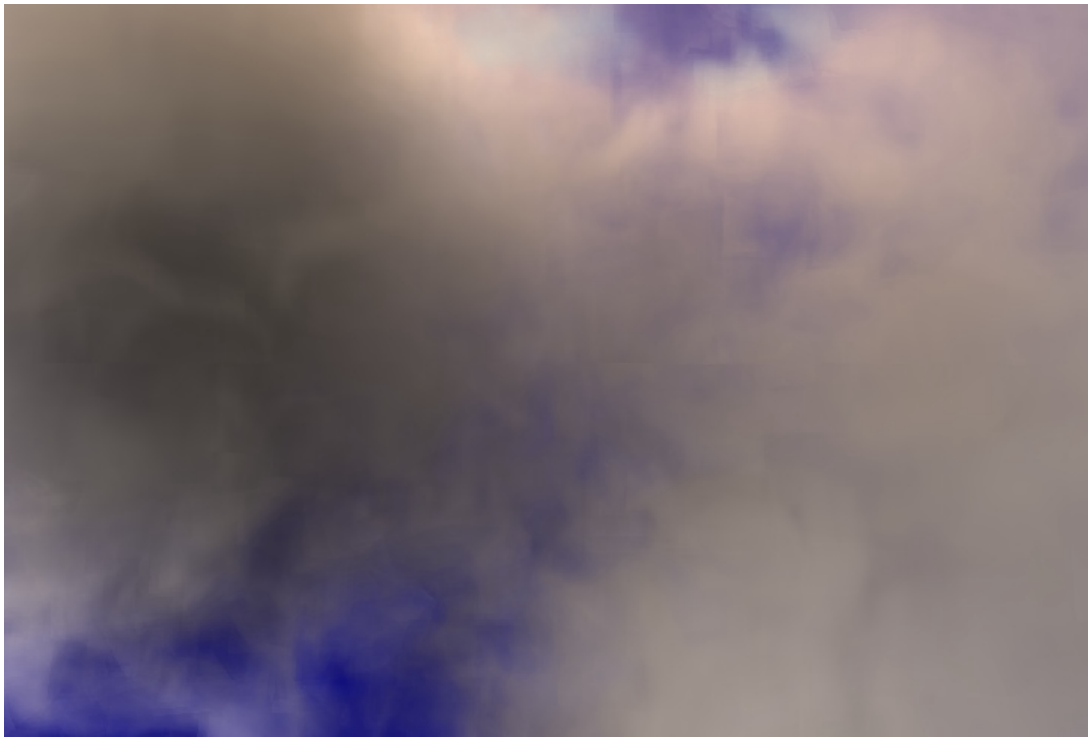
Procedural Techniques and Real-Time Graphics

David S. Ebert

ebertd@purdue.edu



1 An Introduction to Procedural Modeling and Animation

We are working in a very exciting time in computer graphics. The combination of increased CPU power with powerful, and now **programmable,** graphics processors (GPUs) available on affordable PCs has started an age where we can envision and realize interactive complex procedural models.

1.1 Procedural Techniques and Computer Graphics

Procedural techniques have been used throughout the history of computer graphics. Many early modeling and texturing techniques included procedural definitions of geometry and surface color. From these early beginnings, procedural techniques have exploded into an important, powerful modeling, texturing, and animation paradigm. During the mid- to late 1980s, procedural techniques for creating realistic textures, such as marble, wood, stone, and other natural material gained

widespread use. These techniques were extended to procedural modeling, including models of water, smoke, steam, fire, planets, and even tribbles. The development of the RenderMan[1] shading language [Pixar1989] in 1989 greatly expanded the use of procedural techniques. Currently, most commercial rendering and animation systems even provide a procedural interface. Procedural techniques have become an exciting, vital aspect of creating realistic computer generated images and animations. As the field continues to evolve, the importance and significance of procedural techniques will continue to grow.

1.2 Definition and Power of Procedural Techniques

Procedural techniques are code segments or algorithms that specify some characteristic of a computer generated model or effect. For example, a procedural texture for a marble surface does not use a scanned-in image to define the color values. Instead, it uses algorithms and mathematical functions to determine the color.

One of the most important features of procedural techniques is ***abstraction***. In a procedural approach, rather than explicitly specifying and storing all the complex details of a scene or sequence, we *abstract* them into a function or an algorithm (i.e., a *procedure*) and evaluate that procedure when and where needed. We gain a storage savings, as the details are no longer explicitly specified but rather implicit in the procedure, and shift the time requirements for specification of details from the programmer to the computer. This allows us to create inherently multi-resolution models and textures that we can evaluate to the resolution needed.

We also gain the power of ***parametric control***, allowing us to assign to a parameter a meaningful concept (e.g., a number that makes mountains "rougher" or "smoother"). Parametric control also provides amplification of the modeler/animator's efforts: a few parameters yield large amounts of detail (Smith [Smith1984] referred to this as *database amplification*). This parametric control unburdens the user from the low-level control and specification of detail. We also gain the serendipity inherent in procedural techniques: we are often pleasantly surprised by the unexpected behaviors of procedures, particularly stochastic procedures. Procedural models also offer ***flexibility***. The designer of the procedures can capture the *essence* of the object, phenomenon, or motion without being constrained by the complex laws of physics. Procedural techniques allow the inclusion of any amount of physical accuracy into the model that is desired. The designer may produce a wide range of effects, from accurately simulating natural laws to purely artistic effects.

2 Interactive Rendering and Procedural Techniques on PC Hardware

With the advent of programmable hardware graphics pipelines, what are the important factors for generating true, interactive procedural models? The following are several important factors that must be considered:

1. How much programmability is needed and available at the following levels:
    - Vertex

[1] RenderMan is a registered trademark of Pixar.

- Fragment

2. Precision of the programmable operations:
   - Are these 8-bit quantities? 9-bit? 12-bit?
   - Are they fixed range (e.g, 0 to 1)?
   - What precision is needed to not produce artifacts?
   - Are they signed quantities (e.g., 8-bit plus sign bit)?
3. What mathematical operations are available?
   - Addition and Subtraction? (some new boards don't allow fragment subtraction)
   - Multiplication and Division?
   - More advanced operations (sqrt, sin, cos, etc.)?
4. Are dependent operations allowed (can the results of one operation change the next computation)?

All of the above factors greatly affect the type of procedural models that can be implemented at interactive rates. Current hardware allows some programmability of the GPU and we can start to create interactive procedural models. However, the operations that are available at the fragment level are currently limited and the limited precision is a serious limiting factor. The ability to compute a result at the fragment level and use this to change a texture coordinate used in the next texture look-up is a basic capability that is now available and enables basic procedural solid texturing, etc. Unfortunately, to create amazing, complex procedural models in real-time we need the ability to generate new geometry at the fragment level. We could then download to the GPU a small procedural model that creates geometry representing our procedural model in real-time. This is a feature that probably won't be available for several years. But, with clever programming, we can create some very interesting, advanced procedural effects with the programmability available in the latest graphics boards.

3 References

[Pixar1989] Pixar, *The Renderman Interface: Version 3.1,* Pixar, San Rafael, Ca. 1989.

[Smith1984] Smith, A., "Plants, Fractals, and Formal Languages," *Computer Graphics (SIGGRAPH '84 Conference Proceedings),* 18, July 1984.