Autodesk* Maya* now has two dynamics systems: The oldest, known as Dynamics, uses the older solver (Dynamotive), and the newer, nDynamics, uses the Nucleus solver. Because it is multithreaded, Nucleus can better take advantage of multicore processors. The nDynamics system also has the advantage of being a unified solver. For example, if you set a force such as gravity or wind, that force is shared by all of the objects affected by nDynamics in the scene. nDynamics does not yet have a fluid module such as Dynamics has, though, so which one should you use?

Maya* has been set up so that you can use both in the same scene, and experience with both will help you decide which is better in a given situation. In the case of particles effects, which is what this article discusses, it's a matter of personal preference. Those of you who already understand dynamic particles will be relieved to know that nParticles have been created to operate in a similar way and can be controlled with the same expressions and forces—the big difference being that nParticles have some functions built in that you must apply to the older particle system, either in the attribute settings or as forces. In addition, Nucleus has some powerful constraint and collision capabilities. You can use what you learn and already know about dynamic particles to work with nParticles, as well. This article starts out using the Dynamics module to create rain, and then you create smoke as an introduction to using nDynamics.

Creating Rain

Let's create a bit of rain in Maya*. I assume that you have some basic knowledge of Maya*. If you're a beginner, be sure to read Part 1 of this series before going forward in this one.

Creating Particles

Working in the Dynamics module of Maya*, create a plane and use it as a surface emitter. Set the speed of the emitting particles to 0, as shown in Figure 1. Later, you will use gravity to control the particle speed.
Select the particles emitted by the emitter you just created, and open their attribute editor. Click the **ParticleShape** tab at the top of the attribute editor (see Figure 2).

![Image of attribute editor](image)

**Figure 2.** The **Render Attributes** can be found in the attribute editor for the particle, under the tab for the **particleShape** node.

In the attribute editor, beneath **Render Attributes**, choose **MultiStreak** as the Particle Render Type. Click the **Current Render** tab so that you can set the **Multi Count** to 3 and the **Multi Radius** to 0.300. Select the **Color Accum** check box.

![Image of attribute editor](image)
Creating a Particle Lifespan

Now, you need to make a decision as to how long you want each particle to live. Your decision may depend on several factors, one being the distance you need them to travel in your scene. You also might not want each particle to live the same amount of time, aiding in the naturalness of the look of your rain.

In Maya*, there are any number of ways to do everything. You can set the Lifespan Mode to Live forever, Constant, Random range, or lifespanPP. Using Random range, set the lifetime of the particles to 1.5, which equals 36 frames (at 24 frames per second), and introduce a bit of randomness to the lifetime by putting in a Lifespan Random value of 6 (see Figure 3).
Figure 3. Clicking the Current Render Type tab reveals an array of settings for MultiStreak particles, line width, multi count, multi radius, tail fade, and tail size. You will be creating values for the lifespan, the line width, the multi radius, and the tail size. Also note that if you select the **Use lighting** check box (under Tail Size), your particles will be lit by the lights in your scene.
Play your timeline and look to see how long your particles live: See if it works for the scene you are creating. Next, try another way of creating values for the lifetime of your particles. Switch the mode to **Lifespan PP** (per particle), and write this *mel* expression for them in the expression editor: `particleShape1.lifespanPP = rand(4);` (see Figure 4). This expression ensures that each particle lives somewhere between 1 and 4 seconds.

![Figure 4](image)

**Figure 4.** Once you have chosen Expression for the **Lifespan PP**, select the type of expression—in this example, **Create Expression**—from the drop-down list. Right-click the slot of the chosen attribute, then click **Create an Expression** from the list. The expression editor pops up.

### Adding Other Expressions

Now that you have added a per-particle lifetime (LifespanPP), create other expressions. Vary the sizes of the particles by putting *mel* expressions on the tail size and the line width. Add this creation expression to the tail size:

```
particleShape1.tailSize= .7+(linstep(0,lifespanPP,age));
```

This expression adds 0.7 to a value between 0 and 1, depending on the particle’s age. (Refer to Part 1 of this series for information about using linstep.) Add this next *mel* expression to the line width (see Figure 5). It randomizes the line width of the particles:

```
particleShape1.lineWidth= rand(1,1.3);
```
Dynamics and Particle Effects, Part 2

Figure 5. The expression editor with the three expressions you have just created. Notice that all of the expressions are creation expressions. You may decide you want your particles to live longer depending on the distance they need to travel.

Note: You can choose not to use the expressions and—for example—choose a constant tail size of 1 and line width of 1.

Adding the Force, Gravity

Next, with your particles selected, click Fields > Gravity. Accept the default of 9.8 in the gravity attribute editor (see Figure 6).
Figure 6. The Gravity Options window you bring up when first creating the gravity field. Later, you can change the gravity options in the attribute editor of the Gravity field.

In Maya*, the default working units are centimeters. It is best to keep that unit when working with Maya* dynamics, especially if you want to get real-world simulations. If you create your models in correspondence with the unit scale of centimeters, there should be a relationship between real-life physics and the dynamics in your scene. However, even this is problematic. For example, the Gravity field assumes that one unit is meter and sets the default gravity setting to 9.8 magnitude no matter what you have set your unit preference to. If you have been modeling and counting one unit as a centimeter, set the gravity magnitude to 980. For more information on this behavior, see the white paper Nucleus in Autodesk Maya.

Enter -1.000 in the Y direction field and 0.000 in the X direction and Z direction fields. The particles will be going down the y-axis. Now, assign a few other fields to your particles. Place a small amount of turbulence on the particles to make them look a bit more random as they fall. So that your particles are not falling directly downward, put a vortex field on them, letting the y-axis be the one around which they rotate. If you want the particles to slant a bit as they come down, put a magnitude of 10.000 uniform force on the x-axis. Doing so pulls all of the particles in the x-axis direction.

Rendering, Coloring, and Lighting Your Particles

Depending on how you choose to render the particles, there are a number of ways to add color to them. If you use the Maya* renderer, the MultiStreak particles will not show up unless you use the hardware renderer. Use the mental ray* renderer in Maya* so they will show up when you do a
software render. Rendering with mental ray*, you can also assign different shaders to your particles that can change their look. Doing so in the Maya* renderer is not possible, as you can then only render the MultiStreak particles with the Maya* hardware renderer. So, render your particles with the mental ray* renderer inside Maya*.

You can also add per-particle attributes to the color and transparency of the particles (see Part 1 for more information). The transparency could be useful for making the particles more transparent as they near the end of their life. However, for this example, assign a Phong shader to the particles and allow it to assign color and transparency to the particle. Map a ramp onto the color of the shader, and have the ramp go from white to a very light blue. Make the particles 50 percent transparent or fully opaque if you want to render out the particles and use them as a separate element in a composite. There, you can control the particles’ color and transparency.

For best control of the look of your rain, render out the particles as a separate element and composite them into your scene with a compositing program such as The Foundry* NUKE* or Adobe* After Effects*. Figure 7 provides an example.

**Figure 7.** A test render of the rain. You could take this segment into a compositor, as well.

**Creating a Bounce When the Rain Hits an Object**

Create a ground plane for the particles to hit and bounce up from. Select the particles, then the plane. In the Dynamics module, click **Particles > Make Collide**, as Figure 8 shows.
Figure 8. This image shows how you make the particles collide with a geometry. Here, you can also change the resilience, friction, and offset. Later, you can change them in the geoConnector (which can be found as a tab on top of either the attribute editor of the particles or the plane).

Creating the Splash

To create the appearance of a splash when the particles hit the ground plane, choose the particles, and then open the collision event editor by clicking **Particles > Particle Collision Event Editor**. In the Particle Collision Event Editor, shown in Figure 9, set **Event type** to **Emit** and **Spread** to **0.500**. The target particle will be the new particle2 that the event creates. Give particle2 an inherited velocity of about 0.030. Set particle1 (the original particle) to emit three particles and die when it hits the surface. Now, click **Create Event** at the bottom of the editor. You can go back and change these parameters at any time in the event editor by selecting **particle1** and **event0**, and then making your changes.
Select the new particle2 and the ground plane by clicking **Particles > Make Collide.** Doing so makes the ground plane a collision object for particle2. Now, particle2 will bounce up from the plane, not travel through it. Using the Dynamics Relationship Editor, assign the gravity you created earlier to it. Leave particle2 as points, and play your animation to see how it looks (see Figure 10).
Figure 10. A test render of the rain element from After Effects*

The image in Figure 11 was created from several stylized rain elements developed in Maya* and compositied in After Effects*. The two initial particle types were Multithreads and Clouds. Particles that were emitted after the first set of particles hit the plane were given collision properties with a plane. The waterfall effect was created as particles bounced and fell off of the plane.

Figure 11. An image from an animation of stylized rain I created to project behind a dance company performance. Many times, particles can be used to create design elements: Naturalistic effects are not always wanted.
Working with nParticles

The next sections give you a start on using nParticles in Maya*.

Creating Smoke with nParticles

In Part 1 of this series, you created smoke with the dynamic particles in Maya*. Now, you will create some smoke using nParticles in Maya*. Many of the settings are the same, but there are a number of built-in functions to nParticles that you can adjust.

Note: Nucleus has a space scale. If you are modeling in the default centimeters, set the scale to 0.010. If you are modeling in meters, set it to 1.000. For more information, see the white paper Nucleus in Autodesk Maya.

First, go to the nDynamics module. Under nParticles, perform these steps (see Figure 12):
1. Create a directional emitter with a rate of 25 particles/sec.
2. Set the direction to Y by entering 1.000 in the DirectionY field and 0.000 in the DirectionX and DirectionZ fields.
3. In the Speed field, use 0.872 (the default).
4. In Spread field, enter 0.200.
5. Name the emitter smokeEmit.
Figure 12. First emitter settings for smoke using nParticles. Choosing thick clouds gives you some good initial presets.

Figure 13 shows a test render for the smoke you just created.

![Smoke Test Render](image)

Figure 13. A test render of the smoke. Obviously, there is more work to do, but the initial nParticles preset of thick clouds gave you a good start. I used the Maya* software render for this image. Right now, until I adjust the color and incandescence settings, the animation would look blown out in a mental ray render.

Open the nParticles Expression Editor, and set the **Lifespan Mode** to **lifespanPP**. As with the other dynamic particles, there is a **PerParticle(Array)** attributes in which you can insert expressions and ramps. Add the following expression in the **LifespanPP** field (see Figure 14):

```python
nParticleShape1.lifespanPP= rand(6,8);
```
Figure 14. Notice that the Radius PP, Opacity PP, and RGB PP fields already have internal ramps set in them. nParticles gives you these built-in ramps: In the other dynamic particles, you would have to add them yourself.

In the internal ramp for Radius Scale, click at the far right of the ramp to add a position marker. Perform these steps (see Figure 15):

1. In the Selected Value field, enter 2.300.
2. The value of Selected Position should be 1.000.
3. From the Radius Scale Input list, select Age.
4. From the Interpolation list, select Spline.
5. In the Radius Scale Randomize field, enter 0.050.
6. In the Particle Size area, in the Radius field, enter 0.200.

This value in step 6 acts as the envelope for the radius scale. Experiment by changing the number. Setting key frames for it could be an interesting way to create an explosion—something to try out in another nParticles animation.
Figure 15. This is the internal ramp for the radius. You can set different values at the position markers. This ramp has two position markers, but you can add more by clicking in the ramp. Because the Radius Scale Input is Age, the particles will be born with the value of the position marker on the left and die with the value of the position marker on the left.

Figure 16 shows the smoke after another render.
Figure 16. The smoke is getting better. I’m still using the Maya* software renderer.

Now, click Shading > Opacity Scale, and add another position marker to the far right. Perform these steps:

1. In the Selected Value field, enter 0.120.
2. The value of Selected Position should be 0.600.
3. From the Radius Scale Input list, select Age.
4. From the Interpolation list, select Spline.
5. In the Radius Scale Randomize field, enter 0.050.

In this way, the opacity of the nParticles will become less as the particle ages. Let the Opacity setting remain at 0.980 (see Figure 17). If you want thicker smoke, adjust your opacity accordingly.
Figure 17. The smoke with the opacity changes. By making the smoke less opaque as it ages, you avoid the particles popping off when they reach the end of their life.

There are two more internal ramps to adjust—one for color and one for incandescence (see Figure 18). Sometimes, you will find that setting the color to black and letting the incandescence create the color is useful.
Figure 18. Choose the color and incandescence of your smoke. In this example, I have set the Color Input to Radius so that the larger the radius of the particle, the darker it will be. I set the Incandescent Input to Randomized ID so the incandescence of the particles will be random, varying somewhere between the values of the two position markers.

Figure 19 shows the smoke after another render.
Figure 19. In this image, the opacity has been increased, and the emitter rate is 50 particles/sec. The color and incandescence are derived from the ramps in Figure 14. I found that the particles in this animation were popping off too much at the end of their life—fine for an explosion.

**Smoke Variation**

You can create a variation of the smoke you’ve been working on using some per-particle expressions and ramps (see Figure 20). Set your emitter to 50 particles/sec, and then change your internal color and incandescence map to have only one position marker; make its value 0.000. Do the same for the internal opacity ramp.
Figure 20. Here, you can see that I have added ramps in the Per Particle (Array) Attributes area—one for Opacity PP and one for RGB PP. I am keeping the Lifespan PP expression I created earlier, and I am still using the same internal Radius PP ramp. The internal ramps for incandescence, color, and opacity have all been set to black—the 0.000 value—so that the per-particle attributes of Opacity and Color control the color and opacity over the lifetime of the particles.

Now, you can add the ramps for opacity and RGB (see Figure 21).

Figure 21. Here are the ramps you can add for the per-particle attributes—one for Opacity PP and one for RGB PP.

Figure 22 shows the final cloud render.
Conclusion

There is so much more to learn about the Dynamic systems and what you can do with them. Hopefully, this article has whet your appetite for learning more. Future articles in this series will focus on fluid effects, soft bodies, hard bodies, sprites, and nCloth, showing how you can use these tools to create realistic and abstract animations.

About the Author

Audri Phillips is a Los Angeles–based artist currently working and exhibiting in a variety of media that range from computer animation and motion graphics to her more personal work, oil paintings and video art (visual poetry). She has worked for more than 20 years as an artist/digital artist in...
the entertainment industry in Los Angeles at such large studios as Disney Feature Animation, Sony Imageworks, DreamWorks Feature Animation, Rhythm and Hues, Digital Domain, and Electronic Arts as well as a number of small design boutiques. Her production experience includes art direction, design, visual development, storyboards, color and lighting, efx, layout, modeling, and compositing. Currently, she is an adjunct professor at Woodbury University. Check out her work at http://www.vimeo.com/audri/videos.