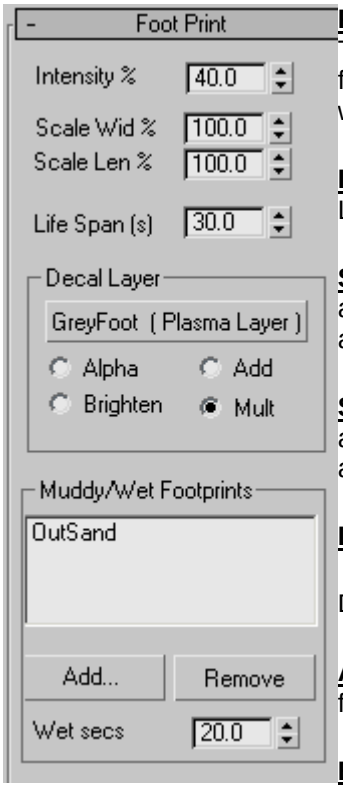
DIRTY/WET REGION

Dirty/Wet – This footstep region component specifies the footprint type that will show on the terrain when walked on.

Attach a Dirty/Wet footprint component to the terrain geometry.

Click Add... to access a list of already-defined footprints, then select the one you want. The footprints are defined

The default amount of time for the footprint to show before fading is 10 seconds. If you want it to be longer or shorted, alter this parameter.



Foot Print – Attach this component to the terrain geometry object(s). The Decal Layer property specifies the texture to be used as the footprint image. The other properties modify the footprint in various ways.

Intensity — Tones down the effects of the settings on the Decal Layer properties.

Scale Wid(th)% — Scales the width of the print size relative to the avatar's foot size. 100% keeps the print the same width, 200% leaves a print twice the width of the foot.

Scale Len(gth)% — Scales the length of the print size relative to the avatar's foot size. 100% keeps the print the same length, 200% leaves a print twice the length of the foot.

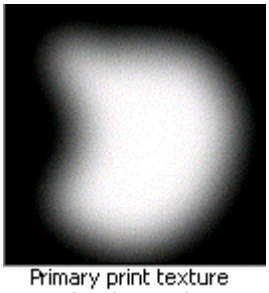
Life Span(s) — The length of time the footprint will remain visible.

Decal Layer

Alpha – Alpha blends the texture with the terrain texture based on the footprint material's alpha channel.

Brighten – Multiplies the footprint texture color with the terrain color, resulting in subtler footprints, such as in dry sand. (This increases the lighting of already lit textures but does not add

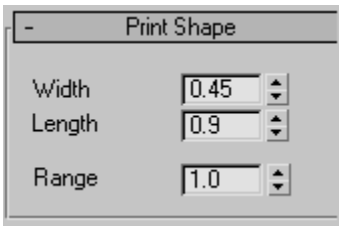
Add – Adds all colors in the the footprint texture to the terrain texture, resulting in darker/brighter/stronger footprints, such as mud on a light walkway.


Primary print texture used in the Garden.

Mult – Use this property for prints that should be subtle, such as wet prints on dark stone. For example, the stone walkway through the pond in the Garden uses this property. The wet prints are made with the texture at right.

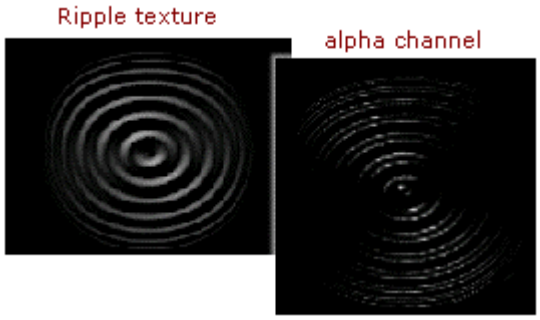
Muddy/Wet Footprints — This property gives you a "chaining" option for footprints. Any footprint specified in this field is enabled for the amount of time in the **Wet secs** field, then is disabled. This is for footprints that when first made, you want a wet looking print that fades, leaving a faint, dry footprint. For example, In the sandy path of the Garden, a footprint called 'inSand' is the primary print. It is a subtle print that's blended with the sand terrain texture. It's Muddy/Wet Footprint property calls for a footprint called 'outSand' which is a sandy decal print that allows the print to be darker for 20 seconds before fading to the primary print.

PRINT SHAPE



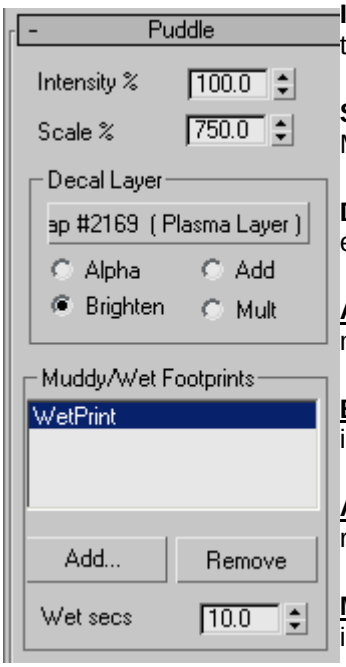
PUDDLE

Puddle – This creates ripples, as on water, from the avatar's feet only. If other body parts need to make ripples (such as when swimming) , use the Ripple component. (It uses lots more engine than Puddle.)



This is the ripple texture used in the Garden (test version) where the avatar walks over the surface of the pond and leaves ripples as footprints. It is referenced by the Puddle component, in the Decal Layer property, as seen in the rollout image below.

The **Puddle component** is attached to a simple piece of geometry that defines the borders of the area (the pond in this example). *Plasma generates prints from the avatar's feet only when this component is used.*



Intensity % – Tone down the effect of the print. The default is 100% of the decal's original intensity.

Scale % – The size of the print in relation to the size of the avatar's foot. Maximum is 10,000%.

Decal Layer – Select the texture to use as the print decal. In this example, the ripple texture shown above was used.

Alpha – Alpha blends the texture with the terrain based on the footprint material's alpha channel.

Brighten – Multiplies the texture color with the terrain color, usually increasing the intensity of the footprints.

Add – Adds all colors in the the footprint texture to the terrain texture, resulting in darker footprints, such as mud on a light walkway.

Mult – Use this property for prints that should be more subtle, such as in sand.

RIPPLE


Ripple
Intensity %
100.0
Scale %
100.0
Decal Layer
None
Alpha
Add
Brighten
Mult
Muddy/Wet Footprints
Add...
Remove
Wet/Dirty secs
10.0
Particles
Add...
Remove
Spew secs
0.25

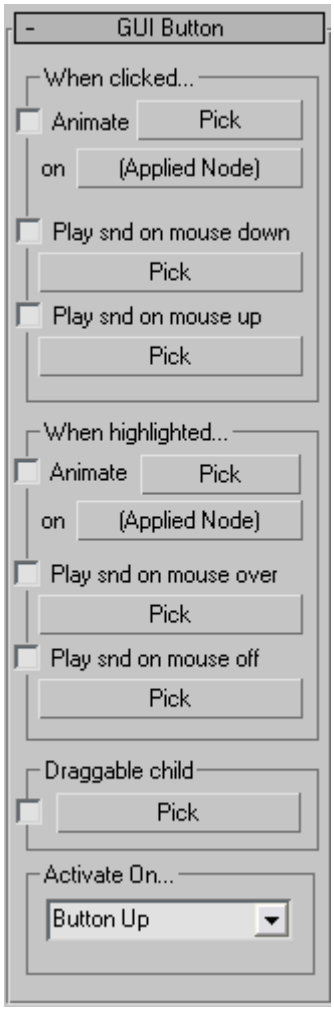
WAKE


Wake
Intensity %
100.0
Scale Wid %
100.0
Scale Len %
100.0
Decal Layer
None
Alpha
Add
Brighten
Mult
Muddy/Wet Footprints
Add...
Remove
Wet secs
10.0
Particles
Add...
Remove
Spew secs

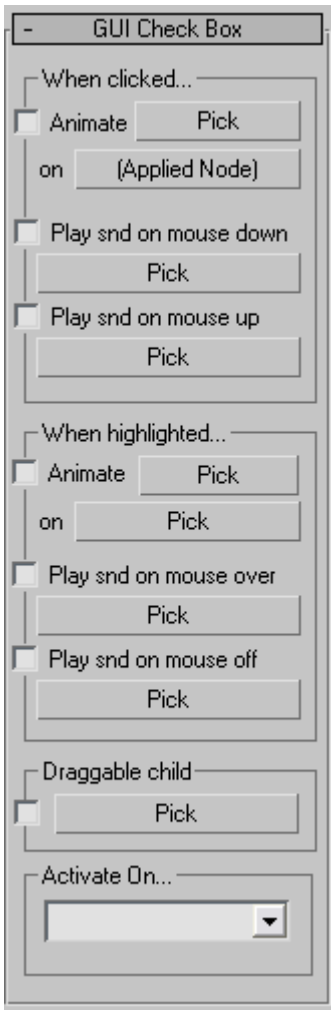
WATER BULLET

GUI

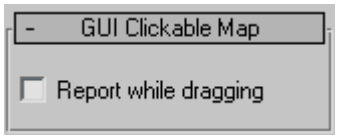
GUI BUTTON



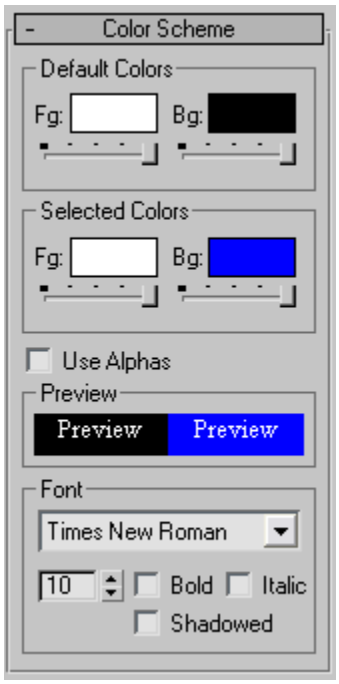
GUI CHECKBOX



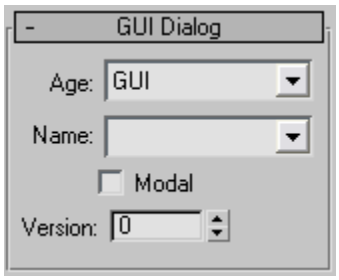
GUI CLICKABLE MAP



GUI COLOR SCHEME

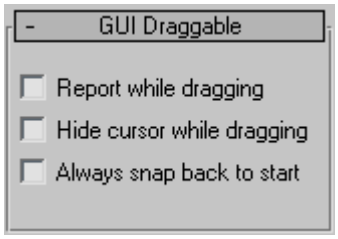


GUI DIALOG

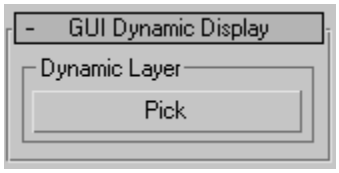


GUI DIALOG DRAG BAR

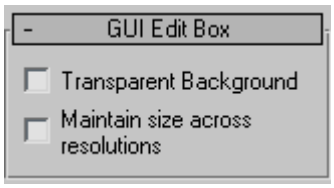
GUI DRAGGLABLE



GUI DYNAMIC DISPLAY

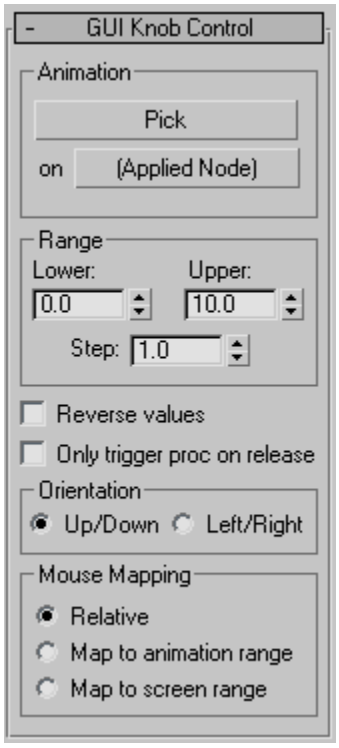


GUI EDIT BOX

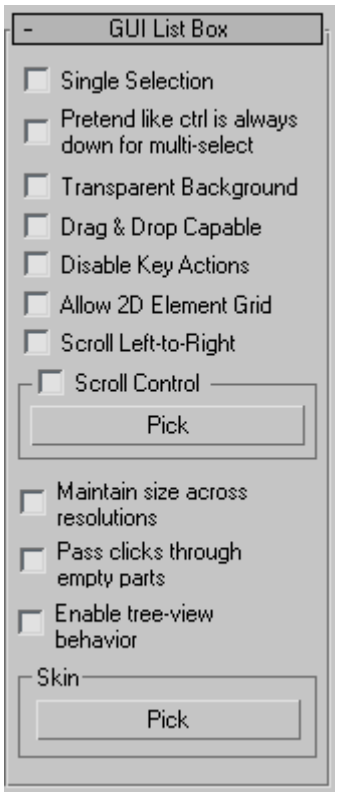


GUI ID TAG

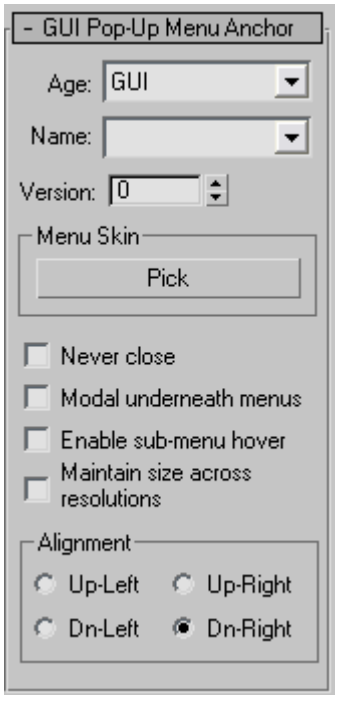
GUI KNOB CONTROL



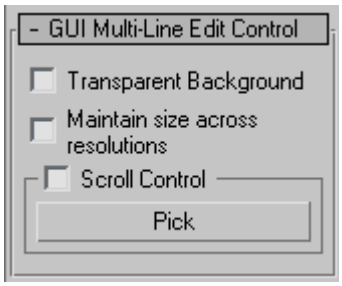
GUI LIST BOX



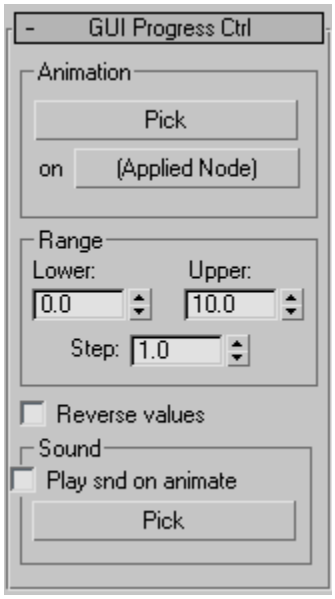
GUI MENU



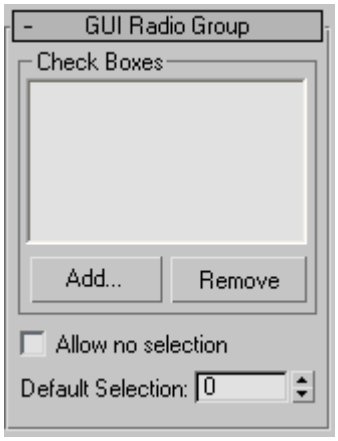
GUI MULTI-LINE EDIT BOX



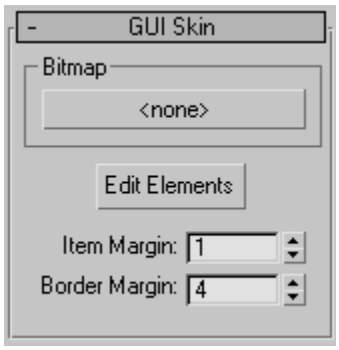
GUI PROGRESS CONTROL



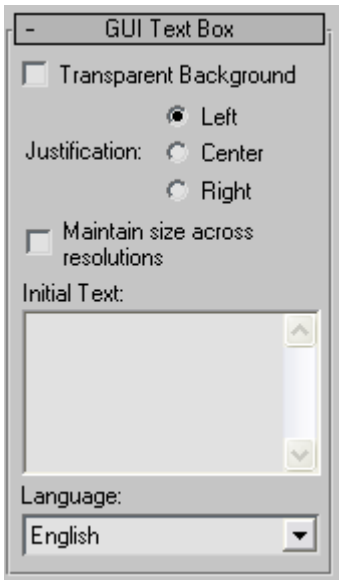
GUI RADIO GROUP



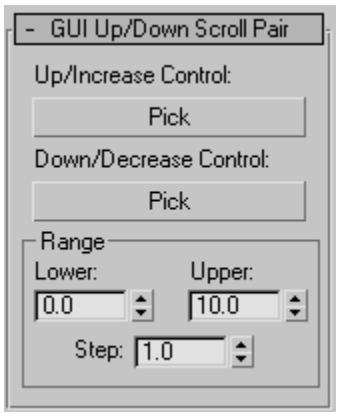
GUI SKIN



GUI TEXT BOX



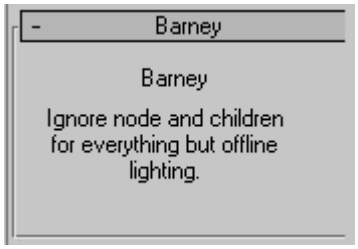
GUI UP/DOWN PAIR



IGNORE

BARNEY

Imagine how a shaded light effects a subject in a photography session. A panel attached to the side of the light is used to adjust the shadowing on the subject. That panel is called a "barndoor" or barney.

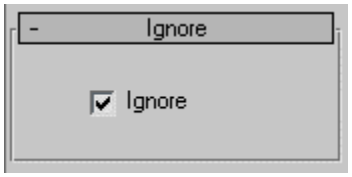


This component has the same kind of effect on Plasma runtime lights (offline lighting). Attach the component to a shape that defines the kind of shadowing you want. The object is not drawn (nor are any child objects); it occludes all runtime lights, throwing a shadow in the shape of the object.

For example, the walls of the garden, being single-sided, have a barney object attached to the outside and over the top. The barney occludes the RT lights as though the walls were double-sided. Using the Barney is far less expensive than making the wall double-sided.

CONTROL MAX LIGHT

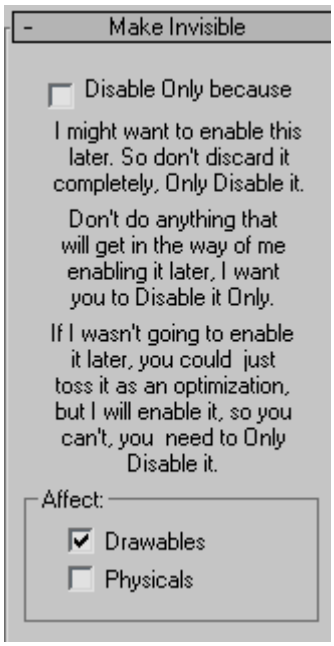
IGNORE



Attach this component to an object so the engine will not export it.

The checkbox is checked by default.

NOSHOW



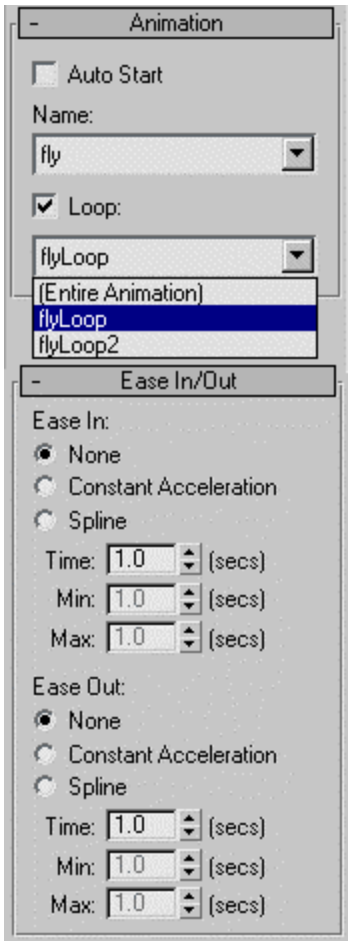
MISC

ANIMATION

- Every animation must have an Animation component attached.

Every animated object must have an Animation component attached to it, but simply attaching the component isn't usually enough to accomplish what's needed for gameplay. You often need multiple animation options on a single object. Plasma 2 makes use of the Max Note Tracks feature where you define pairs of start and end key points. Since you can define multiple pairs of start/end points, you can define segments of a single animation, effectively defining multiple animations on a single object. You will reference Note Track segments for animation from the various Responder components, as well as from the Material Editor panel.

Animation Component example



Auto Start - Check this to start the animation upon game start. Be aware, however, that doing so might mean that your player never sees the animation, since in the Parable world, they might not enter this location when you think they will. It's better to activate the animation with an Detector/Responder.

Name: Select the animation key name from those currently defined in Note Tracks for this object. This is the point in the animation where it'll start. The key is located in Max Note Tracks and must have a start/end pair. For example, the pair *fly@start* and *fly@end* combine to define an animation of the name *fly*, which is what you'll see in the Name dropdown list. If you want the whole animation to play, leave "(Entire Animation)" in the field.

The start key name is of the form *animName*@start. (or @*begin*)
The ending key name is of the form *animName*@end.

More information on Note Tracks →

Loop: Check to activate animation looping.

From the dropdown list, select the animation segment name that defines the start and end of where you want to loop through the animation segment. To loop through the whole animation, leave "(Entire Animation)" in the field.

The key names for looping segments are of the form *animName*@startLoop or *animName*@beginLoop. The ending key names are of the form *animName*@endLoop. (***NOTE****: The name you give the looping segments must be different than the segment name, e.g., birdCirle@begin and birdCircleContinuous@beginLoop, not birdCircle@beginLoop.*)

ADDED 8/02 – Added a checkbox to the animComponent which is disabled unless the node, or some descendant, has a physical component on it. The checkbox specifies whether the animated physical can apply force to other objects (and thus we have to do extra work to update its velocity,etc during the animation). It's on by default, so that things behave as before, but it can be unchecked for things that don't need the extra computation (most notably, animated GUI components that are physical because they're clickable.

Ease In/Ease Out

These parameters allow you to define a smooth start and stop of the animation sequence, for use when the player is expected to view the starting and stopping of the animation. e.g., climbing into the cleft.

None - The animation jumps to full animation speed upon start. (The player may see jerkiness.)

Constant Acceleration - This parameter tells the engine to calculate an acceleration speed based on the time value you enter in the Time field.

Spline - This parameter is very similar to Constant Acceleration, but uses a more complicated calculation to determine acceleration (and is, therefore, more expensive performance-wise). Use this parameter only after trying Constant Acceleration first. If it's still not smooth enough, use Spline.

Time (0 – 99) - This is the time value (in seconds) used to calculate the ease in or ease out time.

Example

Let's say you have a bird object that you want to fly circles around a tree under some circumstances OR to eat an insect from the top of the tree at another time. (These two animations are intended to be used independently of each other, yet they animate the same object.)

1. Animate the bird object to fly a circle around the tree.
2. Note the ending key frame number.
3. Continue animating it to pick an insect from the top of the tree.
4. Now open Note Tracks. Create a Note Track for the bird object and place keys that mark the beginning and end of the circling animation and the beginning and end of the insect eating sequence.
5. Right click each of the keys and type into their successive text panels: *birdCircle@begin*, *birdCircle@end*, *insectEat@begin*, *insectEat@end*. Also, since the circling behavior should be a looped animation, add two more keys: *birdCircleContinuous@beginLoop* and *birdCircleContinuous@endLoop*. (*NOTE: The name you give the looping keys must be*

different than the segment name, e.g., birdCirle@begin and birdCircleContinuous@beginLoop, not birdCircle@beginLoop.)

6. In the Component Manager, attach an Animation component and name it *birdCircle*.

7. Attach a second Animation component named *eatInsect*.

8. Now, with the bird object still selected, open the Component Utility (Max Utilities -> More -> Component Util) and select the birdCircle component from the list. (The bird object must be selected.)

9. In the Animation parameters panel, select *birdCircle* from the dropdown list labeled 'Name'.

10. (Optional... when the player will see the beginning and ending of the animation) In the **Ease In** panel, select **Constant Acceleration** and enter a time of 20. This defines a smooth acceleration from stopped to full speed. Without this, the animation jumps to full speed. You can also define a deceleration speed for a smooth transition from full speed animation to stopped.

11. Since this should be a looped animation, check the Loop checkbox, then select *birdCircleContinuous* from the Loop dropdown list. This tells Plasma where the loop starts. It uses the endLoop note key as the end of the loop.

12. Back up at the Selected Obj list, select the *insectEat* component.

13. Back down to the Animation parameters panel, select the *insectEat* segment from the dropdown list labeled 'Name'. This one is not looped, so don't check the Loop checkbox.

NOTE TRACKS

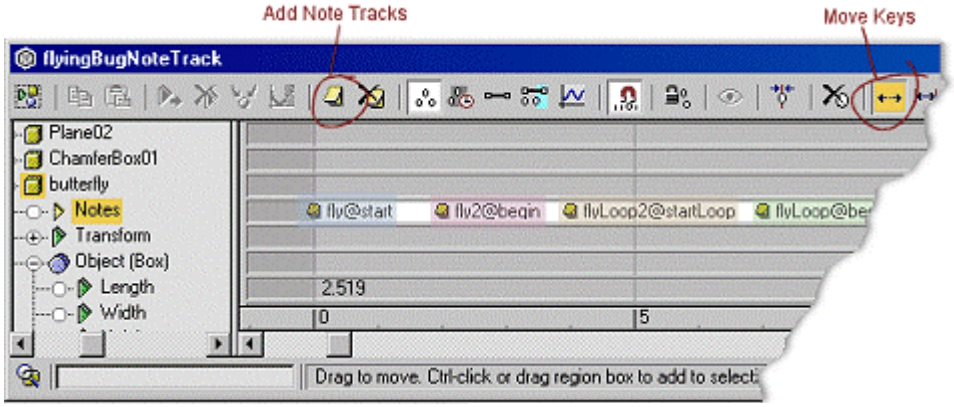
Note: *Complete help is included in your Max help file (3DSMax.chm). Search on 'Track View'.*
If you don't have that file installed, you can download it from here. It's 25M in size.Using Note Tracks

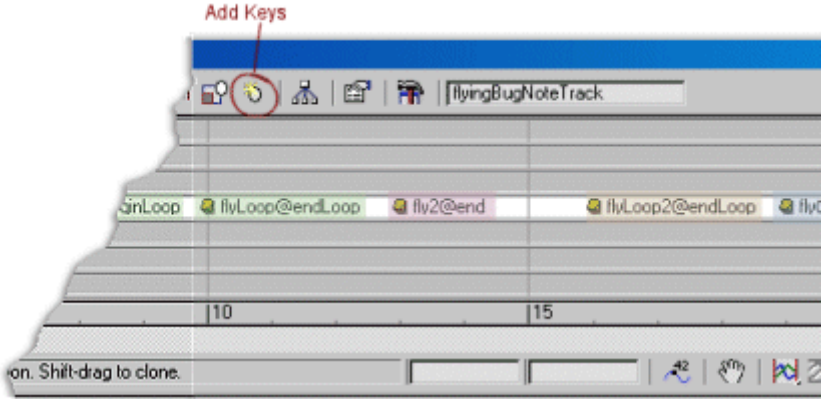
When adding a Note Track for animating, be sure to add it to the proper object. i.e, if you are animating a material, add the Note Tra material, not the object on which it might be applied.

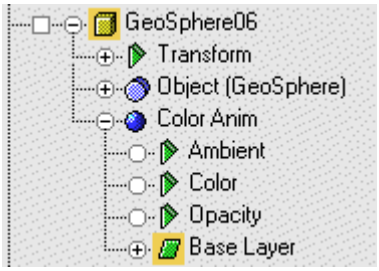
1. *Access Note Tracks by clicking the Track View button in the Max menu.*


Track View Button in Max

2. *In the object tree, select the object you are setting up. In this example, the an butterfly is the object. (Click the object's name, not the icon.)With the object highlighted, click the Add Note Tracks button. A Notes line is added in the objec tree.*







Note: The yellow highlighting indcates the currently selected object or parame example, this graphic shows that GeoSphere06 is selected. And, its material B selected in the Material Editor.

3. *Add keys to the points in the animation where you wa define start and end points.*

(You can move these easily with the' Move Keys' function.)

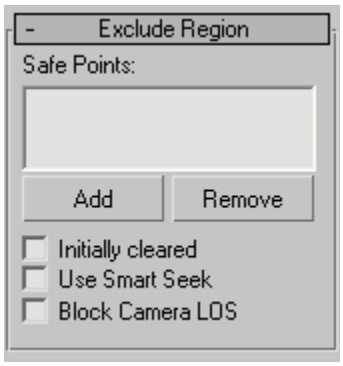
4. *Define each key by right-clicking it to open a dialog, then typing the key name. Follow this style for naming the keys:*
    start point: name@start
    end point: name@end
    start loop point: name@startLoop
    end loop point: name@endLoop
    single marker: name@marker

5. *Repeat step 5 for all the keys.*

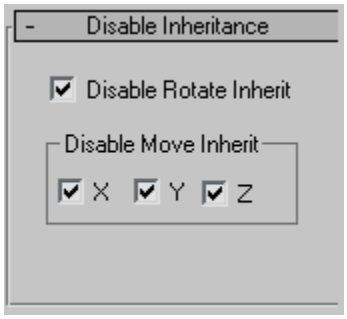
IMPORTANT: *You must attach to the object, a unique Animation component for each start/end pair. For example, if the butterfly object needs a 100 foot flying animation and a 50 foot flying animation, you would create keys in the Notes Track that define the 100 foot sequence (named fly) and another one for the 50 foot sequence (named fly2). Now, to the butterfly object, attach two Animation objects, one for the 100 foot version and the other for the 50 foot version. From the Animation component UI for each component, select the appropriate segment name.*

ANIMATION GROUP

EXCLUDE REGION



FILTER INHERIT



FORCE CLICK 2 TURN

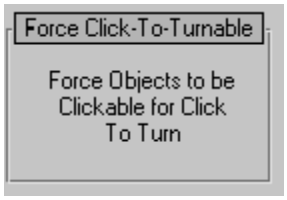
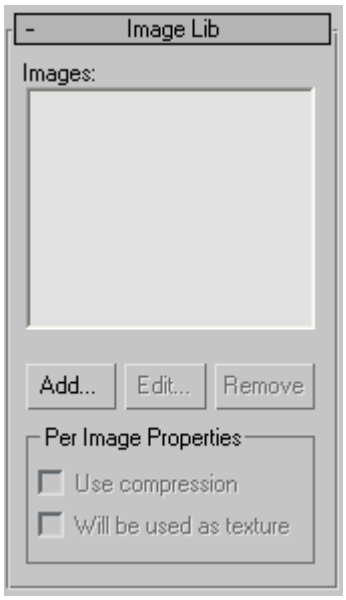
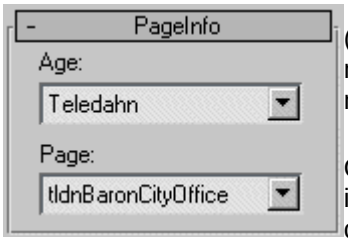


IMAGE LIBRARY



PAGE INFO

- Attach the PageInfo component to every object in the scene to associate it with an Age Description file. Choose Age and Page in the component parameter panel.
- The Age Description file must already be defined and checked in to AssetMan, so it can be selected via the PageInfo component.
- Every object requires this component and can be attached during export.



The PageInfo component sychronizes with the Age description file (.age) in a scene to enable efficient data loading and proper multiplayer behavior. (The Age description file is created and maintained with the Age Description Manager exclusively.)

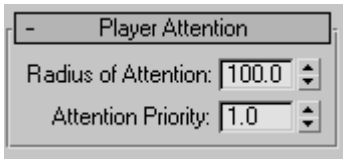
Once a PageInfo component is established in a scene (by attaching it to an object), you can export without manually assigning the component to every object. During export, a dialog comes up that lists all existing Page Info components and lets you assign them to objects from there.

Once you have attached the Page Info component to an object, select the Age and Page properties for the selected object(s) from this property panel. You cannot create them here; they are created with the Age Description Manager (Plasma menu).

The Ages are assets in AssetMan. The Pages are logical groupings of data within the Age. So the tldnBaronCityOffice page defines a logical set of data within the Teledahn age. The Baron City Office is a seperate .max file but its Teledahn Age designation connects it with the other Teledahn scenes.

PERSISTENCE

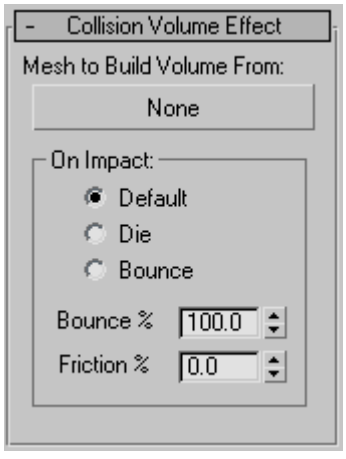
PLAYER ATTENTION



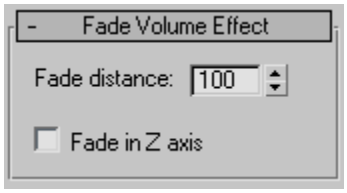
REFERENCE POINT

PARTICLE SYSTEM

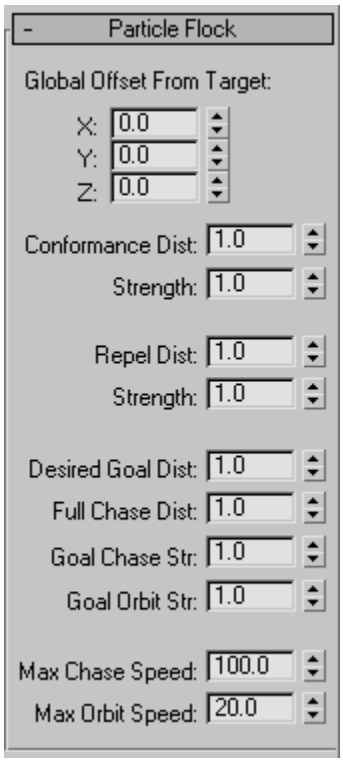
COLLISION VOLUME EFFECT



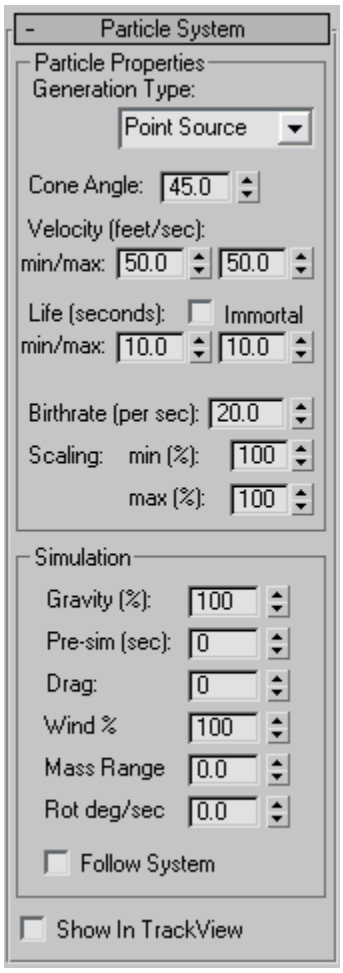
FADE VOLUME EFFECT



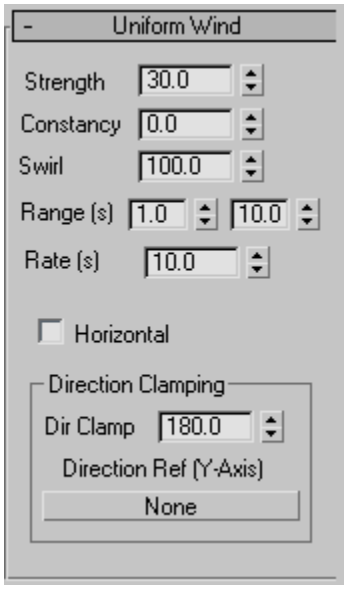
PARTICLE FLOCK



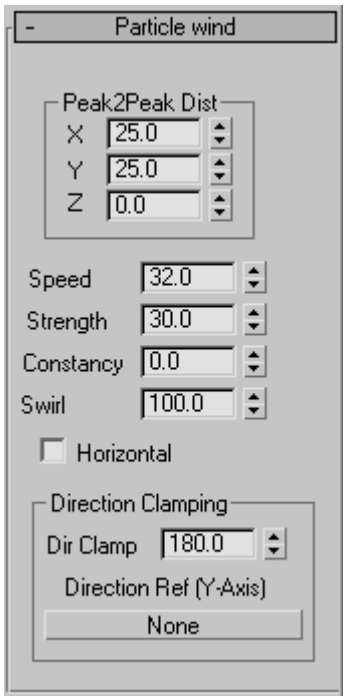
PARTICLE SYSTEM



UNIFORM WIND

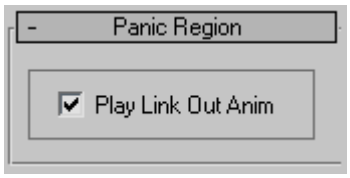


WIND EFFECT



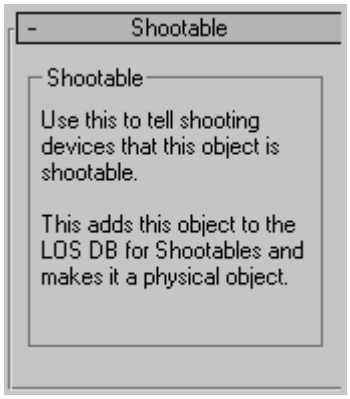
PHYSICS

PANIC LINK REGION



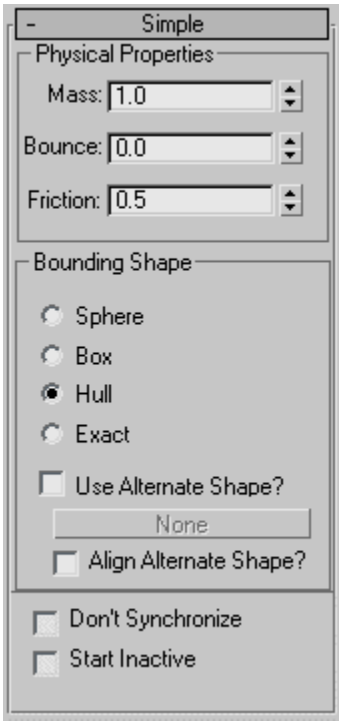
When the player comes in contact with a panic link region the player will be transported back to the beginning of the age. This should only happen if the player falls through the world geometry. In other words, this should happen on an abnormal situation.

SHOOTABLE



This component is for objects that move and collide with other objects. This includes a movement response when collided with, for example, a bush that sways when an avatar brushes by.

Two properties have been added (10/02) that help streamline an Age. 'Don't Synchronize' and 'Start Inactive' give the engine instructions about physical objects that can save a lot of processing and bandwidth.



Mass - The object's weight in pounds (lbs). This value should be, based on real world weights. The default is 1.0 since this is a required property for any moving object.

Bounce - A measurement of the object's hardness (0.00 - 1.0). This determines how objects that collide with it will rebound. E.g., a rubber wall would have very high bounce (.90), while a brick walkway would have almost none (.01).

Friction - A measurement of how slippery the object's surface is (0.000 - 1.000). e.g., a greasy floor would have very little friction (.001) while a sandy beach would have high friction (.8).

Bounding Shape - Required for every physical object, except detectors. Creates a bounding box that is its detector (or collision) shape.

***TIP**: The bounding shape encloses the object which adds to the object's overall size. Ensure that movable objects sit enough above the ground plane so the bounding shape does not break the plane. Otherwise, the object will fall through the ground during runtime.*

Sphere - Encloses the object in the smallest possible sphere.
Box - Encloses the object with the smallest possible box.
Hull - Encloses the object with a "saran wrapped" bounding shape. (It levels out high points in the

surface.)
Exact - Use the object itself as the bounding shape.

TIP: While you can use an object as its own bounding shape (the Exact option), it's not recommended for performance reasons. However, you can use an instance of the object that has the Max Optimize modifier on it, then use that as the custom bounding shape (Use Alaternate Shape). This keeps performance optimal and updates if you change the main geometry. For final production, consider hand-tuning this geometry.

Use Alternate Shape - To use this option, first draw a simple, low poly version of the object (no dummys). Then, click the button and in the viewport, select the bounding shape you just drew.

NOTE: *By using alternate shape you are telling the engine that this object has no purpose in the scene other than specifying geometry for the collision shape of the original object. The alternate shape will not render, nor will it export as a separate object.*

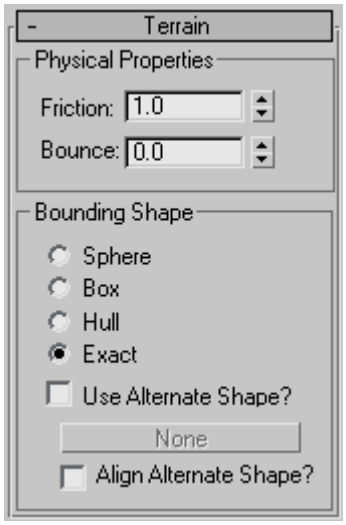
Align Alternate Shape? - A quick way to be sure the alternate shape is aligned correctly with the object it is intended for. At export time, the engine will be sure the two objects are aligned.

Don't Synchronize - Check this property on objects that are purely (or mostly) decorative and are allowed to move differently on different machines.

Here's why: By default, all physical objects are synchronized at load time when run in multiplayer. This is so all players view things the same way. However, for some physical objects this isn't really necessary and would save server processing and bandwidth if those objects were overlooked during synchronization. (For example, leaves blowing around on the ground... it doesn't matter if all players see these the same way.)

Start Inactive - For objects that need to move upon scene loading, leave this property UNchecked. This is so objects such as a rolling rock or falling piano aren't suspended in space.

Here's why: Checking this property freezes the object in its position until it is moved by another physical object or an avatar. This is a good thing to do for objects which are already sitting on the ground at the beginning of a scene. Without this feature, a lot of processing is done to figure out that the physical object is indeed on the ground. Use this component on all objects that are positioned on or very near to the ground in the Max file. It will move normally when the avatar hits it, and you won’t pay for any initialization at launch.



This component is for objects that are expected to be touched (walked on, collided with) but **DO NOT MOVE**. Since this object does not move, Friction and Bounce give the object realistic properties for determining the behavior of a moving object that collides with it.

Friction - A measurement of how slippery the object's surface is (0.000 - 1.000). e.g., a greasy floor would have very little friction (.001) while a sandy beach would have high friction (.8).

Bounce - A measurement of the object's hardness (0.00 - 1.0). This determines how objects that collide with it will rebound (in combination with the object's bounce property value). e.g., a rubber wall would have very high bounce (.90), while a brick walkway would have almost none (.01).

Bounding Shape - Required for every physical object, except detectors. Creates a bounding box that is its detector (or collision) shape.

***NOTE**: The bounding shape encloses the object which adds to the object's overall size. Ensure that movable objects sit enough above the ground plane so the bounding shape does not break the plane. Otherwise, the object will fall through the ground during runtime.*

The Custom selection activates the Custom Bounding Shape button.

Sphere - Encloses the object in the smallest possible sphere.
Box - Encloses the object with the smallest possible box.
Hull - Encloses the object with a "saran wrapped" bounding shape. (It levels out high points in the surface.)
Exact - Use the object itself as the bounding shape..

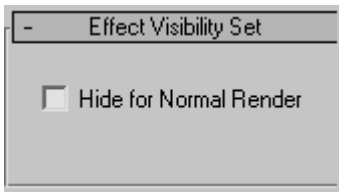
***NOTE:** While you can use an object as its own bounding shape (the Exact option), it's not recommended for performance reasons. However, you can use an instance of the object that has the Max Optimize modifier on it, then use that as the custom bounding shape (Use Alternate Geometry). This keeps performance optimal and updates if you change the main geometry. For final production, consider hand-tuning this geometry.*

Use Alternate Geometry - To use this option, first draw a simple, low poly version of the object. Then, click the button and select the bounding shape you drew.

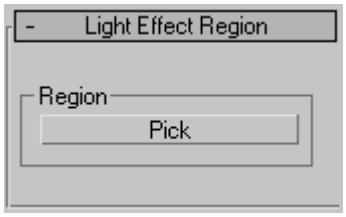
NOTE: *By using an alternate geometry you are telling the converter that this object has no purpose in the scene other than specifying geometry for the collision shape of the original object. The proxy object will not render. Nor will it export as a separate object.*

REGION

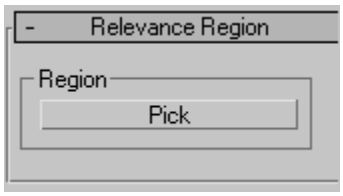
EFFECT VIS SET



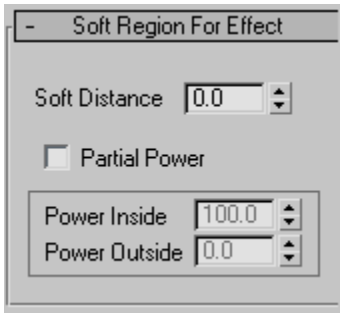
LIGHT REGION



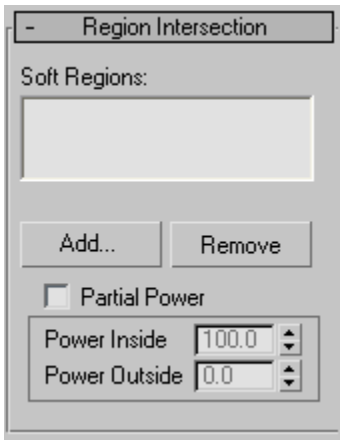
RELEVANCE REGION



SOFT REGION



SOFT REGION INTERSECTION



SOFT REGION INVERTED

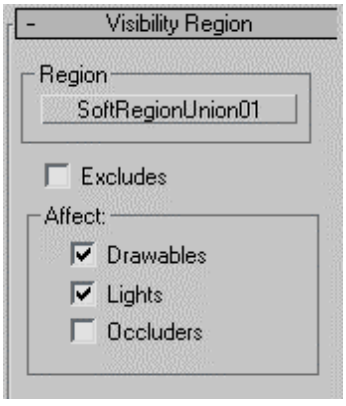
SOFT REGION UNION

VISIBILITY REGION

- When the active camera is inside a soft region to which a Visibility Region component is attached, only those objects with the same Visibility Region component attached will be drawn (or not drawn, depending on the component settings).
- The difference between a visibility region and an occluder is that the occluder blocks visibility of an already drawn object, while the visibility region tells the engine to not draw the object at all.

The Visibility Region component optimizes a scene by specifying the objects to be drawn (or not drawn) when the active camera is within a defined soft region.

For example, from most areas of the neighborhood, the lake and distant city cannot be seen, so those objects are not drawn when the active camera is within the visibility region. However, from some areas like the bridge, DRC walk, and waterfront balcony, the lake is visible. So those areas are enclosed in soft regions that have a different Visibility Region component attached. When the camera is in those regions, the lake and distant city objects are drawn.



1. In your scene, determine which objects must be visible from a particular area and angle.
2. Create a soft region of that viewing area and attach a Soft Region component. Name it accordingly (e.g., softRgnLakeNoVis).

   NOTE: The region can be a single soft region or a union or intersection of regions.
3. Select all the scene objects that are to be visible from inside the soft region defined above. Attach an appropriately-named Visibility Region component (e.g., visRegionLakeNoVis). Be sure to include the runtime lights.
4. Attach the same Visibility Region component to the soft

region.

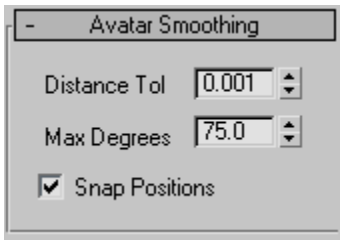
5. Open the Visibility Region component panel. Click the Region button and select the Soft Region you named in step 2.

The **Excludes** checkbox inverts the behavior of the Visibility Region. That is, if checked, the objects with this vis region component attached will NOT be visible (or drawn) when the camera is in the defined visibility region.

Affect - Use these three properties (Drawables, Lights, Occluders) to exclude that group of objects from the behavior of the visibility region. That is, if you have defined an area of a scene as a visibility region but UNchecked 'Lights', the geometry and occluders are not drawn but the runtime lights are still allowed to affect the scene.

RENDER

AVATAR SMOOTH

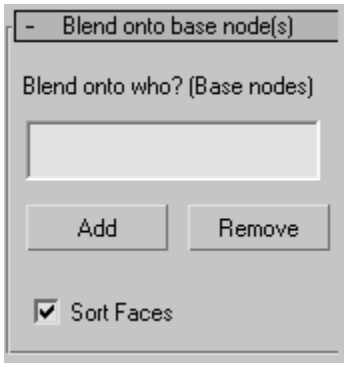


BILLBOARD

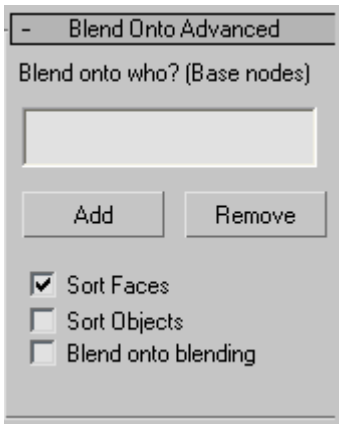
This component has no properties. It specifies an object's Z-axis as always facing the viewer and locked to pivot on the Y-axis. For example, a simple plane with a cloud texture applied with this component attached would look to the viewer like clouds in the sky, regardless of the position from which it is viewed. That's because the Billboard component makes the normal of the object always face the viewer. The object pivots on its Y-axis.

If you need a view-facing object that pivots on more than just the Y-axis, use the Sprite component.

BLEND ONTO

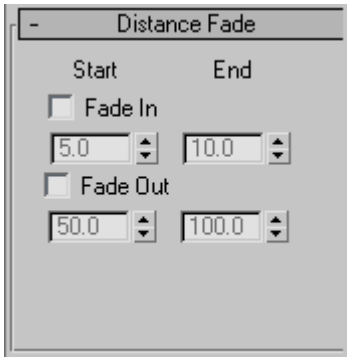


BLEND ONTO ADVANCED



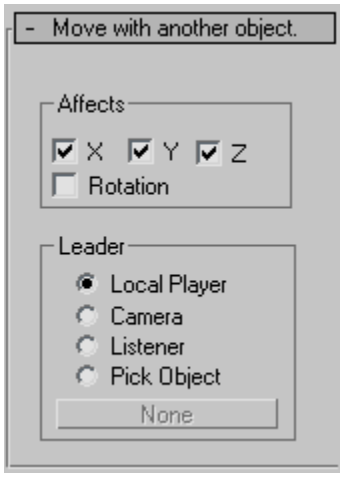
CAMERA VIEW

DISTANCE FADE



DRAW B4 AVATAR

FOLLOW

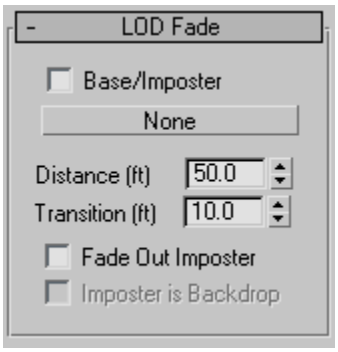


FORCE DYN MAT

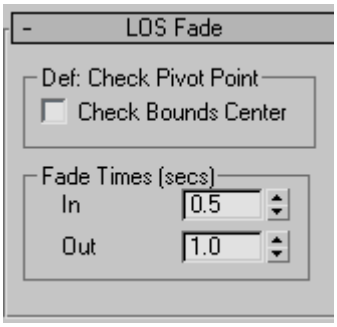
FORCE RT LIGHT

GZ FADE

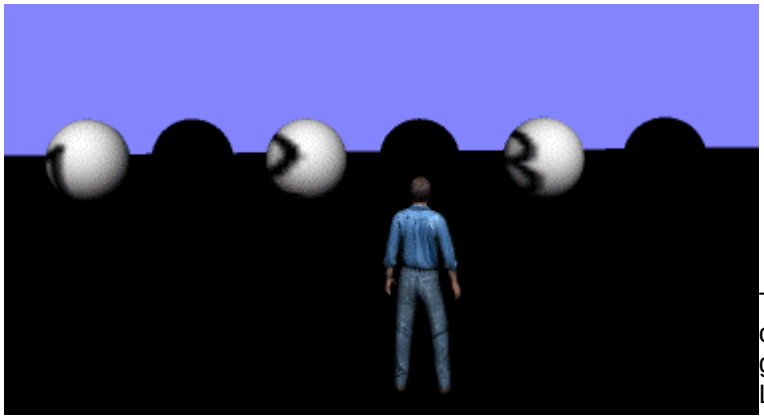
LOD BLEND



LOS FADE

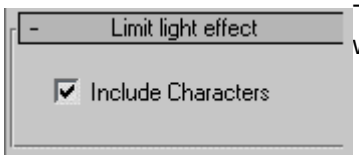


LIGHT GROUP



- Attach a single Light Group component to a Plasma RT light (or multiple RT lights) and the scene object(s) you want lit by that light or lights.
- No other objects will be lit by the particular light or lights.

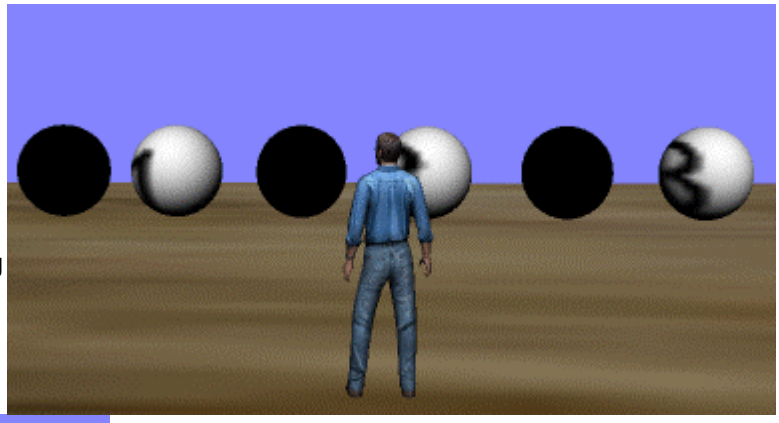
This component lets you selectively light objects with a specific Plasma RT light or group of RT lights. Once an RT light has a Light Group component attached to it, ONLY objects which share the same component will be lit by that light or lights.



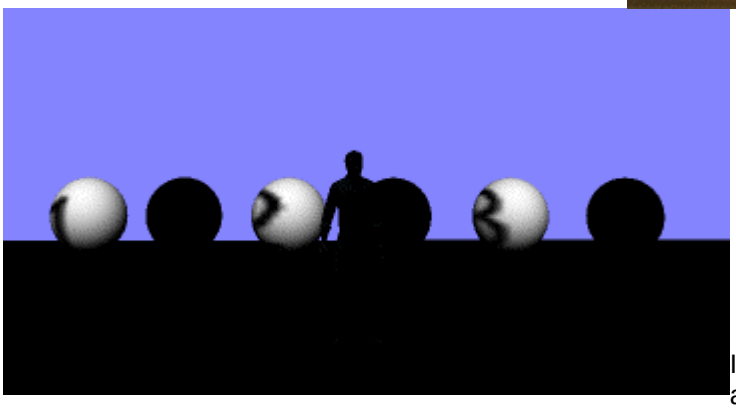
The only parameter is whether or not to light avatars that come within the RT light's range.

Examples:

This sample shows a scene with six spheres and a ground plane, plus the avatar. The scene is lit with a single RT Directional light. The lit spheres and the RT Directional light have a single Light Group component attached. The 'Include Characters' parameter is un-checked.



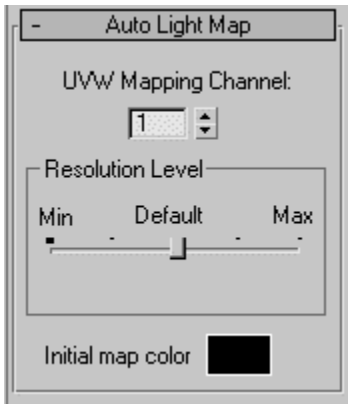
In this version of the scene, the 'Include Characters' parameter is checked, allowing the avatar to be lit by the RT Directional light.



In this final version, the ground plane and the alternate spheres have the Light Group component attached.

LIGHT MAP

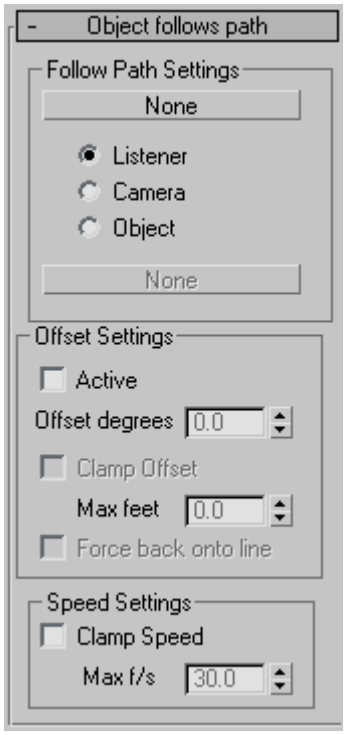
This component lets you control static shadows without using vertex coloring. Based on the scene at export time, the Plasma engine calculates and creates a light/shadow texture layer for the object(s) that have a Lightmap component attached.



On objects that you want the lightmap to appear on, do the following:

1. Generate mapping coordinates.
2. Convert to an Editable Mesh or add an Edit Mesh modifier.
3. Add a UVW Map modifier and select a UV mapping channel for the lightmap.
4. Apply a Plasma material.
5. Attach the lightmap component and select the UV mapping channel you chose for the UVW mapping modifier. You can adjust the shadow's intensity (resolution) and the shadow's initial coloring as well.

LINE FOLLOW



OCCLUSION

- Attach the Occluder component to any object you want used to block visibility of objects behind it.
- The occluder object is not drawn.
- The occluder can be animated or attached to multiple objects.

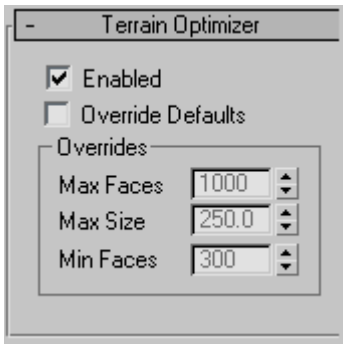
The occluder can be animated or attached to any number of objects.

Check the *Two Sided* property to make the occluder block visibilty from either side of it.

Tips

- Keep the occluder object's poly count low. A plane object doesn't occlude from both sides, so use a low poly box primitive in cases where you want occlusion from both sides.
- If an occluder doesn't totally block the view of an object, the object will be drawn.
- If one occluder doesn't totally occlude an object, and a second occluder doesn't totally occlude an object, but between the two of them the object is totally occluded, the object won't be drawn.

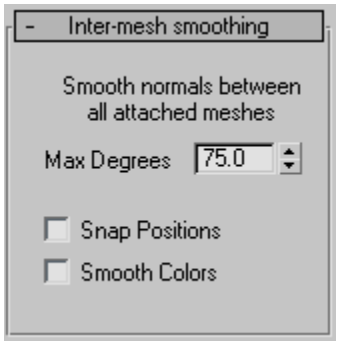
OPTIMIZE TERRAIN



REPRESENTATION

REPRESENTATION GROUP

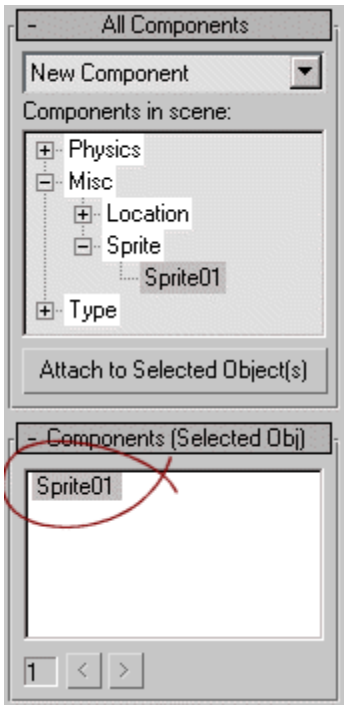
SMOOTH



SMOOTHING BASE

SNAP TO BASE

SPRITE



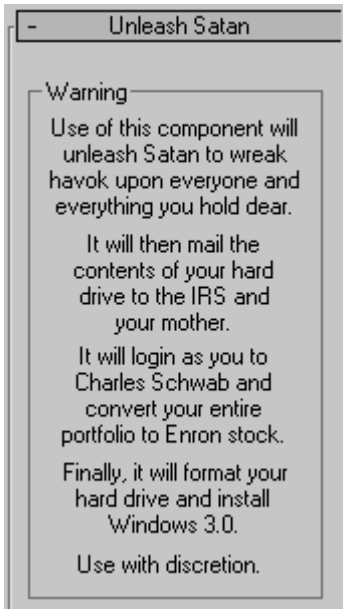
Specifies an object's Z-axis as always facing the viewer from any location regardless of angle. A typical use for this would be a light flare.

If you need a view-facing object that pivots only on the Y-axis, use the Billboard component.

This component has no properties, so no panel is displayed. The image at left shows the component has been attached to the object from the Component Manager.

SWIVEL

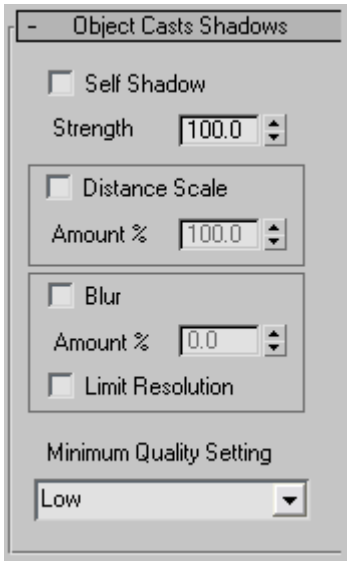
UNLEASH SATAN



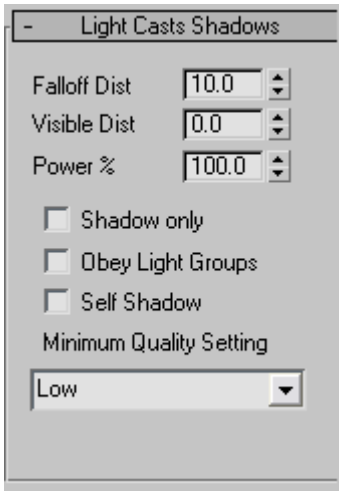
This is an experimental component known only to a few. The more adventurous among you might be tempted to use it anyway. Be warned therefore, that the few WILL know what you've done and have the power to make all the things on the properties panel actually happen.

SHADOW

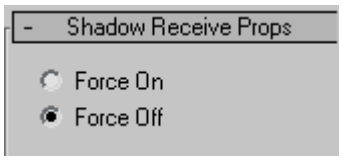
SHADOW CASTER



SHADOW LIGHT



SHADOW RECEIVER



TYPE

SEEK POINT

STARTING POINT

VERTEX/PIXEL SHADING

OVERVIEW

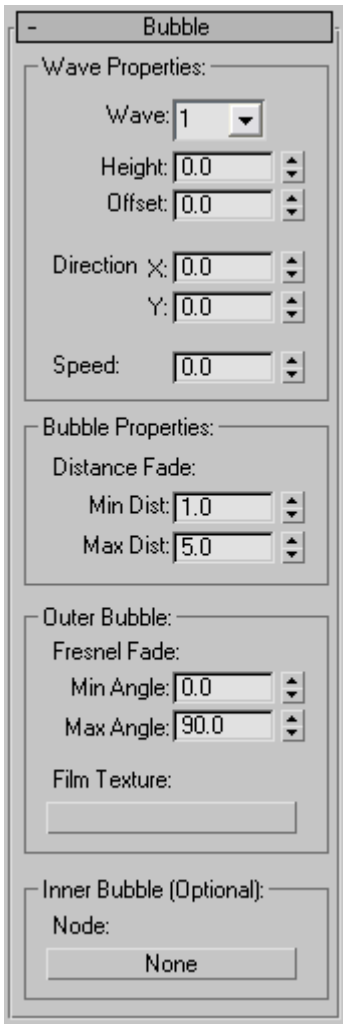
Pixel and Vertex Shaders create ambiance with materials and surfaces that mimic reality. An infinite number of material effects replace the artificial, computerized look with high-impact organic surfaces. By altering the lighting and surface effects, artists are able to manipulate colors, textures, or shapes and to generate complex, realistic scenes.

A Pixel Shader is a graphics function that calculates effects on a per-pixel basis. Depending on resolution, in excess of 2 million pixels may need to be rendered, lit, shaded, and colored for each frame, at 60 frames per second. That in turn creates a tremendous computational load. Modern graphics boards can easily process this load through Programmable Pixel Shaders. Per-pixel shading brings out an extraordinary level of surface detail—allowing you to see effects beyond the triangle level. This provides an unprecedented control for determining the lighting, shading, and color of each individual pixel, allowing them to create a myriad of unique surface effects.

A Vertex Shader is a graphics processing function that manipulates vertex data values on per-vertex basis. These variations range anywhere from differences in color, texture coordinate orientations in space, fog (how dense it may appear at a certain elevation) and point size.

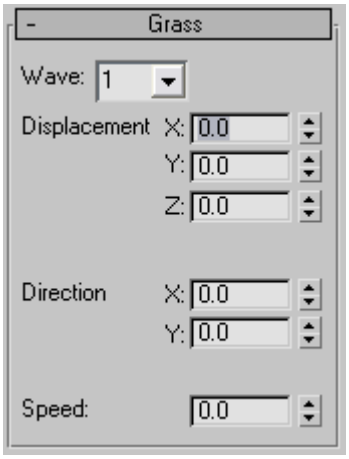
When a vertex shader is enabled, it replaces the fixed-function pipeline for vertices. The shader doesn’t operate on a primitive like a triangle, but on a single vertex. A vertex shader cannot create or destroy vertices; it can only manipulate the vertices.

BUBBLE



The Bubble pixel shader is used for MYST V’s unique vision bubbles.

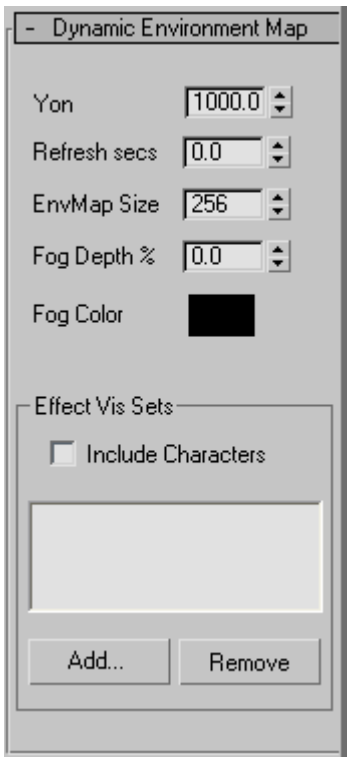
GRASS



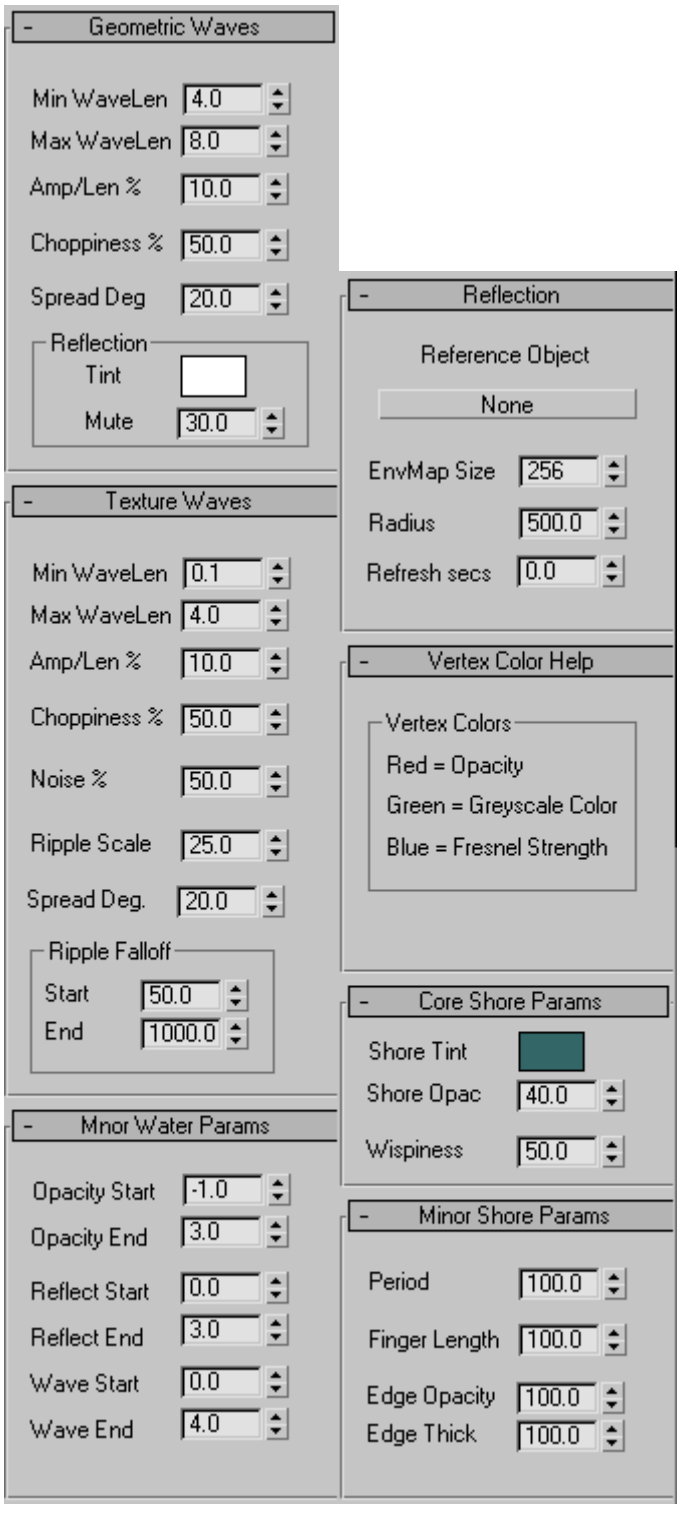
The Grass pixel shader gives life to grass, providing movement such as from wind without having to create and process animations!

WATER

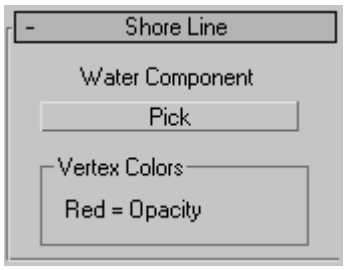
ENVIRONMENT MAP



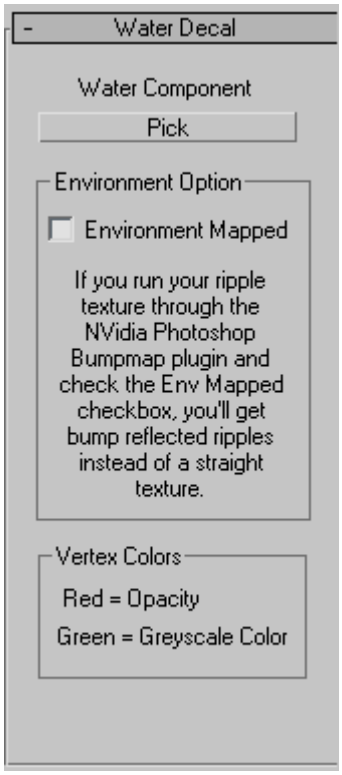
LARGER WATER



SHORE LINE



WATER DECAL



MATERIALS

BUMP MAP

Bump mapping is a form of detail map. Properly used, it will create the illusion of rough surfaces beyond your ability to add polygons, by adding per-pixel shading to runtime lighting. It works in conjunction with the other Plasma lighting capabilities.

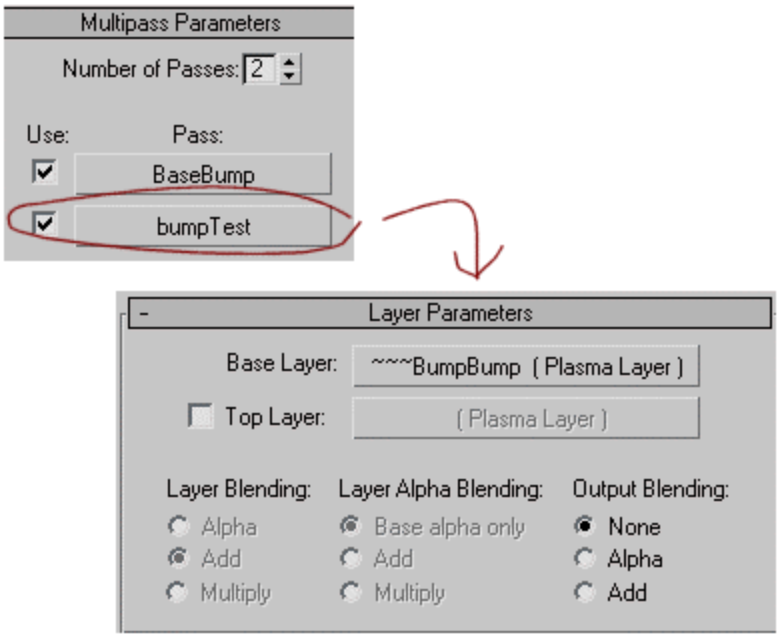
Using Bump Mapping

Restrictions

1. Create the bump map graphic using the Photoshop plugin, NormalMapFilter.8bf, which should be in your Photoshop\Plug-ins\Filters directory.

2. Use the bump map graphic as the single layer of a Plasma Standard Material. Name the layer with three tildes as a prefix. (e.g., ~~~bump map layer name) *This is a temporary requirement until the UI is completed.* It's best to not layer the bump map graphic in the Standard Material, but if you do, be sure to select Alpha in the Layer Blending group.

3. Now use that Plasma Standard Material as the last pass in a Plasma Multi-Pass Material.



4. The object onto which you apply the bump map image needs to be tiled heavily (e.g., 20 on a floor surface, 5 to 10 on smaller objects). (Generate mapping coordinates.)

***NOTE**: A good use would be applying a bump map as a rough rocky texture tiled to have very small texels on the screen. Because the final bumpy effect is dependent on the lighting AND the bump map, the tiling will be very difficult to detect on any non-flat surface.*

Remember that bump mapping is also a shading tool. If you've already burnt your shading into your underlying texture, you won't get much effect from your bump map. Applying a bump map as a layer over the bitmap it was generated from is probably a waste.

Restrictions

- Bump mapping only works with runtime lighting.
- Bump mapping is most effective when the bumpy object and the light(s) are moving relative to each other.
- The object being bump mapped can't have scaling on it. A little scale (uniform or non-uniform) is okay, but a large amount of scale will destroy the effect. The math to correct for scale is easy, but expensive, so just reset the scale on any objects you're bump mapping.
- You can apply mutliple bump maps to a single object with a Multi-sub material, but you can't stack bump maps. Any Max material ID must resolve to, at most, ONE bump map layer, regardless of how many other layers there are for that material ID.

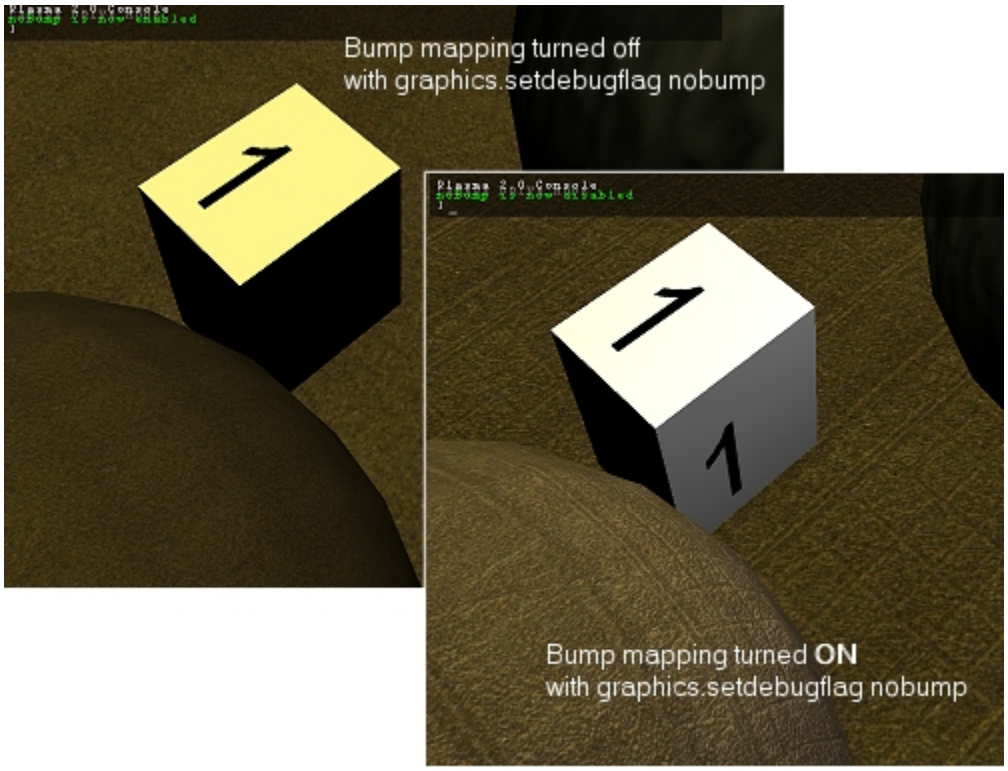
- Bump mapping currently won't work with composite materials.
- Bump mapping requires its own pass to work. You can't apply bump mapping in a way that requires it to render with another layer (like restricting the effect to the other layer's non-transparent alpha).
- Since bump mapping is a blend mode, you can't affect it with vertex alpha or vertex color.
- You can animate the tiling or offset on a bump map, but not the rotation.

The Technical Side of Bump Mapping

You don't need to understand this section to use bump mapping, but it might help if you did. Each texel in the bump map describes the local surface normal, relative to the surface, with Red=X, Green=Y, and Blue=Z. Pure blue means the local normal is aligned with the surface normal - interpolated from vertex normals. Brighter red means the normal is leaning off in the direction of increasing U, brighter green means the normal leans in the direction of increasing V. Dim red or green indicate the normal leans in the direction of decreasing U or V. Neutral values are at 127 (NOT 0), so a solid grey (127,127,127) bump map means nothing is happening.

Console Command

During runtime, check the effects of bump mapping with the console command Graphics.setDebugFlag noBump, as shown in this graphic (*the lighting has changed a little due to the animated omni light*):



In addition to the made-for-Plasma materials, Plasma also handles Max materials in specific ways.

Max Multi-Sub materials are drawn according to these rules:

Short version

Any opaque object with translucent parts will treat those translucent parts as decals of the opaque object.

Long version

- If you have a multi-sub material (MSM) whose sub-materials are all **opaque** (i.e., blend mode of "none"), then any mesh with that MSM is drawn as expected.
- If all of the sub-materials of the MSM have a **blending method other than "none"**, then any mesh with that MSM will be drawn as a blending object sorted with all the other objects in the scene.
- If only SOME of the sub-materials of the MSM have a blending method other than "none" (i.e., there is a **mix of opaque and blending** sub-materials), then any mesh with the MSM applied is treated as a **normal opaque object** and the polygons inside the mesh that have a blending sub-material selected will be treated a **decals** of that mesh. This means that they will NOT sort with other objects in the scene, but ARE guaranteed to be drawn after all opaque triangles in the same mesh.

Composite

The composite material combines standard materials, then allows you to control blending on a per-vertex basis. Using up to three sub-materials, the combined material is applied to each face of the object, then blending is done according to the opacity specified on the vertices. Of course, there are a few rules to remember:

- Objects are allowed a maximum of 8 UV channels.
- On the base material, Output Blending must be set to Alpha for transparency.
- On the sub-materials you **must** set Output Blending to Alpha. (The engine warns about this if you haven't done it.)
- The Plasma Composite Material uses an available UV channel, so be sure one of the eight is left available for it.

Basic Procedure

On the object to which the material will be applied:

1. Generate Mapping Coordinates.

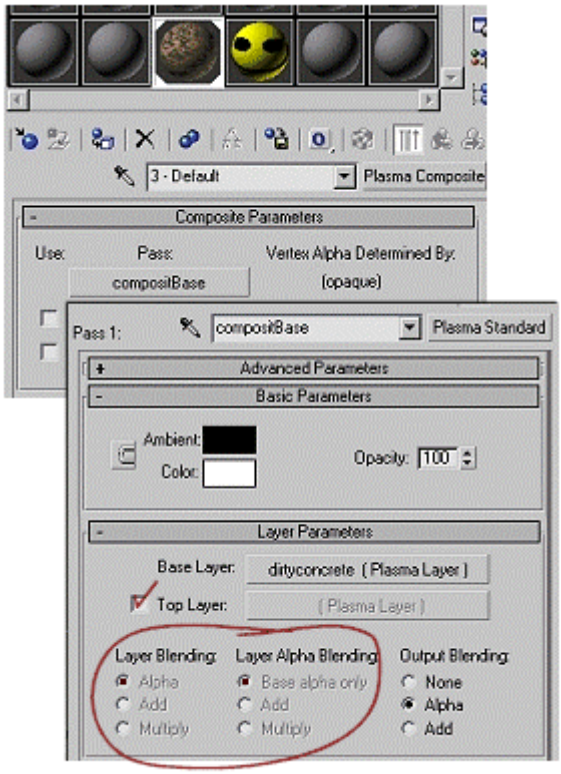
2. Convert the wireframe to an editable mesh.

3. Apply a base material, select 'Alpha' for Output Blending on the Base Layer (Layers Parameters). *(If your base material is a blended material, then you **must** select 'Alpha' for Layer Blending and 'Base alpha only' for Layer Alpha Blending on the 'Top Layer' of the*

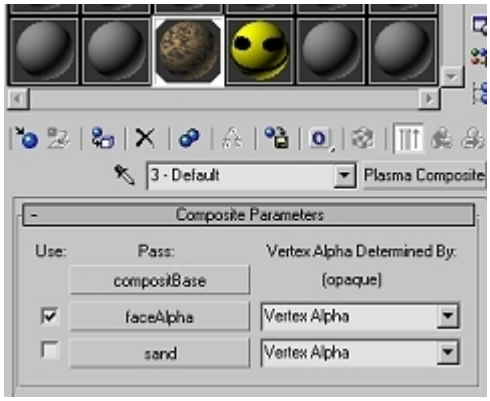
blended material.)

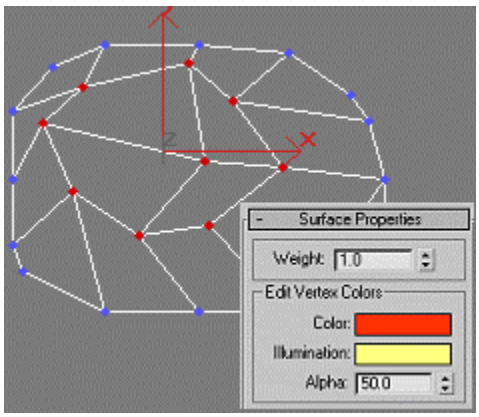
4. For the sub-material, check the checkbox beside its button. On the sub-material's base material, select 'Alpha' for Output Blending (Layers Parameters). *(If your sub-aterial is a blended material, then you **must** select 'Alpha' for Layer Blending and 'Base alpha only' for Layer Alpha Blending on the 'Top Layer' of the blended material.)*

On the geometry, select the vertices for blending. The Max default for vertex alpha (vertex opacity) is 100%. At this setting, the base material will not be drawn at all. 75% vertex alpha would blend 75% sub-material and 25% base. Choosing Inverse Vtx Alpha for the sub-materials in the composite material will invert this ratio. Vertex color and illumination are fully supported.



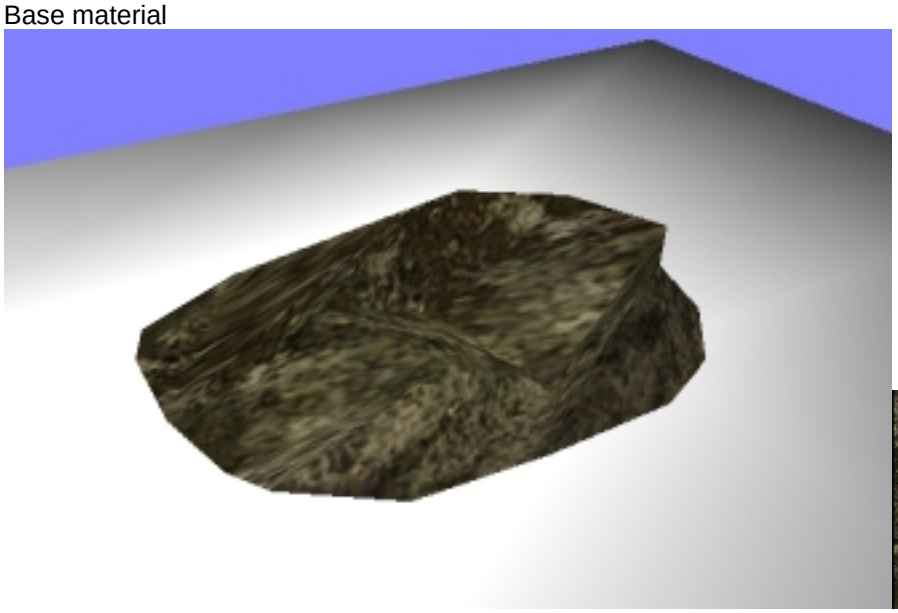
Set these when a material is a blend.



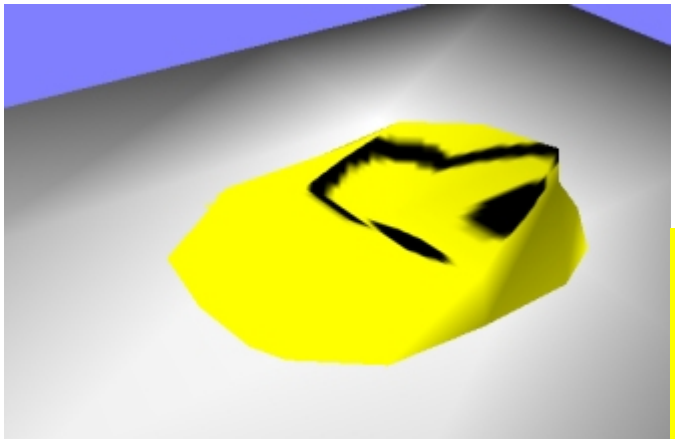


Example

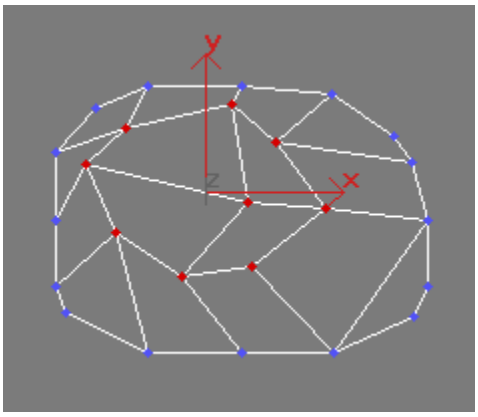
Base material

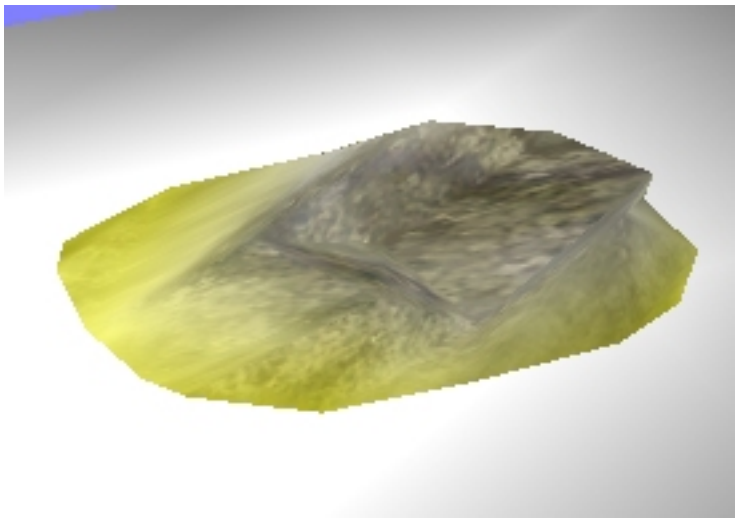


1st sub-material



Inside vertices selected on the editable mesh, vertex alpha set to 5.0



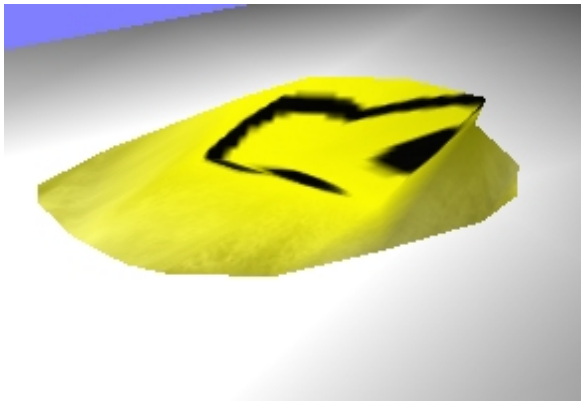


Base material

Output Blending: **Alpha**

1st sub-material
Vertex Alpha

5% vertex opacity



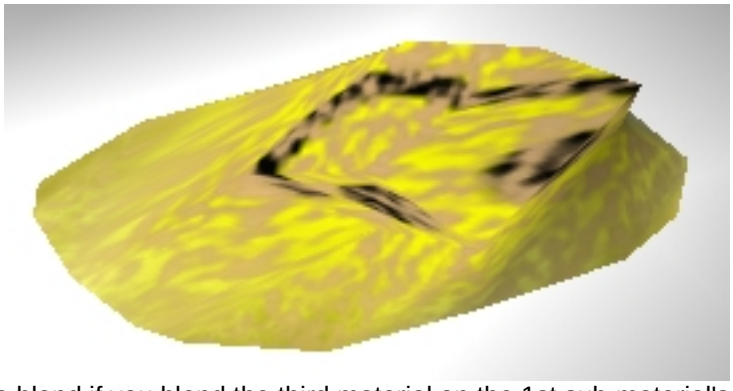
Base material

Output Blending: **Alpha**

1st sub-material
Inverse Vtx Alpha

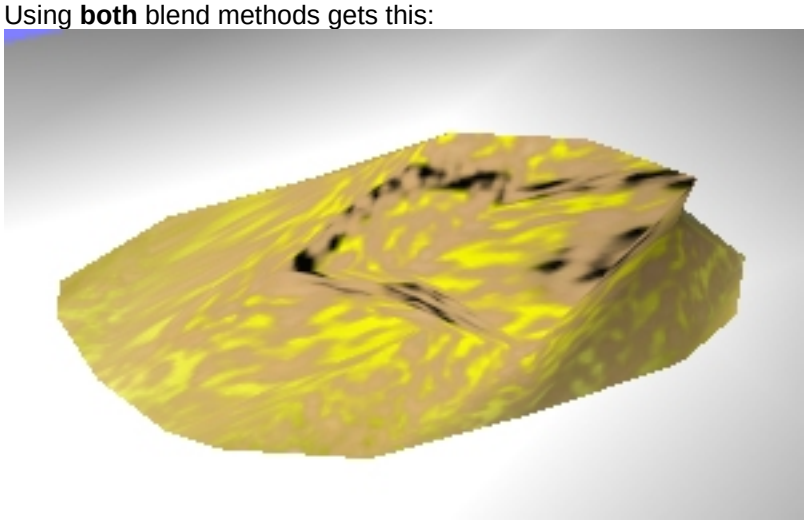
Inverse of 5% vertex opacity
(or 95%)

Throwing a third texture into the mix:



You'll achieve essentially the same blend if you blend the third material on the 1st sub-material's layer parameters
OR
if you use it as a 2nd sub-material. If it's blended on the layer, you must use the blend parameters shown at right.

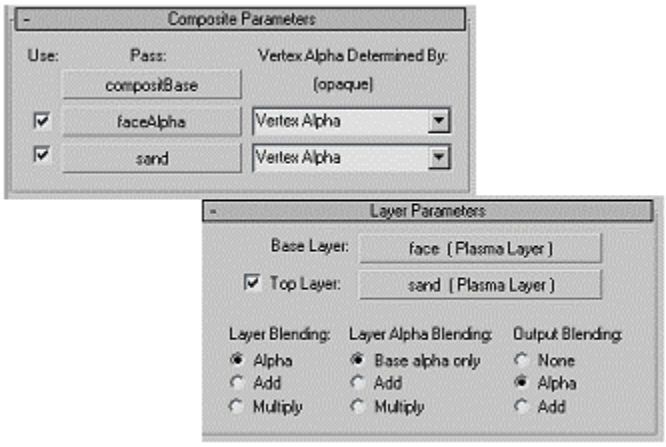
Using **both** blend methods gets this:



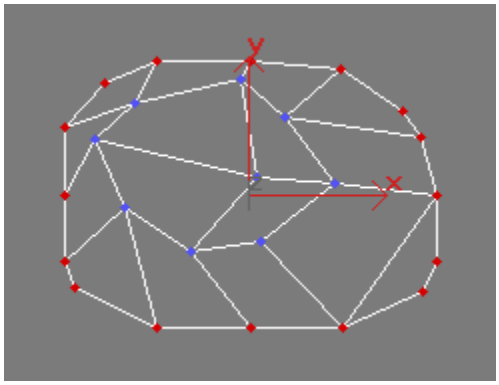
Base material

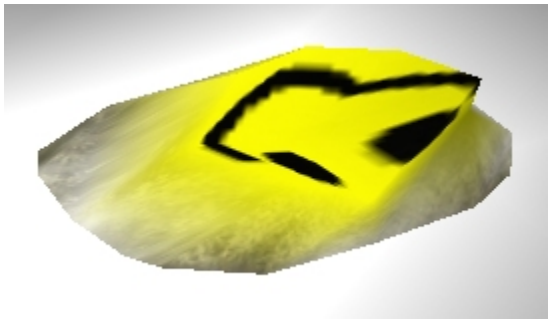
Output Blending: **Alpha**

1st and 2nd sub-materials
Vertex Alpha



Outside vertices selected on the editable mesh,
set to 5.0 on Edit Vertex Colors.



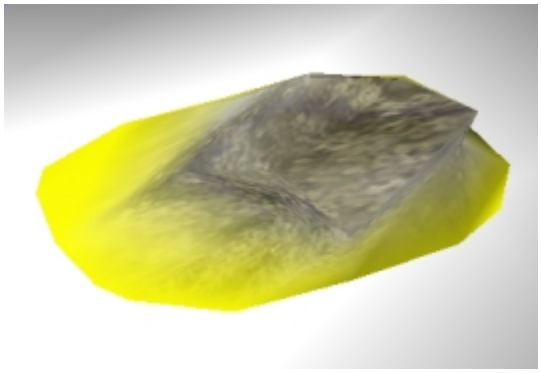


Base material

Output Blending: **Alpha**

1st sub-material
Vertex Alpha

5% vertex opacity



Base material

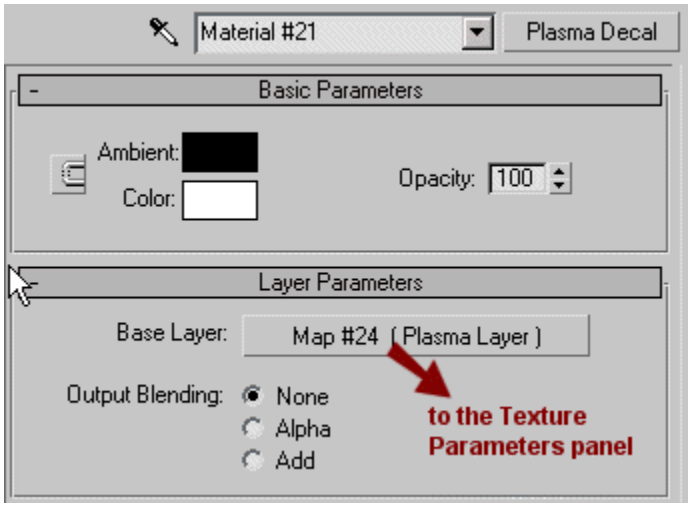
Output Blending: **Alpha**

1st sub-material
Inverse Vtx Alpha

Inverse of 5% vertex opacity
(or 95%)

Notice that this looks the same as the sample above, which has the inside vertices selected and uses the Vertex Alpha setting rather than the Inverse Vtx Alpha.

DECAL



This Plasma material type gives you a method for easily attaching a decal to an existing texture.

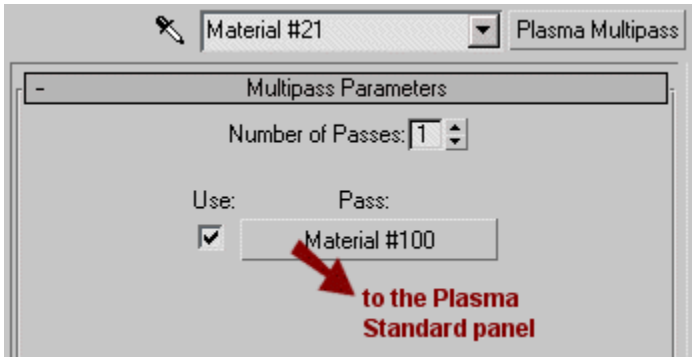
1. Create coincident geometry for the decal (e.g., a plane).
2. Select Plasma Decal as the material, then from there you define the texture map as usual.
3. Link the decal geometry to a parent object. A decal's parent object can be another decal.

   This has the same effect as putting both decals on the same parent object, except it also specifies which decal gets drawn first. Every decal must have a parent node, and thus be guaranteed a non-decal ancestor somewhere down the line (the exporter will warn you if this isn't the case).

To determine draw order for multiple decals on one object, Plasma looks at how many generations there are in the hierarchy between each decal and its first non-decal ancestor. Decals closer to the non-decal ancestor are drawn first (thus underneath) later children.

MULTIPASS

Intended for complex texture blending, this material defines a drawing order for 'sets' of layered materials. Within each 'pass' is the familiar Plasma material with layers.



The engine draws each pass in the order that they are listed on the Multipass Parameters panel. So, if you define two passes and each pass specifies a 2-layer blend, Pass one is drawn, then Pass 2 is drawn 'over' Pass 1.

- Each pass should be a Plasma Standard material; other types are quietly skipped.
- This material is costly.

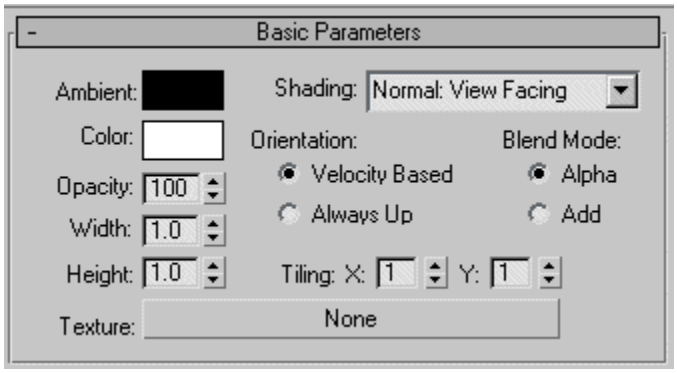
***Example**: The first (bottom) pass is a one-layer material on which the UV coordinates are animated to give the effect of motion. On the layer panel, the "emissive" checkbox is checked,so the material is not affected by the scene lights/shading, and to appear to be glowing.*

*The second (top) pass consists of a simple one-layer material with an alpha channel, to show through to the animated pass "underneath". The "emissive" box on this material is **not** checked, so the opaque parts of the texture are affected by lights and can be shaded.*

PARTICLE

This material is required for a particle system and is applied to the particle emitter object, which can be anything other than a dummy object. The emitter object's geometry is not drawn; it simply defines the location where the particles are created. The material defines the look of the particles.

***NOTE:** Currently, particle material animation is supported for opacity and color only.*



Ambient - Beginning ambient color of the particles. When used with the Emissive shading option, this sets the coloring for the particles. Other lighting is ignored.

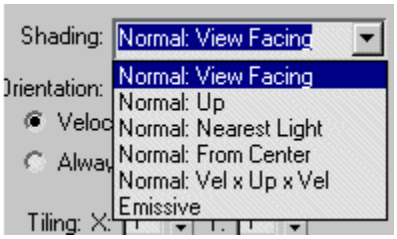
Color - Beginning original color of the particles.

Opacity - Percentage of transparency. 100% is solid, 0% is invisible. Use this to animate the particles in a range of tranparency.

Width - The width of the particle in inches.

Height - The height of the particle in inches.

Texture - The texture file used to give the particles a look. An image that combines a grid of equally sized parts, then tiled provides particles of different looks. For example, the four-part image below, tiled at X-4 and Y-4, would emit each of the four squares as a separate particle.



Shading - Choose the direction of the particles' normals from this dropdown list. Or, choose **Emissive** which tells the particles to ignore lighting and get colors from the ambient color setting.

Orientation
Velocity Based **-** the particles will orient themselves to the current direction of spray.

Always Up - The particle's 'top' edge is always up; there is no tumbling.

Blend Mode
Alpha - Same as for other materials, 32-bit images are blended in the alpha channel.

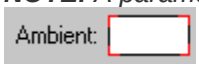
Add -

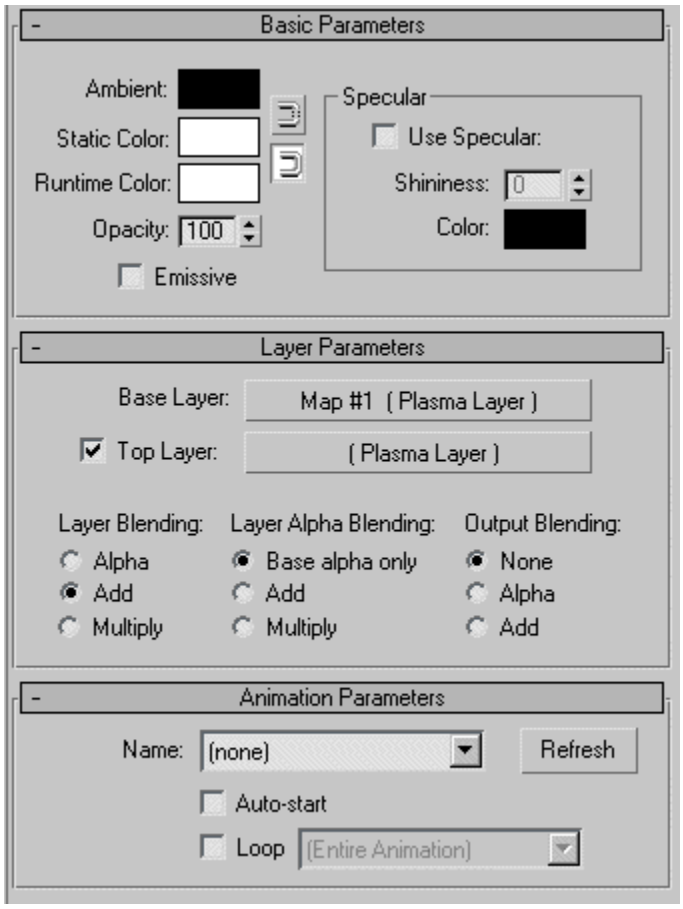
Tiling (default: 1)
By choosing a texture that's designed in multiple, equally sized tiles, and specifying how many tiles are in each direction in these Tiling properties of the material, the particle system will randomly emit the tiles so that they're not all exactly alike. See the Texture property above for an example of this. Also, download and export the sample file for a live demo.

STANDARD

Regardless of which type of material you use (except the Particle material), you'll end up on this panel. The other materials, Composite, Multipass and Decal, all "wrap" this standard material.

NOTE: *A parameter control surrounded by red crop marks, means that parameter is animated.*





Basic Parameters Panel

Ambient - The standard environmental color adjustment to the base color. The default, black, (RGB 0,0,0) adds zero to the base color, resulting in no effect.

Static Color - The diffuse color used for preshaded lighting (static objects - the object neither moves, nor needs lights moving over it). This color can be animated. *See the Diffuse Color note below.*

Runtime Color - The diffuse color used for dynamic lighting (moving objects or objects that will have light moving over them). This color can be animated. *See the Diffuse Color note below.*

Opacity - Standard percentage of transparency. 100% shows the object as completely solid, 0% makes it invisible.

Emissive - Check to specify this material as a light "emitter", which is therefore not going to be lit by either Max preshading lights or the runtime lights. Useful for things like lens flares and glowing things.

Specular - Enabling Specularity tells Plasma to light the material with Plasma Runtime lighting. This can be an expensive method of lighting, so use it with care. Normally, a non-animated object is lit only with static (Max) lighting, but in order to give you more control over specular highlighting, this parameter provides an exception to that standard.

Use Specular: Check this to enable specularity.
Shininess: Range of 0 -100, 100 is the shiniest.
Color: The color of the specular highlights. This color can be animated.

Diffuse Color
The Static Color and Runtime Color selectors, give you separate control over diffuse color when lit by the two types of lighting (static and runtime). By default the two colors are locked as they will usually remain the same. This is because during runtime, an object that has both types of light on it receives a combined light color value (the light colors are added together for a single light color), which is multiplied with the light's color. Keeping them the same value keeps the combined light color the same - as if only one of them was being used.

The only time you'll want to unlock these is for use with specularity. In that case, set the Static Color to the diffuse color you want and set the Runtime Color to black, so when they are added, the result is the Static Color. (static color + 0,0,0 = static color), and the specular highlight color is whatever you select in the Specular Color selector.

Layer Parameters Panel

NOTE: *Holding ctrl while selecting a texture in one the of Max layers will ask you if you wish to remove the texture from the layer (replace it with nil). This is mainly for the clothing material which lets you pick several textures for different things, some of which you may want nil.*

Base Layer: The texture used for the bottom (or only) layer of this material. The button opens the Texture Parameters panel.

Top Layer: The second texture layer, which is blended with the base layer using the blending options in the following parameters. The button opens the Texture Parameters panel.

Layer Blending: The option selected in this group produces a single RGB result (per pixel) for the combined layers. The RGB result that actually gets exported for runtime is determined by the Output Blending parameter.

Alpha - The alpha value of the **top** layer material determines how much color is blended with the base layer's alpha (pixel for pixel). This does not affect the alpha output, only the color blending of the two materials.

Add - The RGB value of the base material is added to the RGB value of the top layer resulting in the single color value for the combined material (pixel for pixel). This does not affect the alpha output, only the color blending of the two materials.

Multiply - The RGB value of the base material is multiplied with the RGB value of the top layer, resulting in the single color value for the combined material (pixel for pixel). This does not affect the alpha output, only the color blending of the two materials.

Layer Alpha Blending: These determine how the ALPHA channels of the two layers will be blended, resulting in a single alpha value (per pixel). The alpha result that actually gets exported for runtime is determined by the Output Blending parameter.

Base Alpha only - The alpha value is determine solely by the alpha channel of the base layer. The top layer's alpha is ignored. This does not affect the color output, only the alpha blending of the two materials.

Add - The alpha value of the base material is added to the alpha value of the top layer resulting in the single alpha value for the combined material (pixel for pixel). This does not affect the color output, only the alpha blending of the two materials.

Multiply - The alpha value of the base material is multiplied with the alpha value of the top layer, resulting in the single alpha value for the combined material (pixel for pixel). This does not affect the color output, only the alpha blending of the two materials.

Output Blending: Once color and alpha blending methods are set in the *Layer Blending* and *Layer Alpha Blending* options, this setting determines how the combined material will blend with whatever is displayed on the screen during runtime. See the example for further clarification (in work 10/23/01)

The exporter checks to be sure these settings are compatible with other layer settings.

None - The material is drawn to the screen without blending with anything behind it. Do not use this option with Multi-pass materials.

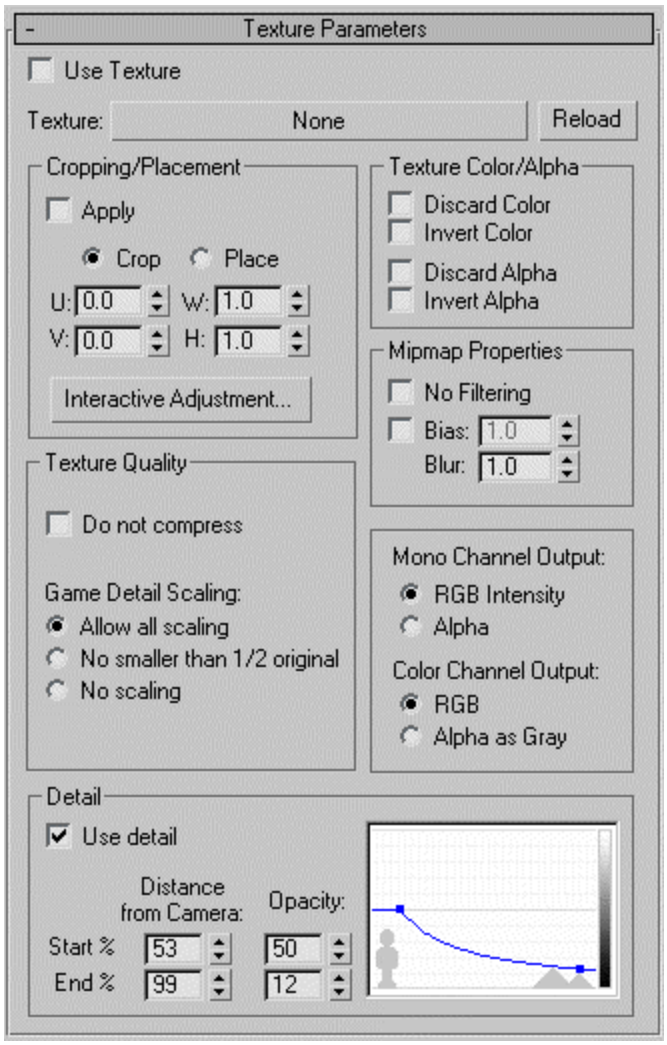
Alpha - This blends the result from the *Layer Alpha Blendin*g parameters with whatever is behind it on screen. e.g., a 50% alpha blend in the combined material output will show 50% of the whatever is behind it on the screen during runtime.

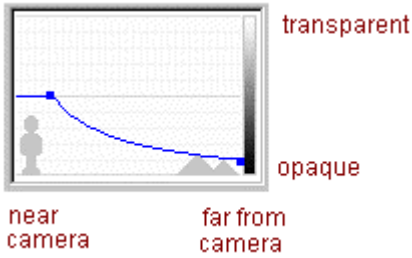
Add - The material's color result from the *Layer Blending* parameters is added to whatever is behind it on screen. This could give a glow to the material if a bright color is behind is, or a dulled result if a darker color is behind it.

TEXTURE PARAMETERS

These parameters are basic to most Plasma textures so you'll see this panel repeatedly. After selecting a Plasma Material in the Material Editor, a panel opens for specifying that material's particular parameters. On each of those type-specific panels is one or more buttons for Layers (or Materials, which then leads to Layers). The Layers button opens the Texture Parameters panel you see below.

(The exception is the Plasma Particle material, which has no use for these parameters.)





Use Texture - Check this to use the parameters on this panel. When unchecked, the texture is ignored by the Plasma engine.

Texture selector - Click to open the AssetMan browser from which you choose a texture file to load.

Shift-click lets you select a texture file from your local machine, skipping AssetMan.

Ctrl-click gives you the option of removing the texture from this layer. This is mainly for the clothing material which lets you pick several textures for different things, some of which you may want nil.

Reload - Click this button when you have made a change to the texture file (in Photoshop or other outside graphics program) and need to update the scene with the modified texture.

Cropping/Placement group -

Apply

Crop

Place

Interactive Adjustment...

Texture Quality group -

Detail group -
These settings are relative to the mapping on object to which this texture is applied. So if you change the size of the object or the tiling of your UVW map, the
actual distances will change accordingly, even though the settings will remain the same.

Use detail - Check to enable the detail parameters. If unchecked, these parameters are ignored during export.

Distance from Camera - Shown on the horizontal axis of the graph, this is how far from the camera the texture layer will be when it's scaled to a single pixel. The percentage values are % of distance. e.g., In the sample image, the opacity will be 50% at 53% of the distance to the camera.

Opacity - Represented on the vertical axis of the graph, 0=transparent, 100=totally opaque.

Detail graph - The blue line represents opacity over distance. If you drag the points on the graph, the values in the spinner controls change accordingly, and vice versa.

GAME LOGIC

OVERVIEW

Game logic is implemented with Detectors and Logic. The Detectors are used to detect different kinds of game events, either player initiated or game initiated.

When these detectors fire the corresponding Logic component is executed.

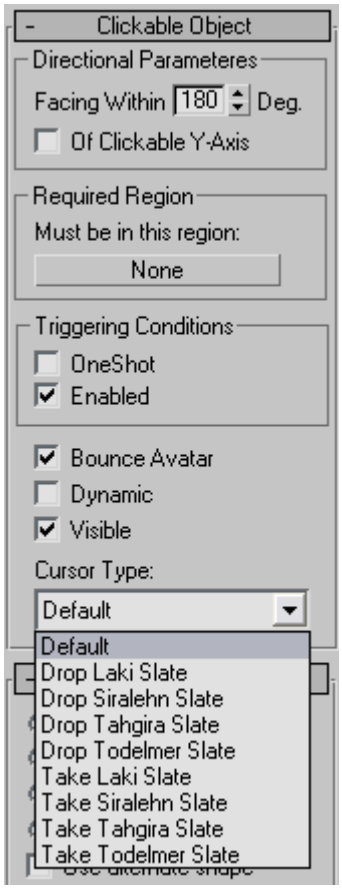
DETECTORS

The Detector component is a way to specify a game event. These can be events that are initiated by the player, such as clicking on an object or an event that happens inside the game, such as an animation event.

When the event happens a Notify message is sent to the component that is waiting for the event. This Notify message will have enough information to determine what the event was and who or what initiated the event.

ANIM EVENT

CLICKABLE



This component sends a message when the object is clicked on by the player (a Responder picks up the message and does something in response). A region is defined around the object which the avatar must occupy before the clickable object can be triggered. It can be triggered with a mouse click or with a use key. During runtime, the cursor changes in appearance when passed over a clickable object.

Directional - When checked, this activator will only be triggered when the avatar enters the trigger area withing the degree setting in the next parameter. Otherwise, any contact with the bounding region will awaken the clickable object within.

Facing Within x Degrees - The avatar must be facing the object within this degree value in order to enable triggering. This prevents triggering when the avatar has merely brushed by.

Use Proxy Object - To use this option, first draw a simple, low poly version of the object. Then, click the button and select the bounding shape you drew.

Must be in region (required) - This required region geometry must have a **Physical Detector component** attached; an error message is shown during export if it does not.

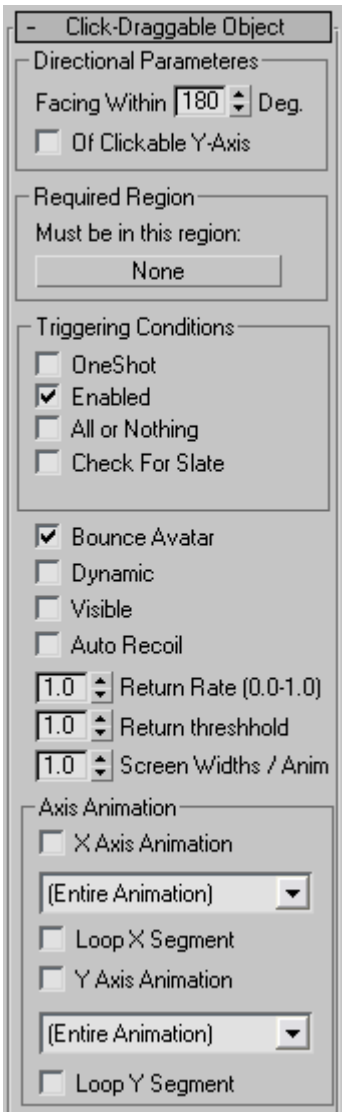
UnTrigger on key/mouse up - Check this to un-click, or un-trigger the activator when the mouse or keyboard key is released. (The activator is triggered only as long as the mouse or keyboard key is pressed.)

Cursor type – You can specify what cursor will be displayed on mouse rollover.

COLLISION SENSOR

DRAGGABLE

Use this on objects that need to move AND send a message for a response, which usually means a lever. Two animations parameters are available, however, only one can be used on a single object.



Directional - When checked, this activator will only be triggered when the avatar enters the trigger area withing the degree setting in the next parameter. Otherwise, any contact with the bounding region will awaken the clickable object within.

Facing Within x Degrees - The avatar must be facing the object within this degree value in order to enable triggering. This prevents triggering when the avatar has merely brushed by.

Use Proxy Object - To use this parameter, first draw a simple, low poly version of the object. Then, click the button and select the bounding shape you drew.

Must be in region (required) - This required region geometry must have a **Physical Detector component** attached; an error message is shown during export if it does not.

X Axis Animation - Check to enable an animation for this object (*you can use either the X Axis anim or the Y Axis anim, but not both*). Using this animation means that when the user drags this object, the animation plays forwards and backwards relative to the mouse movement along the **horizontal** axis of the moniter. This parameter is specified the same as all other animations in Plasma 2, by creating the animation, naming it, then selecting it from the dropdown list. To select a portion of an animation, use Notetracks to place start and stop keys, then select the segment name from the dropdown list.

Loop X Segment - Check to loop the animation while the object is being dragged.

Y Axis Animation - Check to enable an animation for this object (*you can use either the X Axis anim or the Y Axis anim, but not both*). Using this animation means that when the user drags this object, the animation plays forwards and backwards relative to the mouse movement along the **vertical** axis of the moniter. This parameter is specified the same as all other animations in Plasma 2, by creating the animation, naming it, then selecting it from the dropdown list. To select a portion of an animation, use Notetracks to place start and stop keys, then select the segment name from the dropdown list.

Loop Y Segment - Check to loop the animation while the object is being dragged.

All or Nothing - When this is checked, it means that instead of the player being able to 'drag' the object along its animated path in real time and stop it at any point, the animation will play all the way to the end or beginning, depending on which way the player moves the mouse when the object is clicked on. This way the animation plays back at the rate it was animated at instead of being tied to mouse movement directly.

Auto Recoil – This will make the draggable automatically recoil or complete the draggable if the user lets go of the draggable before completing.

Also, if there are stop points specified in the animation notetrack, then the draggable will auto complete to closest stop point.

MATERIAL EVENT

REGION SENSOR

LOGIC

The Logic component is waiting to be triggered by a Notify message that comes from a detector (or another Logic component). When the Logic component is triggered the Logic component will perform the game play reaction to the initial event. This is where the game play logic is implemented.

There are two types of Logic components. 1) Responders which are a list of commands to execute. 2) And Python components that allow the game play scripter to create more sophisticated logic reactions.

Also, Python components can trigger Responders which allows easier updating of animations and sound effects without having to change the python script.

RESPONDER

This component gives artists the ability to design gameplay logic without having to write Python code directly (there may be exceptions). The parameters let you choose the triggering component (Detectors), define groups of commands (each group is called a state), define the order of execution of states and predefined commands, and define the timing of the whole responder sequence, as well as specify command parameters. There is a lot in this component, so an example is included to help clarify its use.

Component Parameters

Detectors - This text box lists the Detectors(s) that, when 'touched', will trigger (or un-trigger) the commands listed in the State list.

To add a Detector, click <Add...> to open a dialog of Detectors defined in this scene. Select the Detector you want and click OK. This is now a 'watched' detector. If more than one is listed, any one of them will trigger the responder commands.

To remove a Detector, select the Detector, then click <Remove>.

State - The term 'state' refers to a command or set of commands that are grouped to create a specific gameplay behavior or series of sounds. The commands are executed in the order they are listed.

You can rearrange the commands by selecting and dragging.

For example, when an avatar touches a linking book, it triggers the book animation, linking sound, avatar fade animation, and transition to the next age. All these commands can be gathered into a single gameplay state, to which you give a meaningful name, such as 'linkOutSequence' or something.

Initially, there is one unnamed state (Default(1)). Rename this to something, then use the <Add> button to add commands to it.

Commands - This lists all commands that are available for running when this responder is triggered (when one of the listed Activators is 'touched'). The object to which you attach this component must be physical (has a Physical component attached).

To add a command, select it from the drop down menu, then click <Add> to copy the command to the text box.

To remove a command, select it in the text box, then click <Remove>.

To propagate the command to a child object,

1. Select the child object(s).
2. In the Component Manager's 'Components in scene' panel, select the appropriate Responder component.
3. Click <Attach to Selected Object(s)>

Command Parameters - The options in this part of the panel differ depending on which type of Command is highlighted in the Commands textbox. Generally, you can select the component that is being animated and the Note Track loop points, if applicable.

Linking to another age is also included as a command parameter. In that case, you can choose the name of the age and a Spawn Point.

The responder commands available are:

- Subtitle
    - Show Untimed
    - Hide Untimed

    - Play Animation
    - Kill Animation
- Material
    - Play
    - Stop
    - Toggle
    - Set Looping On
    - Set Looping Off
    - Set Forwards
    - Set Backwards
    - Rewind
- Random Sound
    - Play
    - Stop
    - Toggle
- Sound
    - Play
    - Stop
    - Toggle
    - Rewind
- Animation
    - Play
    - Stop
    - Toggle
    - Set Looping On
    - Set Looping Off
    - Set Forwards
    - Set Backwards
    - Rewind
    - Fast Forward
- Exclude Region
    - Clear
    - Release
- Enable/Disable
    - Responder
    - Detector
    - Physical
- Visibility
    - Visible
    - Invisible
- Link
- One Shot
- Notify Trigger
- Camera Transition
- Delay
- Footstep Surface
- Trigger Multistage
- Physical Force

PYTHON COMPONENT

The Python File Component is a way to attach a Python Component script to an object. Each individual Python script will specify what the

attributes are which will determine the 3ds Studio Max rollout interface.

See the next section about Python for more detail.

PYTHON

OVERVIEW

Python component scripts are a way to add game logic outside of writing it into the engine.

MODULE STRUCTURE

A Plasma Python component script is a Python module. Each component has a similar structure.

- Attributes that can be specified in Max.
- Class that contains the game logic.

PLASMA PYTHON ATTRIBUTES

When the Plasma Max plugin starts, it reads in the Plasma Python components. It then extracts the attributes and build Max rollout dialogs based on the attributes. This way the Python programmer can create a unique Max interface for each python component.

When the Max users selects a Python component and attaches it to a scene object, the unique rollout dialog is shown and allows them to specify what objects and parameters that the Python code will test or manipulate. This allows the python component to be re-used in different places because of the uniqueness of the attributes.

The attribute types available are:

- ptAttribBoolean – true or false
  - in Max: a checkbox
- ptAttribInt – integer number
  - in Max: a number spinner
- ptAttribFloat – floating point number
  - in Max: a number field
- ptAttribString – a string
  - in Max: an edit box
- ptAttribDropDownList – number selection
  - in Max: a pull down menu
- ptAttribSceneObject – scene object
  - in Max: a scene object picker
- ptAttribSceneObjectList – a list of scene objects
  - in Max: a multiple scene object picker
- ptAttribActivator – a detector
  - in Max: a detector picker
- ptAttribResponder – a responder
  - in Max: a responder picker
- ptAttribDynamicMap – a dynamic texture map
  - in Max: a material picker with just dynamic texture maps
- ptAttribGUIDialog – a Plasma GUI Dialog
  - in Max: a GUI dialog picker
- ptAttribExludeRegion – an exclude region (may not be used in Sasquatch)
  - in Max: an exclude region picker
- ptAttribWaveSet – a wave set pixel shader
  - in Max: a Wave set picker

- ptAttribGlobalSDLVar – a global SDL variable
  - in Max: a pull down of SDL Variables
- ptAttribAnimation – an animation
  - in Max: an animation picker
- ptAttribBehavior
  - in Max: a behavior picker
- ptAttribMaterial
  - in Max: a material picker

PLASMA RUNTIME LIGHT OBJECTS

Understanding Plasma Lighting
Light Group Component

The Plasma runtime lights are required for lighting animated objects, like avatars, and for lights that move across static obects. These lights come with a heavy cost in performance so use them carefully. In order for a runtime light to affect *all* scene objects (static and moving), the light must be animated. The third condition for a runtime light effect on an object is that both the object and the light are enabled for specularity.

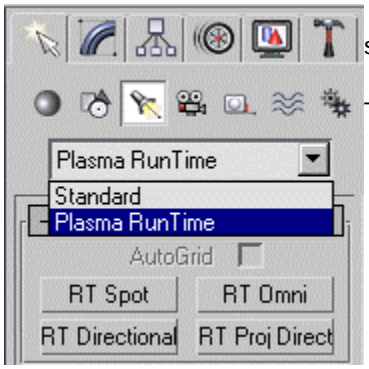
To summarize: objects are affected by Plasma runtime lighting under these three conditions:

- The light is animated
- The object is movable
- The object and the light have specularity enabled

There is a limit of eight runtime lights at any one time for any one object. If you try to use more than that the engine will use the brightest eight lights.

Once a runtime light is placed in the scene, you can right-click on it to access properties in the quad menu. However, since Max controls what goes on these menus, you'll have to remember that *Ambient* and *Find Target* don't do anything for Plasma runtime lights.

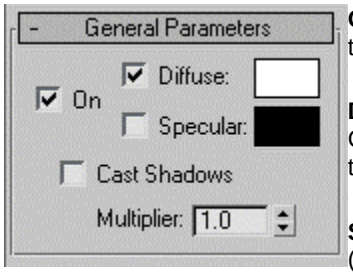
The idea is that the artists are supposed to use MAX lights whenever possible for static lighting and runtime lights only when necessary for moving objects, and even then to optimize their use of the runtime lights since they're far more expensive at runtime.



Access the Plasma lights by opening the Max Light creation panel and selecting **Plasma RunTime** from the dropdown list, as shown in the graphic.

The four runtime lights are each described below, along with the parameters.

Each of the four lights uses the common parameters shown on this panel.



On checkbox - A defined light can be turned off by unchecking this (rather than having to delete it).

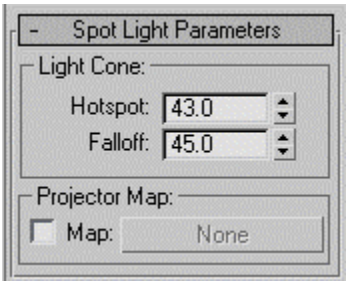
Diffuse - The color that's reflected by the object when this light illuminates it. Check the checkbox to enable it, then double-click the color swatch to access the Color Selector and select the reflection color.

Specular - Enable specularity to place a specular highlight on a static object. (To accomplish this, check Specular on your material, unlock the static and runtime colors and set the runtime color to black (this ensures that no diffuse lighting is added to the object - something you don't want for a static object.)

Cast Shadows - This has to be checked if you want the light to cast shadows.

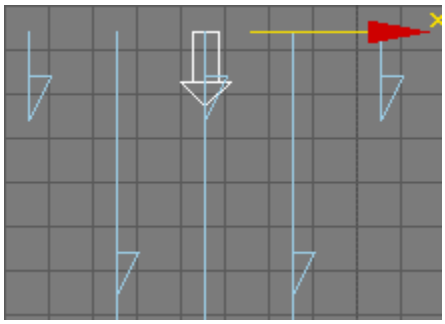
Multiplier - The Multiplier value in every light lets you increase the intensity, or brightness of the light beyond its standard range. You can use negative numbers to reduce the light's intensity.

RT SPOT LIGHT



RT DIRECTIONAL

There are no parameters in addition to the general parameters for this light type.



Use RT Directional lights to illuminate animated objects with a sunbeam-type lighting.
Use Max Directional lights to illuminate static objects with sunbeam-type lighting.

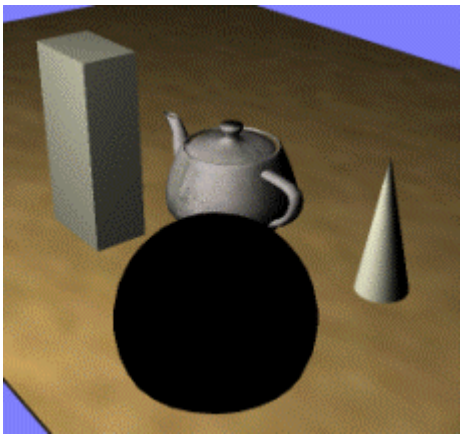
Directional lights cast parallel light rays in a single direction. Directional lights are primarily used to simulate sunlight. You can adjust the color of the light and animate it.

Arrows are used for the viewport symbolism is to show the direction of the light rays.

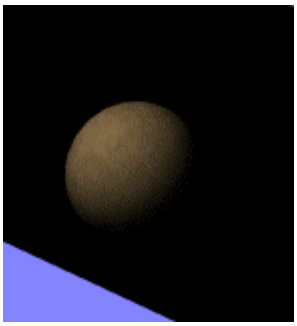
RT OMNI

Use RT Omni lights to illuminate animated objects with all-directional lighting.
Use Max Omni lights for illuminating static objects.

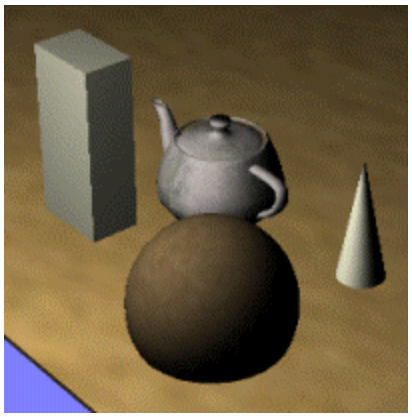
In the samples below, only the sphere is animated.



Only a Max Omni light illuminates this scene, thus the animated spere is not lit.



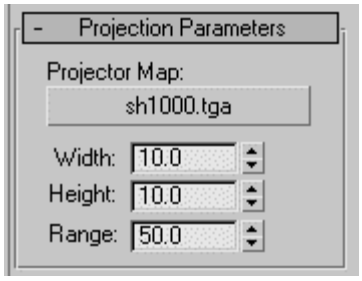
Only a RT Omni light illuminates
this scene, thus only the animated sphere is lit.



Both a Max Omni and a RT Omni illuminate this scene, thus all objects are lit.

RT PROJ DIRECT (PROJECTED LIGHT)

This light projects an image as well as providing light, much like a movie projector does. And, like a movie projector, it requires something on which to project the image, so position the RT Proj Direct light such that it shines on objects exactly as you want the image projected. ProjDir lights have an area of effect defined by width, height and depth. ProjDir lights also "project" a an image along the area defined. This texture is required.



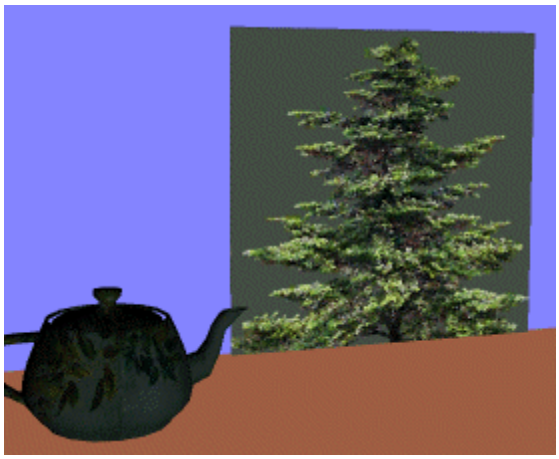
Projector Map - Required. Click the Projector Map button to select the texture map you want to project. The image is projected as it is and ignores the alpha channel.

Width - Specify the width of the projected image. This should fit the object(s) on which the image is being projected since any projection that falls outside of an object will appear cut off.

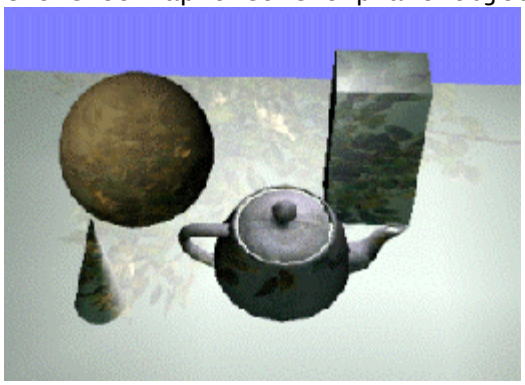
Height - Specify the height of the projected image.

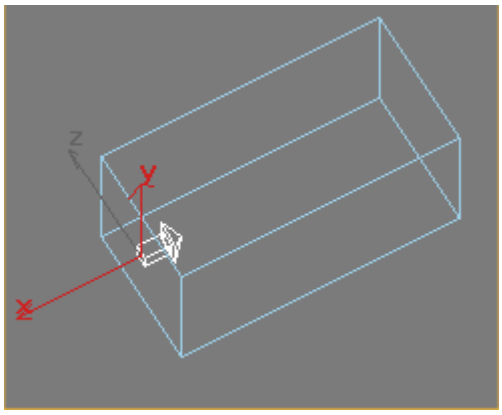
Range - This is the distance between the light object and the object(s) on which the image is being projected. It's best to over reach the target objects a bit to ensure coverage from the projected image.

RT Proj Direct viewport graphic:

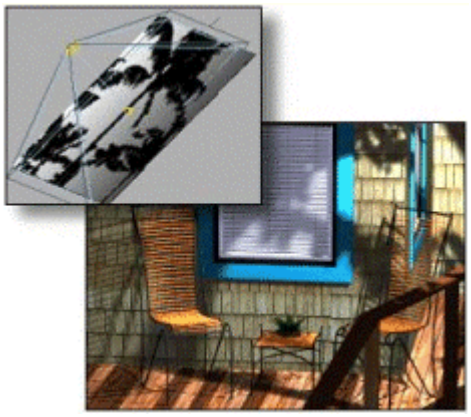


An RT Proj light is projecting a foliage map onto the teapot, and another RT Proj light is projecting the tree map onto the plane object.





In the sample at left, a single RT Proj Direct light is casting its foliage image across multiple objects. This technique could be used with a black & white image to simulate casting a shadow across multiple objects, as this Max example illustrates...



AVATAR STANDARDS

Min/max avatar height	5’-6’6”
Min/max avatar width	1’3”-2’3”
Min/max jump distance (from standstill)	6’-8’
Min/max jump distance (from run)	12’-20’
Min/max jump height	3’-5’
Min/max avatar run speed	16 f/s-33 f/s
Min/max swim speed	3 f/s-6 f/s
Stair riser A	12” deep x 8” high
Ladder rung spacing	1’
Ladder angle 01	0°
Ladder angle 02	10°
Ladder angle 03	20°
Ladder angle 04	30°
Ladder angle 01	40°
Min doorway width	3’
Min doorway height	7’6”
Rail height	3’
Table height	28”
Countertop height	36”
Ramp angle	0-30°
Floor lever height	3’
Wall lever height	4’
Wall button height	4’
Doorknob height	3’
Max fall height	12’
Monkey-bar spacing	3’
Min bridge/path width	3’
Chair/seat proportions	1’6” high x 1’4” deep
Min clearance for crawling	2’

MODULAR LADDER KIT - SPECIFICATIONS

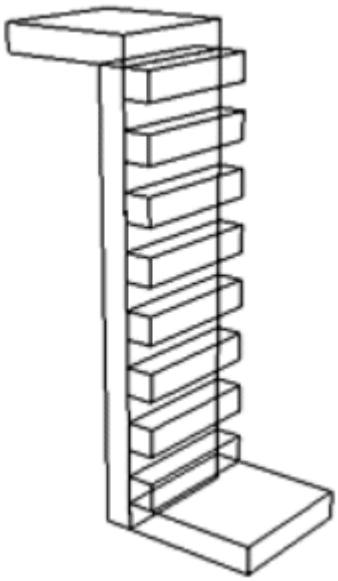
The modular ladder kit is a reusable group of components, geometry, and animations that can be placed in any scene to allow for vertical navigation - ladder climbing.

The following are the dimensions you will need to adhere to when building ladders for your scenes.

- Distance from lower-level ground to top of bottom rung **6 inches**
- Distance from top of top rung to highest-level ground **6 inches**
- Distance between tops of rungs **12 inches**
- Maximum rung thickness **6 inches**
- Distance from wall to front edge of rung **6 inches**
- Minimum ladder width **30 inches**
- Distance between world levels any even number of feet, min. **6 feet** *
- Ladder angles **0 degrees** (straight up and down) **

The ladder kit can be used to traverse vertical distances of six (6) feet or more. Distances shorter than six feet are treated somewha differently, as the avatar no longer has rungs to climb "up" on.

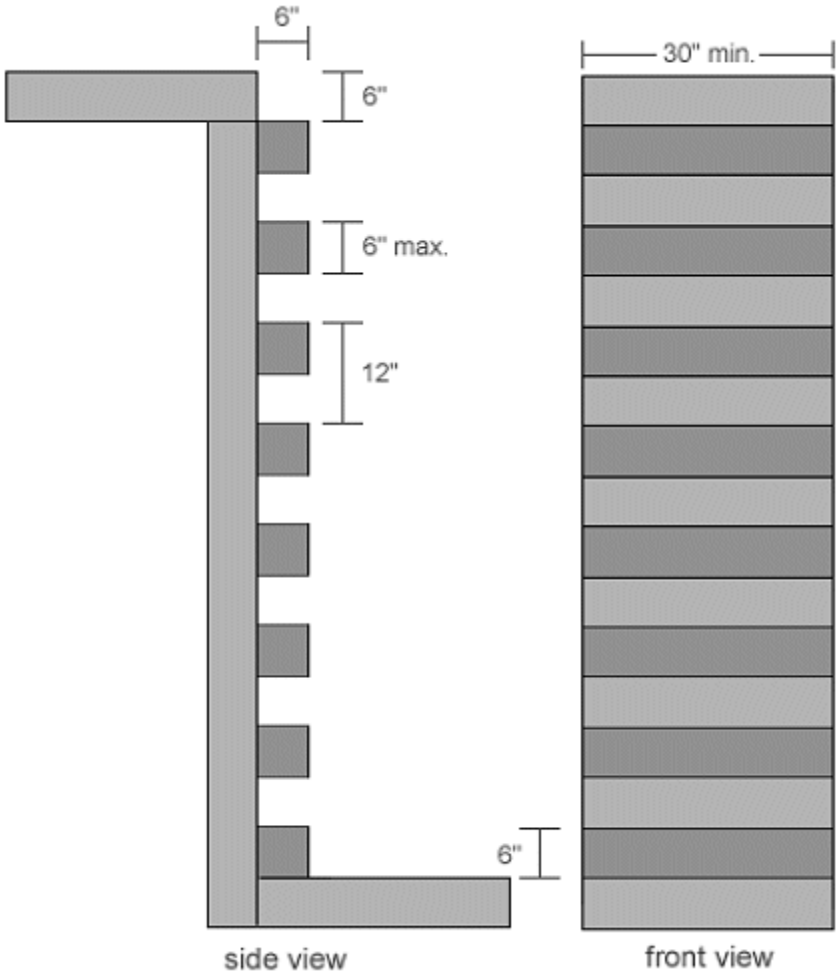
We will provide a group of templates for artists to merge into their scenes. These templates provide geometric guidelines but are no designed to be final art. Use the templates to figure out where to put rungs, where to put the seek point, and how to align with the ground and the floor.



The basic template files are:
12 foot ladder
10 foot ladder
8 foot ladder
6 foot ladder
4 foot stepladder
2 foot stepladder

*two and four feet heights may also be used, but you can not use ladders for climbing. The avatar will instead step or climb onto or clamber over these lev changes. All levels should be an even number of feet apart, or else be ramped.

**eventually, angled ladders will be supported, but not yet.


6"
30" min.
6"
6" max.
12"
6"
side view
front view