Computer Science Honours Project Proposal

A Procedural, Minimal Input, Natural Terrain Plug-in for Blender

by Bruce Alcock <bruce.alcock@gmail.com>

Department of Computer Science, Rhodes University

Supervisors: **Shaun Bangay <S.Bangay@ru.ac.za> Kevin Glass <K.Glass@ru.ac.za>**

---

1. Problem to Be Addressed

Implement a procedural, minimal input, natural terrain plug-in for Blender (an opensource 3D modelling program) which is capable of producing massive, realistic terrains. Currently no such plug-in exists. Many techniques for rendering different types of terrain for different applications such as simulators and games exist, and the plug-in will be based on techniques developed in those fields.

2. Background and Relation to Previous Research

The context of this project is a Text-to-Scene (TTS) application, in which a piece of text is fed in as input, and a realistic animation produced as output with the intention of fully automated movie production. The TTS requires the generation of terrain as a basis for scenes, and due to the whole process being as automated as possible, this requires that the plug-in have minimal specification or manual intervention. Blender is being used as the tool for scripting and lacks any complex model generation procedures.

3. Intended Approach

The project will be executed in the following steps:

1. Investigation into various techniques and decisions concerning requirements for terrain plug-in.
2. Implement terrain generation techniques in an OpenGL application:
   This will have procedurally built in constraints, nothing user specifiable.
   (an OpenGL application is being used at first to give results in real-time and allow for tweaking and optimization of procedures)
3. Implement this algorithm in a python script for Blender
4. Implement a procedural interface for the Blender plug-in:
   Where to place terrain features determined by the artist.
   Size of the terrain specifiable at runtime by the modeller.
5. Implement Level Of Detail (LOD) factors into the plug-in
6. Time permitting: implement some of the extensions (see Section 7)

First Semester:	**Proposed Dates:**
Study material, and formulate method of creating terrain.	25 Feb – 5 Mar
Start project website.	5 Mar – 11 Mar
Implement terrain procedures and OpenGL rendering.	12 Mar – 8 Apr
Create non-configurable Blender plug-in.	9 Apr – 6 May
Start artist configurable plug-in.	7 May
Second Semester:	
Finish artist configurable plug-in.	1 Jul
Implement LOD calculations.	2 Jul – 29 Jul
Testing, Refinements and Extensions	30 Jul – 19 Aug
Write final paper	Due week 2, 4th term

4. Initial Survey of Resources and Literature

Several papers will be used to implement the Blender plug-in:

The terrain geometry will be sourced from two papers:

- *Terrain Generation Using Genetic Algorithms* (Ong, Saunders, Keyser, Legget, 2005): using their idea of tracing out initial lines to represent mountains and then procedurally expanding these to be more detailed and interesting. This seems a good candidate to implement the procedural interface with too, as the user could trace out a rough diagram in Blender and then select this to become terrain.
- *Realtime Procedural Terrain Generation: Realtime Synthesis of Eroded Fractal Terrain for Use in Computer Games* (Olsen, 2004): which details eroding terrain to make flat areas and more natural looking terrain.

LOD in texturing will be implemented using *Texturing Techniques for Terrain Visualization* (Döllner, Baumann, Hinrichs, 2000) using the idea of multi-texture pyramid trees that correspond to the geometry LOD tree, which will inspire LOD geometry too.

5. Preliminary Design Considerations

The terrain plug-in is being developed for Blender because it is open source and as such freely available, highly scriptable and capable. It is also available for a variety of platforms. Since writing plug-ins requires rebuilding source code, the platform to be used will be Fedora Linux as this is probably the simplest and most stable environment to do this in.

Deign considerations for actual script/plug-in:

- Procedural
- Conforming to constraints
- Types of terrain to focus on (hills, valleys, etc)
- Level of detail
- Level of realism
- Preliminary techniques to be investigated (placement, erosion, etc)
- Speed (optimization of procedures)

6. Expected Results

- A configurable natural terrain generation plug-in for Blender.
- Enhanced Python API (will probably need to extend it)
- Interaction with TTS as well as a city modelling Blender plug-in being developed simultaneously.

7. Possible Extensions

- Generating the foliage and other items for the terrain: placing dynamic trees, grass, rocks, etc. on the terrain, making it more of a landscape.
- Implementing world conditions such as clouds, fog, rain, river flow.
- Texturing.
- Investigation into generation at render-time.

8. References

Ong, T.J., Saunders, R., Keyser & J., Legget, J.J. “Terrain Generation Using Genetic Algorithms” from *Proceedings of the conference on Genetic and evolutionary computation* (2005). ACM Press. 25 February 2007. <http://portal.acm.org/ft_gateway.cfm?id=1068241&type=pdf&coll=Portal&dl=ACM&CFID=15681032&CFTOKEN=99776635>

Döllner, J., Baumann, K. & Hinrichs, K. “Texturing Techniques for Terrain Visualization” from *Proceedings of the conference on Visualization* (2000). IEEE Computer Society Press. 25 February 2007. <http://portal.acm.org/ft_gateway.cfm?id=375246&type=pdf&coll=Portal&dl=ACM&CFID=15681032&CFTOKEN=99776635>

Olsen, J. “Realtime Procedural Terrain Generation: Realtime Synthesis of Eroded Fractal Terrain for Use in Computer Games” (31 October 2004). 25 February 2007. <http://www.oddlabs.com/download/terrain_generation.pdf>