

Simple Procedural Geometry Tools Tom Kelly's PhD Proposal

There are many methods for creating procedural geometry. However, in order to be able to generate sufficiently interesting geometry, these methods are of a high complexity. This research is to ascertain whether it would be possible to reduce the complexity of the mechanisms without reducing the complexity of the resulting geometry. The motivation for this is twofold: firstly, there is an increasing need in industry for procedurally generated geometry; secondly, there is a need for such geometry to be under the artistic control of people who are not computer programmers.

Introduction

As the available computing power increases, virtual environments become increasingly complex. This adds to the sense of immersion and realism for those experiencing these environments, but also increases the resources it takes to create them.

As graphical capabilities continue to increase, so does the cost of creating an environment to utilize them. A typical video game development company allocates sixty percent of its staff to create contentⁱ. This leads to increasing costs and fewer innovative products.

Procedural generation of geometry gives a solution to this problem by defining a grammar that evaluates to a geometric object. By making this grammar stochastic or sensitive to itself, each evaluation of the grammar can produce different results. For example a forest of unique trees may be created by the repeated application of a grammar representing a single tree.

Procedural content has the following benefits:

- Automatically generated geometry removes the restriction that the size of an environment is proportional to the time spent creating it.
- The quality of the geometry is consistent at no additional cost.
- In the long term efficient procedural geometry tools could lead to run-time environment generation, giving more variety and less repetition to the user. It offers the potential to generate content that reacts to various stimuli. For example it could respond to the current hardware capabilities, to users' level of expertise or to the length of their attention span.
- In order to fill an environment with limited resources, objects are currently often repeated. Procedural content can generate sufficient variation to avoid repetitionⁱⁱ.

Lindenmayer described his (L) systems in 1968ⁱⁱⁱ as an attempt to describe the development of plants using a formal grammar. To model plant growth convincingly the traditional, sequential replacement of a Chomsky grammar was discarded in favour of a parallel, generational growth model. By 1984 Smith^{iv} was using deterministic L-systems to generate realistic forests and terrains. From this base, progress has been made recently by adding features to the basic L-system.

In 2001 Müller^v developed a city architecture generation tool that uses a self-sensitive constrained L-system, to generate an accurate and adaptable road network. By adding external factors such as trunk roads between population centres the level of realism was further raised. Between the roads another L-system constructed simple skyscrapers. By 2006 Müller^{vi} *et al.* had advanced their work, creating architecture using a sophisticated shape grammar. Shape grammars, as defined by Stiny and Gips^{vii}, allow an input shape to be transformed by choosing from a chain of inference rules. Müller's use of these transformations included a set of constraints to filter a large set of production rules in order to generate intricate and realistic geometry.

As we require more detailed geometry, there is a trend towards increasingly expressive grammars, using context sensitivity at a grammar level and self-sensitivity at a geometric level. Trying to create an intuitive, easily editable language that provides the required inputs for these systems is difficult and this has led to procedural techniques being successful only in limited domains. Indeed some recent systems such as the 'greenery' generator Grove^{viii} avoid an intricate grammar in the search for simplicity. Successful domains of procedural techniques are relatively small in comparison to the

variety of digital art now being created. Some of these domains include the construction of forests^{ix} and texture creation^x.

Proposal

Current procedural modelling techniques tend to use a strong grammar, coupled with weaker, simple primitives to create the desired results. I propose to investigate the premise that more intelligent primitives would allow for a simpler grammar. This would enable the creation of a simpler and more expressive tool for the creation of procedural geometry.

Three-dimensional modelling tools have evolved from simple primitives, such as cuboids and spheres, into sophisticated composite techniques, for example mesh deformers or 'ZSpheres'^{xi}. The proposed research is to examine the consequences of evolving procedural primitives from the simple methods used in geometric grammars today, towards the composite primitives used by artists.

By using procedural primitives more akin to those used by 3D artists, I hope that these artists may be able to define procedural content using many of their existing skills and workflows. Furthermore, by working towards more intelligent primitives we could reduce the complexity and tight domain coupling given by more complex grammars. For example my master's thesis^{xii} stumbled upon the weighted straight skeleton (akin to an artist's extrude tool) as a method suitable for the synthesis of many man-made objects. It has a geometric level of self-sensitivity that removes a burden from the controlling grammar. Presenting a simpler, more powerful grammar to the non-programming user, has the potential to allow procedural geometry to be used by more people in increasingly varied domains.

Work plan

I intend to start work with a review of the current grammars used in procedural content, as well as refreshing myself with some language theory. In this period of reading I want to compare the powerful and brief Turing complete grammars, with simpler regular grammars that offer easier editing. One of the first challenges here will be to identify and experiment with common interfaces between parts of a grammar, and how this affects the way symbols may be combined.

I hope to move on to research and catalogue the primitives currently in use by artists. This will encompass many subjects surrounding the primitives, from the underpinning mathematics, to the range of applications for a function as an artist's tool. Reading on this topic will focus on existing modelling software packages and related design papers.

The third area of research would be to study complete procedural systems as presented to the user and identifying factors that make the systems successful. This will hopefully reveal some 'best practice' workflows and clarify the deficiencies in defining procedural content. As there are so few procedural tools available, a suitable substitute may be the process of using a procedural toolkit from within a programming environment.

At the end of my first year my goal is to have a solid grasp of the current state of the art, both with regard to realism of geometric output and workflow. I will be able to compare existing procedural systems and comment on their similarities to the current art toolkits. Beyond this I intend to prototype and evaluate new concepts for a simpler procedural tool-chain. While it is difficult to test for simplicity, one possible approach is to make an accessible game of the problem^{xiii}. If one can describe an object to an in-game partner by creating a grammar, only the evaluation of which is communicated, then some level of simplicity has been achieved in the grammar.

- ⁱ 'Exuberant youth to sustainable maturity', Department of Trade and Industry, Spectrum
- ⁱⁱ 'Kkrieger' a 64KB game, with deterministic procedural content (<http://www.theprodukt.com/kkrieger>)
- ⁱⁱⁱ Mathematical models for cellular interaction in development, Parts I and II. Lindenmayer, pages 280-315, 1968
- ^{iv} Plants, Fractals and formal languages, Proceedings of the 11th annual conference on Computer graphics and interactive techniques, pages 1 - 10 ,1984
- ^v Procedural Modeling of Cities, Y. I. H. Parish and P. Müller, Annual Conference Series, ACM, pages 301-308, 2001
- ^{vi} Procedural Modeling of Buildings, P. Müller, P. Wonka, S. Haegler, A. Ulmer and L. Van Gool., ACM Transactions on Graphics, ACM Press, Vol. 25, No. 3, pages 614-623, 2006
- ^{vii} Shape Grammars and the Generative Specification of Painting and Sculpture, Stiny, G. and Gips, J., IFIP Congress, 1971
- ^{viii} Grove: A Production Optimized Foliage Generator for "the Lord of the Rings: the Two Towers" Mat Aitken *et al.* Computer graphics and interactive techniques in Australasia and South East, Special short paper presentation table of contents, pages 37 - 38, 2003
- ^{ix} IDV Incorporated's SpeedTree (<http://www.speedtree.com/>)
- ^x Allegorithmic's ProFX Engine (<http://www.allegorithmic.com>)
- ^{xi} A tool for sketching 3D models (<http://www.zbrush.info/site/index.php/ZSpheres>)
- ^{xii} 'City Architecture Generation', Tom Kelly June 2006 (<http://www.cs.bris.ac.uk/home/tk1748/sity.pdf>)
- ^{xiii} Similar to the ESP game (<http://www.espgame.org>)