

Rain using Delta Particle Editor and 3DS Max

By

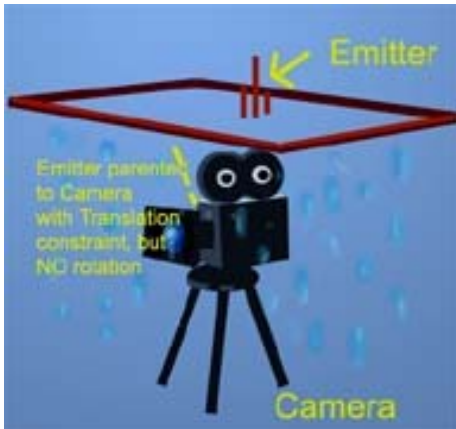
Mr. D at Delta 3D – 3/20/07

Point One: These ways will take a bit of coding work for attachment of *Particle Emitters* and *Mesh Planes* to the *Camera* in the *Scene Graph*. This paper does not cover that aspect, as I do art work and not coding. I have been advised by Team Programmers it should be familiar work for most people familiar with Delta and OSG, you *can not* accomplish this in *Stage* at this time.

Point Two: I am using Delta's Particle Editor, you can try using your own from perhaps your 3D program, and you'll need to figure out equivalent numbers and settings for your particle system. I do find that the 3D Particle Editor provided with Delta seems to actually export more options than what I get with the current *exporters* for the commercial 3D Programs and Blender.

FIRST WAY: For Rain and Snow one way to carry it out is by attaching a Particle Emitter at a fixed height above the camera. Set up to cover an area around the camera, this emitter has its *location/translation locked* to the camera, but *not* the camera's *rotation*. Use this if you have a wide open landscape the player is moving through.

If you've ever played a game where the rain or snow moves with you as you go forward, but the rain pattern changes as you turn. More than likely they are using this system.



Why this system? Mainly because you don't need particles falling all over a level, especially since most of them will not be seen. And in the Emitter over Camera trick you need only generate several hundred particles to achieve rain or snow. As opposed to thousands or even tens of thousands needed to cover a whole area.

First up Rain

We'll start by tackling rain, which means that first we'll do the rain falling, then I'll show one way to fake rain droplets hitting.

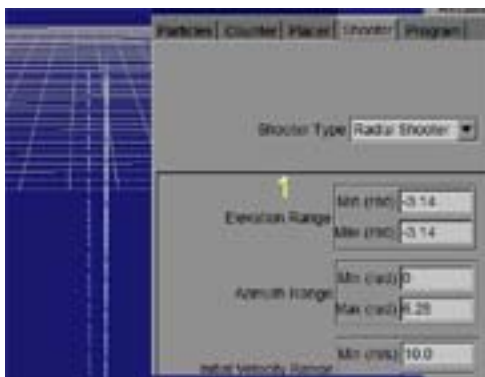


Here is my raindrop texture. I'll just be using a small 128x128 texture. It could be smaller, and I would most likely use just one droplet. But for you to see it a bit easier, plus to remind you that particles billboards are square; so if you made a single droplet at 32x128, it would be stretched out to a square shape distorting your bitmap. It does of course have an Alpha Channel so only the droplet shows in Engine.



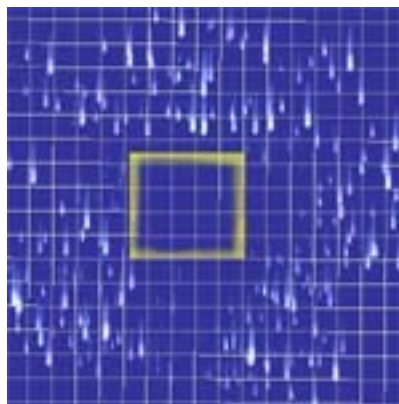
I've open the Delta Particle editor and am on the '*Particles*' **Tab** where the first things to set are

- 1) Alignment to – **Billboard**, so the particles will always face the camera.
- 2) Shape – Can be Quad or Quad Tri Strip.
- 3) Notice the two check boxes '**Emissive and Lighting**', Lighting uses the viewer's OSG light (also in-engine lights will affect) and Emissive makes it rather glow. With neither checked the bitmap simply is the texture's colors unaffected by lighting. For now leave Emissive checked to make seeing your texture easier.
- 4) Box for Browsing to find your texture.



Next we want the particles to be shooting downward, I found that this is easier to set up in the Particle Editor than in Stage.

So go to the *Shooter* **tab** and find Elevation Range. Next set the Min and Max values to -3.14, remember that the Editor is in Radians and not Degrees. This should cause your particles to be shooting straight down.



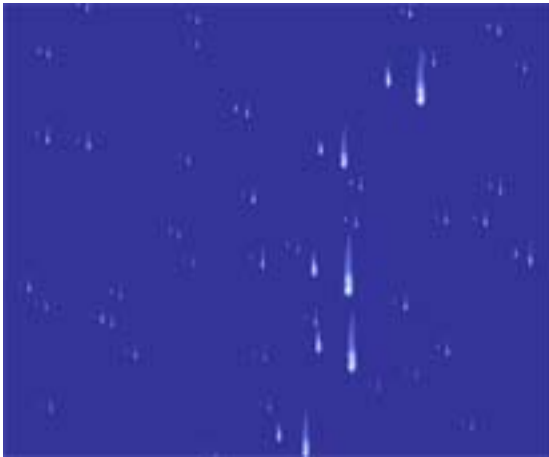
Now make a Rectangular Emitter by going to the *Placer* **Tab** and

- 1) set type to **Sector Placer**
- 2) Below find **Min** and **Max Radius**. To start make the **Max** the full 10, and for the **Min** 3. (You'll find the need to tweak just about all

values by viewing how your *Effect* looks inside your Game Engine.)

Ok now in the picture to the right side (with the really big rain drops) the yellow square shows an open space in the center. Why? This deals with your setting your

Min to 3 and not 0. Recall this is where your camera is sitting and you do not want your rain drops to be falling too close to your camera, because they would look huge due to closeness to the camera. After checking in-engine results you can always readjust if needed.

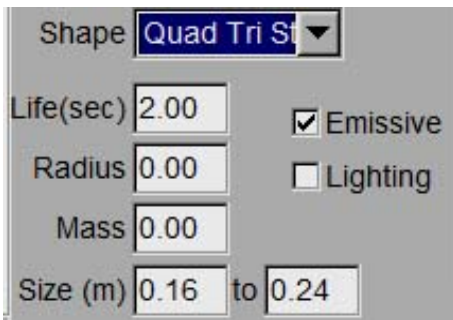


Here I lessoned my **Min** to help show the problem of drops too close to the camera.

Ok so basically you now have rain falling, and now you will start to tweak it to your liking.

So move around your viewer camera to place it towards the center of the emitter, and down a bit. You don't want the emitter sitting right on top the camera.

THINGS TO TWEAK



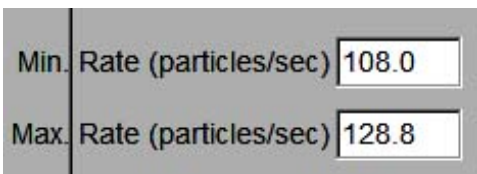
Under Particles Tab

First unless you want glowing rain uncheck **Emissive**. Check lighting if you want scene light to affect the droplets color.

'**Life(sec)**' with rain it's best to have the rain disappear before it hits the ground, we'll fake splashes later, building a system that can create splashes located exactly where a droplet hit can

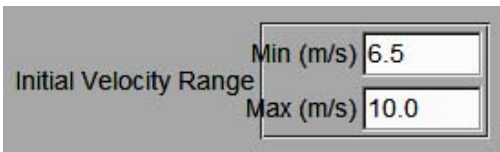
gets complicated.

Size(m) try to make enough of a range so your drops are not all the same size.



Under Counter Tab

Try various numbers in the **Min/Max rate** of particle creation.



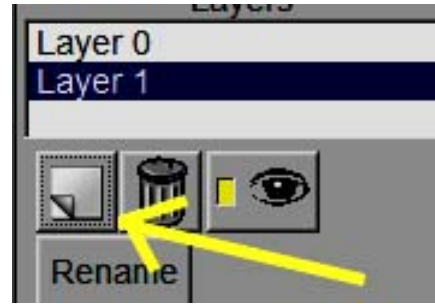
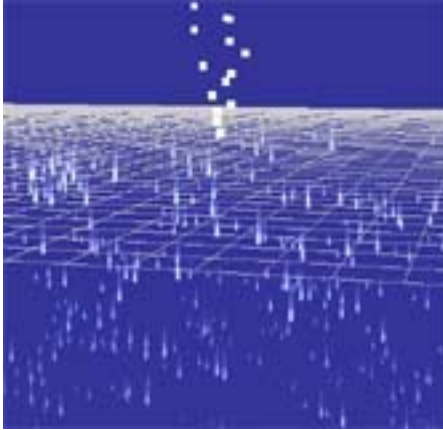
Under Shooter Tab

Play with the **Min/Max** of the **Initial Velocity Range** so drops don't always fall at the same speed.

Note: You're doing all this so your drops don't fall at a regular rates, numbers and patterns. Even though the particles are supposed to be random, most people will notice a pattern after a while.

So how to further stop this from happening?
Remember you can have more than one emitter.

Just hit the **'Add New Layer'** button and a new Emitter is created. Then just follow the same steps as above, but put different numbers in to this Emitter's variables. Even make a different texture if you want.



Here the Second emitter has been added to the rain effect, recall that the **'Eye' icon** hides or shows the layer chosen above.

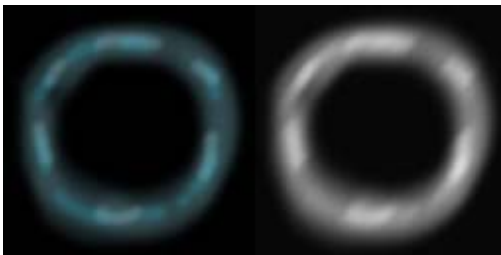
So you can hide the old layer if need be while your working on the new.

On output this is still considered only one OSG particle file even if you have five, six, or more emitters.

Water Striking the Ground

Most games do not show a 'plink' of water hitting the ground, but instead a circular ripple that results when a rain drop hits a wet surface.

So first off make yourself a water ring texture, once more with and Alpha Channel, take a few things in to account however.



Here's the one I made for this demo (128x128) along side its Alpha channel.

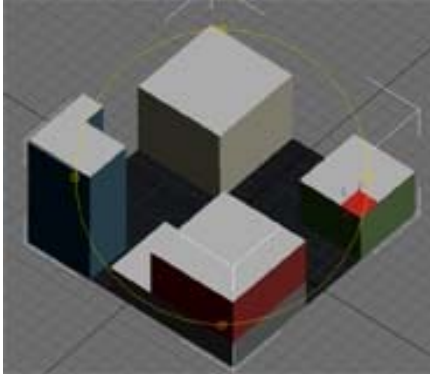
Now I want a ring you can see through, so no bright white Alpha ring, instead nicely grey so my water will be transparent.

The ring's color itself should tend to carry some of the color of your sky and perhaps some from the surrounding environment.

For water has no color, its color comes from the reflections it's casting, but with that said you still might at a touch of greenish-blue because that is the color people believe they should see in water.

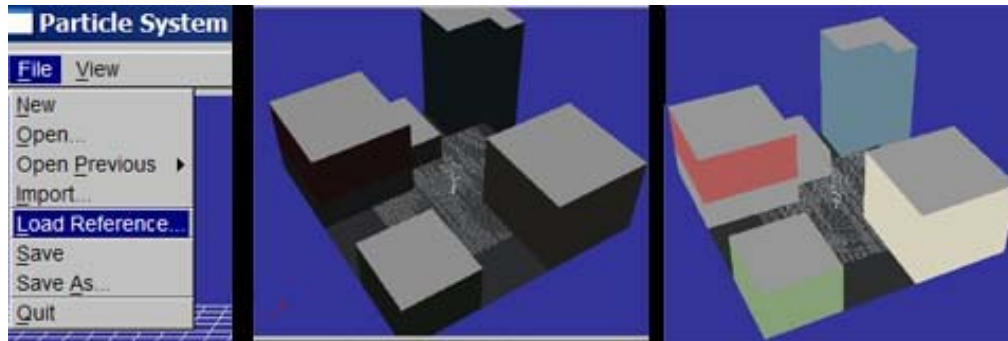
First part to understand about this effect is that it is carried out over a large, evenly angled surface. Usually it is a flat area, such as the surface of a body of water, or object, or a roadway. Remember water rings happen where there is enough standing water for rippling to occur, the ring is not formed by the water drop but by the displacement of the water already there by the droplet.

This is where you need to plan out ahead of time if you want to use this effect in an area, that it is figured into your level's design and the mesh is built to take this into account. Also it's good to have the geometry at least roughed out before hand so it can be brought into your 3d program if you're using that to create your particle systems; or to bring it in as an 'Reference' model in the Delta 3D particle editor.

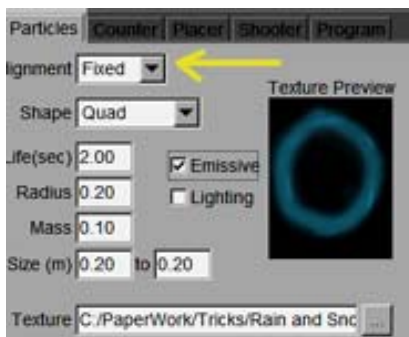


Ok here is a city block with very wet streets. I'll be bringing it into the Delta 3D particle editor for reference to do my rings. My concern is the streets and not the buildings of course.

To be able to bring it in to the Editor I'll first need to save it out in .Osg or .IVE format. Here I'll just use an .Osg file type.

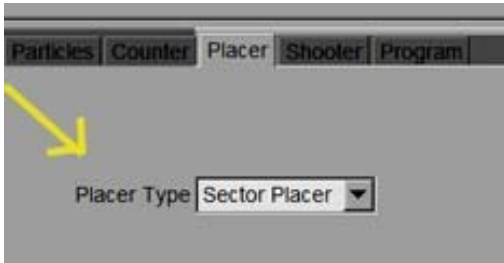


Now the model has been loaded into the *Particle Editor* in the above picture. I've have this showing two different exported version of the model, one with lighting on and the other with lighting off. The particle editor in its current form has a single light shining straight down from overhead, causing vertical faces to show unlit which can cause problems at times determining geometry. If you run in to this problem consider turning off your lighting on export.



Particle Tab:

Make your *Alignment Fixed* this will cause your particles to lay flat on the X/Y plane. Also load in your texture and set your lighting as you like.

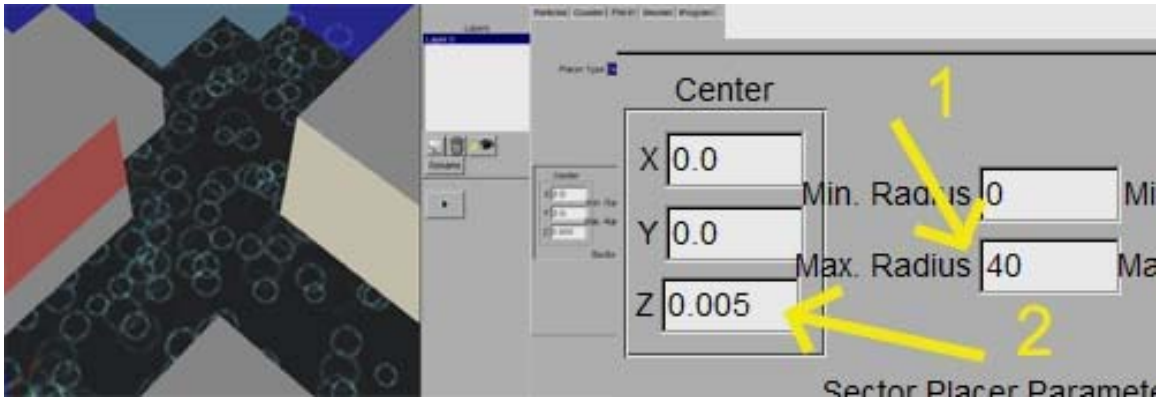
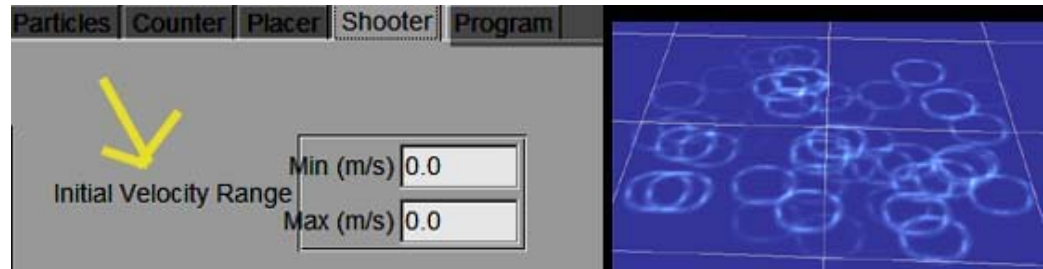


Placer Tab:

Set placer type again to *Sector Place* as you are going to want your particles to cover an area. Don't worry about the Min/Max radius for now.

Shooter Tab:

Lastly in your Shooter Tab locate '*Initial Velocity Range*' and set Min/Max to zero to keep your drop patterns from floating upwards.



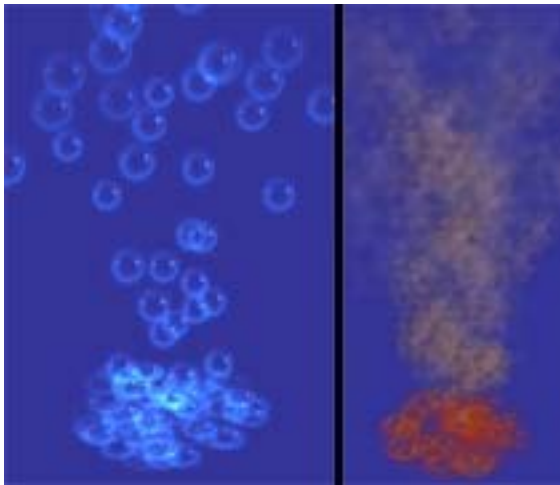
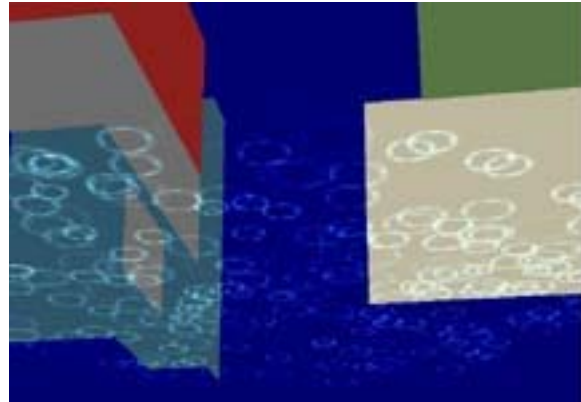
A last few things:

Ok so I made some really big rings again so you can see them in the picture, but as the emitter is lying flat on the XY same as the street. In the **Placer Tab** number **2** first, locate the '**Center**' Input Area and raise the emitter slightly up on the Z so it is above the street.

1 points to the **Max Radius**, this mesh was large but using the mouse to 'slide' the radius value it would only go to 10. It can go higher but must be typed in to do so.

Now in the next picture taken from an angle blow the XY plane you see the rings cover the whole plane, meaning they are inside the buildings.

When seen from above you would not notice this as building geometry hides this, but if in your project you wished to enter the buildings you have to solve for this problem. Such as the floor inside the building is built at a height above the particles. Or you would have to build a number of emitters to cover only road areas. You need to plan in advance so you can build the proper geometry, or lay out a working set of particle emitters. Basically you need to do some *Level Design* work before you model your scene.



Don't Limit Yourself

Don't just think of water rings here for the flat lying emitter. To the right are shown some other possible uses, the first using a bubble texture on the floor to indicate boiling water. Same texture used again to produce the effect of bubbles escaping with a second emitter.

Next I colored the bubble texture red as in perhaps bubbling lava, with this time a wispy texture above to indicate smoke or steam rising up.

A bit hard here to capture the look of each as the effect only looks proper when the particles are animating, but more important is the concept of combining emitters and applying various textures and settings to produce your desired effect.

A previous paper I wrote on using the Particle System can be found here on the Delta site if you need more information on what its buttons do.

<http://www.delta3d.org/article.php?story=20051207101455773&topic=docs>





A FEW LAST POINTS ON RAIN

I made this little scene in *Stage* to talk about a few last things to think about when making rain (if I could only have gotten some better shots, emitters never seem to be emitting when you do your grab) or any other Particle Effect.

First get use to having your particle program, Delta3D or other, open along with your engine or a viewer. You are going to be doing a lot of going back and forth as you tweak your effect to see just how it looks in the *Engine*.

Secondly remember to sell the whole package, meaning take into account besides just your particle effect but also your textures and rain sounds if you're using them. Rainfall amount should reflect the sound heard, if your sound loop is of a light shower and you make a torrential down pour it will look odd, and the opposite if you have a loop of a huge thunder storm going.

Add to this if there is the sound of gusty wind with your rain loop, you might consider angling the rain, or program it so you can manipulate the emitter to swing it about a bit when the wind sound plays.

Textures: Hey, how about a sky that looks grey and rainy, the effect of a cloudy sky means over all your textures are darker, but since no sun to cast well defined shadows, they become less distinct. Also rain washes away dust cleaning objects so they tend to have a bit more saturation of color, and the wetness of objects means they tend to appear more shiny and higher specular values.

Also many materials such as stonework or concrete that have had prolonged exposure to rain and standing water pools, tend to pick up water staining on there surfaces. Look around at buildings and see if you can spot such staining.

‘So I have an emitter attached above the camera raining down, but I want my character to run inside a building where it is not raining.’

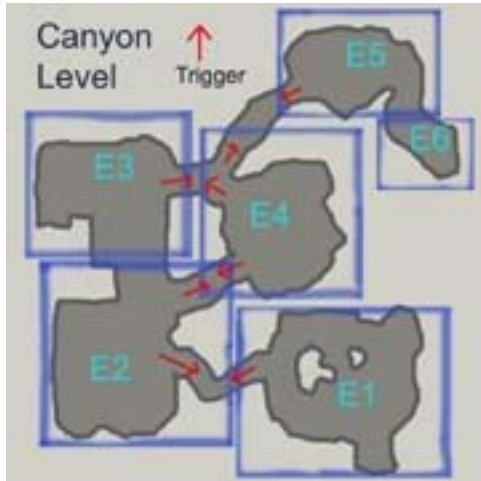
Good question, simple answer – turn the emitter off. This is handled usually by having a trigger setup in the level map that can turn the emitter off when a character enters a dry area. Or as more often happens you gradually lessen the amount of rain as they approach the dry area and it is off before they get there.

This again has to do with Level Design and not strictly game art. Many times Designers will use game play, such as your being attacked , to help distract your attention from noticing the change in weather.

THE RAINY AREA

So you don't want rain above your camera, you just want it raining in a certain areas of your map.

Rain effect is made the same way as above, but in your **‘Sector Placer’** you can set your **‘Min’ to Zero** so it covers the entire area (Hint you can still have a *Min* value if you made for example a building sitting inside the *Sector*).



Now as said before your *Particle Rain* in this type level layout, is usually split up into a number of emitters covering various areas, and to the left is an imaginary level for a first person shooter.

I cover the various areas of the map with a series of **6 Emitters** covering the various sections. Contained with in the map are *Triggers* represented by **Red Arrows** that would act as on off switches. On leaving area E1 the first *Trigger* in the canyon would turn on the particles in areas E2 and E3, since I should be able to see the whole area during play.

As I move down the canyon into area E2 I hit another *Trigger* this one turns off the rain in area E1 if it is on, or turns it on if it is off. If I'm not going to be in an area it is a waste of computing and graphic resources to let the rain effect just run. I might want those resources to provide big explosions during game play in area E2, or there might be water fountains running in E3 I'd rather be using the resources on.

To handle a lot of this it again comes to *Level Design*, whereby the passage through E1 to E2 might be a twisty cave. Why twisty? Because you don't want a player to be able to look back at area E1 and see the rain is not coming down any more. You design your level geometry to control the player's line of sight, and to funnel them as needed to make sure they activate such things as your *Triggers*.

Here the *Rain Effect* is simple, but you need to plan out your *Level Design*, and build *Level geometry*, or at least Test geometry to set up your rain emitters and decide where to place Trigger systems. But be prepared as often you may find you need to modify level geometry or your particle layout to get the best results (or say 'no one will notice, ship it anyway). This happens quite a bit in the game industry.