Chapter 6

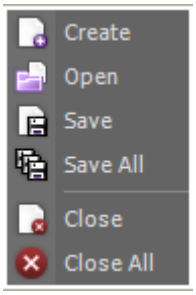
Particular Trails

In this Chapter I'm going to be showing you how to create a Polygon Trail and a Particle System. For the Polygon Trail I'll be following the PolyTrail Demo (PolyTrailDemo-20080905.ste) which can be found in the Code Samples Downloads on the ShiVa Developer Network. For the Particle System I'm going to add a simple "press a button and get an explosion" type of Game.

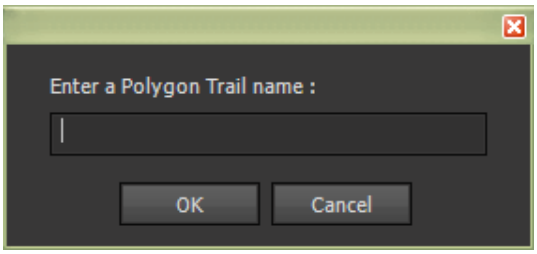
Polygon Trails

The PolyTrail Editor allows the creation and editing of polygon trails. These can be used for such things as spaceship engine exhausts, missile trails etc. and since v1.7 of ShiVa you can now view Polygon Trails in the Scene Viewer.

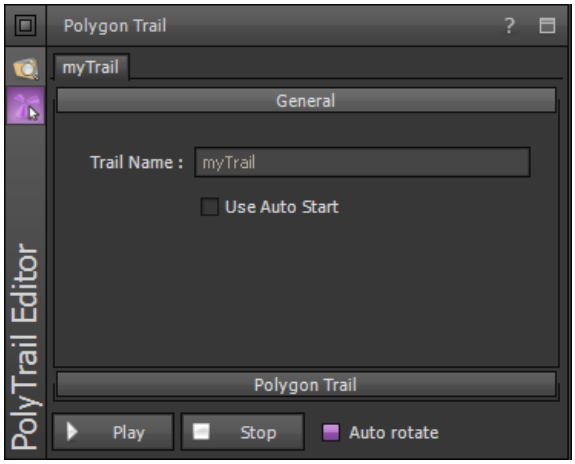
To create a Polygon Trail, open the PolyTrail Editor, click on "Polygon Trail" and then "Create" in the drop-down menu:



This will open the following dialog where you can enter a name for your Trail:



By clicking on "OK", the following will be displayed in the PolyTrail Editor:

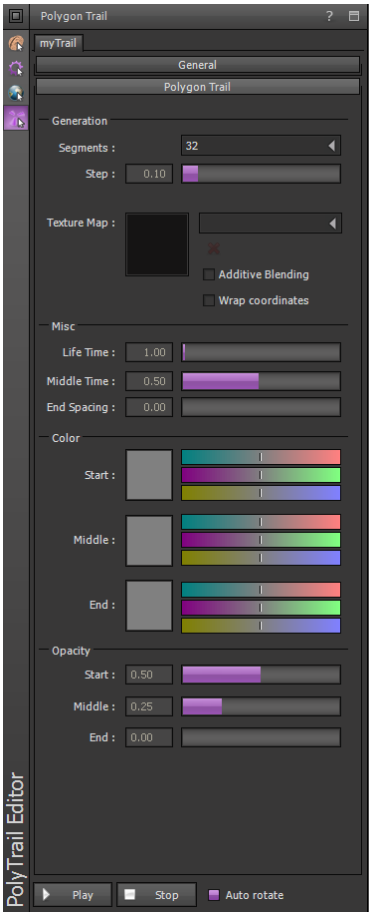


General

You will notice that there are relatively few options in the "General" roll-up. Basically, all you can do is decide if the Trail is to start automatically, or not.

Polygon Trail

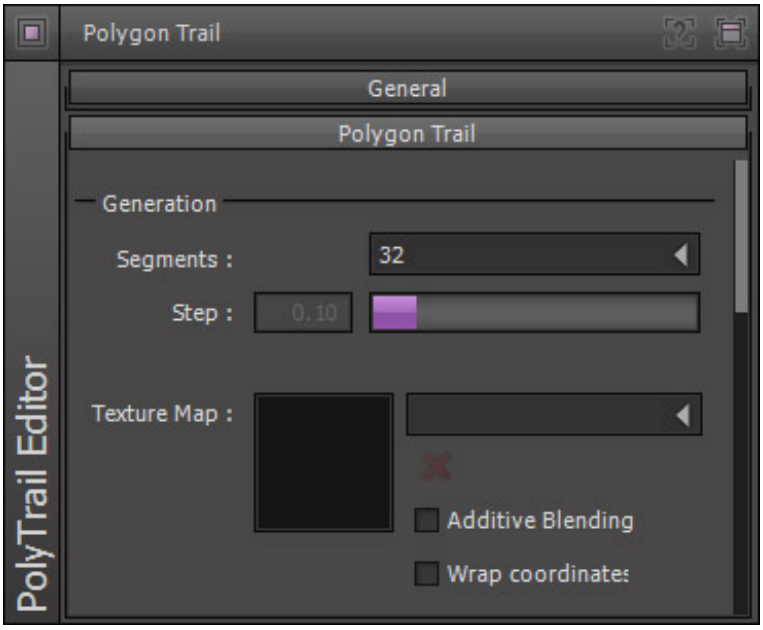
However, if you click on the "Polygon Trail" roll-up, you get the following options:



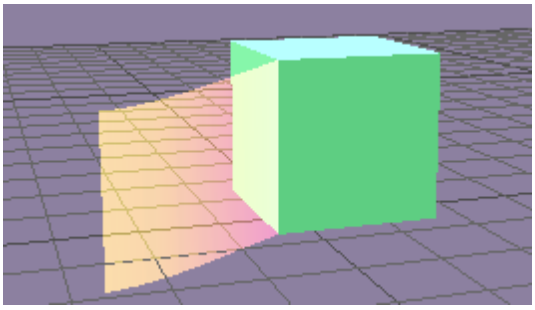
Now, that's more like it! You have lots of options to play with here.

OK, I'll go through the Sections step-by-step:

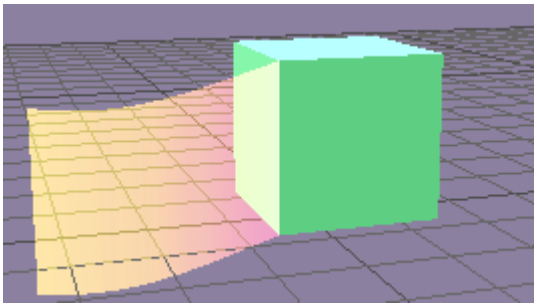
Generation



Segments: This option allows the setting, via the drop-down list, of the maximum number of segments that the Trail will be composed of, and also influences the life time of the Trail. The available values are 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 and 1024.

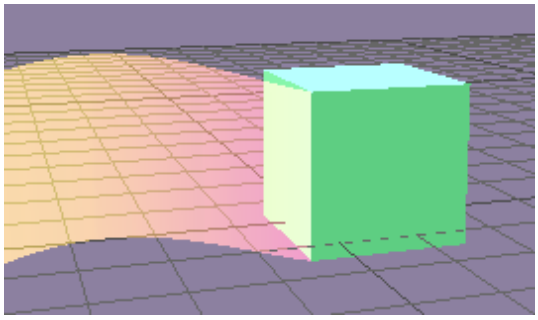


Segments: 32

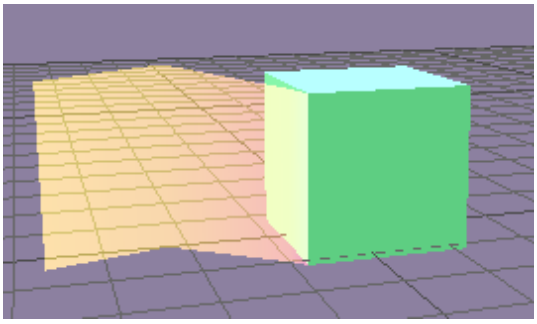


Segments: 512

Step: This option allows the setting of the time taken between each segment generation. The smaller the value, the smoother the Trail will be [0.01 to 1].

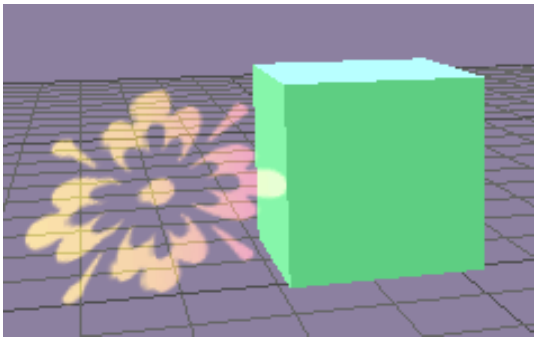


Step: 0.04

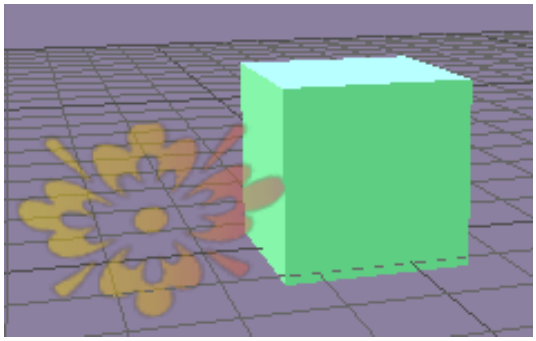


Step: 0.4

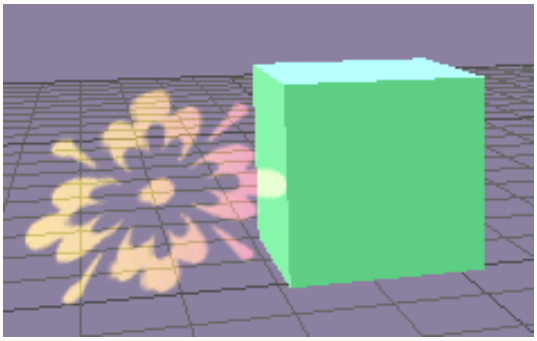
Texture Map: This option allows the setting of a TextureMap to be used to represent the Particles. The selected Texture will be displayed next to the drop-down.
NOTE: all Textures available to be used by the project are displayed in the drop-down list.



Additive Blending: This option, if checked, sets the Trail to use additive rendering.

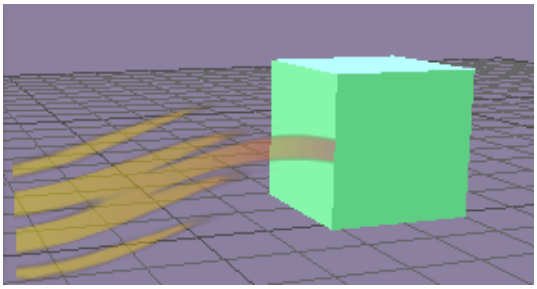


Without additive blending



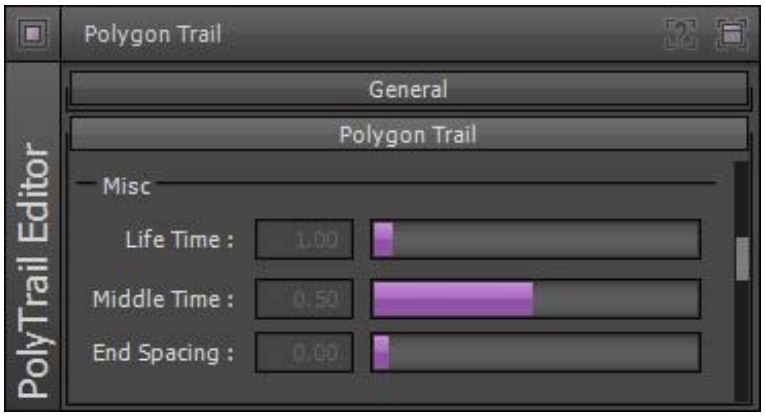
With additive blending

Wrap coordinates: This option, if left unchecked, will cause the Texture to stay on the Trail and act like a flag. If this option is checked, the Texture will scroll at the same speed as the Trail.

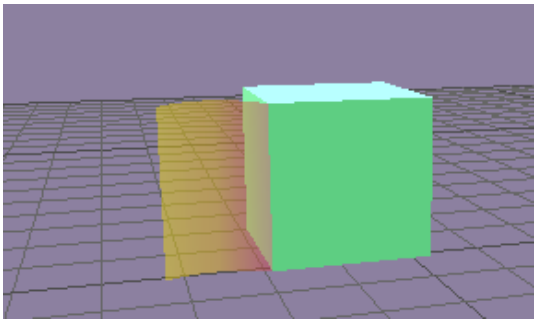


NOTE: to delete the TextureMap, click on  .

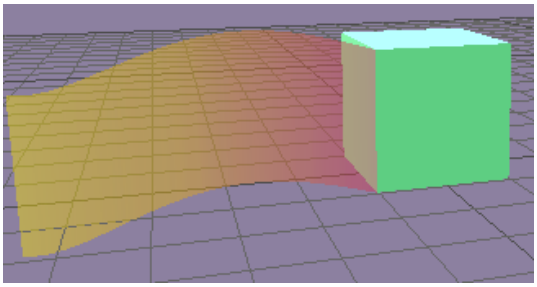
Misc



Life Time: This option allows the setting of the life time of one segment of the Trail (in seconds) [0.01 to 60].

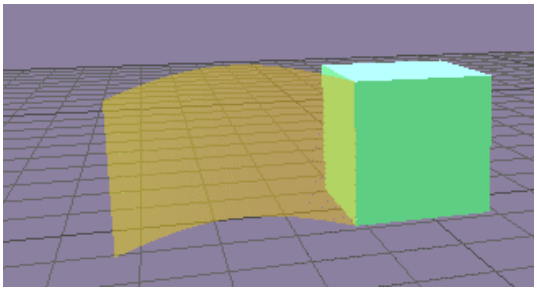


Life Time: 0.4

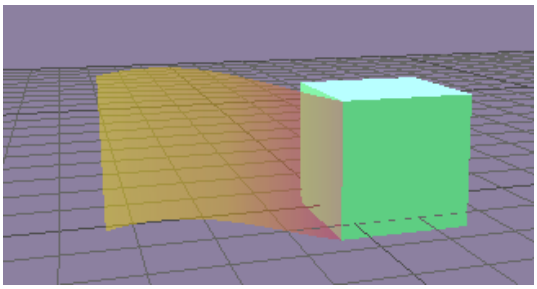


Life Time: 1.61

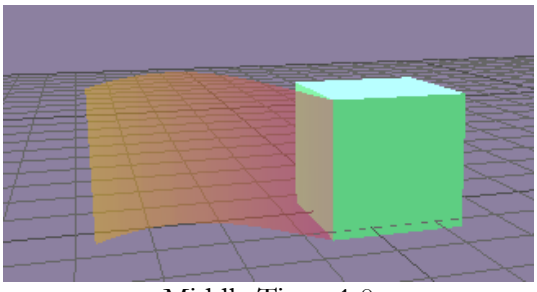
Middle Time: This option allows the setting of the percentage of the Trail that will be affected by the middle colour [0 to 1].



Middle Time: 0.0

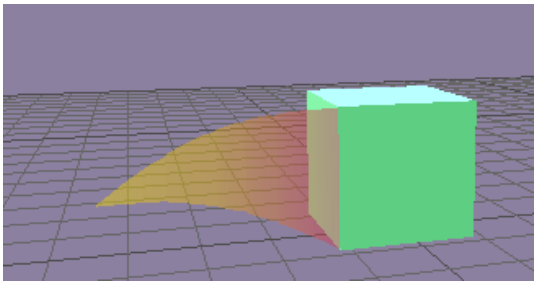


Middle Time: 0.5

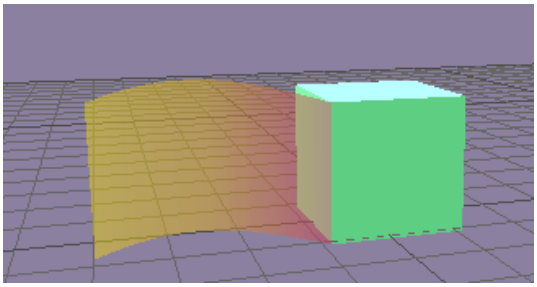


Middle Time: 1.0

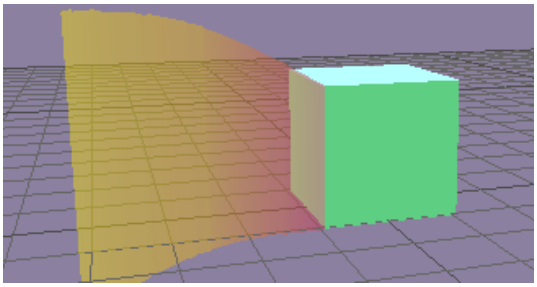
End Spacing: This option allows the setting of the size of the Trail based on the start size (in percent) [0 to 100].



End Spacing: 0.0

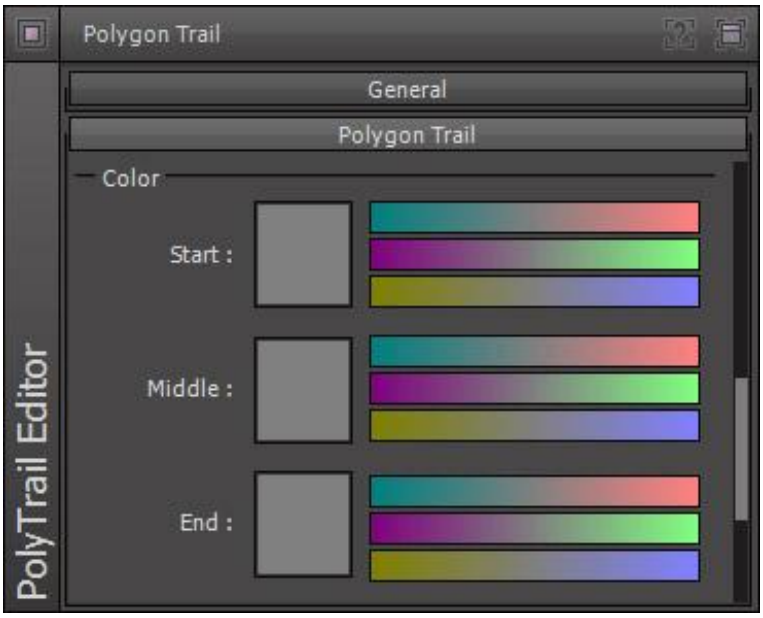


End Spacing: 1.0



End Spacing: 2.0

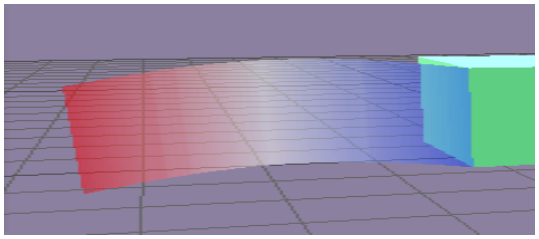
Color



Start: This option allows the setting of the start colour of the Trail.

Middle: This option allows the setting of the middle transition colour of the Trail, as influenced by the "Middle Time" property.

End: This option allows the setting of the end colour of the Trail.



Start = blue; Middle = white; End = red

NOTE: all colours can be selected using standard colour picker dialog (see '**Colour Dialog**' in Chapter 10 – Attributed to You, for more information).

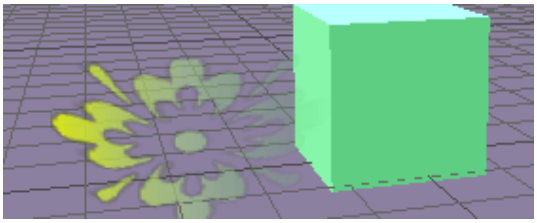
NOTE: the selected colours can also be altered by using the three slider bars to the right of the colour box.

Opacity

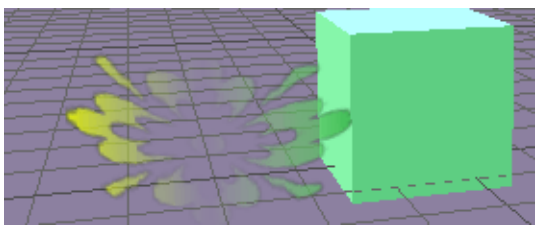
Start: This option allows the setting of the start Opacity of the Trail (in percent) [0 to 1].

Middle: This option allows the setting of the middle time Opacity of the Trail, as influenced by the "Middle Time" property, (in percent) [0 to 1].

End: This option allows the setting of the end Opacity of the Trail (in percent) [0 to 1].



Start=0.0; Middle = 0.5; End = 1.0



Start=1.0; Middle = 0.0; End = 1.0

Now that you know what all the options are, I'll guide you through creating a jet engine exhaust:

Oops! Forgot to mention the buttons at the bottom of the Editor....

The "Play" and "Stop" buttons allow you to see your Particle Trail in the Scene Viewer, by "Playing" and "Stopping" the Trail respectively. The "Auto rotate" check-box, when checked, rotates the Trail around a central point.

OK, back to the exhaust.

To start with select "32" for the "*Segments*" Value. This gives a reasonable PolyTrail.

Then, I would suggest using the following for the "***Misc***" values:

LifeTime: 0.75
Middle Time: 0.50
End Spacing: 0.00 - This is because we want a tapering shape for the exhaust.

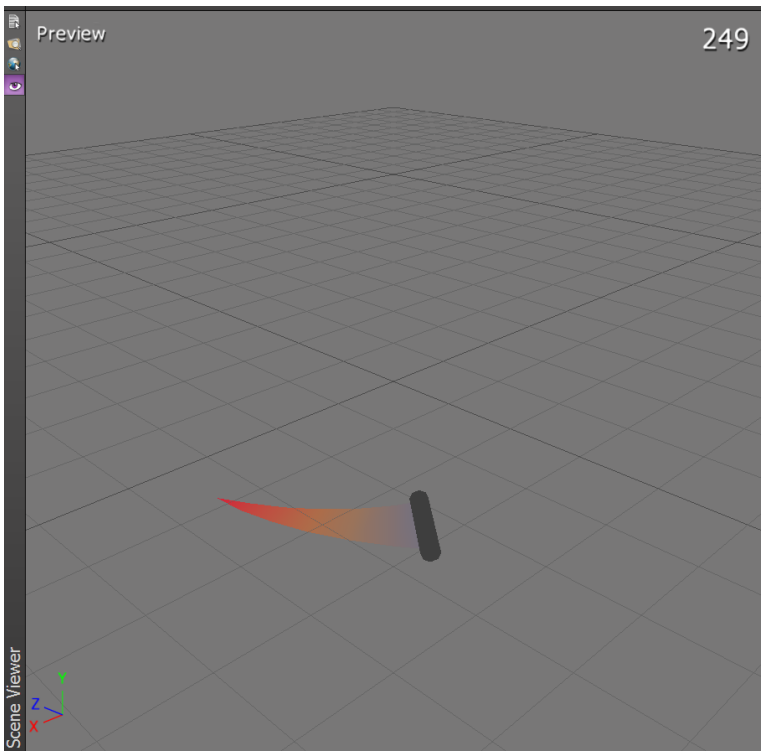
Next, select the following for the "***Color***" values (these can really be anything you want):

Start:	R:70 – G:95 – B:185	-	A blue colour
Middle:	R:220 – G:115 – B:35	-	An orange colour
End:	R:250 – G:5 – B:35	-	A red colour

Finally, enter the following for the "***Opacity***" values:

Start: 0.25
Middle: 0.50
End: 0.75

If you now click on the "Start" button (with "Auto rotate" checked), you should see something similar to the following in the Scene Viewer:



Not too bad! Feel free to play around with the values and see what you can come up with.

Next, I'm going to show you how to use your new PolyTrail in a quick application. In this Chapter, and those that follow, I'll be using ShiVa demos, where possible, to illustrate the concepts of each section. Where a demo is available, you can either download it from the ShiVa Developer Network (SDN) "Downloads" area, or it will be one of the ShiVa demos bundled with ShiVa (don't worry, I'll let you know exactly where to get it from!).

OK, for the PolyTrail, there is a demo available on SDN. It is available in the "Downloads" area, and is called "PolyTrailDemo-20080905.ste".

Once you have downloaded the file, you will need to create a new project in ShiVa. I would suggest for the purposes of these few Chapters, that you create a new folder for each one (i.e. "/ShiVaBook/PolyTrail"). This is done via the "Settings" option in the "Main" menu.

This demo consists of the following files (in addition to those automatically created by ShiVa when you create a new project):

Games	-	PolyTrailDemo
Models	-	Billboard Dummy Ground PointLight
AIModels	-	AutoRotate PolyTrailDemoMain
Materials	-	Ground02 Herbouille ScieRotative_Material000 ScieRotative_Material001
Meshes	-	Billboard_Mesh000 ground_Sphere01-mesh Plane ScieRotative_Mesh000 ScieRotative_Mesh001 TestSphere
Scripts	-	AutoRotate_Handler_onStart AutoRotate_Handler_onStop AutoRotate_State_onEnter AutoRotate_State_onLoop AutoRotate_State_onLeave PolyTrailDemoMain_Handler_onInit PolyTrailDemoMain_Handler_onMouseButtonDown PolyTrailDemoMain_Handler_onMouseMove PolyTrailDemoMain_Handler_onPauseTrails PolyTrailDemoMain_Handler_onStartTrails PolyTrailDemoMain_Handler_onStopTrails

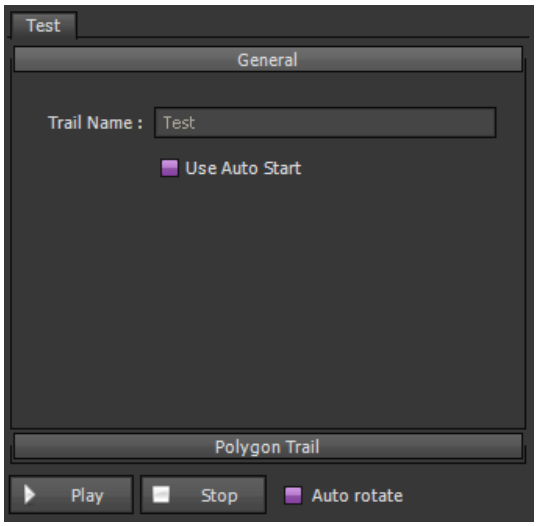
Textures	-	buttPrint f8sgrass grassWalpha Ground00NM q3dm18_Texture001 TXT_SphereMap_Garden
Trails	-	Test
Scenes	-	PolyTrailDemo

The ones that I've highlighted in Red will be the files that I'll be dissecting here. Since this Chapter is about PolyTrails, I think it would be best to start with the "Test" Trail:

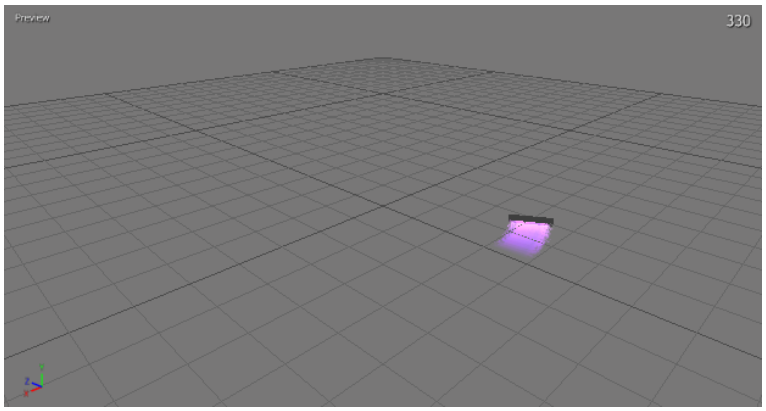
So, open up the PolyTrail Editor, and we'll get started.

If you click on the "Polygon Trail" menu, and then "Open", you can select the "Test" Trail. Don't forget that you can also click on the icon in the middle of the editor to open a Trail!

You will now see the following displayed in the editor:



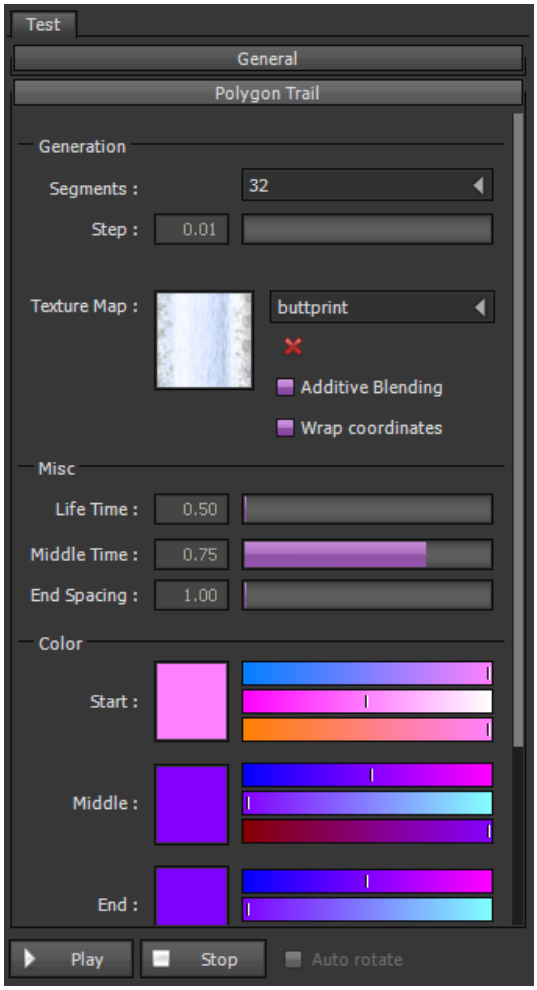
If you open up the Scene Viewer module, you can click on the "Play" button, and you should see something similar to the following:



OK, so how was that created?

Firstly note, in the "General" roll-up, that "Use Auto Start" is checked. This will ensure that the PolyTrail will start as soon as it is instantiated.

Now into the more interesting bit. Open up the "Polygon Trail" roll-up, and you will see the following:

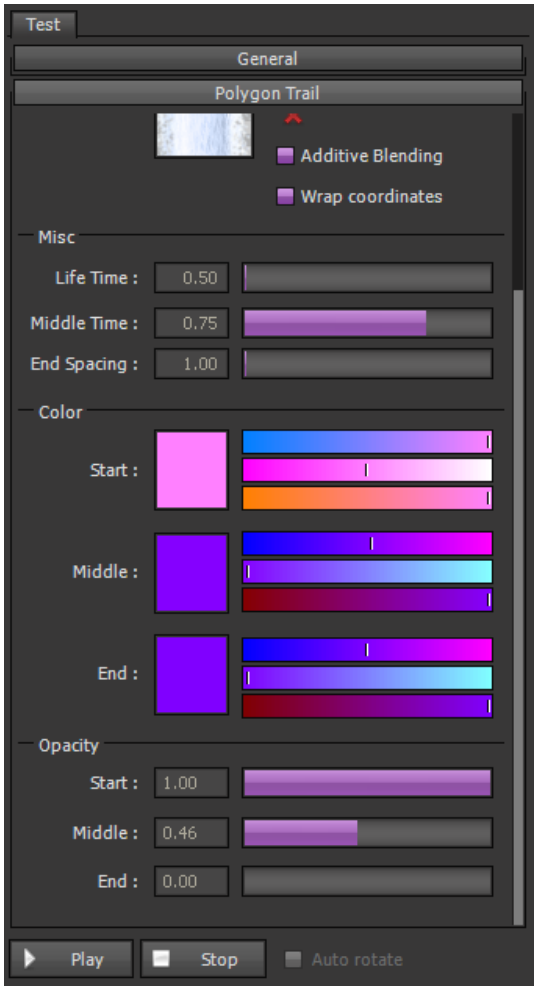


Now, taking it step by step:

In the "Generation" area, you can see that "Segments" has been set to "32" to generate a reasonable PolyTrail. Also, you can see that "Step" is set to "0.01", which basically means that it will be a very smooth Trail. Finally, you can see that a "Texture Map" has been used ("buttprint" – I didn't name this, and I'm sure it means something else in French!). Also, the "Additive Blending" and "Wrap coordinates" options have been checked, meaning that the Texture will be blended onto the Trail (using addition) and that it will look like a flag streaming behind the Object that it is attached to.

In the "Misc" area, you can see that the "Life Time" of the Trail has been set to "0.5", which means that it will only last for a short time. Also, the "Middle Time" has been set to "0.75" which sets the middle colour to be visible for 75% of the life of the Trail. Finally, in this area, "End Spacing" has been set to "1.00", which will give a Trail that is the same size all the way along.

Moving into the "Color" area, you can see that the Trail will start as a pinkish colour, which will turn purple for the "Middle" and "End" of the Trail.



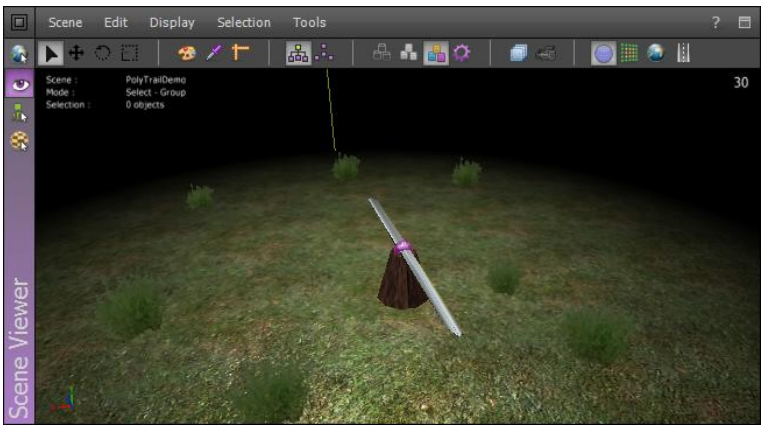
The last area, "Opacity", has the following settings:

"Start"	-	1.00
"Middle"	-	0.46
"End"	-	0.00

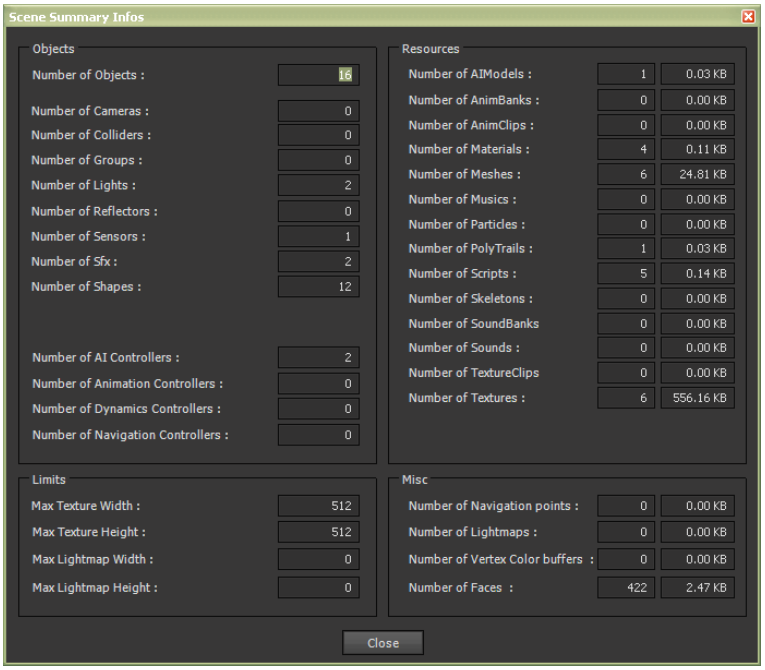
These settings mean that the Trail will start fully opaque, and gradually disappear, as you can see from the original screenprint.

Well, that's it for the PolyTrail itself, so now I'll move on to the Scene that is used in this demo.

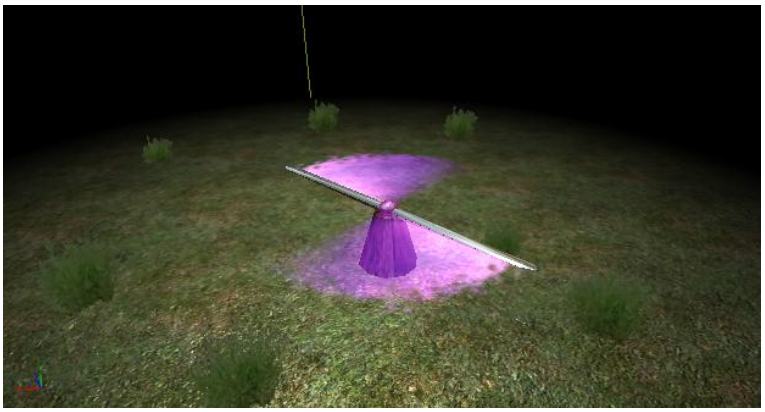
If you switch to the Game Editor, and load up the Scene you will see the following in the Scene Viewer:



If you click on the "Scene" menu, and select the "Infos" option, ShiVa will display the following dialog:

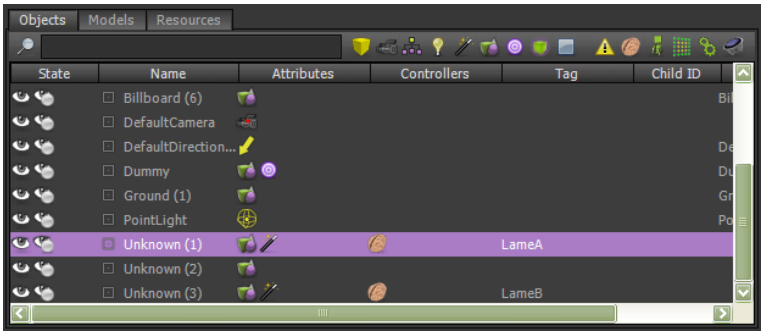


As you can see, there is quite a lot to this Scene, with 16 Objects in total! The one that I'll concentrate on is the "Number of Sfx:" line, which shows that there are 2 effects attached to Objects. Why two you may ask? Well, if you click on the () button, you will see the arms of the windmill-like structure rotate, with a PolyTrail trailing behind each one:



Pretty effective isn't it?

So, how do we actually create this type of Scene? That's quite easy (I won't be going into creating the whole Scene, just how to attach the PolyTrail). If you open up the Scene in the Scene Viewer (after having stopped the Scene running of course!), and then open up the Scene Explorer, you will see the following:

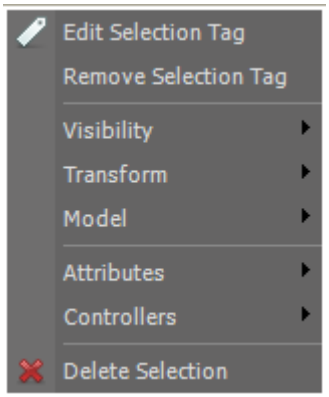


If you look at the Objects "Unknown (1)" and "Unknown (2)", you will see that they both have a little wand () in the Attributes column. This shows that the Object has an Sfx attached, you can also see that they both have AI Controllers attached (in this case "AutoRotate"), and that they have been given Tags ("LameA" and "LameB").

If you double-click on "Unknown (1)", the view in the Scene Viewer will zoom in on the Object:

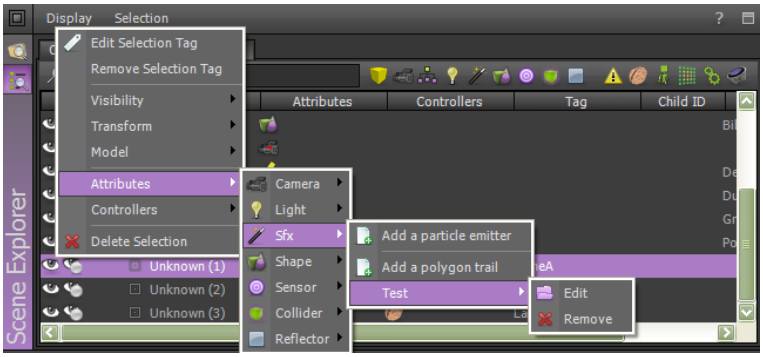
Also, you may have noticed that the Object has green lines denoting its Polygons.

Moving back to the Scene Explorer, by right-clicking on the Object you will be presented with a menu of all the Attributes etc. that you can attach to it:



You will notice that the first two options allow the editing, and removal, of the Tag of the Object. This is where the Tag can also be set (using the "Edit Selection Tag" option), if the Object doesn't already have one attached.

So, how do you add an Sfx? This is done by selecting the "Attributes" option, and then the "Sfx" option that appears in the sub-menu. This generates a further menu, allowing the addition of Particle Emitters, and also Polygon Trails. In the case of the Scene here, you will see that "Test" appears below the option to add Polygon Trails:



As shown above, you can edit or remove the Trail from here as well. If you select the "Edit" option, the selected Trail will be opened in the PolyTrail Editor.

OK, so you now know how to add Polygon Trails to Objects, so what's next?

The next bit, as always, is to script how the Scene will work. For this, I am going to explain the relevant Scripts for this demo.

AutoRotate

This AIModel consists of the following:

Variables

bStopped	-	Boolean	-	Initial Value – "false"
fGlobalSpeed	-	Number	-	Initial Value – 4.000
fSpeedX	-	Number	-	Initial Value – 25.000
fSpeedY	-	Number	-	Initial Value – 25.000
fSpeedZ	-	Number	-	Initial Value – 10.000

States

Idle

Handlers

onStart ( )
onStop ( )

There's not really a lot to this AIModel:

onStart ( )

This Script consists of just one line:

this.bStopped (false)

All this line does is set the LOCAL AIModel Variable "bStopped" to "false".

onStop ( )

this.bStopped (true)

I shouldn't have to explain this!

Idle.onEnter ( )

This Script actually does nothing, since the main line is commented out, so I won't explain anything at this point.

```
local hLinkedItem = this.getObject ( )

if ( hLinkedObject ~= nil )
then
    --object.resetRotation ( hLinkedObject, object.kParentSpace )
end
```

Idle.onLeave ( )

OK, this script is a little longer, but not much!:

```
local hLinkedObject = this.getObject ( )

if ( hLinkedObject ~= nil )
then
    object.resetRotation ( hLinkedObject, object.kParentSpace )
else
    log.warning ( "Bad object..." )
end
```

I'll only comment on one line here, since the rest should be almost second nature to you by now.

This Script basically resets the rotation of the Object (in "Parent Space"), using the "object.resetRotation" Function. Note here that we are using "Parent Space", not the usual "Global Space". This is because the Object has been set up with other Objects as Children of a Parent and, as such, most actions should happen in "Parent Space" so that all of the Objects interact correctly.

Idle.onLoop ( )

Another fairly short Script that basically rotates the Object.

```
if ( not this.bStopped ( ) )
then
```

The first couple of lines check to see if the LOCAL Variable "bStopped" is "false". If it is, then we continue:

```
    local hLinkedObject = this.getObject ( )

    if ( hLinkedObject ~= nil )
    then
```

No comment!

```
        local f = this.fGlobalSpeed ( )

        log.message ( f, " ", this.fSpeedX ( ), " ", this.fSpeedY ( ), " ", this.fSpeedZ ( ) )

        object.rotate ( hLinkedObject,
                f * this.fSpeedX ( ) * application.getLastFrameTime ( ),
                f * this. fSpeedY ( ) * application.getLastFrameTime ( ),
                f * this. fSpeedZ ( ) * application.getLastFrameTime ( ),
                object.kGlobalSpace )
    end
end
```

The last few lines set a LOCAL Variable ("f") to be equal to the AI Variable "fGlobalSpeed", prints the current values of "f", "fSpeedX", "fSpeedY" and "fSpeedZ" to the Log, and then rotates (in Global Space) the Object using these values and the value of the last Frame time.

PolyTrailDemoMain

This AIModel consists of the following:

Handlers

onInit ( )
onMouseButtonDown (nButton, nPointX, nPointY, nRayPntX, nRayPntY, nRayPntZ, nRayDirX, nRayDirY, nRayDirZ)
onMouseMove (nPointX, nPointY, nDeltaX, nDeltaY, nRayPntX, nRayPntY, nRayPntZ, nRayDirX, nRayDirY, nRayDirZ)
onPauseTrails ( )
onStartTrails ( )
onStopTrails ( )

There's not really a lot to this AIModel either:

onInit ( )

application.setCurrentUserScene ("PolyTrailDemo")

Nice easy one-liner that sets the current Scene.

onMouseButtonDown (nButton, nPointX, nPointY, nRayPntX, nRayPntY, nRayPntZ, nRayDirX, nRayDirY, nRayDirZ)

```
local s = application.getCurrentUserScene ( )

if ( s~= nil )
then
```

Fairly standard stuff to start with.

```
   local o = scene.getFirstHitSensor ( s, nRayPntX, nRayPntY, nRayPntZ, nRayDirX, nRayDirY, nRayDirZ, 100 )
```

All that happening in this line is that we are using the values from the Mouse (position & direction of the Ray) to see if there is any Sensor Objects within "100" units of the Mouse Cursor in the direction of the Ray.

```
   if ( o ~= nil )
   then
      this.sendEvent ( "onPauseTrails" )
      this.postEvent ( 0.5, "onStartTrails" )
   end
end
```

If there is, then we send an Event to "onPauseTrails" to pause the Trail, and we also post an Event to "onStartTrails" to restart the Trails. The posted Event is delayed by half a second. So, if you click on a Trail in the demo, then the Trails will pause for half a second, before restarting.

onMouseMove (nPointX, nPointY, nDeltaX, nDeltaY, nRayPntX, nRayPntY, nRayPntZ, nRayDirX, nRayDirY, nRayDirZ)

```
local s = application.getCurrentUserScene ( )

if ( s~= nil )
then
```

Again, fairly standard stuff to start with.

```
   local maxDist = 100
   local o, d, id = scene.getFirstHitSensor ( s, nRayPntX, nRayPntY, nRayPntZ, nRayDirX, nRayDirY, nRayDirZ, maxDist )
```

OK, all we are doing here is almost identical to the previous Handler, but we have used a Variable ("maxDist") in place of the hardcoded "100", to represent the maximum distance of the Ray. This is how coding of values that may change should be done, as it is a lot easier to change one Variable, than to search for each instance of the hardcoded amount!

The only other point I will make is that, this time, we are retrieving not only the Object, but also the distance ("d") and the ID ("id") of the Object. These are used in the following to write a Log message:

```
    if ( o ~= nil )
    then
        log.message ( "A sensor with ID " ..id.." is under the mouse, exactly at "..d.." units from
the camera" )
    end
end
```

In the Log message, you can also see how to use ".." to concatenate Strings.

onPauseTrails ( )

```
local s = application.getCurrentUserScene ( )

if ( s~= nil )
then
    local a = scene.getTaggedObject ( s, "LameA" )
    local b = scene.getTaggedObject ( s, "LameB" )

    sfx.pauseAllTrails ( a )
    sfx.pauseAllTrails ( b )
end
```

Once again, a fairly simple Script that shows how to obtain an Handle to an Object using the Tag applied to the Object in the Scene Viewer ("getTaggedObject"), and how to pause Trails ("pauseAllTrails"). Note that all Objects are referred to using the "scene" API, and that all Trails are referred to using the "sfx" API. This is how ShiVa has been set up, so that each API is called using its name, then a ".", then the relevant Function. This makes the code much easier to read.

onStartTrails ( )

```
local s = application.getCurrentUserScene ( )

if ( s~= nil )
then
    local a = scene.getTaggedObject ( s, "LameA" )
    local b = scene.getTaggedObject ( s, "LameB" )

    sfx.startAllTrails ( a )
    sfx.startAllTrails ( b )
end
```

Since this is almost identical to "onPauseTrails", I'll just point out the use of "startAllTrails" to start a Trail.

onStopTrails ( )

```
local s = application.getCurrentUserScene ( )

if ( s~= nil )
then
   local a = scene.getTaggedObject ( s, "LameA" )
   local b = scene.getTaggedObject ( s, "LameB" )

   sfx.stopAllTrails ( a )
   sfx.stopAllTrails ( b )
end
```

Again, identical to "onPauseTrails", except for the use of "stopAllTrails" to stop the Trails.

Well, that's it for the PolyTrail demo, I'll now explain ShiVa's Particle System, complete with a simple demo.

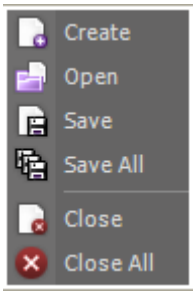
Particle Systems

The Particle Editor module allows the creation and editing of the resources required for a Particle System.

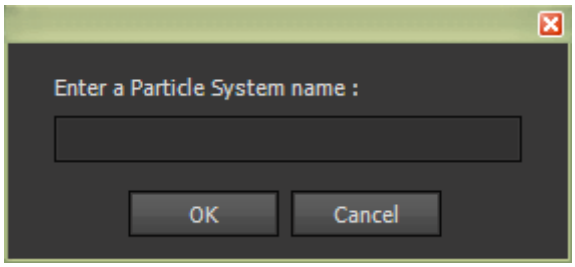
NOTE: particle systems MUST be attached to Objects.

Basically, the Particle Editor allows you to create & edit Particles for effects such as fire, snow & water.

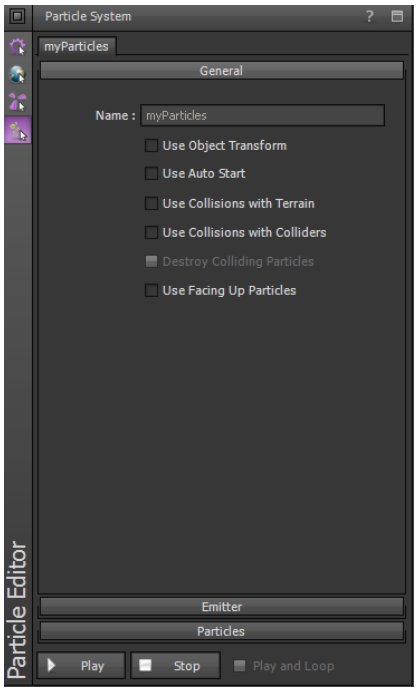
To create a Particle System, open the Particle Editor, click on "Particle System" and then "Create" in the drop-down menu:



This will open the following dialog where you can enter a name for your Particle System:



By clicking on "OK", the following will be displayed in the Particle Editor:



General

The "General" roll-up is slightly more interesting than the PolyTrail Editor, in that you can set various options:

Use Object Transform: This option, when checked, applies the Object's transformation to the Particle System.

Use Auto Start: This option, when checked, automatically starts the Particle generation when it is instantiated.

Use Collisions with Terrain: This option, when checked, enables the Particle System to collide with the Terrain, rather than passing straight through it!

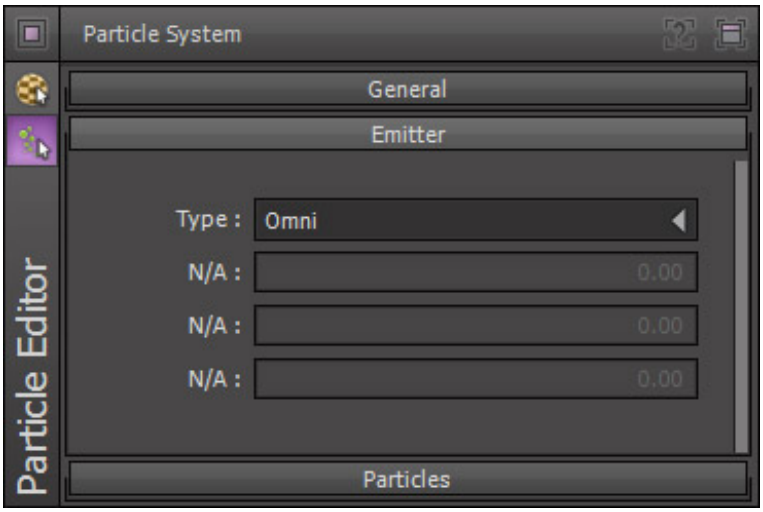
Use Collisions with Colliders: This option, when checked, enables the Particle System to collide with other Objects that have been set as Colliders, rather than passing straight through them!

Destroy Colliding Particles: This option is only available if one of the "Use Collisions with..." options is selected, and will end the Particle's life immediately on collision with either the Terrain, or with Colliders.

Use Facing Up Particles: This option allows the Particles to be created using a planar XZ plus Y.

Emitter

Clicking on the "Emitter" roll-up produces the following options:



Type: This option allows the setting of the type of the emitter. Depending on the selected type, different parameters are available:

Omni:

There are no additional Parameters for this option.

Cone:

Angle: This option allows the setting of the aperture of the Cone, in degrees, when compared to the Z-axis of the transmitter.

Circle:

Angle: This option allows the setting of the Particle emission angle, in degrees, compared to the Z-axis of the transmitter.

Radius: This option allows the setting of the radius of the Circle of the Particle emission.

Disk:

Angle: This option allows the setting of the Particle emission angle, in degrees, compared to the Z-axis of the transmitter.

Radius: This option allows the setting of the radius of the Disk of the Particle emission.

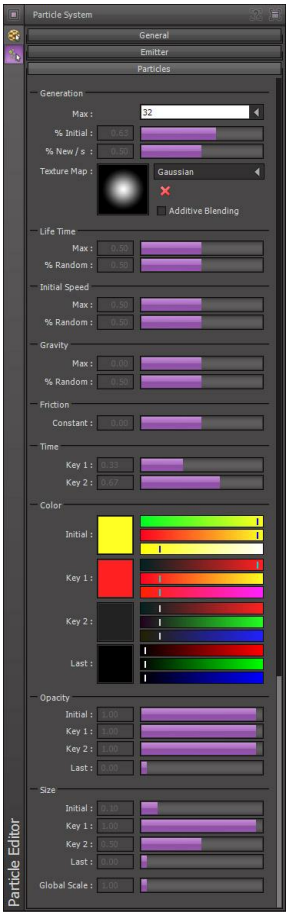
Sphere:

Direction: This option, if positive, will mean that the Particles will be emitted outside of the Sphere, and conversely if negative.

Radius: This option sets the radius of the Sphere of the Particle emission.

Particles

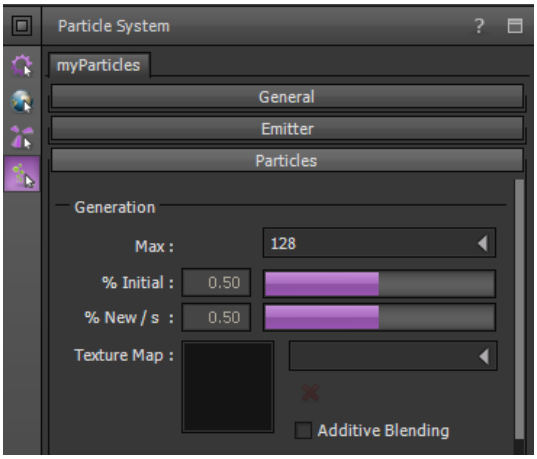
Clicking on the "Particles" roll-up will display the following options:



OK, once again lots of options to play with!

As with the PolyTrail Editor, I'll go through them step-by-step:

Generation



Max: This option allows the setting of the maximum number of simultaneous live Particles. The available values are: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 & 4096.

% Initial: This option allows the setting of the initial percentage of live Particles, relative to the maximum number of Particles [0 to 1].

% New / s: This option allows the setting of the percentage of Particles generated per second, relative to the maximum number of Particles [0 to 1].

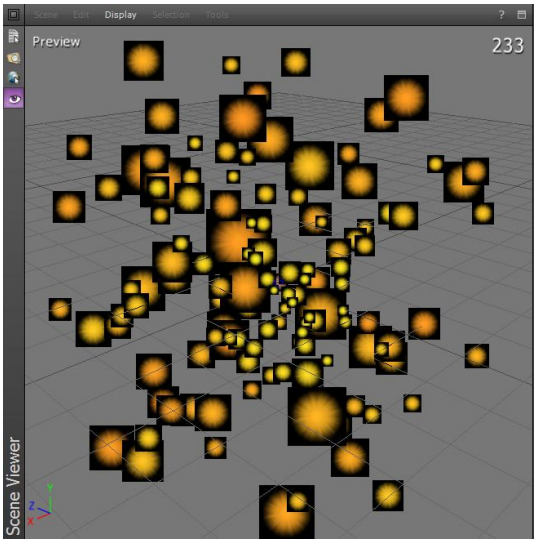
Texture Map: This option allows the selection of the TextureMap to be used to represent the Particles. The selected Texture will be displayed next to the drop-down.
NOTE: all Textures available to be used by the Project are displayed in the drop-down list.

Additive Blending: This option, if checked, sets the Particles to use additive rendering.
NOTE: this is useful for luminous Particles.

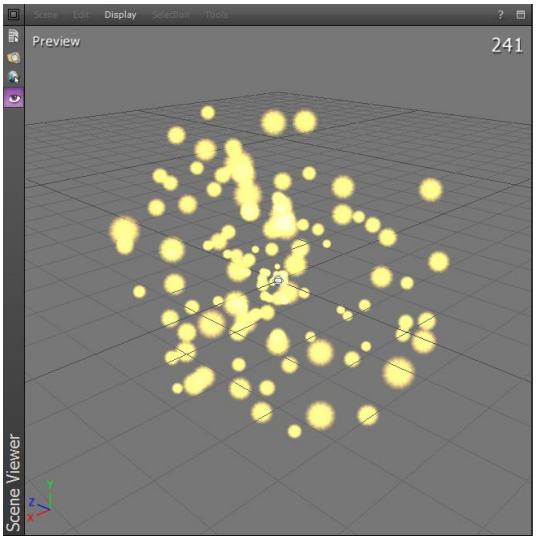
NOTE: to delete the TextureMap, click on  .

Unfortunately, with the Particle System, it is extremely difficult to show how changing some of these values affects the generated Particles. I could show you how it looks with different "Texture Maps", but I'll leave that for you to try out!

The only thing I will show, at this point, is the difference with and without "Additive Blending":



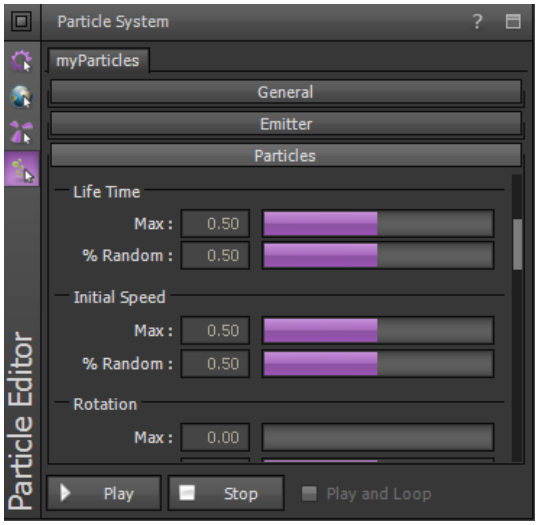
without Additive Blending



with Additive Blending

Well, that looks heaps better doesn't it?

Life Time



Max: This option allows the setting of the maximum life span of the Particles [0 to 1].

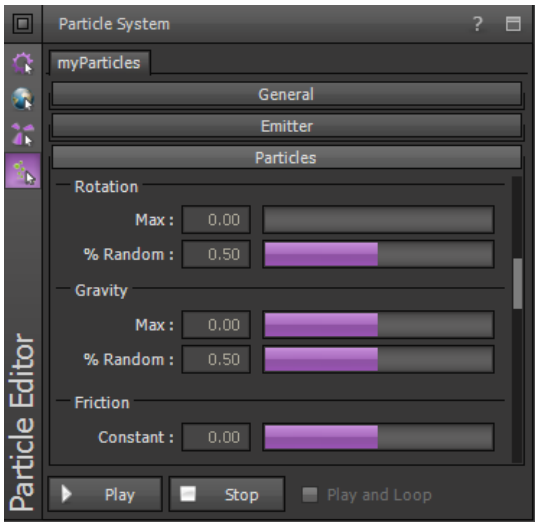
% Random: This option allows the setting of the randomness coefficient of the life time of the Particles [0 to 1].

Initial Speed

Max: This option allows the setting of the maximum initial speed of the Particles [0 to 1].

% Random: This option allows the setting of the randomness coefficient of the initial speed of the Particles [0 to 1].

Rotation



Max: This option allows the setting of the maximum rotational force of the Particles [0 to 1].

% Random: This option allows the setting of the randomness coefficient for the rotation of the Particles [0 to 1].

Gravity

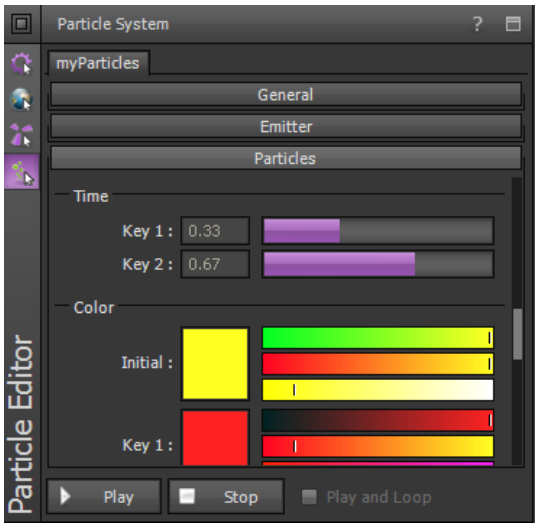
Max: This option allows the setting of the maximum gravitational force of the Particles [-1 to 1].
NOTE: a negative gravity is allowed.

% Random: This option allows the setting of the randomness coefficient for the gravity of the Particles [0 to 1].

Friction

Constant: This option allos the setting of the friction coefficient of the Particles [-1 to 1].
NOTE: a negative value is allowed, which results in acceleration of the Particles.

Time

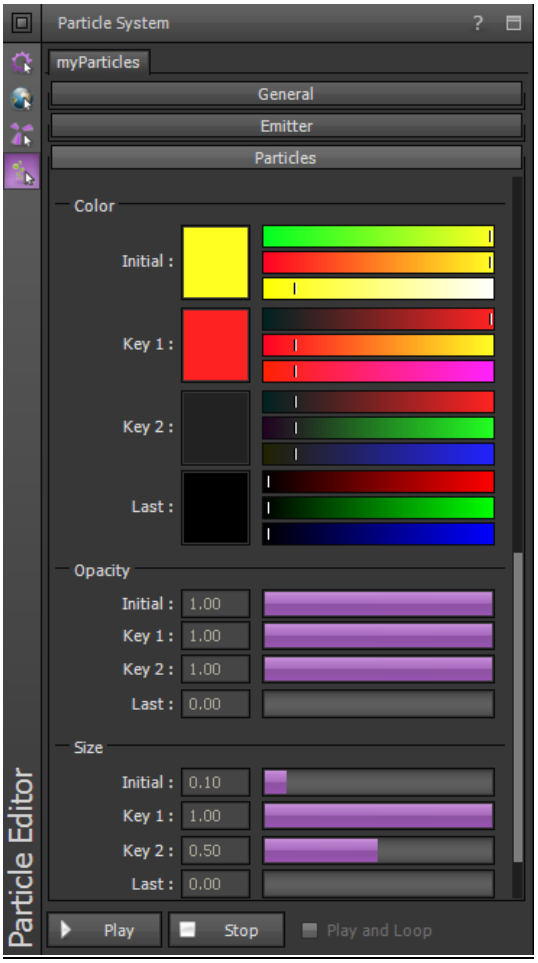


Key 1: This option allows the setting of the first keytime, as a percentage of the life span of the Particle [0 to 1].

Key 2: This option allows the setting of the second key time, as a percentage of the life span of the Particle [0 to 1].

These "Key" values will be used in the next three sections of the roll-up, as you will soon see.

Color



Initial: This option allows the setting of the initial colour of the Particles.

Key 1: This option allows the setting of the colour of the Particles when 'Key 1' is reached.

Key 2: This option allows the setting of the colour of the Particles when 'Key 2' is reached.

Last: This option allows the setting of the final colour of the Particles.

NOTE: all colours can be selected using standard colour picker dialog.

NOTE: the selected colours can also be altered by using the three slider bars to the right of the colour box.

Opacity

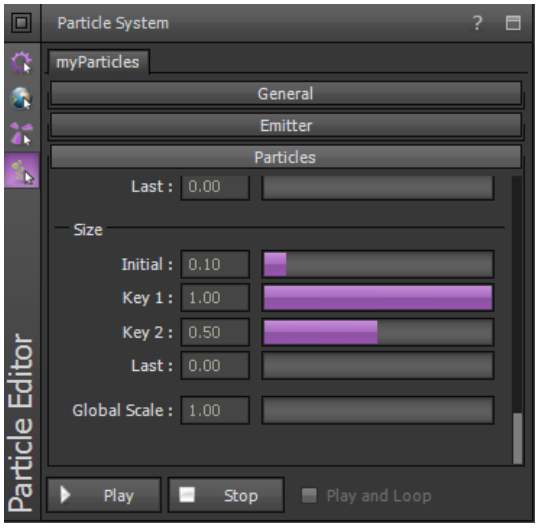
Initial: This option allows the setting of the initial Opacity of the Particles [0 to 1].

Key 1: This option allows the setting of the Opacity of the Particles when 'Key 1' is reached [0 to 1].

Key 2: This option allows the setting of the Opacity of the Particles when 'Key 2' is reached [0 to 1].

Last: This option allows the setting of the final Opacity of the Particles [0 to 1].

Size



Initial: This option allows the setting of the initial size of the Particles [0 to 1].

Key 1: This option allows the setting of the size of the Particles when 'Key 1' is reached [0 to 1].

Key 2: This option allows the setting of the size of the Particles when 'Key 2' is reached [0 to 1].

Last: This option allows the setting of the final size of the Particles [0 to 1].

Global Scale: This option allows the setting of the global scale of the Particles [1 to 32].

NOTE: where any of the above options show a range of [0 to 1], this normally equates to a range of [0% to 100%].

You will no doubt have noticed that there aren't a lot of pictures for the Particle Editor. Since the majority of the above values affect the Particle Editor at runtime, it is very difficult to show you in print, but try some different values for yourself, and you'll soon get the hang of it!

OK, that's it for the Particle Editor. I'm now going to show you how to use it in a quick application.

OK, for the Particle System, there is a demo available (simply called "Fire") in the standard ShiVa demos which are available in the "Downloads" area, and are called "Installation_Samples.zip".

Once you have downloaded and extracted the file, you will need to create a new project in ShiVa.

This demo consists of the following files (in addition to those automatically created by ShiVa when you create a new project):

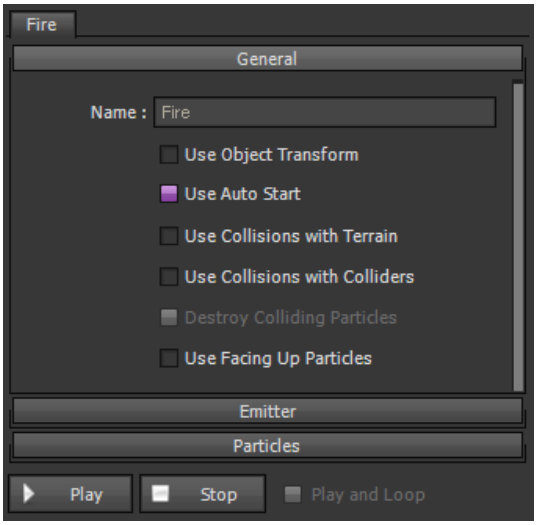
Games	-	Fire
Models	-	StaticPointLight Ball_fire (Shape) Box (Shape) Box (Shape)
AIModels	-	Fire_Main
Materials	-	cornell_box_Material000 cornell_box_Material001 cornell_box_Material002 cornell_box_Material003 cornell_box_Material004 Fire_Ball
Meshes	-	cornell_box_Mesh000 cornell_box_Mesh001 cornell_box_Mesh002 cornell_box_Mesh003
Particles	-	Fire
Scripts	-	Fire_Main_Function_loadScene Fire_Main_Handler_onInit
Textures	-	Fire Flame
Scenes	-	Fire

Again, the ones highlighted in Red will be the files that I'll be dissecting here. Since this part of the Chapter is about Particles, I think it would be best to start with the "Fire" Particle:

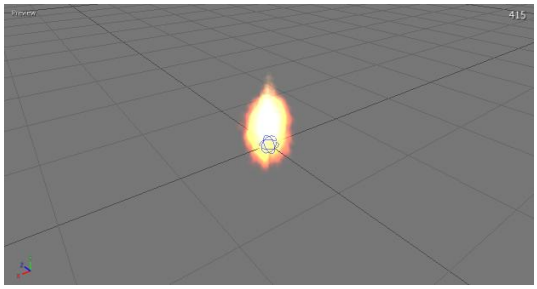
So, open up the Particle Editor, and we'll get started.

If you click on the "Particle System" menu, and then "Open", you can select the "Fire" Particle System.

You will now see the following displayed in the editor:



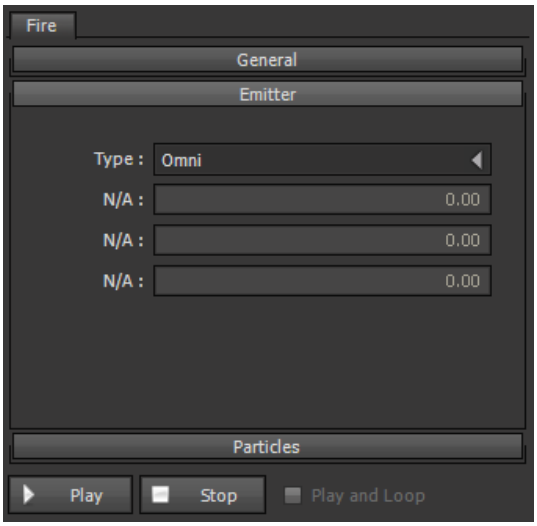
As with the PolyTrail, if you open up the Scene Viewer module, you can click on the "Play" button, and you should see something similar to the following:



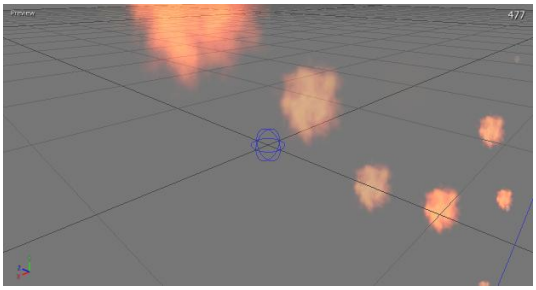
OK, so how was that created?

Firstly note, in the "General" roll-up, that "Use Auto Start" is checked. This will ensure that the PolyTrail will start as soon as it is instantiated.

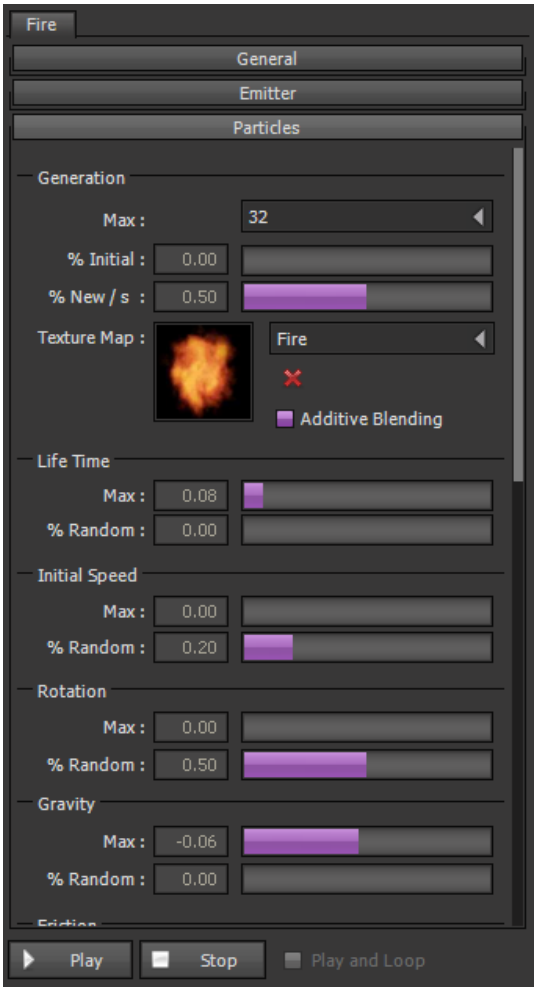
Now, open up the "Emitter" roll-up, and you will see the following:



In this roll-up you will notice that we are using the "Omni" type. This means that the Particles will be emitted in all directions. To see the difference, we will change the type to "Disk", with Parameters of "Angle" – 45 and "Radius" – 25, and we will see something more like this:



Next, open the "Particles" roll-up, and you will see the following:



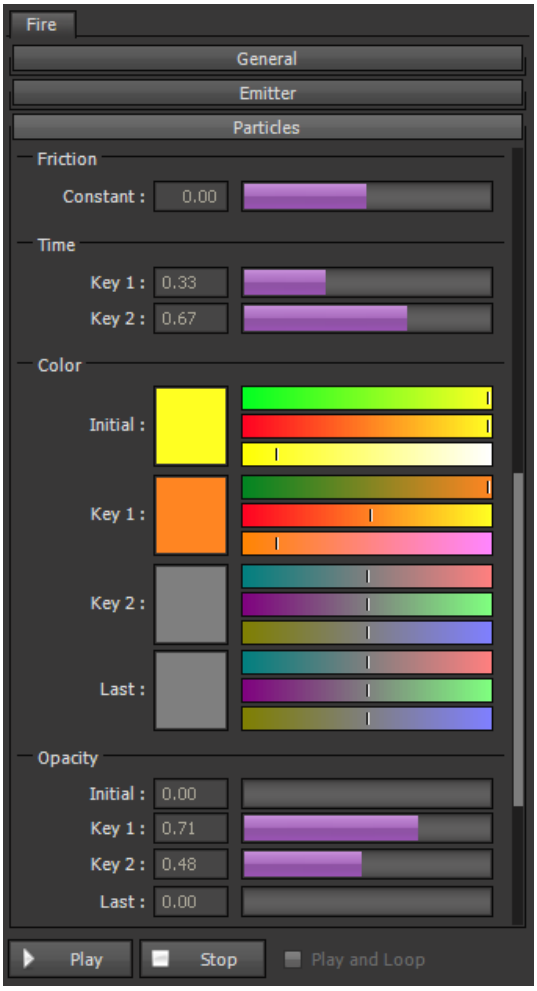
Unfortunately, on my laptop, I can't display all of the options, but I will walk you through them:

Firstly, the "Generation" option is set to "32". So we will have a maximum of 32 "live" Particles at any one time. With "% Initial" set to "0", and "% New / s" set to "0.5", then we will start with no Particles at the start, and 50% of the maximum (ie: 16) will be generated each second.

The "Texture Map" is a nice looking "fire" Texture, and "Additive Blending" is checked, all up producing a very pleasant fire effect.

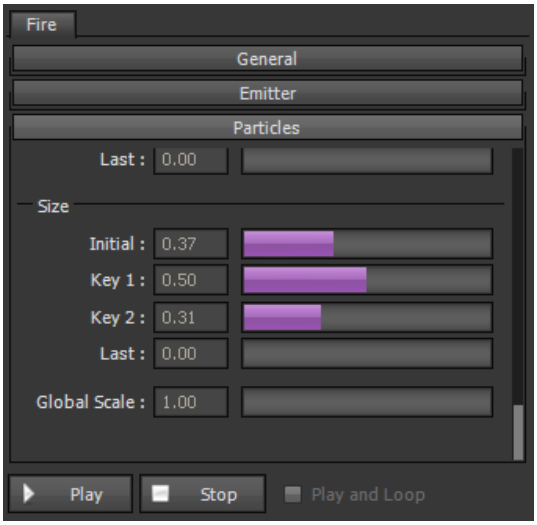
Since most fires have short-lived "particles", the options in "Life Time" are set to "Max" of "0.8" (i.e. short-lived Particles), and "% Random" of "0.0" (to ensure that they all have the same life-span, seeing as it's so short!). You will also probably have noticed that "Initial Speed", and "Rotation" have both been set to "0.0". These have been set to produce quite a good looking fire. You will also have noticed that "Gravity" has been set to "-0.06". A negative gravity means that the Particles will NOT fall back to earth, i.e. they'll continue rising into the night sky (oops! There isn't actually any sky at the moment!).

OK, moving on to the next part of the roll-up:



In this part, "Friction" has been set to "0.0" since the Particles won't really be alive long enough for it to make a difference! You'll also see (in "Time") that "Key 1" is set to "0.33" (i.e. 33%) and "Key 2" to "0.67" (i.e. 67%). This has been done to produce a smooth change in colour at 1/3rd time, and 2/3rd time. Moving on to the "Color" section, you can see that the chosen colours are "yellows" and "oranges", which are typical fire colours. From "Key 2" onwards, the colour will fade to "grey" (that reminds me of a song!). This is to make them fade nicely back to nothing, since "grey" is a nice neutral colour. Finally, "Opacity" has been set to "Initial" – 0.0, "Key 1" – 0.71, "Key 2" – 0.48 and "Last" – 0.0. What this means is that the Particles will start invisible, gradually getting to 71% Opacity, before fading away to nothing again.

The last section of the roll-up sets the "Size" of the Particles:

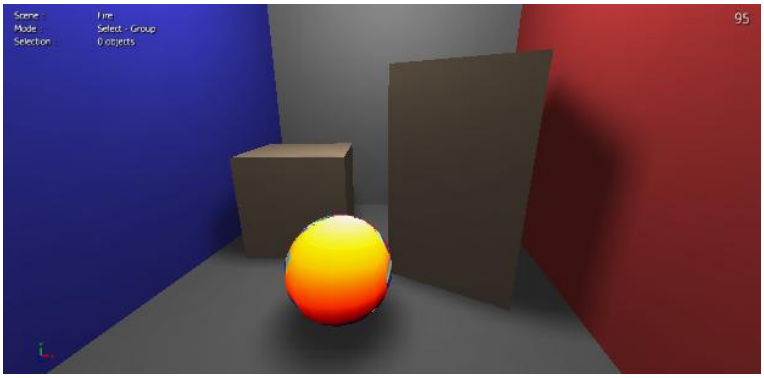


As you can see, the Particles grow, from about 1/3rd full size, to about 50% full size by "Key 1", and then shrink down to nothing by the end.

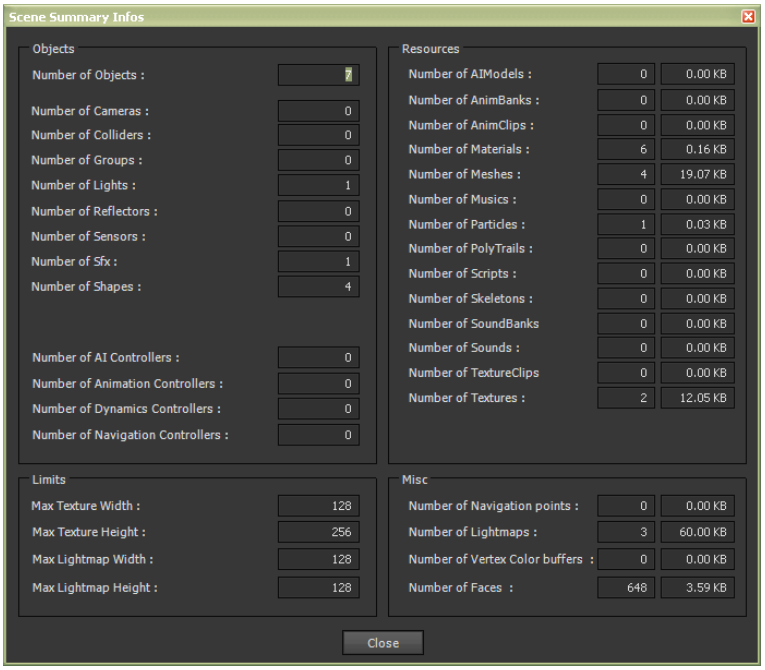
All in all, these settings give quite a pleasant fire effect as we saw originally.

Now, all I've got to do is run through the Scene, and the Scripts, and then we'll have finished this Chapter.

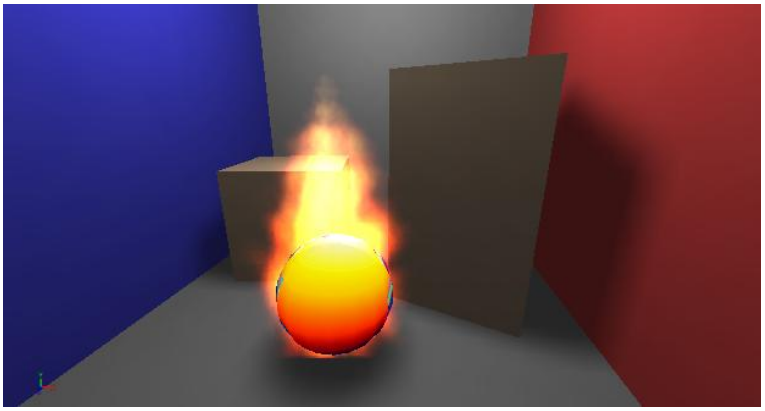
If, you open up the Scene for the demo in the Scene Viewer, you will be confronted with the following:



If you now click on the "Scene" menu, and select the "Infos" option, ShiVa will display the following dialog:

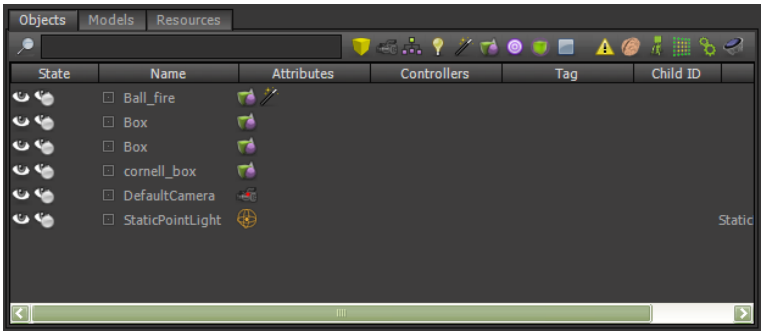


As you can see, there is not quite as much to this Scene, with only 7 Objects in total! The one that I'll concentrate on again is the "Number of Sfx:" line, which shows that there is only 1 effect attached to Objects. So, if you click on the () button, you will see the orange ball suddenly start to produce a nice fire effect:



Pretty cool fire, isn't it?

So, how do we create this type of Scene? Again, that's quite easy (I won't be going into creating the whole Scene, just how to attach the Particle System). If you open up the Scene in the Scene Viewer (after having stopped the Scene running of course!), and then open up the Scene Explorer, you will see the following:



If you look at the Object "Ball_fire", you will see that it has the little wand in the Attributes column, that shows that it has an Sfx attached.

Again, by right-clicking on the Object in the Scene Explorer, you will be presented with a menu of all the Attributes etc. that you can attach to it. Since this is identical to PolyTrails, I won't go through it all again. The only difference, of course, is that instead of selecting "Add a polygon trail", you would select "Add a particle emitter".

OK, so you now know how to add Particle Systems to Objects, what's next?

The next bit is to script how the Scene will work.

The Scripts for this demo are extremely simple, and consist of one Function and one Handler in one AIModel:

Fire_Main

This AIModel consists of the following:

Functions

loadScene ( )

Handlers

onInit ( )

loadScene ( )

This Script consists of just one line:

application.setCurrentUserScene ("Fire")

All this line does is set the current Scene for the User to the "Fire" Scene.

onInit ( )

this.loadScene ( )

All this does is call the "loadScene" Function above.

That's it for this demo! In the next Chapter I'll be introducing Sounds in ShiVa