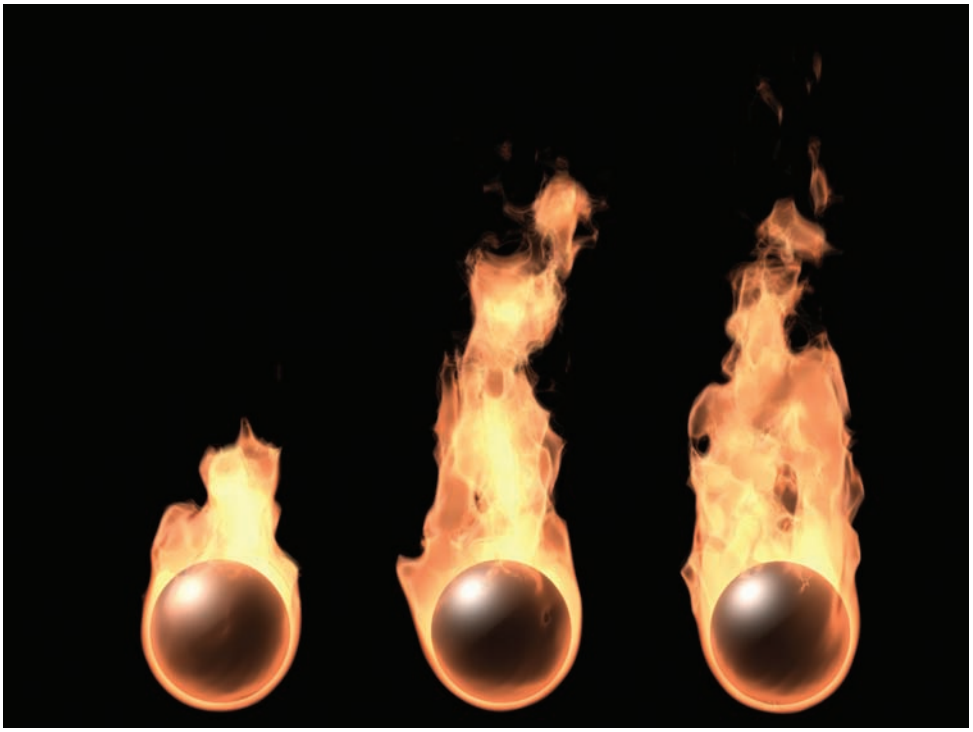
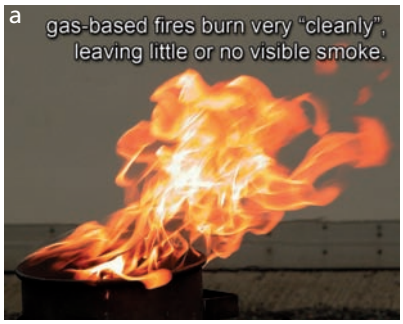
1 Ball of fire

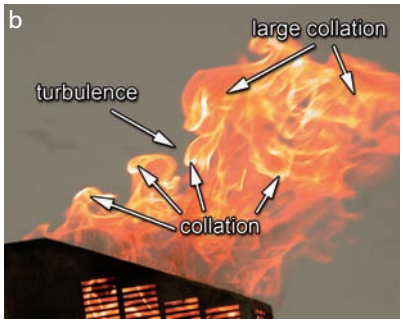


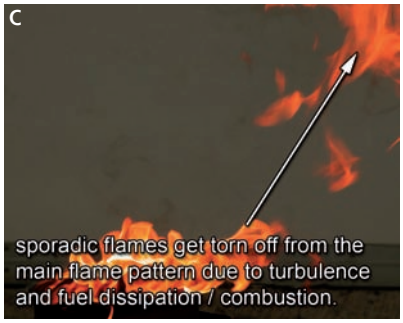
Introduction

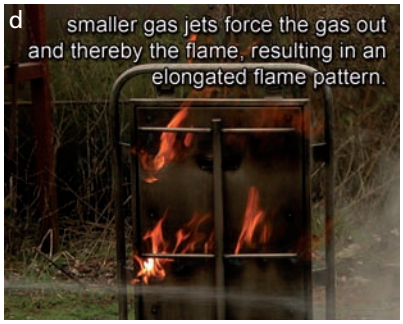
In this tutorial we're going to tackle one of the most common simulations performed with fluid dynamics, that of a metal ball emitting fire. There are several reasons behind this – from the amount of "fuel" the simulation has, its explosive properties, color, and whether or not the system is set to generate smoke. In our system we're going to simulate this type of effect using 3ds Max's native kit, namely Particle Flow, with multiple systems to affect each aspect of the flame shape. The resulting particles will then be surfaced with a Blobmesh Compound Object with a material assigned before rendering.

Analysis of effect









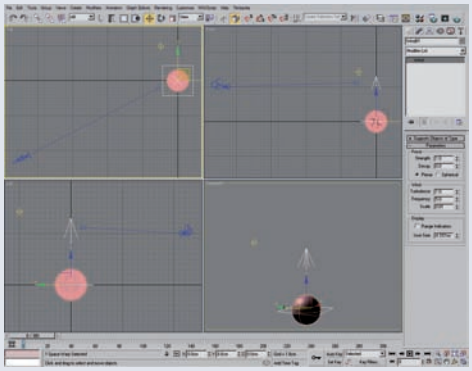
(a) For this particular type of fire effect, we'll base our simulation on a gas-fuelled fire, specifically a non-oxygenated fire and one with a large emission point, which reduces the force and ferocity of the flame. As this flame is gas-based, it leaves very little or no smoke, and as a result it's, aesthetically, very clean burning. The flame is a trademark yellow-orange color with a white core (depending on camera exposure) and a subtle orange perpendicular effect with a fluid-like waveform pattern. (b) This pattern has three main parts – the initial influence being the collation of fire plasma into small pockets that make up the distinctive internal detail of the flame; the second influence breaks up this uniformity and adds larger masses of collation which are then torn off by turbulence (the third influence). It's this turbulent motion that gives this large body of flame its distinctive shape, folding over itself and interacting gracefully with the external environment. (c) Additionally, as the gas burns, we see a occasional lick of flame at the top, with some parts being "torn off" or detached from the main body of the flame. (d) The resulting flame design appears much smoother (aesthetically) than in any other fuel source (wood, oil, etc.).

Okay, first things first. Tutorial number one and we're encountering our first major challenge: the effect we're trying to simulate is totally based on fluid dynamics, that of the motion of flame in relation to its surrounding medium: air. Unfortunately, at this current time of writing, we don't have a fluid system native to the software (20 quid said that by Christmas we've got one, just to put egg on my face!), though there obviously are some excellent plugin solutions out there (see the Appendix for

more information on this). However, we can create a convincing result using the base software. Due to its motion effect, we're obviously going to use a particle system to generate the dynamics of the flame system; however, simply emitting the particles from a single source and exerting a Wind Space Warp on them isn't going to be enough, purely because the Space Warp in question doesn't simulate the desired turbulent motion. It's, however, good enough in this instance to affect the entire simulation to suggest a subtle external force to break up uniformity, creating a light breeze on the flame causing it to distort somewhat and also to be used to drag the particles vertically. The main body of simulation will lie in multiple particle systems. The initial Flame system will be used to position and scatter particles around the Geosphere primitive in the scene, with an SDeflector preventing these particles from intersecting the geometry as they pass around the surface. These particles will be attracted to one another within a small threshold radius, producing collations of particles. These particles will then be affected by a reduced number of particles born from the same location(s), which will produce larger collations and also interact with the main body of the system. Finally, to break up the effect and to design the turbulent refined patterns as seen in the reference material, a time-offset instance of the large influence particles will chase the flame particles, causing them to displace, simulating air rushing in, and producing loops and arcs akin to the reference. The simulation aside, the main crux of the design is shading the particles correctly. Actually we aren't going to render the particles but surface them using a Blobmesh Compound object, which will have a material assigned that uses fog density based on object thickness to drive the brightness of the flame as it progresses through the animation. In the "Taking it further" section, I've adapted this material even further, creating the falloff effect around the edges of the flames. Once you've finished this tutorial, feel free to have a look at this section and the accompanying Taken Further 3ds Max scene file.

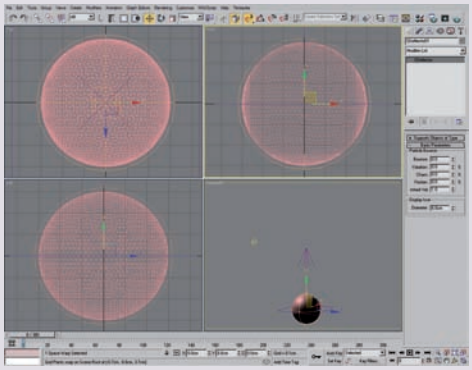
Walkthrough

PART ONE: First we'll load the start scene and add a basic Space Warp before designing our initial particle system.



1 Open the *01_Ball_O_Fire.max* file included with this tutorial and accept any file unit change if prompted. In the Top Viewport, create a Wind Space Warp and relocate it to the origin – XYZ coordinates (0 cm, 0 cm, 0 cm). Navigate to the Modifier tab in the Command bar and set Turbulence to 1. Set its Frequency to 5 and Scale to 0.01

Information: Don't forget to accept any file unit change so that any inputs we give can return uniform outcomes. Otherwise you may experience different results from that in this tutorial. We've added some Turbulence to the Wind Space Warp to simulate external forces not generated originally from the fire itself. The Scale value has been set low so that the resulting waveform is quite large (yes, you read that right …!). I've mentioned about relocating the Wind Space Warp to the origin as we're using the Turbulence setting; the resulting pattern is based on its position in the scene, so by ensuring that it's at the origin we'd get virtually the same resulting effect.



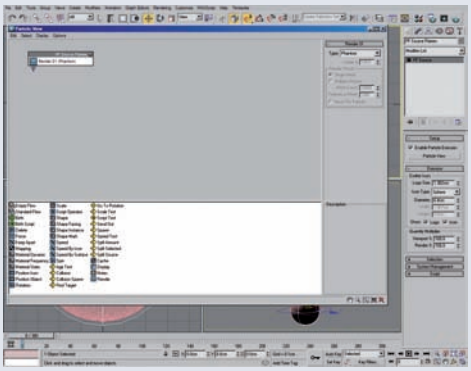
2 Still in the Top Viewport, create an SDeflector Space Warp and relocate it to the origin as before. Set its Diameter to 6 cm so that it's exactly aligned with the Geosphere primative in the scene. Set the SDeflector's Bounce value to 0.

Information: This SDeflector will be used to sculpt the particles around the Geosphere primitive in the scene so that

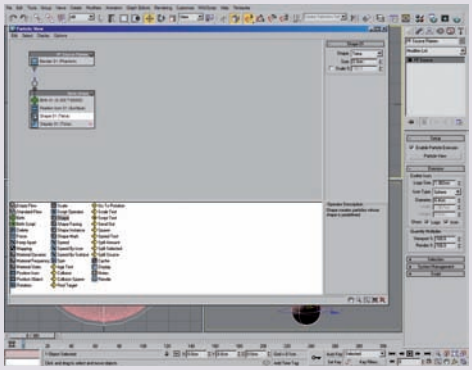
the particles that are born from the PF Source 01 icon in the scene (which is set slightly larger than the SDeflector to give the particles a chance to interact with the surface, else it's possible that some or all of them pass right through it) flow around its surface. The Bounce value is set to 0 so that the particles simply travel over its surface, not bounce off it.

PART TWO: With the Space Warps set up, we'll use the existing Particle Flow icon to build our flame system.

3 Select the PF Source 01 icon in the scene and click on its Particle View button in the Command bar. In Particle View, rename the PF Source 01 icon to PF Source Flames. Click on the Render operator in this event and set its Type to Phantom.

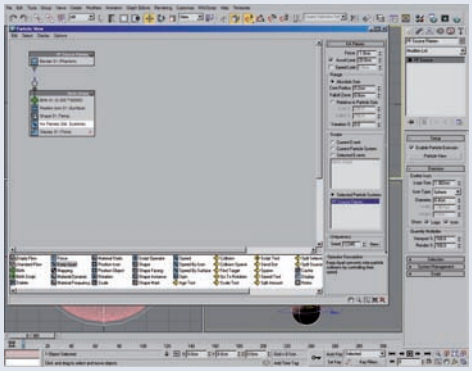


Information: We're using Phantom as the particle position and shape size need to be visible to the renderer so that the Blobmesh object we'll create later on can derive its surface; however, we don't want the particles to be visible in the render.



4 Drag out a Birth operator to the Particle View event display, wire its input to the output of the PF Source Flames event, label the event Flame Shape, and set its Emit Stop value to 300 (the length of the sequence). Set its Amount value to 60 000. Add a Position Icon operator to the event, and in the Location group, choose Surface. Add a Shape operator to the event and set its Size value to 0.3 cm.

Information: We need a fair amount of particles to create a nice effect, hence cranking up the value. We've changed the Location group to Surface so that the particles are simply born over the surface of the Particle Flow icon, not within it, else they'd be born inside the Geosphere! The Shape operator drives the size of the particles referenced by the Blobmesh object, though they won't be rendered.

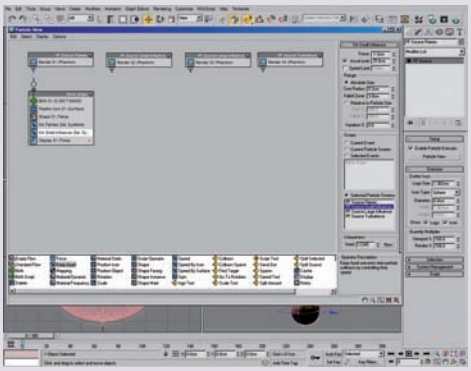


5 Add a Keep Apart operator to the event and label it KA Flames. Set its Force value to −1 cm and set Accel Limit to 20 cm. In the Range group, set Core Radius to 0.2 cm and Falloff Zone to 0.8 cm. In the Scope group, choose Selected Particle Systems and choose PF Source Flames in the list (the only one currently available).

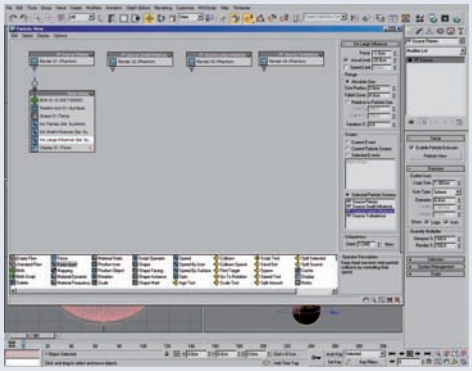
Information: This initial Keep Apart operator has a negative Force value, thus attracting the particles instead of repelling them. This particular operator

makes the flame particles simply attract each other, creating small collations (derived from the Core Radius and Falloff Zone values) and folds in the geometry when surfaced, thus generating small detail in the resulting flame; when the particles collate together, the resulting surfaced geometry will be wider, yielding a brighter color as derived from a material we'll design later on.

6 Copy the PF Source Flames root event three times. Label the first copy PF Source Small Influence, the second copy PF Source Large Influence, and the third copy PF Source Turbulence. Back in the Flame Shape event, add another Keep Apart operator and label it KA Small Influence. Set its Force to −1 cm and Accel Limit to 25 cm. In its Range group, set Core Radius to 0.2 cm and its Falloff Zone value to 3 cm. In the Scope group, choose Selected Particle Systems and choose the PF Source Small Influence in the list.

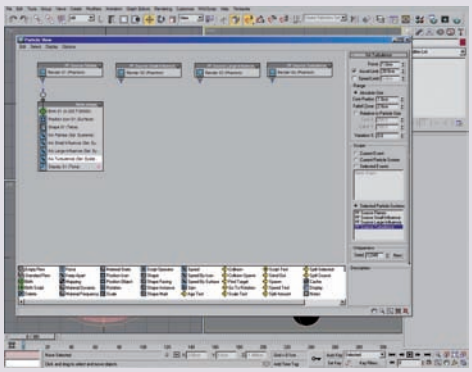


Information: We've set up root events at this stage so that we can get the Keep Apart operators built up in the original system, referencing these new root events (systems). The second Keep Apart operator to be added drags the original Flame particles upwards, creating additional folds in the geometry. The particle system that drives this will have a reduced number of particles so that the attractions are defined as in the reference material.



7 Add another Keep Apart operator and label it KA Large Influence. Set its Force to −1 cm and Accel Limit to 20 cm. In its Range group, set Core Radius to 0.4 cm and its Falloff Zone value to 4 cm. In the Scope group, choose Selected Particle Systems and choose the PF Source Large Influence in the list.

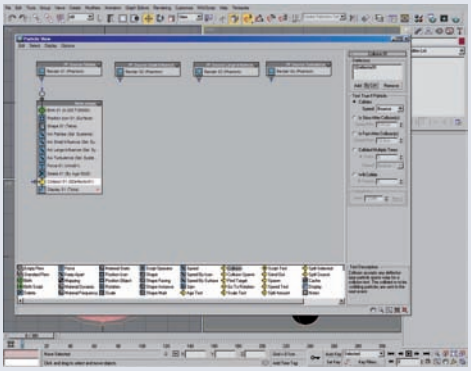
Information: As said before, this new Keep Apart operator affects the motion of the Flame particles further by attracting them towards the particles in the named system, thus resulting in the flame to appear to fold and spiral as the multiple Keep Apart operators fight for influence.



8 Add another Keep Apart operator and label it KA Turbulence. Leave its Force at 1 cm (note the positive value 1 cm this time as opposed to −1 cm previously) and Accel Limit to 20 cm. In its Range group, set Core Radius to 1 cm and its Falloff Zone value to 2 cm. In the Scope group, choose Selected Particle Systems and choose the PF Source Turbulence in the list.

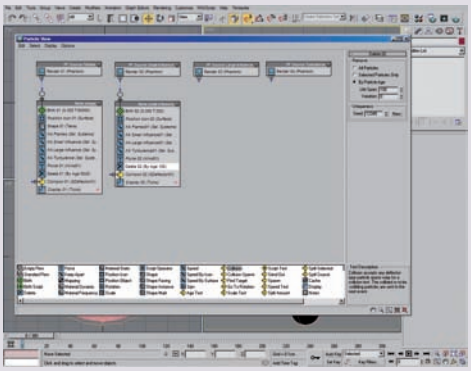
Information: This final Keep Apart operator is unlike others in that it has repulsion instead of attraction. The system it references will have a time offset, a clone of the Large Influence system but with a 5-frame offset and a negative influence to simulate air rushing in behind the flame, partially displacing it.

9 Add a Force operator to the event and add the Wind01 Space Warp to its Force Space Warps list. Set the Influence value to 610. Add a Delete operator to the event and set it By Particle Age. Set the Life Span value to 50 and Variation to 5. Add a Collision test to the event and add the SDeflector to its Deflectors list.



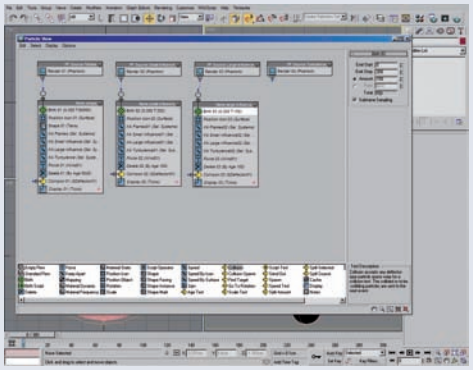
Information: The Wind, as mentioned earlier in this tutorial, is added simply to introduce an illusion of external air turbulence from wind, someone breathing, closing a door… take your pick…! The Collision test loads in the SDeflector so that the particles flow over the surface instead of passing through the Geosphere geometry.

10 Ensure you're at frame 0 and instance the Flame Shape event. Label this new event Flame Small Influence. Make the Birth operator unique and set its Amount value to 350. Remove the Shape operator and make the Delete operator unique. Set its Life Span value to 100 and Variation to 0. Wire the input of this new event to the output of the PF Source Small Influence event.



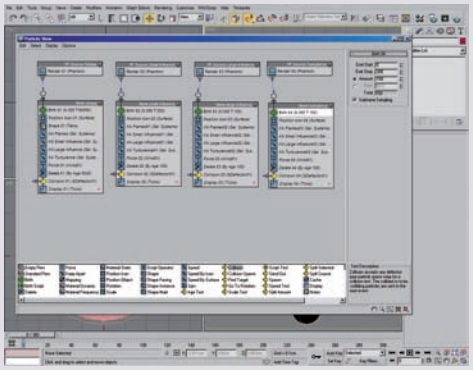
Information: You must ensure you're at frame 0 as you may have particle updates while duplicating a lot of particles affected by multiple Keep Apart operators. Also ensure you amend the settings before wiring to prevent any system lag if you aren't at frame 0. This new event has the same particle emission points, yet a reduced particle count as mentioned before, for

the Flame particles to be attracted to, creating small collations of particles/fire. The Shape operator has been removed as we don't need it in this system, but only in the original one. We've made the Delete operator unique so that the trailing Flame particles are continuously dragged vertically at the top of the flame, creating licks of flame and also ensuring that the flame doesn't try to slow down and attract particles beneath it.



11 Instance the Flame Small Influence event to create a new event and wire the input of the new event to the output of the PF Source Large Influence event. Label this new event Flame Large Influence. Make the Birth operator unique and set its Amount value to 150.

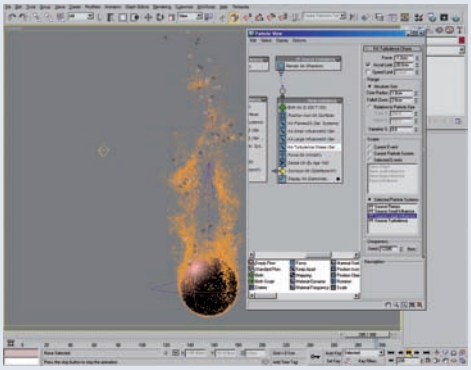
Information: Same deal as before, but with an even lesser amount of particles as this system is going to deal with a larger influence on the Flame particles.



12 Instance the Flame Large Influence event to create a new event and wire the input of the new event to the output of the PF Source Turbulence event. Label this new event Flame Turbulence. Make the Birth operator unique, and set its Emit Start to 5 and Emit Stop to 305.

Information: This time, even though the majority of the system is identical to the previous one, we've set a time offset so that this new system is "chasing"

the previous one. When used within a KA Turbulence operator in the Flames system, it displaces out the particles behind the Large Influence particles, creating the wind to rush in and displace the flame.

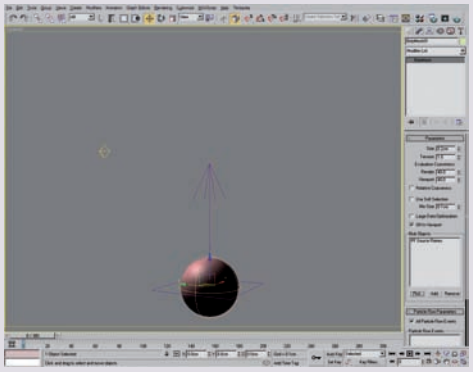


13 Make the KA Turbulence operator in this event unique and label it KA Turbulence Chase. Set its Force value to −1 cm (changed from 1 cm) and choose the PF Source Large Influence system in its Scope group's Selected Particle Systems list. Save the scene, turn off the Autobak feature in your preferences, and click on the Play Animation button to run the simulation. If you face problems like particles suddenly disappearing, see the "Information" section below. In the screenshot, I've also made each Display operator unique and assigned it a color and shape to distinguish the systems.

Information: This Keep Apart operator has had its Force value changed back to a positive value, resulting in an attraction to the Large Influence system. This will, in turn, displace the Large Influence particles along with other particles in the other systems, creating a turbulent effect. We're playing through the sequence to view the simulation to ensure no particles suddenly disappear. Usually this problem occurs with multiple Keep Apart operators dealing with a lot of particles due to the Keep Apart operator's brute force method. The sequence is already set to play every frame consecutively, and if any problem arises (i.e., the system hangs), click back immediately at frame 0 on the Timebar (eventually) so that the system updates and returns to frame 0. We've turned off the Autobak feature so that 3ds Max doesn't attempt to Autosave part way through the playback

sequence or once the sequence has been completed, else it will try to re-run the simulation and attempt another Autobak save at the end and so on and so on... (depending on the time interval between saves you've set in your preferences). To get around the issue of disappearing particles (should you experience it), set a new Seed value in the Position Object operator and re-run the simulation. Worst case scenario, drop the amount of particles in the Flame system, even by just 1000, and re-run the simulation. Actually, to tell you the fact, the Force operator's Influence value was originally set at 600, but I found the Keep Apart operators were failing, so I simply amended the value to 610, which worked. If this value doesn't work for you, just drop it down between 590 and 610; even very small changes can result in the particle positions being recalculated. NOTE: We're running the simulation to ensure particles don't disappear at render time. You can check this in the Viewport as the percentage of particles visible in the Viewport is set the same value as that in the render, plus the Integration Step settings are also the same.

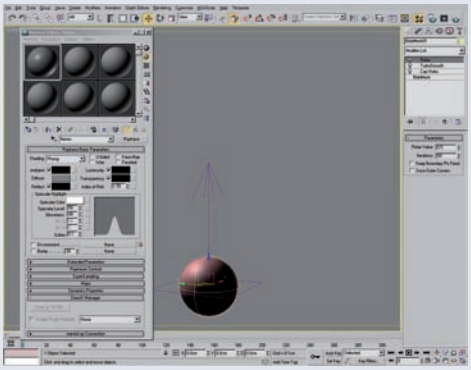
PART THREE: With the particle system now designed, we'll surface the particles before designing a Flame material and assigning it to the geometry.



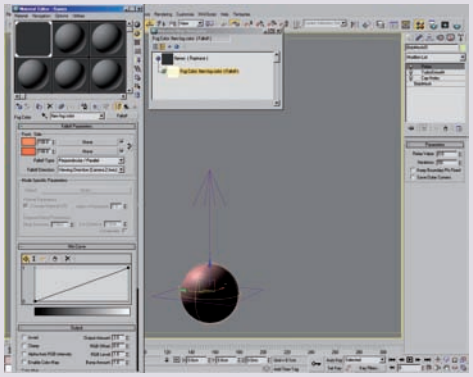
14 Return to frame 0 and enable the Autobak feature. Save the file for safety. Add a Blobmesh operator to the scene. Set its Render and Viewport Evaluation Coarseness values to 40 and enable Off In Viewport. Click on the Pick button and choose the PF Source Flames icon in the scene (with the button depressed, click on the Select By Name icon in the scene if you've trouble clicking on the icon in the Viewport). Click the Pick button again to turn it off.

Information: Make sure you return to frame 0, else you'll surface the resulting Blobmesh at the full extent of the amount of particles in the scene which may slow things down a touch. We've turned off Show In Viewport to speed up Viewport performance. The Evaluation Coarseness value has been increased to bring down the amount of polygons that the Blobmesh will generate, thereby speeding up the surfacing operation; we've set both Render and Viewport to the same value so that if you may decide to view the geometry in the Viewport (by re-enabling Show In Viewport) you can see exactly what you'll get at render time. The reduced mesh density will obviously result in surfacing inaccuracies, but we'll refine the mesh next. As there are multiple PF Source icons in the scene, it's much easier to use the Select By Name feature to assign the relevant object to the Blobmesh. Ensure you turn the Pick button off afterwards, as mentioned above, else you'd accidentally select another object in the scene and add it to the Blobmesh as well!

15 Add a Cap Holes modifier to the Blobmesh01 object. Add a Turbosmooth modifier. Finally add a Relax modifier and set its Iterations value to 50. Turn off Keep Boundary Pts Fixed. Open the Material Editor and add a Raytrace material to a free sample slot. Label the material Flames.



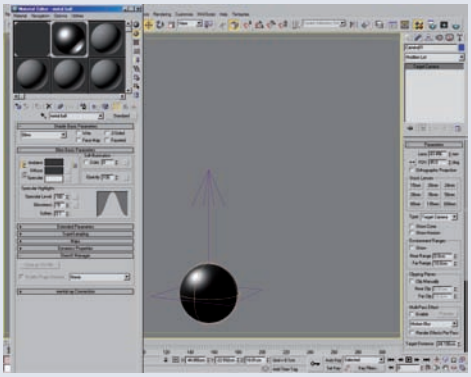
Information: Due to the low mesh density of the Blobmesh, set in Evaluation Coarseness, an occasional hole in the geometry may pop up, which will result in a black mark in the resulting render. Adding a Cap Holes modifier fixes this problem. The Turbosmooth modifier refines the mesh ready for the Relax modifier; the Turbosmooth adds more polygons for the Relax modifier to work on, resulting in a smooth result, which takes only a fraction of the time it'd take to create the same amount of polygons with the Blobmesh object alone! We need these extra polygons to effectively relax and smooth out the main mass of the geometry without losing finer detail of single particles.



16 Set its Transparency color to white and set the Index of Refr value to 1. Set the Specular Level and Glossiness values to 0. Expand the Extended Parameters rollout, and in the Advanced Transparency group, enable Fog. Set its Start value to 1 and End value to 600. In the Fog's Map slot, add a Falloff map and label it Flame Fog Color. Set the Front color swatch to RGB 255,148,93 and the Side color to RGB 255,115,65. Expand the Output rollout and set its Output Amount to 3. Assign this material to the Blobmesh01 object in the scene.

Information: The flame material simply uses the Raytrace material's Fog element to create a smooth texture, with its intensity derived from mesh thickness. This is ideal for our flame effect as the thicker the flame, the brighter it gets, and the thinner the geometry, the lower the intensity. However, we don't want the material to reflect its environment; therefore, we've set the Index of Refraction of the object to 1. We've cranked up the Output value of the Falloff map, which sets two colors – one for the facing geometry and the other for the perpendicular – the sides of the flame, to brighten up the end result. The Fog parameters have been set as such so that the outer edge of the geometry isn't fogged (inset by the Start value of 1 and the inner (End) value set to 600) – a value much higher than the thickness of our geometry as it currently stands, but useful to have in case you decide to amend the scene later on. Should you find the flame too bright for your needs, try increasing this End value to drop down its attenuation.

17 Label a Standard material in a new sample slot Metal Ball. Set its Diffuse color to RGB 55,55,55 and set Specular Level to 100 and Glossiness to 15. Assign this material to the Geosphere01 object in the scene. Render off the scene.



Information: The material assigned to the Geosphere in the scene is just a basic material to simulate a dark metal sphere (hence a large highlight). If you need a more accurate representation, try adding a Raytrace map to its Reflection map slot; the beauty of flame geometry is that it'll be reflected within the Raytrace map!

Taking it further

The end result of this tutorial does work quite well for emitters of this size; however, for larger surfaces and bigger fires a lot of additional work is required. This is due to the amount of heat the "fire" emits, thus affecting turbulence in the air. Additionally, changing the size of the fire requires a change to the amount of fuel, which obviously changes the resulting shape of the fire (see the reference material for examples). For this you'll need to amend the influence systems, plus potentially add additional systems to get the end result look right.

One way to get the end result look a bit more effective would be to speed up its motion; obviously it's currently way too slow and needs speeding up considerably. You could do this in the comp simply by ramping up the playback speed of the footage. If you want this effect in 3ds Max, you'll need to amend the Keep Apart operators so each has more strength over the adjacent particles, and also turn up the Force operator's Influence value, thus dragging the particles upwards faster. We'd have to increase the Keep Apart operators' influence, else the wind may simply drag the influential particles up, not giving the flame particles any chance to catch up!

An additional way is to tweak the assigned material slightly, which I've added as the Taken Further scene included with this tutorial. In this example, using

exactly the same particle and geometry setup, I've simply added a prominent falloff effect to the inner edge of the material. This is achieved by adding a Falloff map to its Transparency map slot and designing the Mix curve to inset the falloff edge a little so that it coincides with the reference material. The brightness of this edge is derived from an instanced version of the map used in the Fog slot in the Luminosity slot. The Output value of this map was dropped down a touch due to the resulting intensity of a (now) partially opaque material being applied onto the fog, which resulted in an overly bright flame. This falloff effect is pretty subtle but does make a lot of difference to the end result, giving the shape of the flame a better definition. However, it isn't without problems. Because of the flame's edges now being prominent, irregularities in the adaptive geometry of the Blobmesh due to the high Adaptive Coarseness values are now visible, even with the Turbosmooth modifier applied. Thankfully, adding some motion blur tends to cure this problem as the particles at the peak of the flame are moving so fast that the falloff effect blends nicely. Ideally the falloff effect should also be linked to the thickness of the geometry, stronger in intensity where the geometry is thick and very transparent where the geometry is thin.

Additionally, even if you decide to keep the material design and speed of the flame as it's, adding a touch of motion blur will make a lot of difference. There are three main ways to achieve a good result – one by using Object Motion Blur (which may take a while and result in artefacts), secondly by using multiple-pass motion blur (set in the Camera), and thirdly by using the Scene Motion Blur feature in the Video Post. One type of motion blur is missing in this list – Image Motion Blur. Obviously this type can't be used as the polygon count is dynamic from one frame to the next, which would result in a blurry mess!

Although this flame color and shape is loosely based on a gaseous fire, you may want to amend the system somewhat by adding some smoke into the fire as the flame licks extinguish. This is actually pretty easy; instead of killing the particles off with a Delete operator, you can send them to another event after the same amount of time they're made renderable (HINT: change the Render operator's Type to Geometry and make the initial event non-renderable in its Properties). The type of particle required for the smoke could be set as a scaling facing particle (Shape Facing operator) or a nice rotating Sphere (Shape operator/Shape Instance operator), and use billboard particles and/or lots of falloff to ensure the geometry edges aren't visible. As these additional particles will interact with the shaded Blobmesh (i.e., they overlap), you may find that the fogging effect applied to the Blobmesh flame geometry is brightened by a fair amount. To get around

this, I'd suggest compositing the smoke separately, else you'll find that your flame is too bright in places and/or suddenly gets brighter as the smoke particles kick in (even if the smoke particles are very dark!). You could try rendering both out in one pass, but you'll more than likely need to amend the Output value of the Falloff map used to control color distribution in the assigned material. Personally, I'd composite the effect as the result is less hassle, plus you can always amend smoke color and density later in the comp!

No matter how much we tweak and amend this system, it has to be said that even at this early stage in the book, (currently) there's no proper native solution to make this type of effect, that is to say there isn't a native fluid system for 3ds Max, so you'll have to resort to a plugin or a third-party solution. However, there's no reason why one couldn't script the motion effectively with a bit of time and effort. Fluid dynamics calculations aren't exactly easy, though if you know MAXScript you should be able to code a system that affects particle positions over time based on surrounding fluid/air viscosity. For more information on computational fluid dynamics and for some very good CG references and examples of dynamics, check out Ron Fedkiw's site at http://physbam.stanford.edu/~fedkiw/.