Integrating procedural generation and manual editing of virtual worlds

Ruben Smelik
TNO Defence, Security and Safety
The Hague, The Netherlands

Tim Tutenel
Delft University of Technology
Delft, The Netherlands

Klaas Jan de Kraker
TNO Defence, Security and Safety
The Hague, The Netherlands

Rafael Bidarra
Delft University of Technology
Delft, The Netherlands

ABSTRACT

Because of the increasing detail and size of virtual worlds, designers are more and more urged to consider employing procedural methods to alleviate part of their modeling work. However, such methods are often unintuitive to use, difficult to integrate, and provide little user control, making their application far from straightforward.

In our declarative modeling approach, designers are provided with a more productive and simplified virtual world modeling workflow that matches better with their iterative way of working. Using interactive procedural sketching, they can quickly layout a virtual world, while having proper user control at the level of large terrain features. However, in practice, designers require a finer level of control. Integrating procedural techniques with manual editing in an iterative modeling workflow is an important topic that has remained relatively unaddressed until now.

This paper identifies challenges of this integration and discusses approaches to combine these methods in such a way that designers can freely mix them, while the virtual world model is kept consistent during all modifications. We conclude that overcoming the challenges mentioned, for example in a declarative modeling context, is instrumental to achieve the much desired adoption of procedural modeling in mainstream virtual world modeling.

Categories and Subject Descriptors

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*; I.6.7 [**Simulation and Modelling**]: Types of Simulation—*Gaming*

PCGames 2010, June 18, Monterey, CA, USA

Keywords

virtual worlds; declarative modeling; procedural methods; manual modelling

1. INTRODUCTION

For three decades now, researchers have been introducing new procedural methods for generating a variety of types of content, ranging from textures to complete virtual cities. All these methods show potential and generate interesting results. In particular for the topic of virtual world modeling, specific procedures have been proposed for many of its aspects and features. However, so far these results together have not resulted in a shift from manual to (semi-) automated virtual world modeling. There are three causes as to why this transition has yet to take place: traditional procedural content generation methods are often complex and unintuitive to use, hard to control, and results generated by different procedural methods are not easily integrated into a complete and consistent virtual world.

In our research, we are addressing these three issues with an integrated declarative modeling approach, which allows designers to concentrate on stating *what* they want to create instead of on describing *how* they should model it. Our prototype modeling system implements this approach, providing a fast and more intuitive way to model virtual worlds, designated *procedural sketching*. Designers interactively sketch their virtual world using high-level terrain features that are procedurally expanded using a variety of integrated procedural methods. Within this integrated framework, we fit together the generated content to form a consistent virtual world, using a consistency maintenance mechanism that automatically manages relationships and conflicts between terrain features.

With interactive procedural methods, such as procedural sketching, designers can quickly obtain a virtual world that matches their requirements on the level of large terrain features and their relations and connections. However, on the more detailed level of individual objects, designers will often want to manually edit and tune the generated results to fit more precisely to their requirements. Such manual edit facilities should work well together with procedural generation methods. This combination, integrated in an interactive

workflow, promises to provide designers with the productivity gain of procedural methods, while still allowing for a fine level of user control and flexibility.

We have identified various categories of manual edit operations and their relation to procedural methods. We also discuss the challenges of bringing these two extremes together, a topic which has, until now, been unexplored. We propose and compare several approaches to address these issues.

Section 2 describes our declarative modeling approach, with the results of procedural sketching and virtual world consistency maintenance. Section 3 focuses on manual editing facilities: the levels of granularity in modeling and the categories of editing facilities. We present a variety of possible manual operations. The challenges we foresee in integrating these operations with procedural generation methods are discussed in Section 4. We propose several approaches and discuss possible solutions for these challenges in Section 5.

1.1 Related work

Procedural modeling has been an active research topic for over thirty years, and has resulted in high-quality results for specific terrain features, such as landscapes [13], vegetation [6], roads [9], urban environments [14] and building facades [12, 7]. However, their industrial application is still limited. As concluded in our recent survey on procedural methods [17], traditional procedural methods provide designers with too little control over the outcome, and typically lack the interactive workflow of manual modeling systems.

Earlier work in providing designers with more control resulted in several non-interactive approaches (see, e.g., the work by Stachniak and Stuerzlinger [21] or Zhou et al. [24]), in which the elevation map generation is constrained by some form of user input, e.g. a line drawing indicating a mountain ridge. With the evolution of the GPU as a device for general purpose parallel processing, interactive user control in elevation map generation has become feasible. Schneider et al. [16] introduce a setup in which the user interactively edits the terrain by painting grayscale images, which are used as the base functions of their noise generator. Using an efficient GPU-based hydraulic erosion algorithm, Stava et al. [22] propose an interactive way for users to modify terrain using several types of hydraulic erosion. To provide users with more control over the exact appearance of mountain ranges, Gain et al. [8] 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. Even more fine-grained control over the shape of mountains is provided by the interactive procedural brushing system introduced by de Carpentier and Bidarra [4]. These GPU-based procedural brushes allow users to interactively sculpt a terrain in 3D using several types of noise.

Extensions of traditional procedural methods are also proposed for generating other features, for instance interactively defining road networks that are used for procedural city generation. Chen et al. [3] propose interactive modeling of road networks by using tensor fields that can create common road patterns (grid, radial, along a boundary) and combine these in a plausible way. McCrae and Singh [11] present a method for converting line strokes to 3D roads that are automatically fit in the terrain. Their system also creates junctions and viaducts for crossing roads. As city districts and blocks are typically defined by the road network, Kelley and McCabe [10] propose an interactive method to generate secondary roads and house blocks based on the primary roads the user manipulates. A similar system by de Villiers and Naicker [5] lets users create a road network and city blocks using sketch strokes, and interprets a set of sketch gestures that modify the properties of the city blocks (e.g. population size, function). Weber et al. [23] present an interactive simulation system for cities growing over time, by expanding streets in the city's road network. An interactive, GPU-based, editing system for a set of Geographic Information Systems (GIS) vector features is presented by Bruneton et al. in [2].

Commercial virtual world modeling tools typically fall either in the manual modeling or in the procedural generation category. However, a noteworthy exception is PixelActive's CityScape [15], which allows for hybrid modeling of urban environments, although it provides a somewhat limited and narrowly focused set of procedural operations.

The research results cited have been important in advancing the field of procedural methods towards interactive and more designer-controlled modeling. However, the presented methods are very much designed for a specific type of terrain feature, and no trivial way is provided to combine their results into a complete and consistent virtual world. Furthermore, the problem of interactively combining procedural generation with manual editing remains unexplored to date.

2. DECLARATIVE MODELING

We believe that it is crucial to provide both procedural and manual operations in an integrated manner, to allow designers to quickly create the rough layout of a virtual world and to allow manual refinements to further improve this virtual world. We have developed an integrated modeling approach, which aims at combining the strengths of manual and procedural modeling, and provides a more productive and intuitive workflow to model virtual worlds. This approach lets designers concentrate on what they want to create instead of on how they should model it, hence it is designated *declarative modeling of virtual worlds* [1]. In declarative modeling, designers simply state their intent using intuitive interaction mechanisms. For instance, they might want a small village on the banks of a river that runs through a valley encompassed by high mountain ranges. Using a combination of specialized procedural methods, this intent can then be automatically translated to a matching 3D virtual world.

We are developing a modeling framework, called SketchaWorld, which demonstrates the feasibility of this declarative approach. Its goals are:

1. to increase designers' productivity, while still allowing them to work in an iterative manner and exercise control over the generation process;
2. to provide an intuitive way for people without special modeling expertise to create virtual worlds that meet their requirements (as was motivated in [18]);
3. to facilitate the application of results from research in procedural methods in an integrated modeling framework.

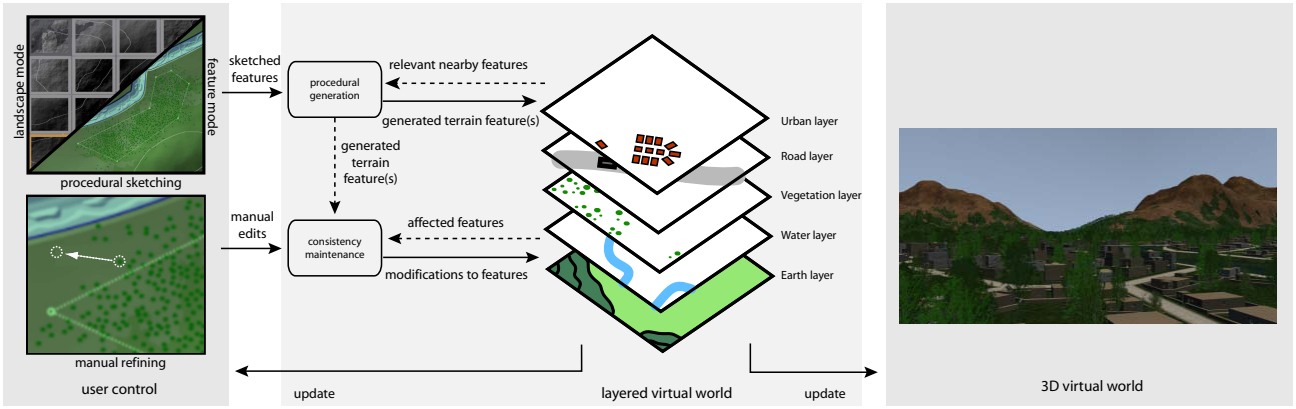

Figure 1: An overview of the workflow of the declarative modeling framework SketchaWorld.

2.1 Procedural sketching

Procedural sketching provides an intuitive and fast way of specifying complete virtual worlds (Figure 1 left hand). Using simple and clear editing tools, designers create a 2D digital sketch: a rough layout of the virtual world. Procedural sketching provides two interaction modes:

Landscape mode Designers paint a top view of the landscape by coloring a grid with ecotopes (an 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 and transition pattern brushes (e.g. from ocean to shore).

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.

As shown in Figure 1, each sketched element is automatically expanded to a corresponding realistic terrain feature using a customized procedure, and taking into account any relevant surrounding features. As the design of a virtual world is very much a creative process, the framework provides an iterative workflow, with support for unlimited undo and redo, and a short feedback loop between edit action and result. Its implementation details are explained in [19].

2.2 Consistency maintenance of the virtual world model

All generated terrain features are grouped in five logical layers, inspired from Geographic Information Systems (GIS) (see Figure 1 middle). Using different layers improves the adaptability of the virtual world model, because changes to one layer do not necessarily have to affect other layers.

Terrain features need to blend in with their surroundings to form a lifelike virtual world. If these features were to be generated separately from their context in the virtual world, designers would have to perform their integration manually, which would harm their productivity and limit their ability to experiment. Because the layered virtual world model is semantically rich, it can be automatically kept in a valid state using a form of consistency maintenance.

The introduction of a new terrain feature into the layered virtual world model discerns two phases, one phase in which the feature is generated and another phase in which the feature is fit in the virtual world (shown in Figure 1). In the first phase, the feature's path, shape or other properties are determined based on the provided user parameters and on relevant nearby existing terrain features. In the second phase, some of the surrounding features are affected by the newly introduced feature and are therefore connected to it, modified in some way or even removed. The consistency maintenance in these two phases is based on *rules* describing the mutual influence of terrain features. They include *priorities* to determine, for instance, which type of features has precedence when features overlap.

Although these maintenance phases somewhat increase the execution time of individual operations, this continuously keeps the virtual world model in a consistent and usable state and it allows the designer to quickly see possible local side-effects of edit actions. Each time a terrain feature is modified, changes to related features are performed automatically as logical side-effects of this change. In traditional modeling systems, any large scale change to a virtual world typically involves so much manual effort and editing steps, that a designer will do anything to avoid it. With our approach, designers are free to experiment with different alternatives, as the consistency and the integration is automatically taken care of. Details of the implementation of consistency maintenance are presented in [20].

2.3 Results

As an impression of how one can use procedural sketching to design a virtual world, we present the intermediate results of a short example sketch session (see Figure 2), in which a designer creates, in a couple of minutes, a natural landscape with a river flowing through a valley and a city on a hill along this river. Figure 2.a shows the basic landscape, sketched in *landscape mode* brushing the ecotope grid: a green valley encompassed by mountains, with some forests defined in the valley and a river flowing through it (see also Figure 2.d). On top of this natural environment, in *feature mode*, some

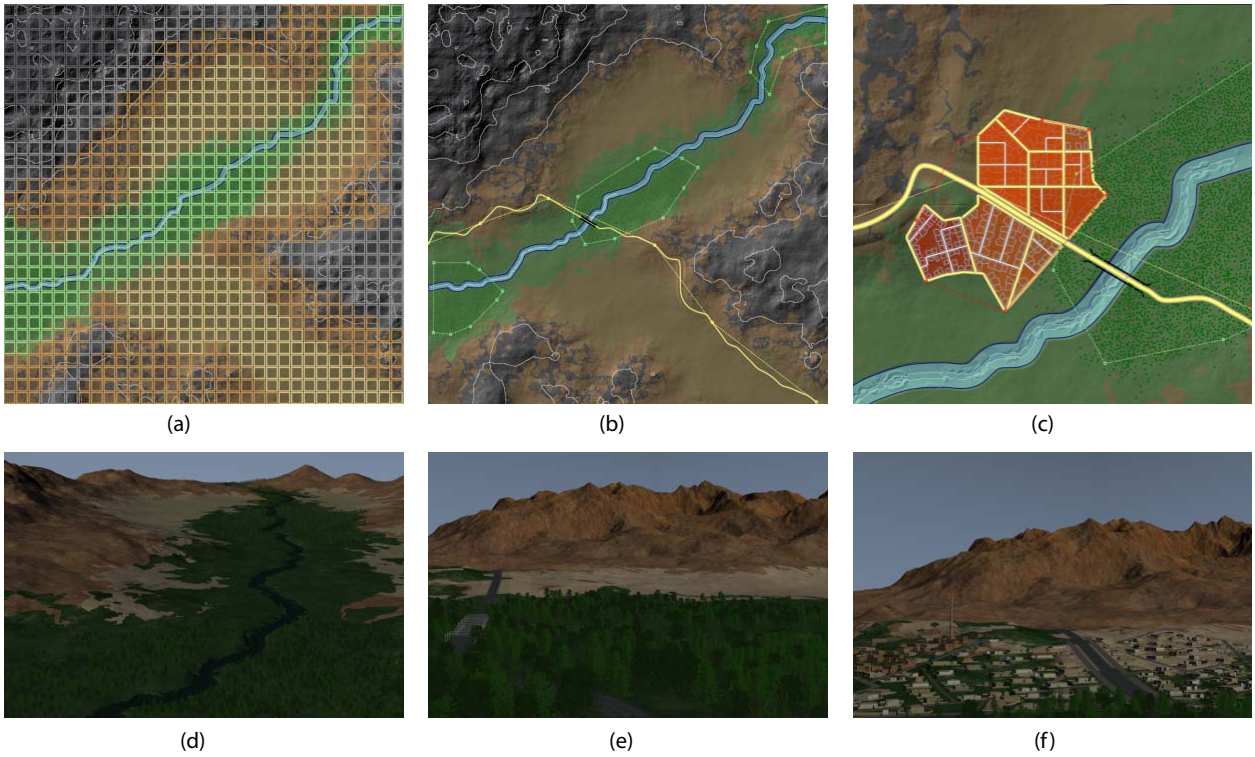

Figure 2: Results of an example procedural sketching session: a) sketch of a natural environment b) road sketched through the valley from east to south, crossing the river c) city outlined on a hill d) resulting natural landscape e) river crossing with bridge f) resulting city on the hills.

man-made features are added. Firstly, the designer coarsely outlines the desired path of a major road Figure 2.b. This road crosses the river at one point, Figure 2.e shows the bridge that was automatically created. Lastly, the designers outlines a small village along this road (Figure 2.c), at the hillside (Figure 2.e).

3. MANUAL EDITING OPERATIONS

To examine how one could integrate procedural methods with manual edit operations, we start by identifying what kind of manual operations one could desire and what are the characteristics of these different types of operations. For this, we discern four levels of granularity in modeling operations for creating virtual worlds:

Coarse level At this level, designers are concerned with the rough layout of the virtual world and specify large scale terrain features such as mountain ranges, rivers and cities. For modeling at this level, procedural techniques are most helpful, as they provide a quick way to fill the world with terrain features.

Medium level This level concerns relatively large refinements to features specified at a coarse level, for instance creating a park within a city feature or changing one of its districts from a residential to a commercial type of district. Although editing entails refining coarse features, these changes typically involve a substantial amount of work, and are therefore very suitable for procedural generation.

Fine level The fine level deals with modifying individual objects (e.g. a single building or tree). These objects typically result from a procedurally generated terrain feature, but a designer could manually place new objects as well. Edit actions on this level involve little procedural generation.

Micro level On the lowest level, designers manipulate geometric meshes, assign textures to 3D models, etc.

Typically, procedural systems only support the coarse level, and sometimes generate only one specific terrain feature. Manual modeling systems often operate mainly on the micro level. Both extremes have clear limitations and they cannot be easily integrated because of the lack of editing facilities that operate on the medium or fine level: generating a coarse layout with a procedural system and then refining this model using a manual system, operating on the micro level, will typically involve much modeling effort.

Therefore, there is a need for a modeling approach that properly integrates procedural generation with medium- and fine-level manual edit operations. The kind of manual operations we consider can be placed in several categories:

1. changing the location of a generated object;
2. locally transforming a generated feature's shape or changing its internal structure;
3. modifying parameters for a specific region in a feature;
4. introducing new objects and features as part of generated features;
5. applying user-defined constraints on a feature;

6. all other kinds of operations.

Table 1 shows a list of examples of manual operations one could consider to provide. As the micro level is already adequately supported by numerous traditional modeling packages, we have considered those kind of edit actions out of our scope. Note the diversity of the operations in scale and complexity. As follows from the table, manual editing in urban environments encompasses numerous different operations and a city feature will, as a result be the most complex feature to edit. As said, many of these example operations are not available in current manual virtual world modeling systems; designers have to achieve these by manually performing a sequence of micro level operations.

Description	Cat.	Level
Soil and elevation		
Brush elevation or soil material data	2	Med
Change terrain roughness in some area	2	Med
Constrain elevation profile	5	Med
Rivers		
Modify the course of a river locally	2	Med
Locally adjust a river's lateral profile	2	Med
Create a natural pass to cross the river	2	Fine
Vegetation		
Add or remove individual trees in a forest	4	Fine
Change species or age of a single tree	3	Fine
Move a tree or change its orientation	1	Fine
Major roads		
Change lateral profile of a section of a road	2	Med
Change a road junction type	2	Med
Add some road equipment (e.g. streetlights)	4	Fine
Plant vegetation along road by a pattern	4	Fine
Add, or remove a bridge	4	Fine
Change type, model or properties of a bridge	3	Fine
Street network		
Remove a small street	2	Fine
Add a street, dividing an existing city block	2	Med
Change street network connections (dead-ends, etc.)	2	Med
Urban environment		
Change a district type	3	Med
Add a specific district or area (e.g. park)	4	Med
Change properties of a building lot	3	Fine
Move a building or urban object (bench, lightpole)	1	Fine
Change a building footprint	2	Fine
Change a building type, style	3	Fine
Change a building layout (rooms, doors, windows)	3	Fine
Insert an existing 3D urban model (church, airport)	4	Fine
Add small urban clutter (garbage bin, bus stop)	4	Fine
Constrain part of a city (trafficability, line of sight)	5	Med
Attach a (semantic) annotation to some feature	6	Fine
Apply a procedural filter to change the visual appearance of a specific area	6	Med

Table 1: Examples of manual operations, sorted by aspect of the virtual world.

In our approach, we implemented the coarse level of modeling with procedural sketching, see Section 2.1. Using only the sketching facilities defined for this level, our framework allows designers to model a complete virtual world in minutes. Typically, these generated results will match the designer's intent to a large extent. The designer can subsequently, if desired, modify the sketched terrain features, by changing their shape, parameters or location and interactively examine the updated results. However, designers will also often want to make smaller scale changes to the generated content. By integrating manual fine-tuning operations as those in Table 1 with both procedural sketching and virtual world consistency maintenance, designers can perform manual editing in a more productive manner, continuously keeping the virtual world model in a consistent state.

4. OPEN ISSUES

The integration of manual editing facilities within any procedural modeling environment poses a number of important challenges for which no standard solutions exist. Here, we identify and characterize three main open issues one has to address for this goal:

1. how to preserve manual edit actions on a terrain feature throughout procedural re-generation?
2. how to balance user control versus automatic model consistency maintenance?
3. how to integrate both procedural and manual operations in the same iterative workflow?

4.1 Preserving manual changes

The first open issue is how to preserve manual edit operations a designer has performed on a terrain feature that is now procedurally regenerated. The motivation preserving these actions is twofold: they state, in more detail, the designer's intent for that specific feature and they can equate to a large amount of designer modeling effort. Losing these modifications every time a feature is regenerated could lead to a perceived lack of user control and frustration. It is therefore important to devise a method of preserving manual changes on generated results, where possible.

In order to be able to preserve manual changes, a modeling system should fulfill the following two requirements:

1. it should be able to associate a sequence of manual edit operations to the specific terrain feature on which these operations applied, so that once this feature is regenerated, all manual changes can be reapplied in order;
2. within such a terrain feature, for each manual edit operation, it should be possible to localize the right location or (sub-)element where the manual operation should be reapplied.

This second requirement is especially challenging for a modeling approach such as ours, where the virtual world model is made consistent after every edit action. Because of these automatic consistency mechanisms, changes to a terrain feature might result in another associated feature to be regenerated. For instance, consider a city generated on the basis of a sketched city outline and suppose that city has had several manual modifications, e.g. some of its streets are rerouted and a park was placed in one of the suburban districts. If a designer now sketches a river running through this city, the city structure drastically changes, typically causing the city feature to be regenerated. Still, as the sketch and the high-level parameters of the city remain the same, the new city will have local resemblance to its previous version, especially further away from the river. In that case, for example, an edit action that introduced a park in one of its districts still might make sense for the new city. The same holds for the modifications to individual streets. If the new city now contains streets that (visually) match with the streets in its previous state, from the designer's perspective, it could be logical to apply the modifications to these matching streets.

Therefore, there seems to be no straightforward manner in which to decide where precisely a preserved manual change has to be reapplied, or whether it should be reapplied at all. Finding a good solution requires lots of experimentation, tuning and evaluation.

4.2 Balance control versus consistency

The second main issue is to balance user control versus model consistency. This issue is most apparent at the fine level of modeling, as this level comes closest to the typical operations found in current manual modeling systems. Without any consistency maintenance, designers have total control, but as a result are themselves responsible for keeping the virtual world consistent. Designers can then accidentally create all kinds of invalid situations (streets through buildings, trees in river beds, etc.), and they would have to clean up these inconsistencies afterwards. All in all, providing no consistency maintenance for manual modeling operations, which is the most common case in current modeling systems, considerably affects designers' productivity. If the mechanism is too strict however, a designer might, for instance, move a generated tree upon a rocky hill, only to see it automatically removed, because the vegetation consistency procedure finds that spot unsuitable for that particular tree species. Designers thus would give away a large amount of control, and could become frustrated with the modeling system.

The challenge is to come up with a solution that, in most manual editing cases, provides the productivity of an automated mechanism and comes up with suitable solutions to feature interactions, but, in specific cases, allows for more extensive control or alternative solutions. This will require, among other things, facilities for restricting the influence of the consistency maintenance. Finding a good solution that provides this balance, while keeping the current simplicity in modeling intact requires lots of evaluation with designers in the loop.

4.3 Iterative modeling workflow

The last major hurdle we identify here is the integration of both the procedural and the manual operations in the iterative modeling workflow. One of the main components of such a workflow is the user edit history, which is used to undo and redo actions. In [19], we explained that, when dealing with procedural operations, storing all (partial) model states in an edit history is impractical due to excessive memory demands. We therefore opted to implement undo and redo support using partial regeneration of the model. For manual editing, it seems more feasible to store local states, as the scope of such an operation is typically much smaller than a procedural operation. However, for medium level operations, or operations involving a great deal of consistency maintenance, this might not hold. The alternative for restoring a feature to its previous state, when undoing a manual edit action, is then to regenerate that feature and then reapply all active manual edit actions to that feature, which is probably not always desired or meaningful, and might in some cases lead to a very slow undo-mechanism. Thus a good mix of these two extremes has to be found, which has a reasonable memory usage and model update performance.

5. DISCUSSION

In this section we discuss possible approaches for supporting manual editing within a procedural system, and we propose some initial solutions for addressing the open issues presented in the previous section. In general, we see three approaches one could take to provide more fine-grained user control in a procedural modeling system:

1. a stepwise approach in which procedural methods and manual editing facilities are offered in two separate modeling phases;
2. a fully declarative approach were designers can more precisely express their intent;
3. a mixed-mode approach where procedural and manual operations are seamlessly combined in an iterative workflow.

5.1 Two-phased approach

A simple approach that one could take in order to avoid some of the previously discussed problems is to split the modeling process in two phases: a *procedural phase*, in which a designer quickly generates a draft version of the virtual world, and a *manual phase*, in which the designer manually tweaks the results until they are satisfactory. In this setup, there is no way to go back from the manual to the procedural phase (they could even be implemented as separate systems), thereby avoiding most problems related to issue 1 and 3 in the previous section. In addition, one could choose to provide total user control in the manual editing phase. If, however, one would whish to provide some form of automated consistency maintenance, a balance still has to be found for that (issue 2 in previous section). The advantage of this approach is that one can keep the user interaction and implementation of the approach at a reasonable level of simplicity. The obvious disadvantage is that this approach avoids the serious problems by disrupting the iterative workflow, which can be quite restrictive and cumbersome in use. In some sense, this approach is very similar to the current practice, whenever some procedural tool is deployed for generating part of the virtual world, thus it is not a satisfying approach.

5.2 Declarative input approach

A more comprehensive approach would be to focus on getting the user input to be more expressive: if designers are better supported to state their intent, and if the procedural methods are able to generate content that matches this intent, there will be less need for manual refinements afterwards.

A way of providing users with more control over procedures, which has often been tried in the past, is to present long lists of low-level input parameters to configure these procedures. As procedure parameters are often quite unintuitive and one needs to have a detailed understanding of the working of the algorithms to be able to set the proper input values. Therefore, this is not a real solution.

A promising alternative direction is to allow designers to set high-level constraints on terrain features. These can be used in the generation process, yielding results that better match designer's intent. Designers are then beforehand able to force certain properties to hold or be present in the resulting virtual world model, instead of having to manually model these afterwards. Some meaningful examples of such constraints might be:

- force a small pass through a mountain range;
- declare a hill to have good visibility over a nearby village or force a specific line of sight;
- declare an area of a city to be dense and narrow (small streets, buildings built close together, lots of urban clutter or more widely constructed with many open spaces);
- force a specific route through a city to be accessible for, e.g., large vehicles;

- declare a specific type of land use or income range within an area of a city feature;
- force a specific grade of tactical cover to be provided in a certain area;
- declare a certain preferred curvature for a road, or a preference for tunnels over bridges.

The advantage of this approach is that it is quite intuitive to work with, and it is able to capture the (high-level) intent of the designer, and these constraints seem to fit well with procedural content generation. The disadvantage of this approach is that it does not provide full, fine-grained user control. We believe that, if designers would have the right constraints at their disposal, this approach could be a very powerful and productive way of working.

5.3 Integrated modeling approach

The most complex and powerful approach would be to come up with a solution that provides the combination of procedural and manual modeling in an iterative modeling workflow. This approach needs to address all the issues presented in the previous section. Here we briefly discuss some of the ideas and mechanisms for addressing these issues.

One of the measures one can take to preserve manual changes to terrain features is to link the relevant subset of the history of manual edits to specific terrain features. Once some action causes the feature to be regenerated, all the associated manual edit actions can be re-applied in order. Depending on the extent of the modifications, a regenerated city might still have the same structure in some relatively unaffected parts. However, minor changes might cause streets, for instance, to be slightly different, while they visually appear to have remained the same. Therefore, in order to avoid losing all manual changes to these features, a feature matching mechanism could be used to find the best matching feature for a manual edit, given some approximation.

Finding a balance between user control and consistency maintenance is difficult, especially for modeling urban environments. If one chooses to provide full consistency maintenance for manual editing, designers will need facilities to influence or limit this process in specific cases. Here we present possible facilities, which are inspired from image processing software, but have more advanced and complex semantics:

Grouping By default, many generated features will be grouped or linked together in some way, for instance, a set of buildings is grouped with a specific road. This grouping entails geometric distance and orientation constraints, so that when a designer moves a road, the associated buildings move along with it. By allowing designers to modify the grouping structure and to introduce new links, they have a more detailed influence on the effects of manual operations.

Locking By locking a certain area or feature, the designer can ensure that it will not be affected by any modeling operations he or she is currently performing. To avoid unrealistic sharp changes in the environment, a fall off area could be established around a locked area, resulting in a more lifelike transition.

Scoping This facility allows designers to examine and modify the scope (i.e. area of influence) for an operation. By restricting this scope, they are able to avoid that an operation has a (negative) effect on a nearby area, if the consistency maintenance rules are too strict or partly unsuitable for that specific modeling instance.

The advantage of this approach is that it combines the fine-grained level of user control that a designer typically desires with the productivity of procedural content generation. The disadvantage of this approach is the complexity in design and implementation, which may also have some effects on the interactivity and performance. Furthermore, it increases the modeling complexity for designers that use these manual editing facilities. Despite these challenges, we believe these manual editing facilities are a requirement to get procedural modeling accepted in mainstream virtual world modeling. We should therefore strive to overcome these challenges, to allow designers to adapt at any time the level of control they desire.

6. CONCLUSIONS

One of the main obstacles for getting procedural techniques into mainstream virtual world modeling is that they offer designers either inadequate or little control to specify their requirements. As a result, they mostly rely on conventional modeling systems, which require enormous manual efforts, but at least provide proven editing facilities to experiment with. Integration of both procedural generation and manual editing operations seems, therefore, a very promising and powerful target, combining the best of two worlds, but so far the topic is as good as unaddressed.

In this paper we discussed several challenges and open issues associated with this integration. According to their level of granularity, we identified a variety of modeling operations that are neither fully procedural nor conventional low-level manual editing. We believe that supporting this kind of operations is essential to bridge the gap between procedural and manual modeling. Particularly difficult are the challenges related to preserving the results of manual operations throughout procedural re-generation, as well as the tension of user control versus consistency maintenance.

Three approaches aimed at solving the above challenges were pointed out and briefly elaborated, but they will clearly require considerable experimentation and evaluation. Our way ahead includes exploring the two most promising approaches: constraint-based declarative input and integrated modeling approach. This research nicely complements our previous results on interactive procedural sketching [19], within the scope of our declarative modeling framework, SketchaWorld, which provides a very powerful and intuitive alternative for the creation of virtual worlds. We believe that by seamlessly integrating procedural modeling techniques with high-level manual editing, designers will be able to freely experiment, switching to and fro between procedural sketching and meaningful manual operations, without worrying about the impact of each edit action on the consistency of their virtual world.

7. 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).

8. REFERENCES

[1] R. Bidarra, K. J. de Kraker, R. M. Smelik, and T. Tutenel. Integrating semantics and procedural generation: key enabling factors for declarative modeling of virtual worlds. In *Proceedings of the FOCUS K3D Conference on Semantic 3D Media and Content*, Sophia Antipolis - Méditerranée, France, February 2010.

[2] E. Bruneton and F. Neyret. Real-time Rendering and Editing of Vector-based Terrains. In G. Drettakis and R. Scopigno, editors, *Computer Graphics Forum: Eurographics 2008 Proceedings*, volume 27, pages 311–320, Crete, Greece, 2008.

[3] G. Chen, G. Esch, P. Wonka, P. Müller, and E. Zhang. Interactive Procedural Street Modeling. In *SIGGRAPH '08: Proceedings of the* 35^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, volume 27, pages 1–10, New York, NY, USA, 2008. ACM.

[4] G. de Carpentier and R. Bidarra. Interactive GPU-based Procedural Heightfield Brushes. In *Proceedings of the* 4^{th} *International Conference on the Foundations of Digital Games*, Florida, USA, April 2009.

[5] M. de Villiers and N. Naicker. A sketching interface for procedural city generation. Technical report, Department of Computer Science, University of Cape Town, November 2006.

[6] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz. Realistic Modeling and Rendering of Plant Ecosystems. In *SIGGRAPH '98: Proceedings of the* 25^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 275–286, New York, NY, USA, 1998. ACM.

[7] D. Finkenzeller. Detailed Building Facades. *IEEE Computer Graphics and Applications*, 28(3):58–66, 2008.

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

[9] E. Galin, A. Peytavie, N. Marchal, and E. Gurin. Procedural Generation of Roads. In *Computer Graphics Forum: Proceedings of Eurographics 2010*, volume 29, NorrkÃűping, Sweden, May 2010. Eurographics Association.

[10] G. Kelly and H. McCabe. Citygen: An Interactive System for Procedural City Generation. In *Proceedings of GDTW 2007: The Fifth Annual International Conference in Computer Game Design and Technology*, pages 8–16, Liverpool, UK, November 2007.

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

[12] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. V. Gool. Procedural Modeling of Buildings. In *SIGGRAPH '06: Proceedings of the* 33^{rd} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 614–623, New York, NY, USA, 2006. ACM.

[13] F. K. Musgrave. *Methods for Realistic Landscape Imaging*. PhD thesis, Yale University, New Haven, CT, USA, 1993.

[14] Y. I. H. Parish and P. Müller. Procedural Modeling of Cities. In *SIGGRAPH '01: Proceedings of the* 28^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 301–308, New York, NY, USA, 2001. ACM.

[15] PixelActive. Cityscape 1.8. `http://pixelactive3d.com/Products/CityScape`, 2010.

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

[17] R. M. Smelik, K. J. de Kraker, T. Tutenel, R. Bidarra, and S. A. Groenewegen. 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.

[18] R. M. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra. Declarative Terrain Modeling for Military Training Games. *To be published in International Journal of Computer Game Technology (IJCGT)*, 2010.

[19] R. M. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra. Interactive Creation of Virtual Worlds Using Procedural Sketching. In S. Seipel and H. Lensch, editors, *Proceedings of Eurographics 2010: Short Papers*. Eurographics Association, May 2010.

[20] R. M. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra. SketchaWorld: A Declarative Framework for Procedural Modeling of Virtual Worlds. *Submitted for publication*, 2010.

[21] S. Stachniak and W. Stuerzlinger. An Algorithm for Automated Fractal Terrain Deformation. *Computer Graphics and Artificial Intelligence*, 1:64–76, May 2005.

[22] O. Stava, B. Beneš, M. Brisbin, and J. Křivánek. Interactive Terrain Modeling Using Hydraulic Erosion. In M. Gross and D. James, editors, *Eurographics / SIGGRAPH Symposium on Computer Animation*, pages 201–210, Dublin, Ireland, 2008. Eurographics Association.

[23] B. Weber, P. Müller, P. Wonka, and M. Gross. Interactive Geometric Simulation of 4D Cities. *Proceedings of Eurographics 2009*, 28:481–492, April 2009.

[24] H. Zhou, J. Sun, G. Turk, and J. Rehg. Terrain Synthesis from Digital Elevation Models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):834–848, July-Aug. 2007.