Steven Batchelor

Ms. Scalf

ENG-111

November 11, 2011

Procedural Generation and the Future of Media Production

In media production and computer science procedural generation has become a commonly used term. Basically, procedural generation is used to refer to content which is generated using algorithmic techniques rather than manual ones. There is a major advantage to this type of content creation in that the content can be generated as it is needed (Ebert 2). This means that an application which takes advantage of procedural generation will require much less memory than those which store every detail of content which was previously created by an artist (Ebert 2). Meshes and textures are the most commonly produced content. However, sound can also be produced with procedural generation and is commonly applied in speech and music synthesis. In my opinion despite the fact that software developers have tried to apply procedural generation methods for a number of years, only a limited set of products have managed to employ the technique in an extensive manner.

Procedural generation has become a common technique which can be used in almost any programming language. In fact, procedural generation is nothing more than mathematics which is the bases of even the most primitive computer programming language. Today, we have a pretty large number of graphics applications that have employed procedural generation in one form or another including video games and content creation suites. One example of these applications would be the commonly used *demoscene* which adopts the procedural generation technique as a way of packaging audiovisual content.

Throughout the last decade or so in which I have been programming as a hobby I have found that I always have to relearn everything I thought I knew. Recently in an interview I conducted for a class assignment I asked a fellow programmer, who works in the computer science industry, what, in his opinion, is the most important characteristic/trait needed to excel in the computer science field. Ryan Murray, the programmer I am speaking of, responded very fittingly by saying, "Always having the attitude of student goes a long way" (Murray). The truth behind his statement cannot be more evident than in the studying procedural generation. Procedural generation has been used throughout the history of computer graphics but recently it has been gaining more attention and countless algorithms have been developed to do almost anything one could imagine. To fully understand every aspect of procedural generation requires much dedication. There are more algorithms and theories out there which use procedural generation than can be covered here. Not to mention all of the hardware specific implementations and optimizations. In this paper I only intend to cover a brief summary of some of the implementations where I believe procedural generation could truly shine, and I will also try to explain them in plain English with as little techno jargon as humanly possible.

One of the main implementations which I feel could have the largest impact on video games in particular would have to be procedural planets. However, when creating the procedural planets for a real-time virtual simulation such as a video game, it is important to note that a high-frame rate is usually required for smooth animation. This is usually achieved through dynamically decreasing or increasing the planet's level of detail depending on its position relative to the observer. The unnecessary detail is not drawn since the observer will not be expected to see it. Basically, to do this the standard approach begins with some type of tree data structure such as a quad-tree which can form panels for each side of a tetrahedron, cube, or

octahedron. As the observer approaches, the panels will split into three or more smaller panels. The panels are warped in order to create a sphere, but special care is given to ensure they are properly connected to the edges of surrounding panels.

Some type of noise algorithm is then used to distort the surface of the sphere and generate a terrain. Most commonly *perlin noise* is used but there are many other options to generate the noise. A good example would be the algorithm I developed which uses a three dimensional texture and smooth texture filtering also called trilinear filtering. My approach is better optimized for modern computer graphics hardware but by no means is it a better general purpose method. After the mesh has been constructed and the noise has been applied the procedural planet can be observed in three dimensions. Proper calculations are also needed to present things like rivers, movement of the sun or even rotation of the earth and satellites depending on the intended planet. Clouds, trees, plants, and even cities can also be generated on the surface of the planet using procedural generation (Ebert 568). Some well know applications use procedural planets include *MojoWorld* and *Terragen* but for the most part these applications are not intended for real time simulations. However, many independent game developers and hobbyist programmers have developed their own implementations which can indeed render in real time. Beyond planets, procedural generation can even produce procedural galaxies and use these to create a procedural universe (Ebert 571).

Computer animations are yet another area I feel procedural generation could have a huge impact. Procedural animation is one method that has been commonly used to produce more realistic motions. Through procedural animation, it is possible to simulate the movement of specific things such as particle systems, water, smoke, and fire (Mahoney). Character animation can also be procedurally generated with astounding results. The animated product simulates a

real event thereby making it more or less realistic. When it comes to video games, animation has been used to achieve complex performances including death or collapse of a character. This is achieved through the use of interconnected rigid elements and bodies which have been programmed in such a way that they incorporate Newtonian physics.

Today we have more complex animations produced through procedural generation such as the use of characters which can walk, pick different things up, and drive vehicles. Animations in real-time have also been adopted and can even punch, dodge and react to the specific environment. Through procedural generation, it has become extremely possible to simulate almost everything through animation and it is the reason modern animated games have become more enjoyable and advanced. Therefore, through animation programming, it can be possible to create anything and make it more realistic. This has also been incorporated in the field of engineering to simulate plans and explain how completed projects will work.

One of the greatest procedural methods for improving computer graphics would have to be the use of procedural textures. Though they are heavily implemented in computer-generated imagery for movies they have rarely been used in video games. I believe perhaps this is simply due to relatively cheaper cost of producing art than complex algorithms for every type of texture used in a video game. A procedural texture is basically an image that is computer generated and tends to imitate a real one. This is usually achieved through the use of algorithm which is purposely intended to simulate and create a real depiction and representation of a given natural element. This is done for materials such as metals, marble, wood, stone, and granite. In order to achieve the natural color and look of the simulated textural image, the use of turbulence functions and fractal noise is integrated. Such functions have to be incorporated because of their randomness and numerical representation which a common aspect of nature. They offer a major

advantage over image based textures in that when an object is textured with an image it tends to produce noticeable seams but this is not a problem with procedural textures since they are produced randomly (Ross 273).

Currently, there are a number of programs which have been found effective in the creation of varied textures through procedural texturing. These include *DarkTree*, *Filter Forge*, *Algorithmic Substance* and *Texture Garden.* With these programs, it can also be possible to create self-organizing textures which start from random noise. Using self-organization a structured pattern is developed in a randomized manner. A good example of a self-organizing production of procedural generation is the reaction-diffusion produced using specified programs such as *Algorithmic Substance.* Procedural textures can be produced through solid texturing which is a process used to generate textures and *Perlin noise* forms the basis function (Ebert 413). We also have genetic and cellular texturing which have become very common today (Ebert 135).

It has been possible to produce music which is entirely created through the use of procedural generation. However, even though it has been possible to produce music through procedural generation for a number of years it has been rarely used in any type of media. The common term used for this kind of music is *Generative Music* and the term was described and popularized by Brian Eno. Another thing about this kind of music as described by Brian is that it remains ever changing since it is system-created (Collins and Brown). The production of this kind of music revolves around structural, procedural, emergent, and interactive theories. Currently we have an increasing number of software and computer systems that have been created and properly written in such a way that they can successfully create and compose

generative music (Collins and Brown). Some of the software which can be used to create this kind of music includes *SSEYO, FractMus, Nodal, and Bloom* among others.

Basically, what this means is that today we have programs which are capable of producing generated music. This kind of music has also been incorporated in animated films and games thus arousing in the audience the best sense of feeling and attachment depending on what is on the screen. However, some individuals strongly believe that generative music is the best kind of music while others argue that the real music should be originally performed by a musician (Collins and Brown).

Procedural programming has become a common practice and as a result it has been adopted for several generations. This form of procedure revolves around different routines and functions and has made it possible to produce different procedural elements such as animations, textures, planets, and even music. As a result, these aspects of production have been adopted in different areas such as film production, music industry, and also in the production of computer games. Procedural generation has become the talk of media production today and it has helped the industry grow at a very fast rate. Future developments especially in the areas I have covered could have a huge impact on movies, video games, and almost any other types of media.

Works Cited

Ebert, David. Texturing & Modeling: A Procedural Approach. New York: Morgan Kaufmann, 2002.

Mahoney, Diana Phillips. Procedural animation. Computer Graphics World, 20.5 (1997): 39. Academic Search Premier. Web. 29 November. 2011.

Murray, Ryan. Personal interview. 2 October 2011.

Nick Collins and Andrew R. Brown. Generative Music Editorial. Contemporary Music Review, 28.1 (2009): 1-4. Academic Search Premier. Web. November. 2011.

Ross, Brian J. Procedural Texture Evolution Using Multi-objective Optimization. New Generation Computing, 22.3 (2004), 271-293. Academic Search Premier. Web. November. 2011.