EGVE Symposium (2008)
B. Mohler and R. van Liere (Editors)

A Proposal for a Procedural Terrain Modelling Framework

R.M. Smelik [1], T. Tutenel [2], K.J. de Kraker [1], R. Bidarra [2]

[1]Modelling & Simulation Department, TNO Defence, Security and Safety, The Netherlands
[2]Computer Graphics & CAD/CAM Group, Delft University of Technology, The Netherlands

Abstract

Manual game content creation is an increasingly laborious task; with each advance in graphics hardware, a higher level of fidelity and detail is achievable and, therefore, expected. Although numerous automatic (e.g. procedural) content generation algorithms and techniques have been developed over the years, their application in both games and simulations is not widespread. What lacks is a unifying modeling framework that combines these techniques in a usable manner. We propose to develop a new, high-level framework for automatic generation of virtual worlds (e.g. game levels, simulation terrain) that requires intuitive user input and results in a rich 3D terrain model.

Categories and Subject Descriptors (according to ACM CCS): I3.5 [Computer Graphics]: Computational Geometry and Object Modelling I.6.7 [Simulation and Modelling]: Types of Simulation — Gaming I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism — Fractals I3.5 [Computer Graphics]: Applications

Keywords: automatic generation of virtual worlds, simulation terrain, game-level design, procedural modelling

1. Introduction

Creating a terrain for a (serious) game or simulation is a laborious, repetitious task. While the first 3D game levels were small indoor maps that could easily be designed by hand, current game levels increasingly consist of outdoor terrain, and their size and visual fidelity grow with the advancements in hardware.

Military simulators have always used large outdoor terrain databases, but their level of detail was originally low; in some cases limited to a coarse terrain based on a DEM-file with a 2D satellite image draped across. Nowadays, much more attention is paid to the details of 3D terrain, e.g. vegetation and man-made structures. This increase in detail is necessary in order to achieve the level of fidelity that ground-based simulators need. It gives cues of sizes and distances in the terrain, essential for immersion and situational awareness [Wel05]. Moreover, it strongly influences a tactical situation (cover, line-of-sight).

For both games and simulations, the effort in time and money required for designing terrain by hand has become so high, that it is desirable to automate the design process as much as possible.

A promising approach to automating terrain construction is to use procedural content generation methods. Over the years, numerous procedural methods for generating, among other things, terrain and terrain features have been developed. Classical examples are height (or elevation) map generation algorithms, based on noise or fractals (e.g. [Mil86, Per85]), or more physically based: thermal or hydraulic erosion, plate tectonics, ridges and rivers (e.g. [EWM*03], see [Ols04] for a good overview of different methods). Much progress is made in automatic distribution of vegetation and plant model construction, using e.g. species, ecosystems and resource competition for distribution, and grammar systems for simulating plant growth [DHL*98]. Procedural methods are not limited to natural terrain: for generation of urban environments (cities, road networks, buildings with their facades and interiors), see e.g. [MWH*06, WWSR03, HBW06]. These examples show the potential of current procedural methods: They can be used to automate

terrain construction. However, the application of procedural methods currently suffers from two important drawbacks:

1. It is not clear how to tune individual procedural algorithms to work well together; there is no tool or integrating framework that combines these various algorithms in a usable way.
2. The parameters of these algorithms and tools (e.g. noise octaves, persistence) often require an in-depth knowledge of the algorithm to predict the effect of a parameter on the outcome. A user is virtually unable to declare his *intentions*; he typically has little control over the generation process and is forced to use a *trial and error* approach, as was noted in e.g. [SS05].

These drawbacks indicate that current procedural methods alone are not yet enough for automating terrain construction. An *ideal* terrain modeller is to be able to generate a large variety of realistic, natural terrains, controlled by a small number of *intuitive* user input parameters, while still allowing the user to perform detailed fine-tuning of the generated terrain, or (partial) regeneration. This shifts the paradigm from manual terrain *construction* into terrain *declaration*. In order to approach this ideal modeller, we identify requirements for an integrating terrain modelling framework and describe the workflow of such a modeller.

2. Modelling Workflow Requirements

This section identifies high-level requirements that follow from analysing how terrain and game level designers currently work. Figure 1 shows the typical terrain modelling process. Manual terrain modelling for games and simulations is usually a process of iterative refinement. A game level designer often starts out with a sketch of the terrain, which indicates the overall view and the location of important terrain features (see **sketch** arrow), although sometimes an idea or storyboard is the only starting point for the modelling process. Terrain features may be distinctive terrain elements (e.g. a mountain), but could also be related to game flow or training objectives. Following this, the designer builds a 3D terrain from *coarse*, e.g. a height map, to *refined*, with buildings, trees, shrubs, see **construct** and **construct and refine** arrows. This is done in an experimental manner (involving e.g. undo / redo). The final terrain model will be remotely based on the sketch, but typically much has been changed in the process. An accurate 3D view of the terrain is essential, as the designer visually evaluates both the suitability and the aesthetics of the terrain (constantly), and because the process involves a great deal of detailed fine-tuning.

Manual terrain modelling is a lengthy and laborious process; we want to speed up and automate this process without disturbing the creative working method of designers, and we want to make procedural methods accessible to users that have no intimate knowledge of the underlying algorithms. For this, we propose a framework that fits the current modelling process, see Section 3, but employs procedural methods for terrain generation to accelerate it. To overcome the drawbacks that hinder the adoption of procedural methods, stated in the introduction, a procedural modelling framework has to fulfil the following high-level requirements (and any more detailed requirement that can be derived) on the workflow, usability, and the control the user has over the terrain generation:

1. Terrain Sketching: The framework should require high-level user input in the form of a terrain sketch, or rough layout of the terrain. Here, the level designer should be able to declare whereabout important, sizeable terrain features are located.
2. Framework Usability: The user input should consist of intuitive, result-oriented parameters, which are mapped to the typical procedural algorithm parameters.
3. Terrain Generation: After the designer has declared the terrain he has in mind, the framework should generate a high-resolution terrain that follows the user-specified features at large, but has, on a small scale, a high level of detail and variations. It should adequately match the original specifications; by itself, it should not (randomly) introduce features that have a large impact on the terrain (e.g. a mountain appearing in the middle of a grassland).
4. Terrain Editing: The workflow should support further manual editing and fine-tuning, as well as regenerating (areas of) the terrain.
5. Terrain Visualisation: The framework should provide a clear 3D view of the terrain.
6. Framework Results: The terrain model should be exportable to a format usable in games and simulations.

3. Terrain Modelling Framework

Figure 2 gives an overview of the proposed modelling workflow, that supports the typical design process:

1. Layout of the terrain by the user, using a rough terrain map and global terrain parameters (**sketch** arrow);
2. Procedural generation of a layered, detailed terrain map ((**re-)generate** arrow). Further refinement of this map by the user (**modify** arrow);
3. Automatic export to, among other things, a 3D terrain model (**export** arrow).

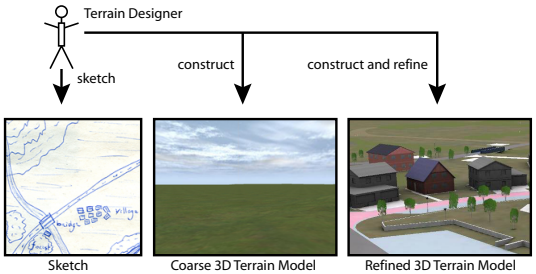

Figure 1: *An overview of the current modelling workflow.*

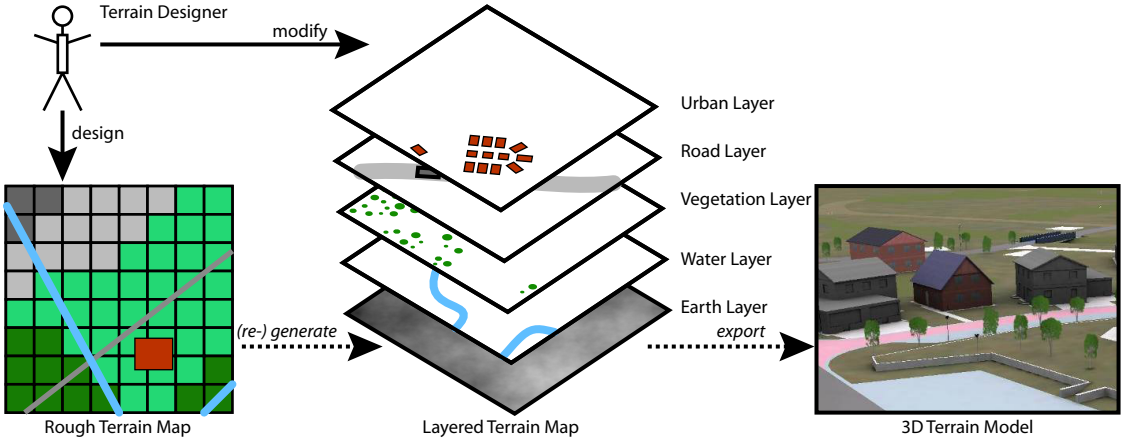

Figure 2: *An overview of the proposed terrain modelling workflow.*

To guide the procedural modelling process, the user supplies a sketch of the terrain he has in mind, in the form of a parameterized rough terrain map. Based on this rough terrain map a layered, detailed terrain map is automatically generated. This map contains, among other things, elevation information, and natural and man-made terrain features. The designer is able to manually edit the layered map. Where desired, areas of the map can be regenerated. Finally, the layered map is automatically exported to a 3D terrain model. The process is iterative: the user can go back and forth between the rough and detailed map, and manually refine the layered, detailed terrain. This fits the creative working method designers are used to.

Below, we discuss the workflow from sketch to a 3D terrain model in more detail. The designer of a new terrain (of e.g. 5 x 5 km) starts with a grid-based rough terrain map (e.g. 50 x 50 cells). A cell in this grid will later be amplified to a large square piece of terrain (e.g. 100 x 100 m). The idea of procedurally expanding a small grid map was proposed in [RP04]. The user assigns to each grid cell an *ecotope* (describing the type of terrain, e.g. some specific kind of desert, hills or forest), as used in e.g. [Hag06]. Additional properties of terrain cells can be defined by the designer (e.g. tree density). On top of this terrain grid, the designer places major terrain features with their own properties, e.g. a city with its population size.

After the designer has specified a rough layout of the terrain he has in mind, the framework expands it to a highly detailed terrain. The designer can edit this terrain manually in a way resembling common terrain editors. But he can also choose to regenerate certain areas of the terrain. Care is to be taken to preserve manual changes, when areas of the terrain are regenerated. The framework should also prevent the user from distorting other areas of the terrain, when editing some area. This requires a layer mechanism: as is the case in, for instance, image editing software like Photoshop, using independent layers improves the adaptability of the designed terrain, because changes to one layer do not distort other layers (e.g. moving or deleting a river does not leave a ground elevation artefact). As depicted in Figure 2, we suggest five layers for the terrain map: two man-made layers and three natural layers. They are stacked as follows:

Man-made terrain layers:

1. Urban Layer: cities, towns, farms;
2. Road Layer: highways, local roads, bridges.

Natural terrain layers:

3. Vegetation Layer: bushes, trees;
4. Water Layer: rivers, lakes, oceans;
5. Earth Layer: elevation and soil types.

Based on the rough layout of the terrain, each layer is to be generated using suitable and in most cases existing procedural methods, tailored to the layer. The framework should generate smooth transitions between terrain ecotopes; e.g. from a forest to a plain, the tree density will slowly decline. On each layer, the designer can perform manual adjustments using layer-specific tools (e.g. road layout or elevation modification tools). To be able to determine the suitability of the terrain, the user must have both a schematic 2D and 3D view of the terrain or individual layers.

Naturally, a layer will be influenced by other layers. There are numerous interactions between layers and each type of interaction must be detected and solved. For instance, vegetation must not appear at locations where buildings are located. When a river on the Water layer crosses a highway on the Road layer, a bridge can be introduced or the road can be rerouted (see Figure 3).

After the designer has tuned the terrain to his satisfaction and the framework has solved any layer interactions, the terrain should be exported to a usable format, with the five terrain layers merged into one terrain model. Elevation corrections (flattening, see e.g. [LB06]) of the Earth layer will have to be performed to accommodate for these terrain features.

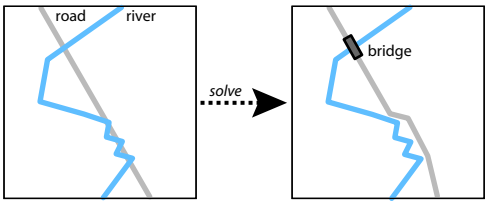

Figure 3: *Interaction between Water and Road layers.*

This 3D terrain model is created automatically and defined according to a well-known standard (e.g. OpenFlight, COLLADA). Export options are different levels of fidelity (low for a flight simulator vs. high for a infantry trainer) or a navigation map for a Computer Generated Forces package.

4. Conclusions and Future Work

We have presented high-level requirements and proposed a workflow for a procedural terrain modelling framework. It fits an iterative design process, and supports experimentation and incremental improvements. Based on high-level user input in the form of a rough layout of the terrain, the framework provides automated terrain generation using procedural methods, resulting in a detailed 3D terrain map. For the generation, we can combine many of the existing procedural algorithms in a new, coherent, and useful way. The detailed terrain map can be manually edited and fine-tuned, and exported into a format usable in games and simulations. The framework can also be used to augment geospecific GIS source data with geotypical detail in a 3D terrain model.

Our research is now focussed on the realisation of such a framework, which poses several important design challenges. A large number of complex interactions between the five layers will emerge, and solving an interaction between two layers may have side-effects for other layers. For some interactions, simple rules will suffice (e.g. remove trees that are in the way of man-made terrain elements); in some cases, constraint solving methods might be necessary to correctly place terrain features (e.g. maintaining road connections when regenerating a mountainous terrain).

Another challenge is to map the terrain (feature) parameters to the different procedural terrain generation algorithms. Most algorithms have several parameters, the values of which greatly influence the quality of the generated content. When the algorithms are combined, finding adequate parameters will be even more difficult.

Because of the increasing demands in virtual world modelling, we believe it is essential to move from the paradigm of terrain *construction* towards *declarative* terrain modelling. Although it is a challenging task, we expect that the realisation of a procedural terrain generation framework will contribute to this paradigm shift.

Acknowledgments This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

References

[DHL*98] DEUSSEN O., HANRAHAN P., LINTERMANN B., MĚCH R., PHARR M., PRUSINKIEWICZ P.: Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM Press, pp. 275–286.

[EWM*03] EBERT D. S., WORLEY S., MUSGRAVE F. K., PEACHEY D., PERLIN K.: *Texturing & Modeling, a procedural approach*, 3rd ed. Elsevier, 2003.

[Hag06] HÄGGSTRÖM H.: *Real-time generation and rendering of realistic landscapes*. Master's thesis, University of Helsinki, August 2006.

[HBW06] HAHN E., BOSE P., WHITEHEAD A.: Persistent realtime building interior generation. In *Sandbox '06: Proc. of the ACM SIGGRAPH Symposium on Videogames* (New York, NY, USA, 2006), ACM, pp. 179–186.

[LB06] LATHAM R., BURNS D.: Dynamic terrain modification using a correction algorithm. In *IMAGE* (July 2006).

[Mil86] MILLER G. S. P.: The definition and rendering of terrain maps. *SIGGRAPH Comput. Graph. 20*, 4 (1986), 39–48.

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., GOOL L. V.: Procedural modeling of buildings. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 614–623.

[Ols04] OLSEN J.: Realtime procedural terrain generation. Technical Report, University of Southern Denmark, October 2004.

[Per85] PERLIN K.: An image synthesizer. *SIGGRAPH Computer Graphics 19*, 3 (1985), 287–296.

[RP04] RODEN T., PARBERRY I.: From artistry to automation: A structured methodology for procedural content creation. In *Proceedings of the 3rd International Conference on Entertainment Computing* (Eindhoven, The Netherlands, September 2004), pp. 151–156.

[SS05] S. STACHNIAK W. S.: An algorithm for automated fractal terrain deformation. *Computer Graphics and Artificial Intelligence 1* (May 2005), 64–76.

[Wel05] WELLS W. D.: Generating enhanced natural environments and terrain for interactive combat simulations (genetics). In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2005), ACM Press, pp. 184–191.

[WWSR03] WONKA P., WIMMER M., SILLION F., RIBARSKY W.: Instant architecture. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM, pp. 669–677.