Interactive Creation of Virtual Worlds Using Procedural Sketching

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

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

Abstract

Procedural modelling is an attractive alternative to cut down the costs of manual content creation for virtual worlds. We discuss our declarative modelling approach to the creation of 3D virtual worlds, which integrates a variety of procedural techniques in order to enable a non-specialist user to interactively create a complete 3D virtual world in minutes. In particular, we introduce procedural sketching, a novel paradigm which allows designers to quickly specify and see the effects of their procedural modelling operations, and describe its main features as implemented in our prototype system SketchaWorld. Two main interaction modes are described, for specifying the landscape and terrain features, respectively. Our approach automatically fits all generated terrain features with their surroundings, for example by smoothing out rough terrain for roads, or creating a bridge to cross a river. It is concluded that this approach provides designers with the productivity gain of procedural methods, while still allowing for fine user control and actively supporting iterative modelling.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism I.3.4 [Computer Graphics]: Graphics Utilities—Paint systems I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

1. Introduction

The use of 3D virtual worlds is widespread: they are found in entertainment and training games, movies, visualizations, etc. Modelling systems for virtual worlds typically stem from entertainment game development. These tools offer designers total *control*. With skill and dedication, designers can create literally any world exactly the way they want to. This level of control also has a clear downside: a designer *has to* model every aspect of the virtual world in detail. As virtual worlds become more and more detailed, the workload to create content this way rapidly increases. Furthermore, these systems require extensive 3D modelling experience to be effectively used. However, the community of users of virtual worlds is diverse, as is their 3D modelling expertise. Potentially, there are many virtual world designers, for instance, gaming enthusiasts creating new game levels, and training instructors designing a tailored training curriculum. In practice, though, only expert 3D designers are currently able to create them.

There is a need for new modelling alternatives with both higher productivity and reduced complexity. Automatic procedural modelling is very promising in this regard. Research in this area has already resulted in numerous interesting methods, especially for virtual world modelling, as recently surveyed in [SdKT*09]. However, procedural modelling in itself is not enough: generated results are often too random to allow for effective modelling and the type of user input it requires is not intuitive for designers to work with. In recent years, the research direction is shifting to more interactive and user controllable procedures. Here we mention some noteworthy examples of terrain feature generation either by means of sketching or interactive editing:

- Schneider et al. [SBW06] introduce a setup in which the user interactively edits the terrain by painting in greyscale the base functions of their noise generator;
- Using an efficient GPU-based hydraulic erosion algorithm, Stava et al. [SBBK08] propose an interactive erosion-based terrain modelling method;

- Gain et al. [GMS09] introduce a sketch-based height-map generation method in which users sketch the silhouette and bounds of a mountain in a 3D interface, and the generator creates a matching mountain using noise propagation;
- de Carpentier and Bidarra [dCB09] introduce GPU-based procedural brushes that allow users to interactively sculpt a terrain in 3D using several types of noise;
- Chen et al. [CEW*08] propose interactive modelling of road networks by the use of tensor fields that can create common road patterns (grid, radial, along a boundary) and combine these in a plausible way;
- McCrae and Singh [MS09] present a method for converting line strokes to 3D roads that are automatically fit in with the terrain;
- To address the complexity of shape grammar creation for building generation, Lipp et al. [LWW08] propose a shape grammar editing system, in which the effects of new rules are interactively visualised.

The downside of these methods is that they are designed to generate one specific aspect of 3D virtual worlds, and the authors do not explore their suitability for other terrain features. No method to date provides an integrated framework that allows one to procedurally model a complete virtual world, ranging from mountains to man-made structures. We identified three requirements for such a framework:

1. intuitive controls and an accessible user interface;
2. automatic integration of all generated content into a complete and consistent end result;
3. an interactive workflow with smart editing facilities.

Starting from these requirements, we have proposed a new virtual world modelling approach [STdB08]. It aims at reducing the complexity of virtual world modelling and increasing designers' productivity, while still allowing them to work in an iterative manner and exercise control over the generation process. User input consists of a 2D digital sketch: a rough layout map of the virtual world, created using simple and clear editing tools. Each sketch element is procedurally expanded to a corresponding terrain feature. These generated terrain features are placed in logical layers of the terrain model and are automatically fit with their surroundings. Once satisfied with the generated results, designers can save the 3D virtual world model and export the world to other relevant formats, such as GIS data.

This paper focuses on the third requirement mentioned above: *interactivity*. We present interactive *procedural sketching* facilities for virtual worlds, and discuss many of its design choices and implementation considerations.

2. Interactive procedural sketching

An interactive workflow that effectively supports creativity is essential for the usability of a modelling system, be it for experienced game level designers or for non-expert users. This kind of workflow requires the system to execute user

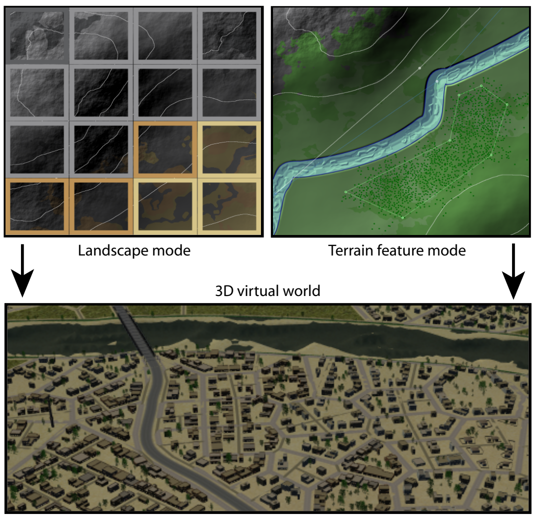

Figure 1: *Procedural sketching of virtual worlds.*

actions rapidly, to remain responsive even during complex edit operations, to support undo and redo, to have a clear user interface and offer a good visualisation of the results.

2.1. User interaction and visualisation

Procedural sketching provides two interaction modes:

Landscape mode Designers paint a top view of the landscape by colouring a grid with ecotopes (a small area of homogeneous terrain and features). These ecotopes encompass both elevation information (elevation ranges, terrain roughness) and soil material information (sand, grass, rock, etc.). The grid size is adjustable and the brushes used are very similar to typical brushes found in image editing software, including draw, fill, lasso, magic wand, etc.

Feature mode Designers place elements like rivers, roads, and cities on the landscape using vector lines and polygon tools. This resembles the basic tools found in vector drawing software: placing and modifying lines and polygons is done by manipulating control points.

To directly see the effect of sketch actions on the virtual world model, users sketch on top of a 2D view of the generated terrain layers (with familiar options like hiding layers, zooming, etc.). This view is updated immediately as new results are generated. Depending on the interaction mode, an overlay is displayed representing relevant elements of the user sketch. Figure 1 shows the grid overlay that assists with painting the landscape (left) and shows the line drawing overlay for displacing a river (right).

2.2. Asynchronous procedural generation operations

As designers typically model a virtual world in an iterative manner, we need to provide a short feedback loop between

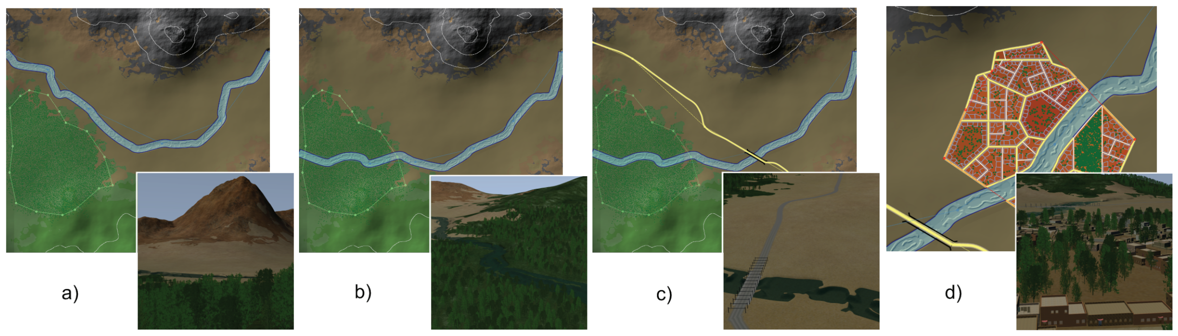

Figure 2: *A procedural sketching session and resulting 3D virtual world (insets): a) natural environment mountains, river and forest. b) the river is rerouted to run through the forest. c) a road is placed across the river. d) city created along the river banks.*

sketch action and the visualisation of generated results. To this end, each sketch action is executed separately and the results of this partial generation are displayed. This allows designers to quickly see the effect of their edit operations and work towards the desired end result.

However, sketch actions may often execute at non-interactive rates. Although improvements in hardware and new approaches such as GPU computing significantly alleviate this problem, operations affecting large regions or requiring complex algorithms (e.g. city generation) take longer than desirable and, therefore an asynchronous setup was implemented. Although edits to the sketch itself are made immediately and control is returned to the designer, the corresponding sketch action is placed in a process queue.

Essential to iterative design is the ability to undo and redo actions. For this, designers are presented with the well-known *history* found in many image editing software, with which they can step through a list of all their editing actions, undoing or redoing one or more actions. As many editing operations do not have a well-defined inverse operation for undoing their results, a common approach is to store a (partial) state for each editing action, and restoring this state when undoing the action. Obviously, this approach is demanding in memory use. For domains as bitmap editing, techniques as compression and disk paging are enough to ensure that such memory requirements are not prohibitive. However, for virtual world modelling, this would quickly lead to an explosion in memory use. Therefore we opted to implement undo and redo by partial generation and restoration steps, executed in a similar way as normal sketching actions. At the expense of additional computation time, it makes unlimited undo and redo of procedural operations feasible.

The edit history that is manipulated by the designer is a data structure maintained separate from the process queue, although sketch actions in the history can be linked to actions in the queue. This separation is necessary to guarantee robustness and prevent overwrites (which would leave the terrain model in an inconsistent state) even if time-consuming sketch actions are, for instance, quickly undone and redone. The edit history shows the status of sketch actions, determined by whether a linked action found in the process queue is executing, waiting for execution, or done.

Special consideration is required for random number sequences, as they are used by most procedures in making their decisions. For instance, a procedure that locally routes rivers or roads chooses a direction randomly proportional to its suitability (based on elevation constraints, obstacles, distance to target location, etc.). To ensure exactly the same results redoing an undone action, the state of the random number generator (i.e. its starting seed and position in the random sequence) must be saved and restored for each sketch action.

2.3. Terrain feature integration and consistency

Terrain features like roads and cities seldom exist in isolation, instead they blend in with their surroundings. During the generation of a terrain feature, the local landscape and nearby features are taken into account, e.g. to choose a suitable path for a road or river. After this, integration steps adapt local terrain features to the newly created feature, for instance by removing trees on a generated road's path. Each time a terrain feature is modified, changes to related features are all performed automatically as logical side-effects of the change. To allow the designer to quickly see the local side-effects of sketch actions, these integration steps are performed and the results are displayed immediately.

For each sketch action, we have identified the local region and type of terrain features it affects or is affected by and in what way. When a terrain feature is generated, the identified adjustments are performed, modifying or (partially) regenerating the affected terrain elements. Although this somewhat increases the execution time of sketch actions, it keeps the virtual world model in a consistent and usable state.

In traditional modelling systems, large scale changes to a virtual world typically involve so much manual effort and editing steps, that a designer will mostly want to avoid it.

With our approach, designers are free to experiment with different alternatives, as the consistency and integration steps are taken care of automatically.

3. Results

Interactive procedural sketching was implemented in SketchaWorld, our prototype virtual world modelling system, using C#, C++ and CUDA, and OpenSceneGraph for 3D visualisation. In Figure 2 we illustrate its functionality with an example session involving several consistency maintenance steps. Figure 2.a shows a natural environment with mountains, a forest and a river. After sketching river's path, a suitable course is plotted through the landscape and the river bed and banks are carved into the terrain. In Figure 2.b the designer has displaced this river to let it run through the forest. Automatically, the river is locally regenerated and embedded, removing any tree on the river's bed and banks. In Figure 2.c a major road is introduced. To integrate this road, a bridge is inserted at the crossing with the river, and the road is integrated into the terrain to form a road embankment. Finally, when the designer outlines a small city, its districts and secondary roads form around the river, as shown in Figure 2.d. If the designer were now, for instance, to move this river, it would affect the underlying terrain, the crossing road and the city structure. For a better impression of SketchaWorld and its interactive workflow, we refer to the accompanying online video with real-time footage.

4. Conclusions

We presented a novel declarative modelling approach to the creation of 3D virtual worlds that integrates a variety of procedural techniques. We described several interactivity features of SketchaWorld, a prototype system implementing this approach and, in particular, we introduced *procedural sketching* as a powerful paradigm that significantly increases the usability of procedural modelling techniques. Two main interaction modes have been described that provide a fast and intuitive way for both experts and non-specialist designers to declaratively create virtual worlds: *landscape mode*, in which a top view of a landscape is painted on a grid of ecotopes (including e.g. elevation ranges, terrain roughness, and soil material information); and *feature mode*, in which elements like rivers, roads, and cities are placed on the above landscape (e.g. by manipulating control points on lines and polygons). The short feedback loop between each sketch action and its generated results makes it easier and more intuitive to create whole 3D virtual worlds, thus increasing productivity and reducing content production costs.

One of our main challenges ahead is to further enhance the level of control provided to designers, who will often wish to manually edit and fine-tune entities on a more detailed level than terrain features, to more precisely fit their requirements. We are currently investigating possible approaches and constraints to the integration of such manual editing facilities with procedural sketching in our interactive workflow.

The procedural sketching facilities of SketchaWorld described here provide designers with the productivity gain of procedural methods, while still allowing for fine user control and actively supporting iterative modelling. Therefore, they represent a small but clear step towards making procedural techniques suitable for a variety of applications, such as simulations and serious games.

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

[CEW*08] Chen G., Esch G., Wonka P., Müller P., Zhang E.: Interactive Procedural Street Modeling. In *SIGGRAPH '08: Proceedings of the 35th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2008), vol. 27, ACM, pp. 1–10. 2

[dCB09] de Carpentier G., Bidarra R.: Interactive GPU-based Procedural Heightfield Brushes. In *Proceedings of the 4th International Conference on the Foundations of Digital Games* (Florida, USA, April 2009). 2

[GMS09] Gain J., Marais P., Strasser W.: Terrain Sketching. In *I3D '09: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2009), ACM, pp. 31–38. 2

[LWW08] Lipp M., Wonka P., Wimmer M.: Interactive Visual Editing of Grammars for Procedural Architecture. In *SIGGRAPH '08: Proceedings of the 35th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2008), ACM, pp. 1–10. 2

[MS09] McCrae J., Singh K.: Sketch-based Path Design. In *GI '09: Proceedings of Graphics Interface 2009* (Toronto, Ontario, Canada, 2009), Canadian Information Processing Society, pp. 95–102. 2

[SBBK08] Stava O., Beneš B., Brisbin M., Křivánek J.: Interactive Terrain Modeling Using Hydraulic Erosion. In *Eurographics / SIGGRAPH Symposium on Computer Animation* (Dublin, Ireland, 2008), Gross M., James D., (Eds.), Eurographics Association, pp. 201–210. 1

[SBW06] Schneider J., Boldte T., Westermann R.: Real-Time Editing, Synthesis, and Rendering of Infinite Landscapes on GPUs. In *Vision, Modeling and Visualization 2006* (November 2006). 1

[SdKT*09] Smelik R. M., de Kraker K. J., Tutenel T., Bidarra R., Groenewegen S. A.: A Survey of Procedural Methods for Terrain Modelling. In *Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)* (Amsterdam, The Netherlands, June 2009). 1

[STdB08] Smelik R. M., Tutenel T., de Kraker K. J., Bidarra R.: A Proposal for a Procedural Terrain Modelling Framework. In *Poster Proceedings of the 14th Eurographics Symposium on Virtual Environments EGVE08* (Eindhoven, The Netherlands, May 2008), pp. 39–42. 2