An Instructor-Centered Declarative Approach to Modeling Synthetic Environments

Frido Kuijper, Rob van Son, Ruben Smelik
TNO
The Hague, The Netherlands
frido.kuijper@tno.nl, rob.vanson@tno.nl, ruben.smelik@tno.nl

ABSTRACT

Instructors often have to deal with the fact that available synthetic environments do not match their specific training requirements. Geo-specific terrains are not guaranteed to provide the desired setting. Dedicated geo-typical terrain databases can help solve this issue because they can be modeled explicitly to match training requirements. However, existing editing environments for this type of terrains are often laborious and expensive in use. Procedural modeling techniques may help in rapidly building geo-typical terrains, but tools that support these techniques are difficult to configure and do not allow for direct integration of the instructor's intentions into the modeling process.

In this paper, we present the results of a PhD study into the use of a novel declarative method for modeling synthetic environments. This method proposes a workflow that exploits procedural techniques to rapidly generate terrains, controlled by intuitive declaration of the terrain from an instructor's perspective. At the basis of the method is a concept of interactive sketching of terrain features and constraints at a higher level of abstraction, close to the instructor's intents, while maintaining a short feedback loop on the resulting terrain.

We have evaluated the use of this approach for a variety of training simulators currently in use within the Dutch DOD. Our observations show that the approach provides an effective instructor-centered alternative to current methods. Challenges exist in creating varied content, as well as in providing appropriate methods to express user intent in terms of terrain features and constraints, while ensuring consistency in the resulting model. We describe our solutions to these challenges and discuss the value of our method for the military training simulation community.

ABOUT THE AUTHORS

Frido Kuijper is a senior research scientist at TNO, a Dutch research institute. He holds a Master's degree in Computer Graphics from Delft University of Technology and has 20 years experience in the field of Modeling & Simulation for military applications. Previous work of Frido has been in the field of networked simulation architectures and virtual environment technology. Currently, he is the lead scientist for TNO in research that aims at the generation and application of synthetic natural environments for Defense and Civil Security applications.

Rob van Son is a research scientist for TNO in the Netherlands. Currently, he's performing research within TNO's modeling and simulation research program on synthetic environment modeling techniques, focusing on synthetic natural environment generation and representation. He holds a Master's degree in Computer Science from Utrecht University.

Ruben Smelik is a research scientist for TNO in The Netherlands. He is a PhD candidate at Delft University of Technology on a project entitled "Automatic Creation of Virtual Worlds". The project - the main source for this paper - is part of the Dutch research program "Game Research for Training and Entertainment (GATE)". Ruben holds a Master's degree in Computer Science from the University of Twente.

An Instructor-Centered Declarative Approach to Modeling Synthetic Environments

Frido Kuijper, Rob van Son, Ruben Smelik
TNO
The Hague, The Netherlands
frido.kuijper@tno.nl, rob.vanson@tno.nl, ruben.smelik@tno.nl

INTRODUCTION

Synthetic environments are one of the prerequisites for training simulation. A model is required that describes the virtual environment in which a training exercise takes place. When specific mission preparation is the purpose of the simulation, the virtual environment shall be geo-specific, i.e. it shall resemble the real mission environment. In this paper we focus on the class of environments that are dedicated to basic training and education, where resemblance with a real world location is not the issue. The issue we want to address, is that an instructor should be effectively and efficiently supported to obtain a synthetic environment that best suits the training goals and harmonizes with the setup of the exercise the instructor has in mind.

The research presented in this paper is based on a PhD study that has been concluded in 2011. Parts of the results have been published earlier, see Smelik et al (2008-2011). This paper adds to these publications by summarizing and more specifically evaluating our approach to synthetic environment modeling from the perspective of defense and security related training simulation.

PROBLEM STATEMENT

When preparing simulation exercises in the context of a training curriculum, for each exercise, the instructor has to obtain a synthetic environment that supports the training goals of the exercise. Most commonly, instructors work with *what is available*. They scan available environments and look for locations in these environments that are suitable for the exercise. The search process is quite often done in a quasi-random way and the judgment whether a location is suitable is a subjective one, usually performed by studying the map or by inspecting the simulated world in 3D. If the search fails, the fallback is to have a dedicated synthetic environment built. This requires complex modeling tools, experts and - most importantly - budget. In all of these cases, it is our observation that the process of obtaining a suitable synthetic environment for training simulation is determined by the limitations of available environments and the budgetary constraints of building one.

In our research, we have tried to overcome these limitations that instructors are faced with. Rather than working from *what is available* or hitting the wall of costly model development, we highlighted the fact that instructors preparing a simulation exercise for training or education purposes, always have a clear idea about what kind of synthetic environment they need to support the exercise. Taking this as a starting point, we defined our problem statement to be the following question:

Can we develop an alternative approach to modeling synthetic environments for training simulation that:

- is driven directly by the specific ideas the instructor has about the required environment?
- can be implemented in a tool that is accessible for non-modeling-experts (such as the instructors themselves)?
- results in a productivity gain and cost reduction compared with traditional manual modeling approaches?

AVAILABLE APPROACHES

Three different approaches can be distinguished when summarizing currently available methods and tools that support the modeling of synthetic environments for training simulation:

- automated processing of GIS data;
- manual interactive editing;
- procedural modeling tools.

The most commonly used approach is through *automated processing of GIS data*. Various commercial tools are available that allow for importing of elevation data, imagery and vector data and translating this data into a coherent 3D synthetic environment. The advantage of these tools is that generation is fully automated and that correlated output to various simulator components can be generated. Under certain conditions, the automated translation of GIS data into a 3D model can even be deferred to the simulation runtime stage (this approach has been applied in the past by various simulator systems and is still propagated in numerous systems. The downside of relying on automatic translation of GIS data is that, before this automated process can be started, a complex task has to be fulfilled of setting up the numerous components and parameters of the processing chain.

The second main approach is the one that fully relies on *manual editing* of the 3D synthetic environment. Through an interactive WYSIWYG (what you see is what you get) interface, the user shapes the terrain by brushing and crafting the ground profile, followed by brushing, drawing and dragging the features on top of the terrain skin. The advantage of these tools is the fact that the user works in a short feedback loop. The resulting 3D terrain is immediately visible after an editing action. On the downside, producing large detailed terrains requires great effort.

The third approach - *procedural modeling tools* - we identify here is actually not an independent approach in itself, but rather a class of methods that provides building blocks for the previous two approaches we mentioned. Procedural modeling tools exhibit the capability of generating a (part of a) synthetic terrain on the basis of a small set of input data. Some examples for which procedural generation tools can be found include, see Smelik (2009b):

- terrain skin generation, see Miller (1986), Musgrave (1989);
- river network generation, see Kelley (1988), Prusinkiewicz (1993), Teoh (2008);
- plant model generation and vegetation distribution, see Prusinkiewicz (1990), Deussen (1998);
- road network generation, see Sun (2002);
- building generation, Parish (2001);
- city generation including roads and buildings, see Wonka (2003), Müller (2006), Finkenzeller (2008).

Procedural modeling tools can be very powerful and increase the productivity of modelers, but in general they come with a complex user interface. Quite often, the modeler has to tune a set of non-intuitive parameters, see Smelik (2009b). Also, most of the procedural tools that are available, only address a part of the entire modeling workflow. Hence, the resulting workflow consists of a series of chained tools, making the process only manageable for skilled experts.

When striving for a solution to the problem as we stated it, currently available approaches fail to deliver adequate solutions. The main reason for this failure is the complexity of the tools, requiring highly skilled - and costly - personnel. A different approach is required to deliver tooling that is manageable by non-modeling-experts and allows for efficient and effective synthetic environment generation that directly solves the needs of the instructor.

PROPOSED DECLARATIVE APPROACH

Basic concept

The approach we propose is *instructor-centered*: it starts with the instructor that needs a synthetic environment to perform a certain training simulation exercise. It is our experience that the instructor always has a clear idea about what kind of environment is required.

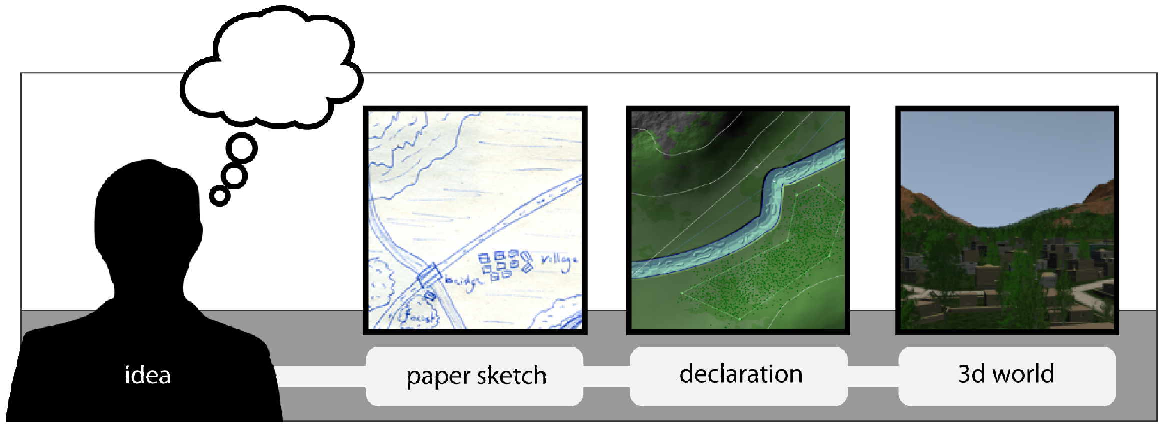

Figure 1. The basic concept of our approach: the ideas of the instructor are formalized into a declaration of his intentions, followed by automated modeling from these declarations using procedural modeling techniques.

Typically, the instructor is able to quickly sketch the outline of the terrain and identify the main characteristics of the desired terrain. The basic concept of our approach is to catch these ideas of the instructor and formalize this into a specification. From this specification, the synthetic terrain is derived via procedural techniques, as outlined in Figure 1.

We call this a *declarative approach*: the instructor is enabled to *declare* his intentions for the synthetic environment. It is key to the success of the approach that this declaration is done in a very intuitive and accessible way that is close to the way the instructor is used to work. The challenge is to, at the same time, provide enough flexibility and control that enable the generation of effective and appealing synthetic environments.

The user perspective: sketching the world
The declaration is in fact a specification of the desired environment. We describe a sample to illustrate our approach a typical verbally expressed declaration as could be quoted from an instructor, usually paired with a quickly drafted sketch on a piece of paper:

"I want a terrain resembling Afghanistan terrain. The terrain shall be 16 km x 16 km, mainly flat in the center, but surrounded by hills. There shall be a small town along a river. The main road of the town crosses the river in the center of the town. The bridge shall be visible from a specific threat point on the hills."

In our concept, we have devised four means of expression to formalize this type of declaration and thus enable procedural generation of the environment:

- **Choose a template**
  "I want a terrain resembling Afghanistan terrain". The user gets to choose a *template* for his environment. This template reflects the entire style of appearance of the environment. The types of buildings, vegetation and all other features will be influenced by the selected template. Provided, obviously, that a selection of content is available, the template choice lets the user control the appearance of the resulting environment without having to browse through large libraries of content.

- **Sketch the landscape**
  "The terrain shall be 16 km x 16 km, mainly flat in the center, but surrounded by hills.". After setting the size of the terrain, the user will be provided with a map of the terrain that is divided into grid cells. The characteristics of the landscape can now be specified by interactively painting *ecotopes* (areas of homogeneous terrain and features) into the grid cells. These ecotopes encompass both elevation information (elevation ranges, terrain roughness) and soil material information (sand, grass, rock, etc.). In the sample declaration, the user would choose to paint ecotopes named *arid flat lands*, *arid hills* and *arid low mountains* into the designated grid cells of the landscape sketch. By doing so, a complete landscape can be generated that includes both elevation profile as well as appearance and surface characteristics.

- **Sketch the features**
  "There shall be a small town along a river. The main road of the town crosses the river in the center of the town.". This part of the declaration is formalized by sketching features into the terrain. Using common line and polygon drawing tools, the user can place features like roads, rivers, lakes, vegetation and build-up areas *on the map.*
  The main difference between our approach and common interactive editing tools is that the drawing of lines and polygons is done by sketching rather then exact drawing. The input can be provided in a rough way, with very few points. The procedural generation algorithms will ensure that the final features will be modeled in a detailed and plausible way.

- **Specify constraints**
  "The bridge shall be visible from a specific (threat) point on the hills.". The final part of the verbal declaration we took as our sample is one of the most interesting parts. It has most of all to do with the particular intentions the instructor has in mind with an exercise. The instructor wants to introduce a threat in the scenario and therefore needs a position in the terrain that has line of sight to a location where the trainees will be exposed to the threat. Through the introduction of semantic constraints, we enable that generation of the synthetic environment will automatically comply with the intentions the instructor expresses in the constraints.

By these four concepts, we provide the means to formalize the declaration of the instructors intent. Figure 2 illustrates how these concepts work in practice on the sample declaration we used above. A few things are important to notice here. Firstly, the four concepts are not necessarily operated in ordered sequence. Terrain modeling is an iterative process and our approach adheres to that lemma. It should be possible to sketch parts of the landscape, sketch features and specify constraints in any order and refine parts as needed. Secondly, it is stressed that the interactive editing process as depicted is fundamentally different from common interactive editing in WYSIWIG

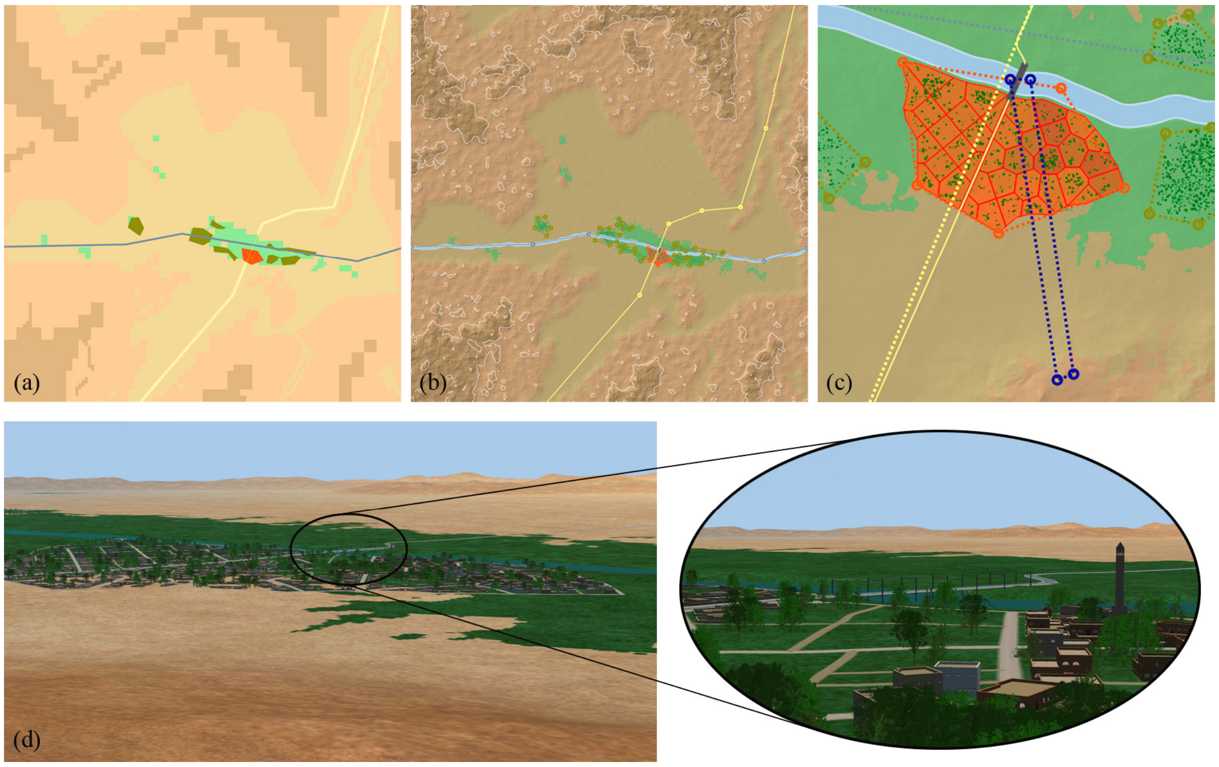

Figure 2. The declarative approach in action: specify the terrain by (a) sketching ecotopes into grid cells and sketch the required features. The landscape and features are then procedurally generated (b). A line of sight constrained (c) is specified (the blue area) to make sure that the bridge is visible from the threat location (d) as illustrated in the resulting 3D model as seen from the threat location.

modeling tools. In our approach, what is edited is not the terrain itself, but the *specifications* of the terrain. The terrain itself is automatically generated through procedural generation from the specifications. The resulting landscape and feature characteristics are feedback directly (in a short feedback loop to enable interactive editing) to the user in the 2D map representation the user is interacting with.

Under the hood: procedural generation

As mentioned, the user is not directly editing the synthetic terrain. Under the hood of our concept is a set of procedural generation techniques that generate the actual synthetic environment according to the user's specifications. We now outline the techniques that we apply and the mechanisms that are required for the techniques to be effective. See Smelik (2011b) for a detailed description.

Semantic layers and consistency maintenance

At the heart of the technical approach is a semantic world model that represents the synthetic environment in five different layers: urban, road, vegetation, water and landscape layer (see Figure 3).

The landscape layer describes the terrain skin elevation profile and soil characteristics. It provides the foundation for feature placement in the four layers on top of it. Obviously, these layers cannot be dealt with as independent parts of the world. It is essential that the method deals with interactions between the layers according to clear and manageable rules. Whenever the user specifies a new feature in the environment, it shall be considered in relation to other features that are on the same location or nearby. Also, the elevation profile underneath the requested feature may be unsuitable for the feature. To detect and resolve these interactions, a feature can issue two types of requests:

- *claim*: A feature can make a claim for an area of terrain in order to place its objects. The consistency maintenance mechanism resolves interactions - two features claiming overlapping areas - on the basis of priorities. Conflicts can either be resolved by denying the claim with the lower priority, or by deciding that the two features will *connect*.

Examples of resolution by *connection* are two roads that overlap and generate a crossing, or a road and a river where it is decided to build a bridge across the river.

- *modification*: In general, the features will adapt their structure and geometry to the landscape profile. However, features may also request a modification of the landscape profile and/or soil type on a claimed area. Think for example of a building that requests to flatten the terrain underneath it. Or a hydrographic feature that requests to include its bed and banks into the landscape elevation profile.

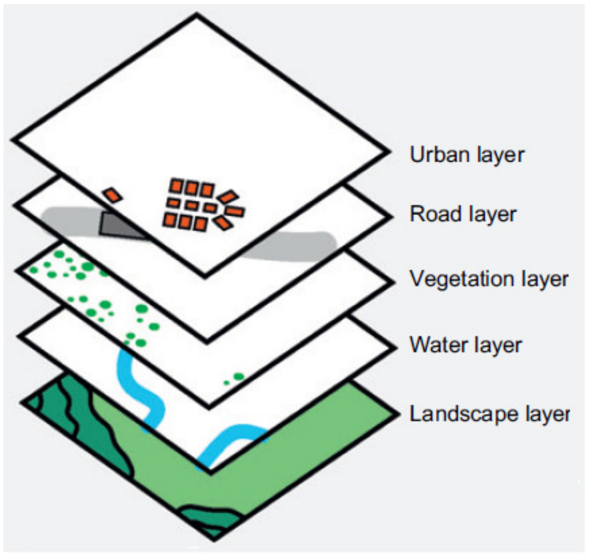

Figure 3. The synthetic environment model is structured according to five stacked semantic layers.

Landscape generation

The landscape data is initially generated from the landscape sketch that provides for each grid cell an ecotope definition. Without bothering the user with complex configuration interfaces, the ecotope definition translates into a set of parameters that determine elevation range and terrain roughness. A combination of several flavors of fractal noise, such as ridged multi-fractal noise (Musgrave, 1989), mixed according to the roughness factor, are used to determine the elevation value within the desired min-max range. The distribution of soil material is based on the ecotope value and by mapping the elevation value to a lookup table.

Water feature generation

Rivers and lakes are generated with procedural generation. Rivers - and likewise other linear hydro-graphic features like canals and ditches - are determined by a specification polyline as input by the user as well as the landscape elevation profile. The actual river path is found by finding a suitable path between each of the points on of the specified polyline (see Figure 4). This search algorithm tries to find a path that has declining elevation along the path, a minimum of strong curves and it avoids features that have higher priority than the river. The path found is smoothed by fitting a Catmull-Rom spline to it. The segments of this spline are attributed with a river width value that depends on local slope. The landscape elevation profile is then locally adapted to fit the river's bed and bank. The algorithm is very efficient and allows for quick feedback to the user. It is the users responsibility to choose appropriate points for the specification polyline. If these points follow an impossible elevation profile, the results will not be plausible. Due to the speed of the feedback, the chosen polyline points can easily be interactively adjusted.
Lakes are specified by its coarse area. The outline is then slightly perturbed by noise techniques to obtain a more natural border of the lake. The landscape profile is subsequently modified to accommodate the border and bathymetry of the lake.

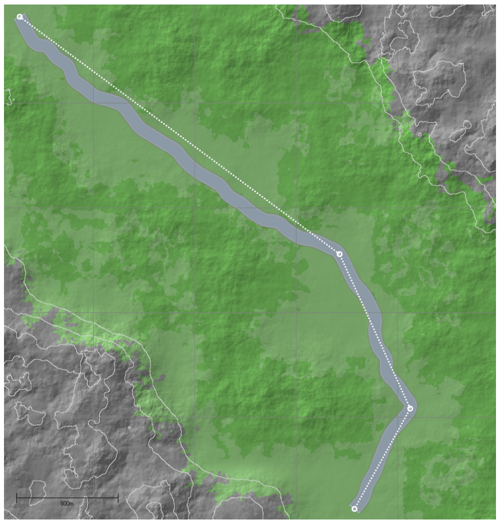

Figure 4. From a simple input polyline, a more detailed plausible river path is determined.

Vegetation feature generation

Similar to lakes, forests can be specified by drawing a coarse areal. This area is then perturbed with noise. An iterative procedure based on the method introduced by Deussen et al. (1998), which simulates the competition of plants for natural resources as space and water, determines a distribution of trees of different species in this forest (see Smelik, 2010) for specific details on this procedure). Additional input for this procedure is the local elevation profile and soil material, as the definition of a tree species includes preferences for certain soil types, as well as elevation and slope ranges, as illustrated in Figure 5. Through consistency maintenance, any individual tree that conflicts with another feature with higher priority will be removed.

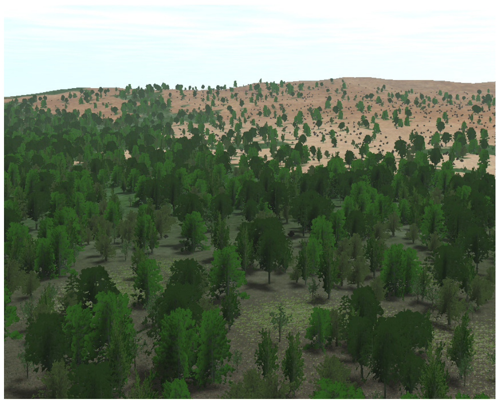

Figure 5. Trees are planted according to a defined preference for certain soil types as well as elevation and slope ranges.

Road feature generation

A road feature is declared by coarsely specifying its path. Its generation procedure is based on the path plotting algorithm introduced by Kelly et al. (2007). This algorithm iteratively finds a smooth path between a set of control points of a polyline defined on an elevation map. It prefers an even change in elevation from start to end, while guaranteeing all control points to be visited and the path to deviate only within a limited range from the specification. The algorithm was extended to avoid unacceptably sharp turns and slopes, to connect to existing features, such as rivers, if necessary, and to avoid potential feature conflicts with negative consequences for the road. Although the procedure uses a scoring mechanism to determine a path, it is not optimizing a cost function, as for instance the A*-based path finding method by Galin et al. (2010), and therefore is not guaranteed to find an optimal path in all cases. However, our procedure typically runs more interactively while staying close to the coarse path sketched by the user, thereby providing more fine-grained control.

Urban feature generation

The purpose of urban feature generation is to allow the user to generate an entire city with just a few clicks. A template choice determines the main characteristics of the city, including the type of city structure (e.g. radial, grided) and the population size.

A detailed description of our urban feature generation can be found in Groenewegen (2009). Basically, the algorithm structures the feature into a number of *clusters*, i.e. areas the feature succeeds to claim within the specified city boundary. The clusters are then divided into *districts* with a specific function, e.g. commercial, industrial or residential. The placement of the types of districts across the clusters follows an iterative algorithm that takes into account the interaction between the types of districts (the different types are defined to attract or repulse each other), the suitability of soil type and the distance to nearby road and river features. After the districts have been placed, the road network within the city is created, followed by subdividing the open spaces within the road network into suitable parcels and building lots. Each road and building feature will issue a modification request to adjust the local elevation profile.

Buildings are procedurally generated for each building lot. The type of building will depend on the specifications of the city and the function that was assigned to the parcel during the structuring of districts. Many methods have been proposed to implement procedural building generation, Parish (2001), Wonka (2003), Müller (2006), Finkenzeller (2008). Most of these methods generate buildings that adapt to the designated lot shape and can also be controlled to a certain extent in e.g. height. Figure 6 shows a sample of buildings we have generated.

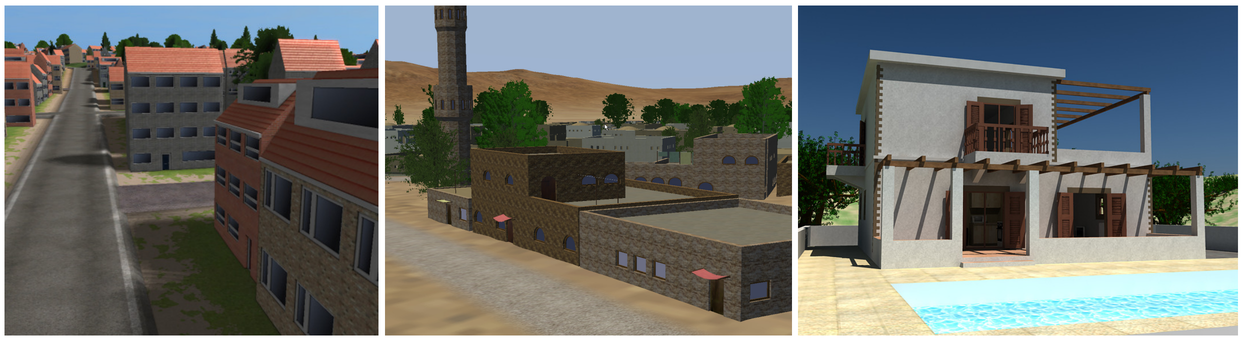

Figure 6. Some sample urban features. Shape grammar based building models allow virtually any type of building, like the detailed villa on the right hand side.

Full flexibility is obtained when using programmable building generation, like shape grammar based building generation as in Müller (2006) that is included in our method. Buildings can be made to look as good as the nice villa shown in Figure 6. However, the detail trades rapidly against processing time. Building generation typically becomes a bottle-neck in the feedback loop to the user.

Constraints solving

The specification and solving of semantic constraints on the environment is a broad theme that includes many possible different constraints and complex algorithmic challenges.

We have studied a number of constraint types, using GeoBML concepts. Hieb et al (2006) designed GeoBML to provide a formal description of terrain characteristics for geospatial battlefield applications. Given the fact that this formal language has the capability to *describe* tactically relevant characteristics of a real environment, it can be assumed that it is also capable of *prescribing* these characteristics for a synthetic environment. In our research, we addressed three types of constraints:

- *line of sight constraint*: there shall be intervisibility between two areas;
- *route constraint*: there shall be a route between two points;
- *choke point constraint*: the designated area shall be a choke point (i.e. surrounded by elements that provide cover for enemy), see Figure 7.

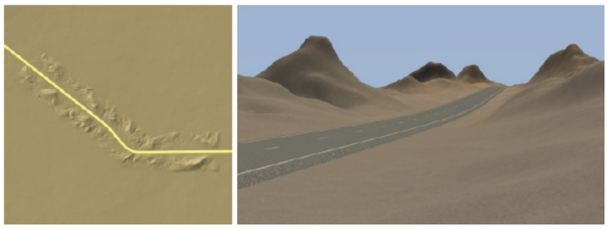

Figure 7. A sample *choke point constraint*: a piece of road is turned into a choke point by modifying the landscape around it.

Within the scope of this paper, we cannot address the constraint solving in detail. In Smelik (2011a) it is described in more detail. In essence, each constraint is assigned an extent (the area in which the constraint influences the terrain). Constraint solving comes down to modifying the landscape and/or interacting with the features that have claims within the extent of the constraint.

IMPLEMENTATION

For research and evaluation purposes, we have implemented the proposed approach in a self-contained software tool that enables the interactive sketching of environment specifications, the procedural generation of the environment model and the export of this model.

The tool has been implemented in C# and C++ and uses OpenCL to run parts of the procedural generation on the GPU. The GPU programming generally delivers an order of magnitude gain in performance, thus making interactive sketching feasible. An example is the generation of the elevation and soil profile, which is done completely on the GPU. OpenSceneGraph is used to build a 3D visual representation of the environment and enables an iterative design process based on the final 3D appearance of the model.

The current performance is acceptable for small and medium sized landscapes (up to 1000 km2). Most procedural operations typically take between 100 ms and 3s. However, when modeling larger landscapes, continuously updating the 3D geometry of the virtual world becomes a bottleneck that hinders the interactive feedback loop.

Implemented export modalities include OpenSceneGraph and Collada for the resulting 3D visual model of the environment as well as GIS raster and vector data that describe the environment.

EVALUATION

Evaluation method

Several iterations of our implementation have been evaluated with the intended target user group. This group consists of military training instructors and training developers and modeling experts from both military and entertainment gaming contexts.

We have used a hands-on qualitative analysis to evaluate the applicability and usability of our approach. The evaluation consisted of a free-play modeling session and the execution of a specific modeling use case in which users were instructed to sketch a relevant and realistic training scenario on paper and model it. During these sessions we have recorded observations. Afterwards, we have conducted interviews which contained questions about the following topics:

- applicability of our approach in relation to the user's current activities and workflow;
- ease of use;
- quality of output.

Results

The users' response to our approach on a conceptual level was generally positive. The match between their - as they described it - actual thinking and working process and our proposed workflow makes terrain modeling *easier and more accessible* for non-expert users. One specific user suggested that this approach, if easy-to-use and capable of producing a synthetic environment in a few hours, could potentially extend their work package from scenario creation in existing terrains to modeling of new terrains.

A point of feedback that was put forward explicitly by military users is the fact that the available content - i.e. terrain types and feature classes - should match the discourse typically used in a specific military setting.

Most users envision this approach as the ideal starting point for creating coarse geo-typical terrains, after which they would still need the capability to refine the environment model manually in detail. The users specifically mentioned the unusual combination of a quick workflow for generating unexpectedly extensively detailed models (see Figure 8).

DISCUSSION

Comparison with other Methods

When comparing our approach to existing tools that employ *automated processing of GIS data*, the most notable difference is *the suitability for a particular class of terrain*. Whereas the latter is most suited for the modeling of geo-specific environments, our approach appears better suited for the modeling of geo-typical environments. The inherent non-deterministic character of our approach is not suited for accurately modeling an existing part of the world. However, for geo-typical environments, our approach does not require the laborious process of creating detailed GIS data from scratch, removing additional time, expert work and budgetary penalties to the workflow.

A declarative approach permits, by definition, a smaller *degree of control* over the appearance of the environment model compared to automated GIS data processing tools. These tools require the user to specify exactly how to model specific features, while in a declarative approach control is asserted by and limited to choice of template and the declaration of features.

Compared to *manual editing tools*, creating a geo-typical terrain using a declarative approach is significantly less time-intensive. Inversely, manual editing tools allow for a much greater degree of control over the appearance of the environment.

Existing *procedural generation tools* are comparatively limited in scope. These tools focus on a limited set of feature categories (mostly terrain elevation and terrain skin) and do not provide an integrated set of procedures that interact together.

Compared to all existing methods, we believe that none of them is easy-to-use and intuitive - in particular for someone who is not a modeling expert. In the evaluation of our approach we see that different types of users require little time to accustom themselves to the user interface and model the environments that they envision.

Advantages of our approach

Based on the comparison of our declarative approach and other existing methods, we can summarize the advantages as follows.

First of all, the declarative approach shows to be an approach that allows for a better match between military training instructors' ideas and needs and the eventual environment model. Although we have not performed an extensive study as to what these ideas are exactly, our observations show that the sketch-based approach closely matches the terrain sketching that instructors do before a training exercise.

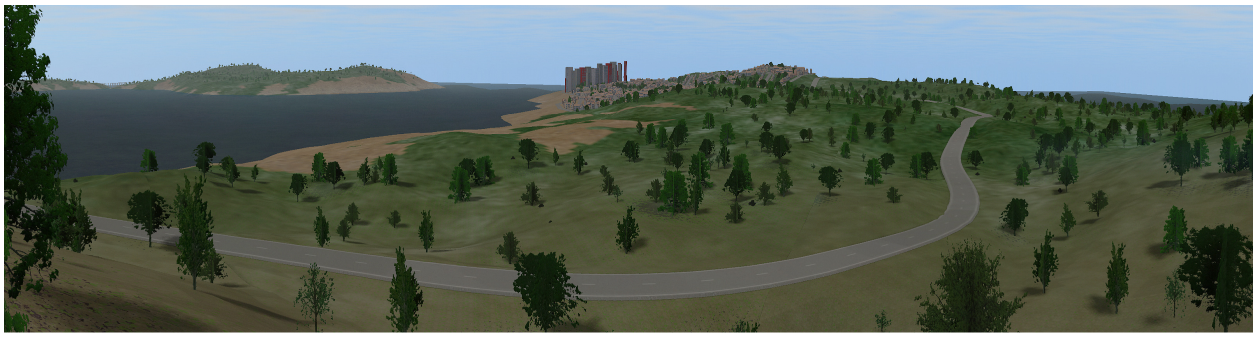

Figure 8. A sample terrain as built with our prototype. This type of output can be achieved within 15 minutes.

Secondly, our approach shows to be easy to use. All users that evaluated the tool needed little time to accustom themselves to the tool and to start creating an environment model.
Third, a declarative approach allows for a quick start to any modeling workflow. All evaluators were able to create a terrain of 16 km x 16 km with a high level of detail (including build-up areas) from scratch in less than an hour.
All of the advantages mentioned above together contribute to a fourth benefit. The reduction of the need for expert manpower, ease of use and quick workaround time significantly reduce the costs involved in creating an environment model.

Disadvantages of our approach
Inherent to the declarative approach is that the user has limited control over the details of the final environment model. Given the non-deterministic character of the procedural generation, some results may please the user while other may not. This disadvantage is minimized by the short interactive feedback loop of the method that allows the user to quickly refine his declarations. Also, we allow the user to influence feature parameters, either for an entire feature or locally. Further, we are investigating more fine-grained mechanisms for locking features or objects from modification, grouping objects together and introducing local geometric constraints.

A second disadvantage of our approach is the dependency on content. By lessening the degree of control over the (detailed) appearance of the generated environment model, demands for high-quality and fitting content become very high. If this content is not available or not sufficient, the user has no capability to compensate for this.

Critical success factors
We identify a number of critical success factors for the application of a declarative approach for modeling environments for training simulation.

First, *applicability* is a key factor. For a declarative approach in particular, it is essential that the available tools and methods correlate with the users' perception, routines and needs. This implies that the required content should be sufficiently rich and complete to meet specific needs.

We expect that typical users are not waiting for another tool to be added to their already extensive collection of tools used for modeling training simulation environments. Instead, for this approach to be accepted, it should be part of an integrated *end-to-end capability* in which users can refine their models manually and output several correlated environment representations at once.

The last factor to consider is *compatibility* with existing training simulators. This implies that data output shall not only be compatible with commonly used simulator platforms such as VBS2, but also should be able to serve as input for existing modeling workflows.

Way ahead
Currently TNO and re-lion are combining efforts to integrate our approach with their manual WYSIWYG builder tool. This manual editing interface shall be complementary to the declarative phase, therefore allowing the user to quickly declare and generate an environment model and consecutively manually edit its details. Also, in this project, we strive to create the capability to generate output for a number of training simulators currently in use by the Dutch Department of Defense, i.e. VBS2, Steel Beasts, and the Small Caliber Arms Simulator.

For different applications, different templates and content shall be developed. In particular, a template representing a Dutch rural environment is currently under development for a levee patrol training simulator.

CONCLUSIONS

We have presented an approach to synthetic environment generation that puts the instructor in the driving seat. The declarative approach enables non-modeling-experts to quickly build a detailed geo-typical terrain model that closely matches the needs of the instructor. We solve the common problem in training simulation that instructors have to work with the limitations of available terrain databases or are faced with (too) costly developments of dedicated terrain models. Conditional to this statement is that our approach shall be turned into a tool that provides an end-to-end capability with sufficient content, allowing instructors and training developers to build their own synthetic environments. We expect high user acceptance when the method is integrated within a WYSIWYG manual terrain editor, resulting in a tool that provides both the productivity of our method as well as the full user control of the manual editing.

ACKNOWLEDGEMENTS

This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO).
We thank Rafael Bidarra and Tim Tutenel of Delft University of Technology and Klaas Jan de Kraker of TNO for their co-operation in this project.

REFERENCES

Belhadj, F. & Audibert, P. (2005). Modeling Landscapes with Ridges and Rivers: Bottom Up Approach. *GRAPHITE '05: Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, p. 447-450.

Deussen, O., Hanrahan, P., Lintermann, B., Měch, R., Pharr, M. & Prusinkiewicz, P. (1998). Realistic Modeling and Rendering of Plant Ecosystems. *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, p. 275–286.

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

Galin, E., Peytavie, A., Marchal, N. & Gurin, E. (2010). Procedural Generation of Roads. *Computer Graphics Forum: Eurographics 2010 Proceedings*, *29*, p. 429-438.

Groenewegen, S.A., Smelik, R.M., de Kraker, K.J. & Bidarra, R. (2009). Procedural City Layout Generation Based on Urban Land Use Models. *Eurographics 2009: Short Papers,* p. 45–48.

Hershey, D.J., Klein, J.A., McKeown D.M. & Starmer, W.J. (2011). Automating the Generation of Large Scale Environments. *Proceedings of IMAGE 2011, Scottsdale, Arizona, June 2011.*

Hieb, M.R., Powers, M.W., Pullen, J.M. & Kleiner, M. (2006). A Geospatial Battle Management Language (geoBML) for Terrain Reasoning. *11th International Command and Control Research and Technology Symposium*, paper I-110.

Kelley, A.D., Malin, M.C. & Nielson, G.M. (1988). Terrain Simulation Using a Model of Stream Erosion. *SIGGRAPH '88: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques,* p. 263–268.

Kelly, G. & McCabe, H. (2007). Citygen: an Interactive System for Procedural City Generation. *Proceedings of the Fifth Annual International Conference in Computer Game Design and Technology*, Liverpool, UK. p. 8–16.

Miller, G. S. P. (1986). The Definition and Rendering of Terrain Maps. *SIGGRAPH '86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, 20*, p. 39-48.

Müller, P., Wonka, P., Haegler, S., Ulmer, A. & Gool L.V. (2006). Procedural Modeling of Buildings. *SIGGRAPH '06: Proceedings of the 33rd Annual Conference on Computer Graphics and Interactive Techniques*. p. 614–623.

Musgrave, F. K., Kolb, C. E. & Mace, R. S. (1989). The Synthesis and Rendering of Eroded Fractal Terrains. *SIGGRAPH '89: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, p. 41-50.

Parish, Y.I.H. & Müller, P. (2001). Procedural Modeling of Cities. *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques,* p. 301–308.

Prusinkiewicz, P. & A. Lindenmayer (1990). *The Algorithmic Beauty of Plants*, Springer-Verlag.

Prusinkiewicz, P. & Hammel, M. (1993). A Fractal Model of Mountains with Rivers. *Proceeding of Graphics Interface '93*, p. 174–180.

Smelik, R.M., de Kraker, K.J., Tutenel, T. & Bidarra, R. (2009a). A Case Study on Procedural Modeling of Geo-typical Southern Afghanistan Terrain. *Proceedings of the IMAGE 2009 Conference*, July 13-16, St. Louis, MO, USA.

Smelik, R.M., de Kraker, K.J., Tutenel, T., Bidarra, R. & Groenewegen, S.A. (2009b). A Survey of Procedural Methods for Terrain Modelling. *Proceedings of the CASA Workshop on 3D Advanced Media in Gaming and Simulation (3AMIGAS)*, June 16, Amsterdam, The Netherlands.

Smelik, R.M., Galka, K.A., Kuijper, F., de Kraker, K.J. & Bidarra R (2011a). Semantic Constraints for Procedural Modelling of Virtual Worlds. *Proceedings of PCGames 2011 - Workshop on Procedural Content Generation for Games*, 28 June - 1 July, Bordeaux, France.

Smelik, R.M., Tutenel, T., de Kraker, K.J. & Bidarra, R. (2008). A Proposal for a Procedural Terrain Modelling Framework. *Proceedings of Eurographics Virtual Environments Symposium*, May 29-30, Eindhoven, The Netherlands

Smelik, R.M., Tutenel, T., de Kraker, K.J. & Bidarra, R. (2010). Declarative Terrain Modeling for Military Training Games. *International Journal of Computer Games Technology 2010,* paper ID 360458.

Smelik, R.M., Tutenel, T., de Kraker, K.J. & Bidarra, R. (2011b). A Declarative Approach to Procedural Modeling of Virtual Worlds. *Computers & Graphics, 35 (2)*, p. 352-363.

Sun, J., Yu, X., Baciu, G. and Green, M. (2002). Template-based Generation of Road Networks for Virtual City Modeling. *VRST '02: Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, p. 33–40.

Teoh, S.T. (2008). River and Coastal Action in Automatic Terrain Generation. *CGVR 2008, Proceedings of the 2008 International Conference on CG & VR*, p. 3–9.

Wonka, P., Wimmer, M., Sillion, F. & Ribarsky. W. (2003). Instant Architecture. *SIGGRAPH '03: Proceedings of the 30th Annual Conference on Computer Graphics and Interactive Techniques*, p. 669–677.