Continuous Level of Detail

Using MRM and SDS To Achieve High-Quality Graphics in a Fraction of the Time

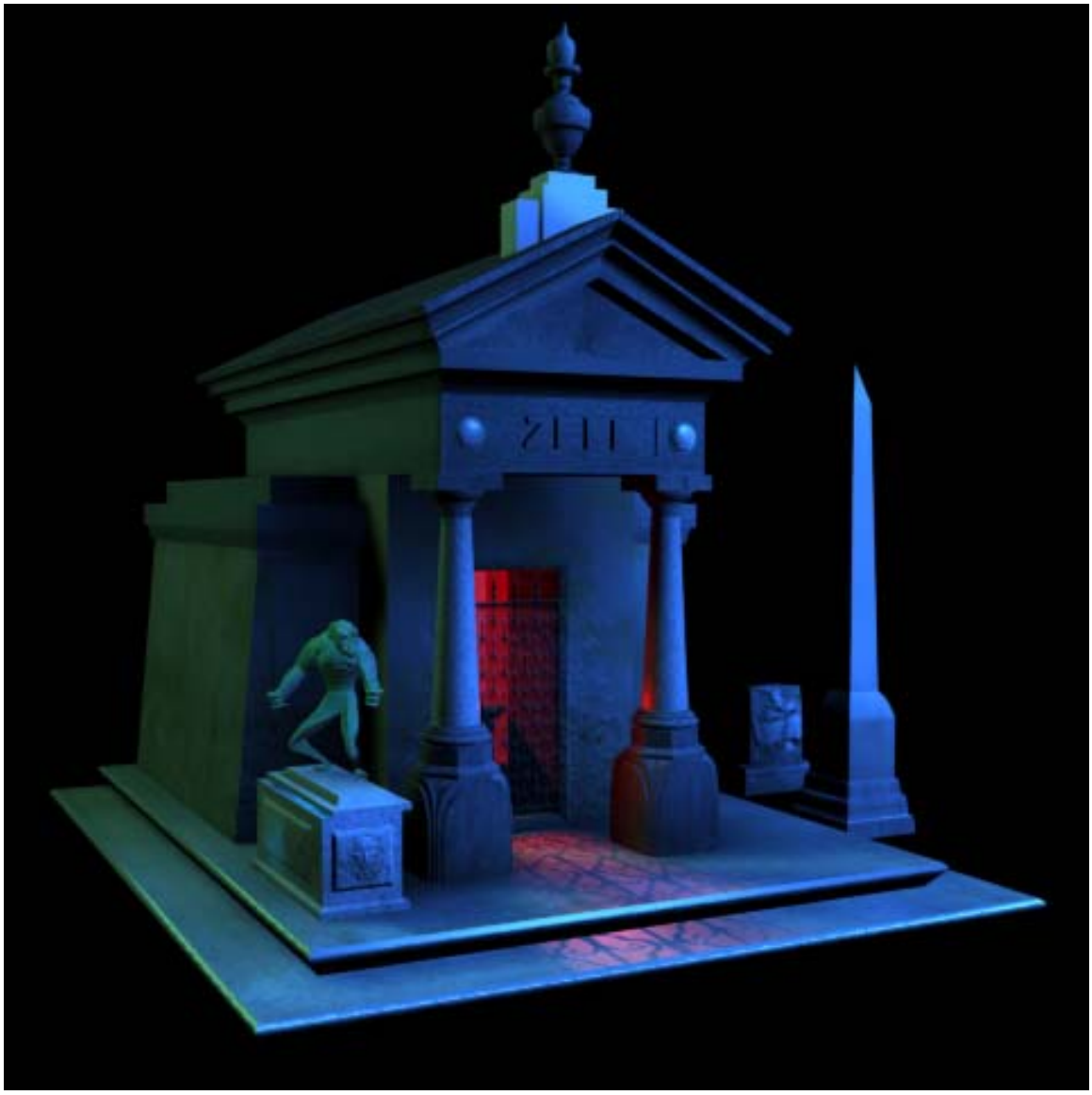

Figure 1. Beneath the textures and lighting it's really just an arrangement of geometric shapes (and in this case, tons of polygons). Responsiveness and look are two major components of enjoyable game-play. To keep a good balance of both, use only the polygons you need, and only when you need them.

It Looks Good, but It Runs Like a Slug

Well, that's a problem. We'd love that film-quality look, but our characters need to run, kick, and jump when we tell them to. Just as important, we'd like our games to play well and look good on as many different machines as possible. Polygon count isn't the only factor in determining the responsiveness of your game, but it's a very good place to start.

If you've already built 3D assets for either feature work or games, you already know the concept of "heavy" and "light" models: a heavy model contains many subdivisions or polygons, and is more difficult to move around than a light model containing few subdivisions or polygons.

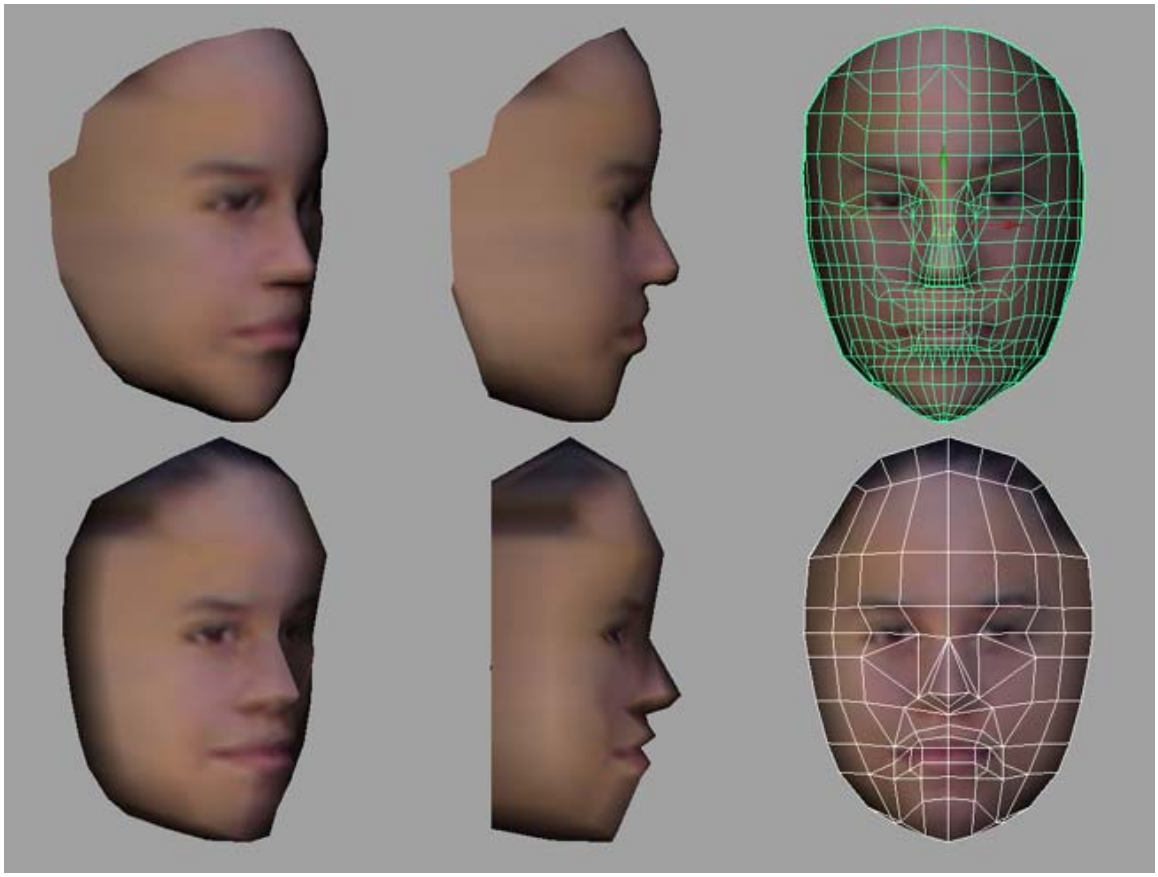

Figure 2. High and low poly-count models with the same face texture. Profiles suffer greatly on light models, but harsh shadows must also be dealt with in fuller views. Wire overlays on right show the underlying polygonal structure of both faces.

What's more, a scene populated with heavy models is much harder to traverse than one populated with much lighter models. For internet-based games, we also have to consider the effects of narrow bandwidth and slow download times on our content.

Unfortunately, building low-poly models doesn't solve everything. Few things look worse than a nicely-painted human face texture plastered across one flat polygon (see Figure 2). The trick is to use only as many polygons as you need.

This once translated into many painstaking hours of building separate versions of each model at various resolutions of detail. Models were swapped in and out, mostly based on their proximity to the camera—heavy models for close-ups, and light models for medium to long shots. Not only did it waste valuable production time, there were often visible "pops" as models were swapped during game play.

The best situation, of course, is to author your assets once, then let the game engine automatically and continuously adjust the level of detail as needed. Our focus technology here, Macromedia Shockwave* 3D, uses two related but distinctly different techniques for doing just that: Multi-Resolution Mesh (known as MRM), and Subdivision Surfaces (known as both SubDiv Surfaces and SDS).

Multi-Resolution Mesh (MRM)

Intel Labs <<link to www.intel.com/labs>> developed a special modifier, the #LOD modifier, for Macromedia Director* Shockwave Studio using their Multi-Resolution Mesh technology. The modifier intelligently adjusts the number of polygons in a model, while keeping the texture firmly in place. Keep in mind that MRM doesn't add complexity – it won't make a model more detailed than it was to begin with. It will, however, lower the complexity, and then bring the model back to its original form, as needed.

Therefore, the workflow for MRM begins with building the model at the highest level of detail **necessary**. I stress "necessary" because it's never a good idea to build models heavier than they need to be, especially if the intent is to download or stream them over the Internet.

What happens next depends mostly on your modeling or animation package. If you're a Discreet 3ds max* user, for example, you'll normally add the MRM modifier into your model's stack before it's exported with the Shockwave 3D plug-in. On the other hand, the Alias|Wavefront Maya* Shockwave 3D exporter does that automatically. It's best to check the documentation for your specific application and plug-in version.

Note: You can also add the #LOD modifier to exported Shockwave 3D elements on a "per model" basis through Lingo* scripting. It's also important to note that the #LOD modifier is not available for primitives (such as boxes and spheres) generated by Director itself.

After export, you'll then import your object into Director in a similar way to any other cast member. The only difference is that 3D objects are imported into Director specifically with the format "Shockwave 3D."

Once your object is a cast member, it's time to give Director some rules for how you want the level of detail handled for that cast member.

Working with the #LOD Modifier

The best way to illustrate this is through an example. The purpose of this example is to show the effect of MRM on an object. The spheres were generated in a separate 3D modeling package and are not Director primitives (see note above).

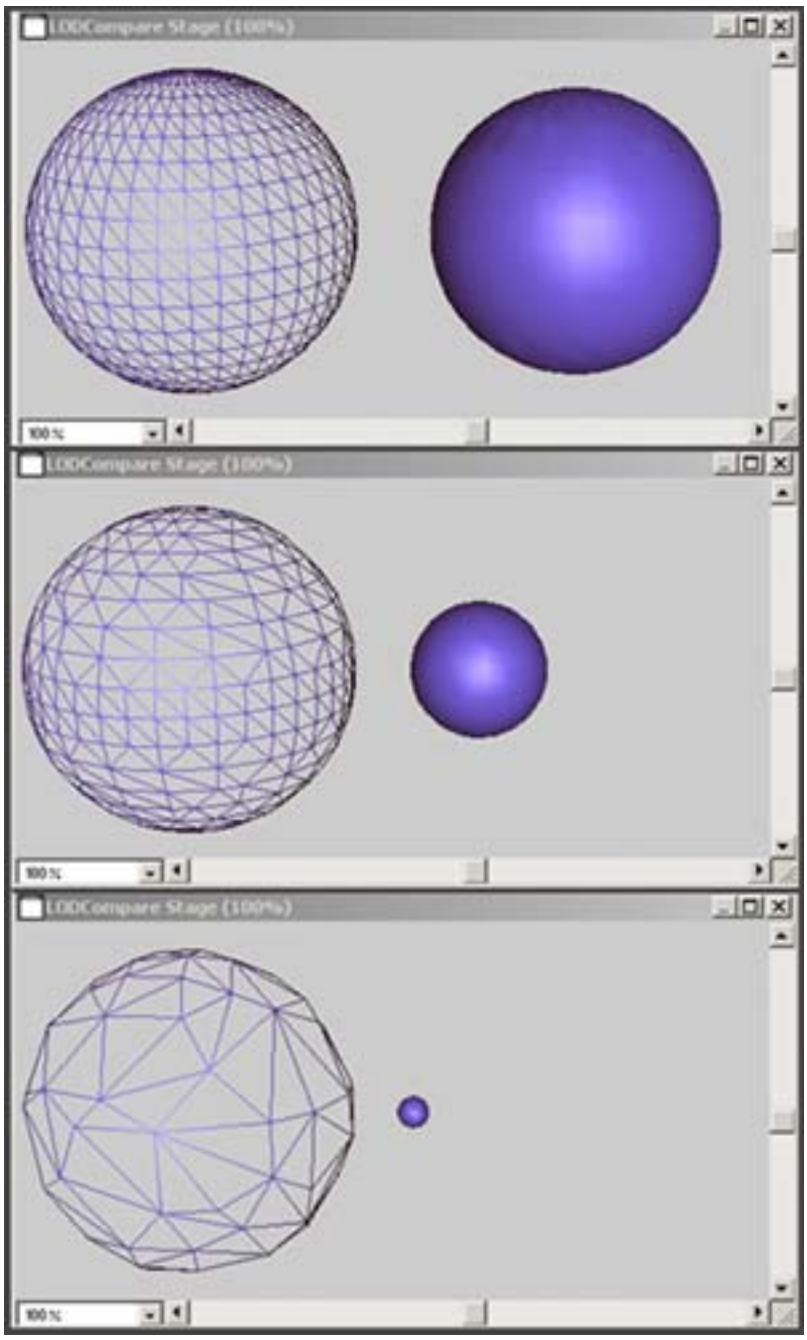

Figure 3. Multi-Resolution Mesh. Example 1. Compare Script.

The resolution of the stationary wireframe sphere (on the left) replicates that of the solid sphere.

As the movie plays, the solid sphere moves away from the camera, and then back again.

The #LOD Modifier has been set to automatically reduce the polygon count of both spheres as the solid sphere grows more distant. Even with a significant reduction in complexity, the detail isn't really missed.

New Properties

We've reprinted the script for Example 1 at the end of this section, complete with comments. The form and syntax of the script are nothing new, but the three properties of the #LOD Modifier most likely are new, so we'll discuss them here.

Auto

- Form: myModel.lod.auto

Default: TRUE (1)
With Auto set to true, polygon counts will automatically be reduced based on an object's proximity to the camera (near the camera = highest count, far = fewer polygons). Exactly how quickly (and how aggressively) polygons are removed is set by the Bias property.

Bias

- Form: myModel.lod.bias (functional value from 0 - 100)
- Default: 100
- Think of bias as "how few polygons do I want?" Lower values mean low polygon counts, or a very aggressive use of automatic polygon reduction. A value of 100 shouldn't affect your model's complexity at all.

Level

- Form: myModel.lod.level (functional value from 0 - 100)
- Default: 100
- The percentage of the original mesh you'd like rendered when Auto is set to FALSE (0).

Example 1 Script. Compare

```
global scene, MyModel, MySecondModel

on preparemovie
  member(1).resetworld() -- Reset the 3D cast member
end

on startmovie
  clearglobals  -- Clear all global variables
  scene = member("MyScene")  -- DeReference 3D scene castmember
  MyModel = scene.model("MyModel")
  MyModel.cloneDeep ("MySecondModel") -- Create second Model from first
  MySecondModel = scene.model("MySecondModel")
  MyModel.translate (-150,0,0) -- Move Original model to the left
  MySecondModel.translate (150,0,0) -- Move Copy to the Right
  MyModel.resource.lod.auto = 0
  MySecondModel.resource.lod.bias = 25
  MyModel.resource.lod.level = MySecondModel.resource.lod.level
  MyModel.shader.renderstyle = #wire
end

on enterframe
  updateposition()
  updateLOD()
end
```

```
on updatePosition
  global stepsize
  global directionToMove
  global numberOfSteps

  if voidp(stepsize) then stepsize = 30
  if voidp(directionToMove) then directionToMove = -1
  if voidp(numberOfSteps) then numberOfSteps = 0
  if numberOfSteps > 300 or numberOfSteps < 0 then
    directionToMove = directionToMove * -1
  end if

  MySecondModel.translate(0,0,-
stepsize*directionToMove,member(1).camera(1))
  numberOfSteps = numberOfSteps + directionToMove
end

on UpdateLOD
  MyModel.resource.lod.level = MySecondModel.resource.lod.level
end
```

Subdivision Surfaces (SDS)

Unlike MRM, which begins with a model of very high resolution and drops complexity as needed, the Subdivision Surfaces technology created by Intel for Director goes exactly the opposite way—it actually adds polygons as needed to give a more rounded appearance to your models. The main benefits here are quicker development time—it generally takes less time to fashion a less complex model—and faster download times due to smaller model size.

Note: It's important to understand that many authoring packages exist, and they don't necessarily agree on naming conventions. Similar tools and functions often have different names in different applications. Conversely, some tools and functions that have little or nothing in common can share the same, or similar, names. Subdivision Surfaces is a prime example. In the Shockwave Studio realm, SDS is a method for increasing the roundness of a model. 3D modeling packages, on the other hand, use the name Subdivision Surfaces to describe an actual geometry type.

Used in the right way, and on the right objects, Subdivision Surfaces can give a great look with a small download size. Will it make bad models good? Sometimes, but not usually. Building a model that takes full advantage of SDS takes a little forethought and some practice.

Artists, remember this: when you use Subdivision Surfaces in Shockwave Studio, you're really only building your model partway—you're depending on the computer to finish it. Yikes! This sounds fine to programmers, but it's a challenging concept for some artists, because it sounds like a total loss of control over their creation. Just remember there's an Art to doing it right.

So what does it take? And what sorts of things look good with SDS?

As with Multi-Resolution Mesh, SDS is best explained by example. Figure 2 illustrated how unrealistic low-poly faces can look from the side; faces are one of the biggest problems in keeping things "real" in a game, especially on Web-based games. We want that "low-poly" performance, but we hate that sharp-angled, "low-poly" look. Low polygon counts lead to little or no curvature in objects like facial profiles, which our senses tell us should look more organic and smooth to be believable.

So we'll use a nose to demonstrate. What SDS does is "intelligently" round objects, or parts of them. It's smart enough not to round things like the border edges of an object (which would immediately create ugly seams). It makes educated guesses on everything else, but the artist needs to provide the education.

So how do you (the artist) teach SDS (the computer) what to do in a given situation? It's really a matter of polygon placement and angles. Closely positioned polygons receive little or no rounding; angles that are too sharp, or not sharp enough, may round unpredictably (see Figure 5).

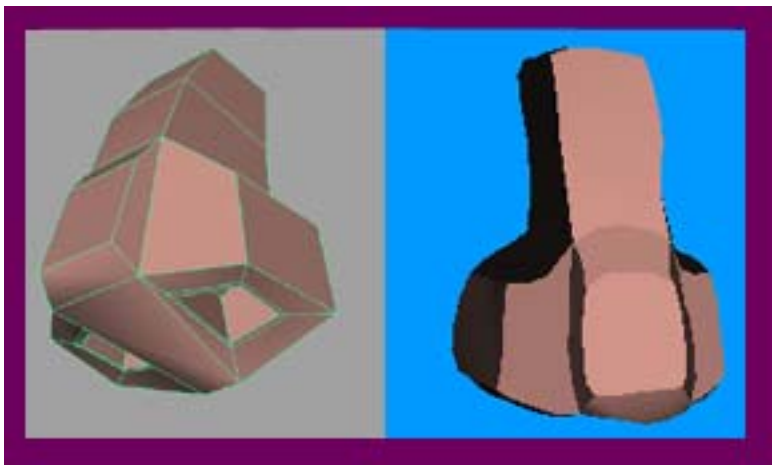

Figure 5. Just a little too blocky. SDS has smoothed the nose, but we really haven't given it enough information to work with in terms of polygons and angles. It obviously doesn't know what we want at this point—the rounding is unpredictable—look at the upturned nostrils.

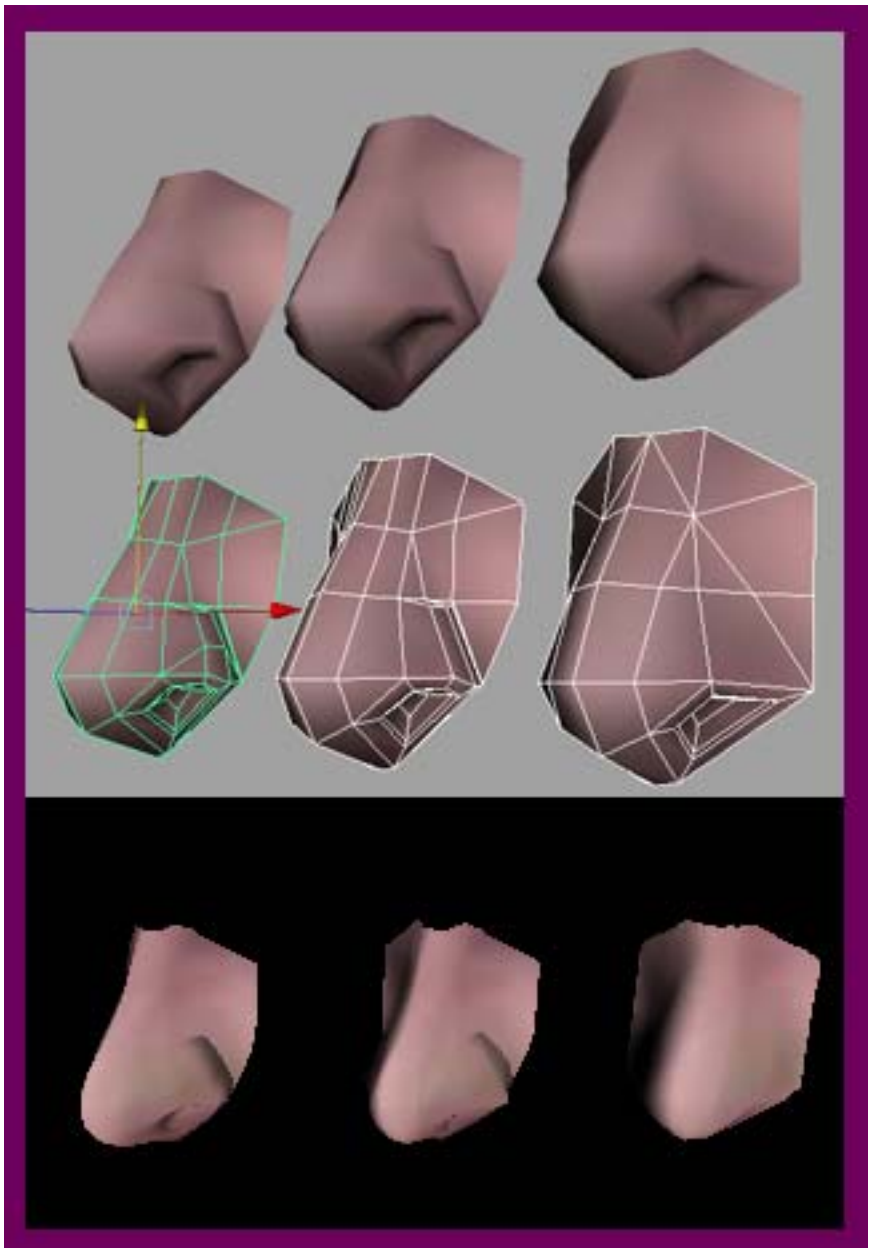

Figure 6. Various noses…

Without getting too much into modeling techniques here, it does take a little practice to coax SDS into giving you exactly what you want. Just making a simple, blocky model and giving SDS free reign is not going to do it.

In general, if you want sharp angles, double up on the polygons in that area. Help determine the slope you want by splitting polys to simulate the curves you need. Noses in the gray area are as they appear in Maya*. Below they have been subdivided twice in Director* Shockwave* Studio.

You've Modeled it – Now What?

Using SDS is fairly straightforward from a Lingo scripting point-of-view. Check out the script, SubdivApply (provided below). This script applies sds.depth and shader.renderstyle properties to the models we've brought into Director Shockwave Studio. It also changes the depth of subdivision and the style of the rendering output (wireframe or shaded), based on the keys pressed.

```
property pDepth, pStyle

on beginsprite me
  pDepth = 0
  pStyle = #fill
  repeat with i = 17 to 19
    member(1).model(i).addmodifier(#sds)
  end repeat
end

on enterframe me
  case the keypressed of
    "1":
      pDepth = 1
    "2":
      pDepth = 2
    "3":
      pDepth = 3
    "4":
      pDepth = 4
    "5":
      pDepth = 5
    "0":
      pDepth = 0
    "w":
      pStyle = #wire
    "f":
      pStyle = #fill
  end case
  repeat with i = 17 to 19
    member(1).model(i).sds.depth = pDepth
    member(1).model(i).rotate (0,1,0,#self)
    member(1).model(i).shader.renderstyle = pStyle
  end repeat

end
```

Conclusion

In conclusion, MRM and SDS are great techniques for reducing your graphics development time while still allowing you to create realistic 3D images. One product that includes both techniques is Macromedia* Director 8.5 Shockwave Studio, which can be purchased and downloaded at www.macromedia.com.

About the Authors

Steve Pitzel is a professional animator and technical marketing engineer for Intel Corporation and has been a computer graphics instructor and animator for six years. He began his graphic arts career in college as an editorial cartoonist and courtroom sketch artist. After converting from pencil to mouse, he went on to convert others, teaching 3D applications such as Softimage * , PowerAnimator * , and Maya * to traditional cell animators and computer graphic artists for Disney Feature Animation, Sony Pictures

Imageworks, VIFX/Rhythm & Hues and UCLA. He was a lead animator for the CBS feature, The Nuttiest Nutcracker*, and a senior artist for Mattel.

When Steve isn't animating, he's usually writing. St. Martin's Press published his first novel, Wizrd, in 1994 under his pen name, Steve Zell. Steve is currently working on his second novel.. You may contact Steve at steve.pitzel@intel.com.

Clint Taylor is a Technical Marketing Engineer in Intel's Microprocessor Research Lab. Clint can be reached at clinton.a.taylor@intel.com.