A CASE STUDY ON PROCEDURAL MODELING OF GEO-TYPICAL SOUTHERN AFGHANISTAN TERRAIN

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

Tim Tutenel, Rafael Bidarra
Delft University of Technology
Delft, The Netherlands

ABSTRACT

A cost-effective method of military training is game-based training, for which custom geo-typical terrain models are often most suitable. However, for training instructors, scenario preparation is seriously slowed down by the complexity of current terrain modeling tools and methods. They would benefit from a quick and easy to use automated terrain modeler. The purpose of our declarative modeling framework is to offer an intuitive and fast way for instructors to model geo-typical terrain.

We investigated how procedural methods can be developed and applied to quickly generate specific types of geo-typical terrain models. In this case study, we focused on generating terrain models that contain the typical natural and man-made features of southern Afghanistan. By analyzing these features and by developing and integrating procedural methods in our sketch-based terrain modeling framework, we found that one can quickly achieve convincing results. We present new and tailored methods that are fast and effective and that can generate a complete and consistent geo-typical Afghanistan-like terrain model. The results strengthen our belief that the framework is a firm step towards a declarative, automated terrain modeling process. Moreover, these procedural methods can be tailored to generate other types of geo-typical terrain.

INTRODUCTION

Military training instructors increasingly often employ simulations and modified entertainment games to train personnel in a variety of skills and tactics. One of the difficulties instructors face when using this technology is creating customized scenario content, including platforms and entities, Computer Generated Forces scripting, and 3D terrain databases. Terrain plays a key role in military training scenarios, and the features of available terrain models constrain the types of different training scenarios an instructor can create.

In contrast to, for instance, mission rehearsal scenarios, training scenarios frequently take place on geo-typical terrain. It would be very helpful for instructors if they were able to create their own geo-typical terrain databases. In that case they could conceive terrain databases that are tailored for certain training objectives, allowing them to develop well-constructed, effective and efficient training curricula. However, current manual terrain editors are both too complex and too time-consuming to be useful for instructors; automatic terrain generation methods show a lot of potential, but currently available tools still lack user control and intuitive editing capabilities. The underlying reason for this is that these tools often require an in-depth knowledge of the algorithms to predict the effect of a parameter value on the outcome. This results in a trial and error process for obtaining the desired result.

As an overall consequence, instructors cannot create new terrain, nor can they easily make changes to existing terrain models. Therefore, they are forced to choose from a fixed set of pre-made terrain databases. Although these can cover many possible settings, they do significantly limit the number of potential training scenarios and decreases the variety in the training curriculum.

In order to solve the above problems, we developed a novel modeling framework for creating geo-typical terrain. Our intent is not only to significantly speed up the terrain modeling process but, more importantly, to provide a way for people without special 3D modeling expertise, such as training instructors, to rapidly create terrain models that meet their requirements. We believe that for this purpose a declarative approach is best suited: it allows instructors to focus on declaring *what* they want, instead of focusing on *how* they should model it.

Typically, instructors do have an idea of the layout of a terrain that fits their training scenario. Our framework allows them to express this idea using a sketch interface and creates the terrain model accordingly. In this way, the framework lets instructors focus on declaring the terrain they need, without bothering them with modeling tasks. The framework provides automated modeling by integrating a variety of

Presented at the **IMAGE 2009** Conference
St-Louis, Missouri, July 2009.

procedural content generation methods and making them accessible and user-friendly.

Most existing procedural terrain generation methods focus on Western Europe- or North America-like terrain, while for military training purposes a terrain that resembles actual and possible mission areas has more value. As, for The Netherlands, southern Afghanistan is currently an important mission area, this case study demonstrates the automated modeling of geo-typical southern Afghanistan terrain. It leverages our procedural terrain modeling framework to generate terrain that resembles and has the typical features of the southern Afghanistan terrain and culture.

The remainder of this paper is organized as follows. First, we discuss the state of the art on procedural terrain modeling. Next, we give an overview of our new terrain modeling approach. After this follows the case study. We describe which typical Southern Afghanistan features we have reproduced and show our procedural methods for generating both the natural environments and urban areas. To show the results, a selection of images from the generated terrain models is presented. Lastly, we summarize our contributions and indicate areas for further research.

BACKGROUND

Procedural modeling has been an active research topic for at least thirty years. A major topic within procedural modeling is the automatic generation of terrain models, which started with natural phenomena such as terrain elevation and growth of plants in the 1980 and 1990's and extended its focus to urban environments at the start of the new millennium. Here, we discuss some of the most influential work on procedural terrain generation. For a more in-depth survey on these procedural methods, see [18].

Automatic generation methods for geo-typical elevation maps are often based on fractal noise generators. Examples include Perlin noise [12, 13], which generates noise by sampling and interpolating points in a grid of random vectors. Rescaling and adding several levels of noise to each point in an elevation map results in natural, mountainous-like structures. Further transformations on elevation maps include simulations of physical phenomena, such as erosion. Thermal erosion diminishes sharp changes in elevation, by iteratively distributing material from higher to lower points, until the talus angle, i.e. maximum angle of stability for a material such as rock or sand, is reached. Erosion caused by rainfall (fluvial erosion) can be simulated using a similar, but more complicated, algorithm, taking into account rainfall, water evaporation and the amount and velocity of water (containing dissolved material) that flows out to lower points (see, for example, Musgrave's PhD thesis [9]).

Except for rivers, procedural water bodies, such as oceans and lakes and their connections, stream networks, deltas and waterfalls, have received little attention to date. Belhadj and Audibert [1] present a combined algorithm for elevation maps with mountain ranges and river networks. They generate a mountain range by placing Gaussian curves along a path. Next, they place river particles along the top of the mountain range and let them flow downwards according to simple physics, creating a river stream network in the valley.

For vegetation, there are procedures for generating 3D tree and plant models and methods for automatic placement of vegetation on a terrain model. Prusinkiewicz and Lindenmayer [15] discuss the use of the L-system, an often used grammar rewriting system, for procedural plant models. Starting from the root, these grammars grow a plant by adding increasingly smaller branches and ending with the leaves.

Deussen et al. [4] describe an ecosystem simulation model to populate an area with vegetation. The input of this simulation model is the elevation map and a water map, several ecological properties of plant species, such as rate of growth, and, optionally, an initial distribution of plants. Based on this and taking into account rules for competition for soil, sunlight and water, a distribution of plants inside an area is iteratively determined.

Procedural modeling of urban environments begins with generating a suitable network of roads and streets. One method for this, is using templates, as proposed by Sun et al. [19]. They reconstruct several patterns frequently found in real road networks, using a corresponding template for each pattern: a population-based template (implemented as the Voronoi diagram of population centers), a raster (e.g. a modern North American city) and radial (e.g. European city core) template, or a mixed template. Parish and Müller [11] use an extended L-system to let a road network grow. The L-system roads try to connect population centers and form specific road patterns, such as the ones discussed above. Their L-system is extended with rules that prefer to connect new proposed roads to existing intersections and rules that check road validity, considering impassable terrain and slope constraints. Streets are inserted into the remaining areas as grids.

Buildings are to be placed in the polygonal regions enclosed by streets. Subdivision of these regions results in lots, for which different subdivision methods exist, see e.g. [7]. The lot shape is used as the basis of the footprint of a building. By simply extruding the footprint to a random height, one can generate a city of skyscrapers and office buildings. To obtain more varied building shapes, Müller et al. [8] apply shape grammars. They start with a union of several volumetric shapes which defines the boundary of the building. This shape is then divided into floors and the resulting facades are subdivided into walls, windows, and doors, following the production rules of the grammar system.

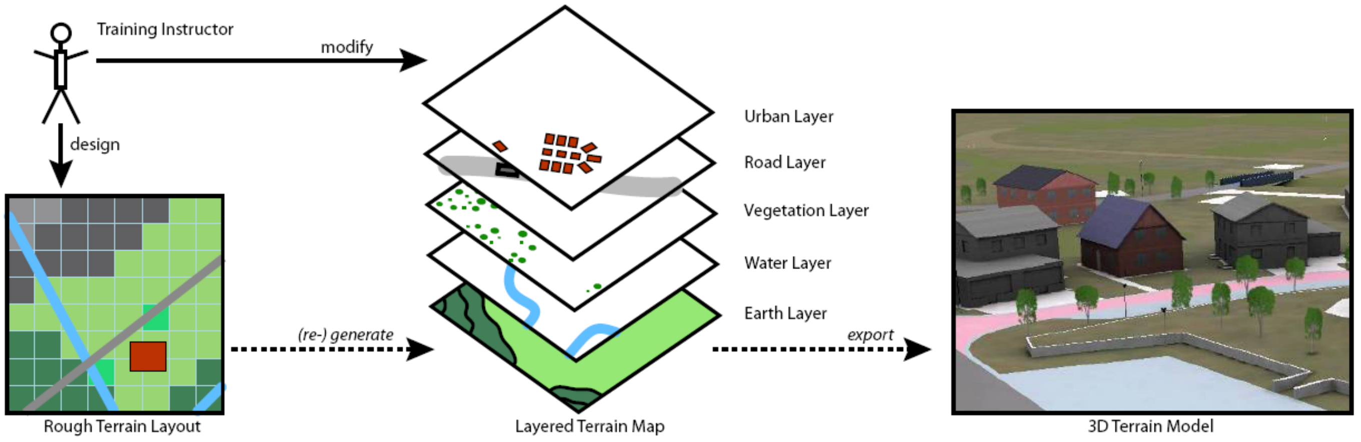


Fig. 1: Overview of phases in the declarative terrain modeling workflow.

TERRAIN MODELING APPROACH

We propose a new approach for modeling geo-typical terrain. Our intent is not only to significantly speed up the terrain modeling process, but, more importantly, to provide a quick way for people without special modeling expertise to create terrain models that meet their requirements. We believe that for this goal a declarative approach is best suited. This declarative terrain modeling approach (focusing on "what do I want?") is essentially different from the common constructive approach (focusing on "how do I model it?"). It is ideally suited for training simulations, in which often the scenario designers are end users, such as instructors, and not experienced 3D modelers.

We have developed a modeling framework to support this declarative approach. In previous publications, we have identified key requirements for this framework and shown its potential for military training games [17, 16]. Instructors have an idea for a particular terrain that fits their training scenario. Our framework allows them to express this idea using a sketch interface; it then automatically creates the terrain model accordingly. The framework thus lets instructors focus on declaring the terrain they need, without bothering them with 3D modeling tasks or difficult tuning of parameters of generation algorithms.

Sketch-based Modeling

The typical modeling workflow in our framework is as follows (see Fig. 1). Users compose a digital sketch of the rough layout of the terrain. The basic outline of a terrain is declared by specifying which ecotopes occur where. An ecotope describes both the type of terrain, e.g. a specific type of desert, hills or mountains, and the associated range of elevation. This part of the sketch is drawn on an ecotope raster, a regular grid of small cells, with each cell representing an area of e.g. one or two hundred meters square. Each ecotope has its unique color; therefore drawing the raster is quite similar to painting a small pixel bitmap. Next, users declare the location of important terrain features, such as forests, major roads, rivers and urban areas. These are drawn as vector elements, similar to the point, polyline and polygon area shapes in ESRI Shapefiles. If desired, for each polygonal sketch element, a small set of attributes can be set (e.g. city size, average river width, etc.).

Once the users are satisfied with the rough terrain layout, the framework generates a high-resolution terrain map that complies with the specified features at large, but has, on a small scale, a high level of detail and variations in elevation, vegetation, etc. One can view the resulting terrain in 3D and modify the rough layout where desired. Afterwards, the terrain model can be automatically exported to data formats and models relevant to training simulators.

Terrain Generation Procedure

The generated terrain map is structured in several logical layers, both natural layers and man-made; see Fig. 1. Using different terrain layers improves the adaptability of the terrain, because changes to one layer do not necessarily have to affect other layers. We distinguish five layers in the terrain map, stacked as follows:

- Urban layer: cities, towns, housing, airports, factories;
- Road layer: highways, local roads and streets, bridges;
- Vegetation layer: forests, bushes, trees;
- Water layer: rivers, lakes, oceans;
- Earth layer: elevation data and soil information.

Although the layers are kept separately in this editing phase, many layers have interdependencies. To generate a consistent terrain, the generation process of the layers is ordered in such a way that a layer can take into account features of other layers. For example, generating plants and trees for the Vegetation layer takes into account the proximity of rivers in the Water layer and the properties of soil and elevation in the

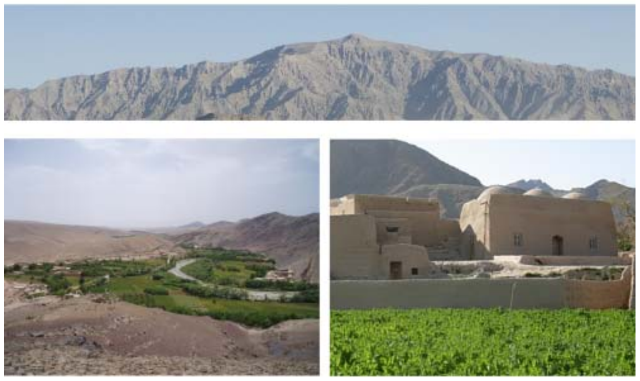


Fig. 2: Southern Afghanistan features: mountain ridges, wide rivers with green zone, farm (quala).

Earth layer. The major roads generation method for the Road layer will have road sketch elements but also the previously generated Earth layer as input, to be able to determine where valid roads can be placed, e.g. not too steep ascending roads.

Still, to obtain a fully consistent and valid terrain, an integration phase is necessary after the generation process. This includes local corrections (e.g. flattening terrain before placing a building), significant modifications (e.g. carving a road embankment through a mountain range), and more complex changes (e.g. introducing bridges to river crossing roads). The framework is responsible for integrating all features correctly in the base terrain, and detecting and resolving any inconsistencies. By maintaining the terrain consistency in this way, exporting the layered terrain model to, for instance, a 3D model is a straightforward automated process.

Creating a detailed terrain map based on the rough layout of the terrain is a form of data amplification, i.e. automatically expanding a small dataset into a large one. Amongst the most used data amplification algorithms are procedural content generation methods, discussed earlier. We are using combinations of existing procedural methods, which have been tuned to work well together, to expand sketch elements to terrain layers.

Framework Implementation Details

Our framework is implemented in C# / C++ .NET, with 3D visualization in OpenSceneGraph. To realize rapid terrain generation, a large part of our generation and merging process is performed in parallel on the GPU using NVidia's CUDA [10], a C-like programming language for GPU computing. With CUDA, one can launch a large set of light-weight threads on the GPU, performing the same computation task (a C program called a *kernel*) in parallel. It is similar to, but far more flexible than, shader-based GPU computing.

Currently, the framework can export a 3D terrain model as a paged terrain database in OpenSceneGraph's file format, and export the terrain as GIS raster and vector data, which can be used for further processing in e.g. TerraVista [14] or VBS2's Visitor [2].

SOUTHERN AFGHANISTAN FEATURES

The reference material for our case study has been, for the most part, maps and photographs of the southern Afghanistan area. From this we took interesting terrain features to recreate. See Fig. 2 for some example photographs of features typical to this area.

Afghanistan has a striking elevation profile. It is characterized by large mountain ridges, wide valleys and sharp changes in elevation.

Another noticeable feature are rivers and their influence on the surrounding area, resulting in a green, very dense and varied vegetation zone, which sharply contrasts with surrounding dirt terrain. The water level of a river ranges from completely dry in some months, making the river passable, to high, claiming nearby land and increasing the importance of available bridges.

Southern Afghanistan cities often have a structured center and more informal settlements, farms and farmland near the outskirts of the city. The road network is mostly structured and paved near the center and is less structured with dirt roads towards the sprawling outskirts. The typical style of farm housing is one-storey loam buildings, houses are almost always circumvented with a solid, defensive wall.

NATURAL ENVIRONMENT

The following sections describe how we reproduce the typical features found in the natural environment of Afghanistan: mountain ridges, rivers and green zones. We also discuss seasonal influences on the natural environment.

Mountain Ridges

The impressive and distinctive shapes of mountain ridges in Afghanistan were formed during centuries of erosion. To generate these features, an erosion simulation algorithm could probably be implemented that delivers satisfying results. However, the anticipated running time of such an iterative algorithm is too long for our type of application, in which an instructor would like to quickly evaluate the results of his sketch input. Therefore we have devised a more efficient shape imitation algorithm that delivers adequate results; see Fig. 3.

The design of the shape imitation algorithm is as follows. Starting from a vector line sketch element, we generate a skeleton of the ridge. We convert the vector line into a spline using Catmull-Rom interpolation (see [3]). We sample a number of control points at this spline to obtain a set of skeleton segments. At each control point we elevate

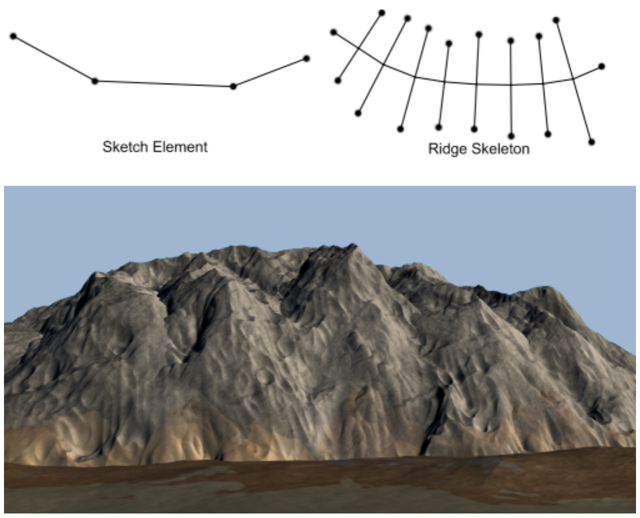


Fig. 3: Mountain ridges: sketch element, generated skeleton of the ridge, screenshot of resulting ridge.

the point to a controlled random height and place two perpendicular lines (of a length relative to the control point's elevation). This results in a skeleton for the mountain ridge.

These skeletons are the input to our earth layer generation algorithm, described in [16]. This algorithm uses noise-based perturbation (random offsets in the x-y direction) and noise scaling (affects the elevation profile) to generate the mountain ridge shape. It has been implemented using CUDA, which makes it fast enough for interactive terrain development. The algorithm results in mountain ridges that have a natural and convincing eroded look, see Fig. 3.

Rivers and the Green Zone

A river flows from high altitude terrain, such as mountains, to the lower valleys. A steep local slope results in water flowing at high velocity, resulting in compact streams, while on relatively flat terrain, the river spreads out wider and meanders around.

A river is procedurally generated in the following way. Starting from a simple line sketch element, the river generation algorithm plots a path through the terrain, following the steepest local slope downwards. By limiting the spread of the points considered in the path planning, we can ensure that the plotted path stays on track with the line sketch element. If the local elevation profile does not allow a descent, the river carves through the terrain. This is implemented by subtracting a 3D river spline, which follows the path and has a desired elevation profile, from the Earth layer. This way, our rivers are not limited to flat surfaces, but can run down hills and mountains.

A main property of the river sketch element is the river width. The actual 3D river spline varies around this base width, depending on the local slope. To model the typical rivers in Afghanistan, with dry beds within the river, we generate a noise-based elevation profile within the river. This is performed during the merge of the river and the terrain, and is implemented in a CUDA kernel. See Fig. 4 for the resulting naturally looking river with dry beds and streams.

We model the green zone by computing a river bank polygon on both sides of the river, based on the river's course and local slope and width, and some random offset. Within this polygon, the river results into a green zone of vegetation. The underlying earth layer is modified accordingly from sands and stones to grassy terrain, and an iterative vegetation simulation is run to populate the banks with trees (see [16] for details on this algorithm, which was based on [4]).

The water level of the river ranges from completely dry in some months, making the river passable, to high, claiming nearby land and increasing the importance of available bridges. We implemented the seasonal influence on both the water level of the river and the vegetation models in the green zone. By changing the season parameter, users can obtain a 3D model of the terrain in a different season. See Fig. 4 for 3D models of a river and vegetation zone during different seasons.

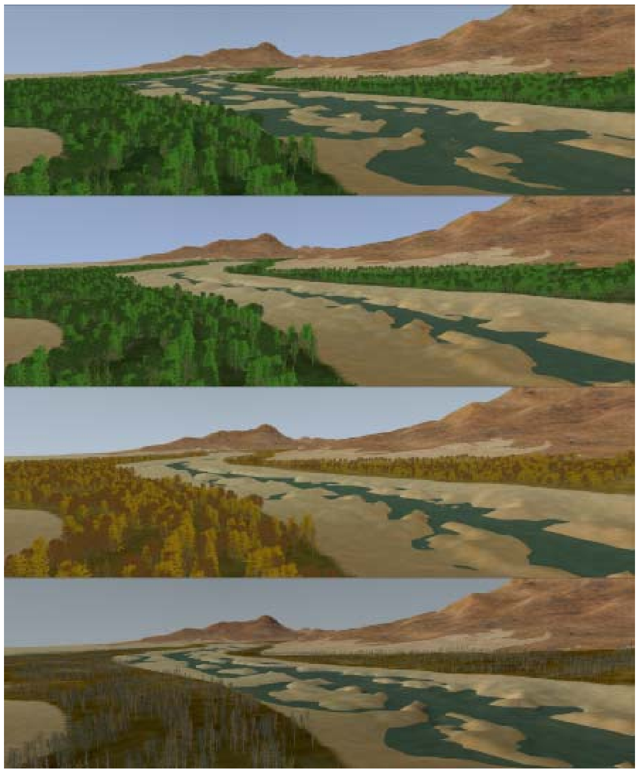


Fig. 4: A river with green zones on both banks in the seasons spring, summer, autumn and winter, showing the variation in water level and vegetation.

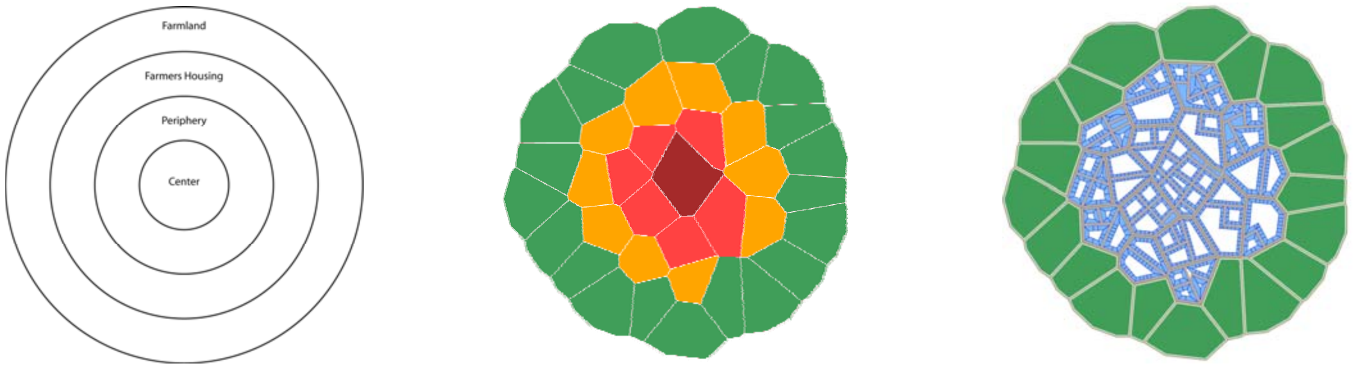


Fig. 5: Southern Afghanistan city structure: concentric zone model, generated district distribution, road patterns and lots per city cell.

URBAN AREA

Next, we treat the typical man-made features: the structure of cities, the generation of a road network and integration of the roads with the terrain layers, and building models.

City Structure Model

Procedural models of cities, towns and villages often lack a believable structure, offering little more than random buildings placed along random roads. In [6], we presented a method for generating a subdivision of a city into districts, according to existing urban land use models for North-American and West-European cities.

Unfortunately, for the small cities in southern Afghanistan, no such scientific model exists. Therefore, based on our work in [6], we derived a scaled-down model of these cities. It represents the city structure, consisting of four concentric zones (see Fig. 5):

1. Center: multi-storey, solid stone buildings, shops, regular road patterns;
2. Periphery: single or multi-storey buildings, sometimes circumvented by high walls, radial road patterns;
3. Farmers housing: loam houses, circumvented by walls, several small sheds per lot, irregular roads;
4. Farm lands: fields separated by informal roads.

Each zone is subdivided in a number of districts. Based on the declared village size, i.e. the diameter attribute of the village sketch element, we determine the total number of districts to be placed. We allocate districts to the zones using a rule-based distribution template. Next, we place the center points of the districts in the concentric circles of the zones, at random positions but with somewhat evenly spacing within a zone (to avoid very small or very large resulting districts).

From the center points, we need to obtain the district boundaries as a polygon. A Voronoi graph (see [21]) is a natural choice here. To make the boundaries somewhat more natural, we generate far more district centers than needed, and merge Voronoi cells based on criteria related to the perimeter and the shape of the cell (e.g. no triangle shapes). To obtain a natural bound of the village, we place a ring of "ghost" districts at the village bounds, and remove all those districts after obtaining the Voronoi graph. Fig. 5 shows an example village district distribution.

Road Network

The above district distribution results in a coarse layout of the urban environment. To refine this, we generate a road network graph. We discern three types of roads in our framework:

1. Primary Roads: roads that provide connections between different cities or villages;
2. Secondary Roads: major roads within urban environments;
3. Tertiary Roads: streets that run between housing blocks.

Primary roads are generated from line sketch elements. The sketched line points are input for a path finding algorithm (that was based on [7]) which plots a more detailed path through the generated Earth layer, while trying to preserve an even change in elevation along the route. When a primary road intersects a river, the road is split and a bridge is inserted, perpendicular to the river's heading; see Fig. 6.

The plotted points are converted to a 3D spline to obtain the exact shape of the road. The road's footprint is merged into the terrain in a CUDA kernel. For the verge on both sides of the road a transition zone is established to avoid unrealistically sharp changes in elevation. Here the elevation is adjusted by interpolating between the elevation of the road spline and the current elevation of the terrain. This creates a naturally smooth transition between the road and the surrounding landscape.

The purpose of secondary roads is to provide connections among urban districts. Secondary roads are generated by simply taking the boundary polygons of the

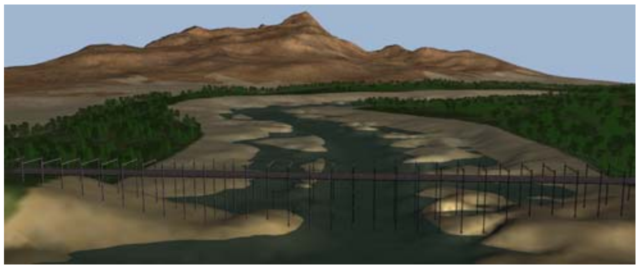


Fig. 6: Roads: Bridge generation and integration into road network.

districts and converting them to road splines. At the intersections, the geometry of the road's end is converted in order to fit with all connecting other roads. The road's footprint is merged into the terrain in a CUDA kernel, but no transition zone is used, as these roads normally have less impact on the surrounding terrain.

Tertiary roads connect housing blocks within a district. They are generated based on patterns (e.g. grid, radial, mixed), which follow from the type of zone in which the district lies. The algorithm iteratively tries to place new roads, starting in perpendicular directions from the largest secondary roads in a district. For each proposed road, validity constraints are checked: the road length should be within some range and a minimum is also defined for the angle between this road and a connecting road. If the new road does not adhere to these constraints, it is discarded. Next, the road endpoint is examined. If the endpoint is near an existing intersection, it is snapped to this intersection. If the new road itself intersects another road, an intersection is created, which becomes the road's new end point. (After moving an end point all road validity checks have to be reevaluated.) A valid road is added to the network and, from its endpoint, in turn new roads are proposed in directions depending on the desired road pattern.

This process results in realistically looking road networks that fit the 3D terrain as well as present village structures. Fig. 5 shows the road network generated for an example Afghan village.

Buildings

Once the road network of a village is constructed, a graph algorithm that finds all cycles in the road network graph can be applied to obtain all the areas enclosed by roads and streets. These areas, which we call *city cells*, are destined to be built upon. For this, we need to subdivide the available land in *building lots*.

There are several approaches to determine a set of lots in a polygonal city cell. Often used is the subdivision approach (e.g. [11]). Lots are obtained by iteratively subdividing the cell into two, using a cutting line from the middle of and perpendicular to the largest edge in the polygon, until a certain minimum size of the polygons is reached. With some random offsets, a varied collection of

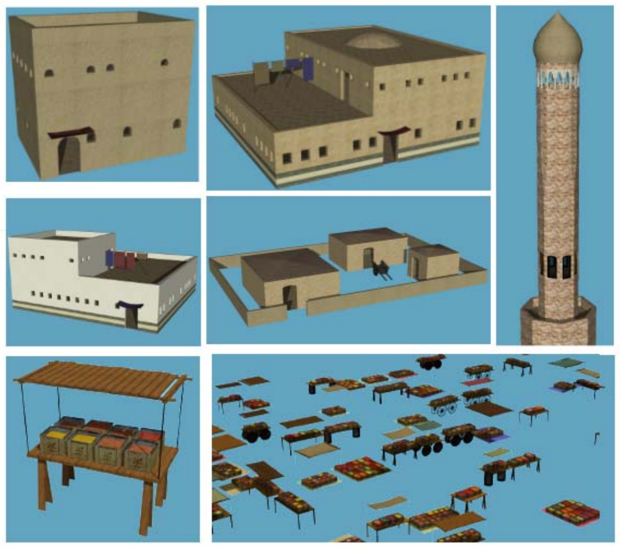


Fig. 7: Generated building models for the city center, periphery and farm lands. The generation procedure can also generate differently shaped models, such as the minaret and the market stalls shown.

building lots can be acquired. However, this method works best on convex, rectangular-like, polygons. For concave and irregular city cells, the results are often undesirable (unsuitable lot shapes) or the algorithm fails altogether.

A solution for this could be to, as a preprocessing step, split any concave or irregular polygon into several convex and rectangular-like polygons. However, as we prefer to have a method that works for both convex and concave shapes, we have designed a different approach for lot determination: we create lots perpendicular to each street connected to the cell. Each edge of the cell is offset inwards by a configurable size (depending on the type of buildings to be placed in the cell). For most city cells with some edges situated close to each other, offsetting the edges results in errors in the polygon: overlapping or intersecting lines, invalid polygons, etc. A fixing algorithm removes all these errors and results in the inner polygon, containing the area of the city cell where no buildings will be placed. Next, based on the city cell and

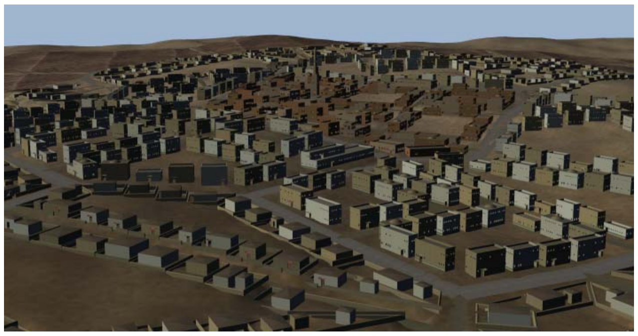


Fig. 8: View on the generated 3D model of the city in Fig. 5.

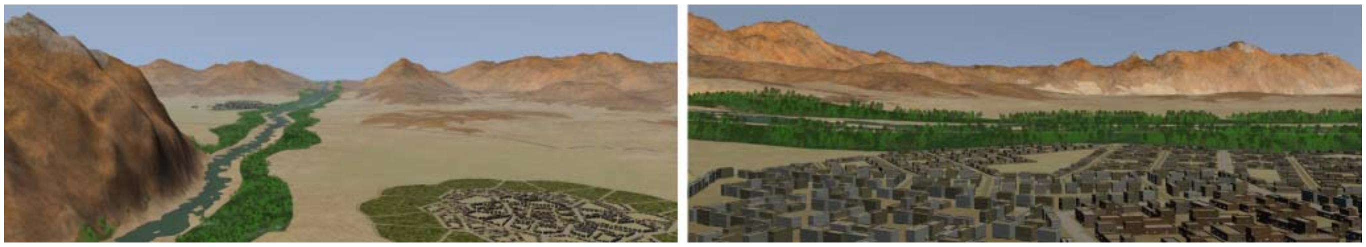


Fig. 9: Case study results: a generated terrain model with two cities, connected via road over a river, surrounded by mountain ridges.

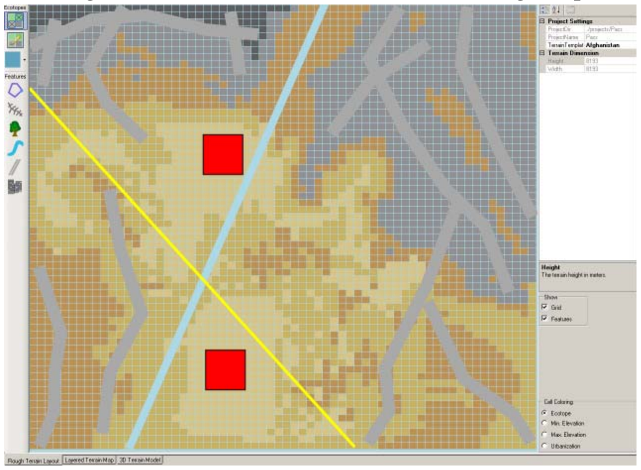
inner polygon, lots are constructed by connecting the inner polygon edges at a regular interval with the related cell's edges; see Fig 5.

This method has the advantage that it handles any city cell shape, while still delivering mostly rectangular lots. A possible disadvantage could be that lots are uniformly shaped along a road.

Once we have a set of lots, we can extract footprints of buildings to be built on these lots. Rectangular boxes are fitted within the lot and combined, resulting in a 2D footprint that can be extruded to a certain height. Depending on the building template, another storey is added with either the same shape (village center) or a smaller shape (periphery). The buildings resulting from these two templates vary in shape, size and appearance. For farm housing, the approach is slightly different. Besides the main farm, several smaller utility buildings are placed. To find the positions of these buildings on the lot, we use a solving system; see [20].

We now have a basic geometric representation of the buildings. Several rules are applied to give the houses more details: windows are placed along the walls, to reach the second floor a staircase is added, doors are placed, oriented towards the nearby road, and, if applicable, the lot is circumvented with a loam wall.

Fig. 7 shows instances of the three building templates



Project Settings
ProjectDir: ./projects/Pacs
ProjectName: Pacs
TerrainTemplat: Afghanistan
Terrain Dimension
Height: 8193
Width: 8193

Height
The terrain height in meters.

Show
[✓ Grid]
[✓ Features]

Cell Coloring
[✓ Ecotope] Min. Elevation Max. Elevation Urbanization

Fig. 10: Rough Terrain Layout: Sketching the terrain model shown in Fig. 9.

defined for the district types within an Afghan village. Further improvements could be interior layout generation, as described in [20], or urban clutter generation, as shown in [5]. Fig. 8 shows a section a generated city.

RESULTS

All procedures that generate Southern Afghanistan terrain features have been integrated in our terrain modeling framework. Shown in Fig. 10 is the sketching interface on which one can draw ecotopes and terrain features. The example sketch involves two villages (red icons), connected by a major road (yellow line) that crosses a river (blue line). The terrain is enclosed by mountain ridges (gray lines). Fig. 9 shows two panorama views of the generated terrain model, the first one looking from the south city towards the village in the north, the second one looking from the northern city to the east. Visible in both views is the green zone near the river and the enclosing ridges.

The automated process from sketch to a paged OpenSceneGraph terrain database typically takes a couple of minutes in our current implementation. To illustrate the performance of the different steps, the table below shows the running time of the procedural generation processes per layer and the geometry creation for the (8 km. by 8 km.) terrain shown in Fig. 9 and Fig. 10.

Process	Layer Generation	Geometry Creation
Earth	14.39 s.	142.87 s.
Water	0.05 s.	0.04 s.
Vegetation	8.32 s.	4.80 s.
Road	0.28 s.	2.80 s.
Urban	43.65 s.	80.80 s.
Merge Layers	5.72 s.	-

CONCLUSIONS

Producing appropriate terrain models is crucial for the effectiveness of military training scenarios, but non-expert terrain modelers are seriously hindered by the difficulty in use of current tools and methods. We presented a declarative modeling framework that overcomes this problem by offering an intuitive and fast way for non-expert users to model geo-typical terrain.

The potential of this framework is illustrated in a case study on generating terrain that resembles the typical natural and man-made terrain elements found in southern Afghanistan. This case study includes defining procedural methods for generating mountain ridges, rivers and green zone and small cities with fitting building models, and integrating these into our sketch-based modeling framework. The study has resulted in convincing terrain models and has shown that, using our framework, it is possible to:

- Quickly develop new, custom procedural methods for generating certain geo-typical features;
- Rapidly generate complete and consistent geo-typical terrain models with these features.

Future improvements include increasing the amount of variety and level of detail of our generated terrain features, especially in the urban environments, in which building interiors and urban clutter could very much enhance the immersion and believability. We will also continue to explore the use of terrain semantics, such as seasonal influences discussed here. A continuing challenge will be defining methods for maintaining consistency when integrating terrain layers.

This case study illustrates that our framework is a firm step towards a declarative, automated terrain modeling process.

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).

AUTHOR BIOGRAPHIES

Ruben Smelik is a researcher at the Modelling, Simulation and Gaming department of TNO Defence, Security and Safety in The Netherlands. He is a PhD candidate, on a project entitled “Automatic Creation of Virtual Worlds” in close cooperation with Delft University of Technology. This project aims at developing new tools and techniques for creating geo-typical virtual worlds for serious games and simulations. It focuses on the specification and maintenance of virtual world semantics, procedural modeling of virtual worlds and runtime adaptive environments. The project 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. His research interests include computer graphics, natural environment modeling and military simulations.

Klaas Jan de Kraker is a member of the scientific staff at TNO Defence, Security and Safety. He holds a Ph.D. in Computer Science from Delft University of Technology. He has a background in Computer-Aided Design and Manufacturing, collaboration applications, software engineering (methodologies), meta-modeling and data modeling. Currently he is leading various simulation projects in the areas of simulation based performance assessment, collective mission simulation, multifunctional simulation and serious gaming.

Tim Tutenel is a PhD researcher in the Game Technology group of the Computer Science department of Delft University of Technology. In his PhD project, “Semantics in Game Worlds”, he develops generic methods for specifying and maintaining semantics in both the design phase of game worlds and in game play. In particular, he is researching how semantics can improve procedural generation techniques. The project is part of the Dutch research program “Game Research for Training and Entertainment (GATE)”. Tim got his master’s degree in Computer Science at the Hasselt University in Belgium. His research interests include game world semantics, emergent game play and procedural content generation.

Rafael Bidarra is associate professor Game Technology at the Computer Graphics and CAD/CAM Group of Delft University of Technology, The Netherlands. He graduated in electronics engineering at the University of Coimbra, Portugal and received his PhD in computer science from Delft University of Technology. His current research interests include: procedural and semantic modeling techniques for the specification and generation of virtual worlds and game play, serious gaming, semantics of navigation, and interpretation mechanisms for in-game data.

REFERENCES

[1] Belhadj, F. and Audibert, P. (2005). Modeling Landscapes with Ridges and Rivers: Bottom Up Approach. In *GRAPHITE '05: Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 447 - 450, New York, NY, USA. ACM.

[2] Bohemia Interactive Australia (2009). *Virtual Battlespace 2*. Available from http://www.vbs2.com

[3] Catmull, E., and Rom, R. (1974). A class of local interpolating splines. In *Computer Aided Geometric Design*, pages 317–326, New York, NY, USA. Academic Press.

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

[5] Giuliani, J. L., LaDieu, J., and McKeown, D. M. (2008). Parametric Generation of Street Level Details for Urban Visualization. In *Proceedings of the IMAGE 2008 Conference*, pages 104 - 116, St. Louis, Missouri, USA. IMAGE Society.

[6] Groenewegen, S. A., Smelik, R. M., de Kraker, K. J., and Bidarra, R. (2009). Procedural City Layout Generation Based On Urban Land Use Models. In *Short Paper Proceedings of Eurographics 2009*, Munich, Germany. Eurographics Association.

[7] Kelly, G. and McCabe, H. (2007). 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.

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

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

[10] NVIDIA Corporation (2008). *NVIDIA CUDA Compute Unifed Device Architecture Programming Guide 2.0*.

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

[12] Perlin, K. (1985). An Image Synthesizer. In *SIGGRAPH '85: Proceedings of the 12st Annual Conference on Computer Graphics and Interactive Techniques*, volume 19, pages 287 – 296, New York, NY, USA. ACM.

[13] Perlin, K. (2002). Improving Noise. In *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 681- 682, New York, NY, USA. ACM.

[14] Presagis (2009). *TerraVista*. Available from http://www.presagis.com/products/content_creation/details/terra_vista

[15] Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, NY, USA.

[16] Smelik, R., de Kraker, K. J., Tutenel, T., and Bidarra, R. (2009). Declarative Terrain Modeling for Military Training Games. *Submitted for publication*.

[17] Smelik, R., Tutenel, T., de Kraker, K. J., and Bidarra, R. (2008). A Proposal for a Procedural Terrain Modelling Framework. In *Poster Proceedings of the 14th Eurographics Symposium on Virtual Environments EGVE08*, Eindhoven, The Netherlands.

[18] Smelik, R. M., de Kraker, K. J., Tutenel, T., Bidarra, R., and Groenewegen, S. A. (2009). 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.

[19] Sun, J., Yu, X., Baciu, G., and Green, M. (2002). Template-based Generation of Road Networks for Virtual City Modeling. In *VRST '02: Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 33 - 40, New York, NY, USA. ACM.

[20] Tutenel, T., Bidarra, R., Smelik, R. M., and de Kraker, K. J. (2009). Rule-based Layout Solving and its Application to Procedural Interior Generation. In *Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*, Amsterdam, The Netherlands.

[21] G.F. Voronoi (1908). Nouvelles applications des paramètres continus à la théorie de formes quadratiques. In *Journal für die reine und angewandte Mathematik*.134: 198–287.