Consistent Real-Time Procedural Terrain
Using Multiple Algorithms

Motivation
Consistent real-time terrain is useful in many applications, from games to simulations to visualizations to previews of prerendered CGI. Hand-crafted terrain is expensive in time during creation, in storage, and in rendering. Procedural terrain is an answer but single algorithms generally lead to insufficiently realistic and complex results, whereas combining multiple algorithms can lead to consistency problems. Further, procedural techniques suffer lack of control compared to hand-crafted content. This project is focused on creating procedural terrain, traversable and renderable in real-time, using multiple algorithms but retaining consistency.

Background
Procedural modeling and texturing is an approach to 3D content creation that relies on algorithmically-generated geometry and textures rather than hand-crafted content. It has major advantages regarding scalability but must sacrifice control. A particular target for procedural graphics has been terrain, since terrain requires scalability and can be simulated with some realism via fractal-based approaches. The scalability works on multiple dimensions: procedural terrain can be generated to an infinite extent (assuming only a certain portion is visible); such size requires no more work than a small patch of terrain; it requires only the storage of the visible geometry and the algorithm; with an appropriate level-of-detail manager it can support high fidelity from closest to farthest zoom.

However, procedural generation of terrain suffers from the above-mentioned deficits in realism or consistency and in control. Generating terrain from a single algorithm yields unrealistic results, where mountain ranges are indistinguishable from small piles of rock and where the terrain appears completely uniform in its roughness across its expanse. Another limitation is that elevation is generally created via a height map (a representation of elevation in the form of a grayscale image where brightness indicates elevation) which precludes the possibility of overhangs. Further, besides these general differences in height there are few recognizable features like valleys, canyons, riverbeds, lakes, and the like. Adding complexity to the terrain requires the use of additional algorithms, and this can lead to a lack of consistency in the output. Consistency is also a problem if one wishes to add other sorts of features, such as vegetation,
buildings, or craters, which must blend correctly into the generated terrain. Here consistency requires control of the local terrain so as to not feature discontinuities and to provide proper surfaces for the additional features. Finally, if the additional features are placed procedurally, one might still wish some control over that placement without fully specifying each location.

**Project Overview and Goals**

Much work is at present being done on procedural graphics, both on the individual algorithms and on bringing them together consistently. This project does not seek to break new ground on the procedural algorithms themselves, but rather to combine multiple algorithms in a way such as to yield consistent terrain graphics with sufficient control over both the general terrain and the placement of features. A further combination of algorithms is necessary since the viewpoint is not static; some level-of-detail management must be undertaken to allow dynamic creation and reduction of geometric detail in the terrain. Thus the project will involve not only the implementation of the various algorithms but also their selection, and the documentation of that selection process and the benefits and costs and consequences of the choices.

The particular form of this project will be the terrain system of a flying game, which terrain will feature basic elevation, additional topographical aspects (valleys, riverbeds, and the like), and added features (craters, building complexes) applied to the base terrain. The terrain will feature dynamic complexity that scales, culls, and manages level of detail given the view frustum, supporting closeness to the surface from a meter to millions of kilometers. In order for a high level of detail to be supported, in addition to level of detail management as much computation as possible will be offloaded to the GPU. To allow real-time demonstration of how the project handles control and consistency, real-time changes in control parameters and added features will be possible, as well as simply moving the viewpoint.

The goal of this project will therefore be a complete real-time terrain generator that includes all these features and an examination of the combination of algorithms that best produce that terrain system.

**Roadmap**

There will be three major milestones for the project:

- By spring break, March 8: A working implementation of the base terrain algorithm with interactive control over its constants

- By third week post-break, April 12: Implementation of at least two additional terrain features, such as valleys, craters, collections of buildings, etc.
• By end of the Reading Period, May 1: Finished executable and report. If time permits procedural texturing of the generated terrain will be added to the procedural modeling.

**Deliverables**

This project will deliver source code, any content used, and a program executable on the workstations in the Graphics Lab, featuring real-time terrain with the aspects outlined above. With this program will be submitted documentation as to its implementation and use. The terrain, the implementation choices, the methods of ensuring consistency and sufficient control, and lessons learned from the choices and interaction of algorithms will be documented and discussed in a report. In addition the report shall feature the goals and an overview of the project, background on procedural graphics, and any deviations from this proposal. Finally, still images and videos of the program in operation shall be provided.

**Key Sources**


