Declarative Terrain Modeling for Military Training Games

R.M. Smelik
TNO Defence, Security and Safety, the Netherlands

T. Tutenel
Delft University of Technology, the Netherlands

K.J. de Kraker
TNO Defence, Security and Safety, the Netherlands

R. Bidarra
Delft University of Technology, the Netherlands

Military training instructors increasingly often employ computer games to train soldiers in all sorts of skills and tactics. One of the difficulties instructors face when using games as a training tool is the creation of suitable content, including scenarios, entities and corresponding terrain models. Terrain plays a key role in many military training games, as for example in our case game Tactical Air Defense. 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 still lack user control and intuitive editing capabilities. We present a novel way for instructors to model terrain for their training games: instead of constructing a terrain model using complex modeling tools, instructors can declare the required properties of their terrain using an advanced sketching interface. Our framework integrates terrain generation methods and manages dependencies between terrain features in order to automatically create a complete 3D terrain model that matches the sketch. With our framework, instructors can easily design a large variety of terrain models that meet their training requirements.

Keywords: *3D environments; military training; procedural content generation; serious games; sketching interface; terrain modeling*

1. INTRODUCTION

3D computer games have grown tremendously in size, detail and visual realism. Game technology has matured rapidly and is nowadays frequently used outside the entertainment domain in so-called serious games. One important application area for serious games is training and instruction. Military instructors use games as a tool for training their personnel, because games provide a visually realistic, immersive training environment and are very affordable.

There are several examples where games are successfully applied to military training. A popular example is the military training system Virtual Battlespace 2 [1], which is based on the commercial game Armed Assault and is used by, among others, the US, UK, Australian and Dutch army for training soldiers in basic infantry tactics. Steel Beasts [2] is an armored vehicle simulation game that has been employed for years to train tactical vehicle movement and combat. Tactical Iraqi [3] is a game that teaches soldiers to interact with Iraqi people in their language and following their cultural manners. Although more a recruitment tool than purely a training game, America's Army [4] is one of the classic examples of serious games.

We examine the training game Tactical Air Defense, and discuss the importance of terrain in this and other military training scenarios. We show that common methods and tools for terrain modeling are not suitable for end-users of training games (e.g. training instructors), and describe how this hinders the effectiveness of game-based training. Next, we present our solution for this: a framework for declarative, automated terrain modeling. User can declare a terrain using an easy to use sketch-based interface and the framework automatically generates a matching terrain model. We explain how we use and adapt existing procedural methods for generating terrain. An example training scenario illustrates how the declarative approach of our framework simplifies terrain modeling. Lastly, we discuss some challenges ahead for improving this framework.

2. THE ROLE OF TERRAIN IN MILITARY TRAINING GAMES

We examine the role of terrain in military training games by analyzing Tactical Air Defense, an illustrative example of a serious game for training air defense personnel. We give the relevant background on the case and discuss how instructors define scenarios in this game. The

FIGURE 1: a) Vehicle with mounted Stinger missile system. b) Tactical Air Defense map view with sight diagrams of Stinger teams (colored circles) and radar coverage (surrounding polygon).

features of the terrain turn out to be a key factor in air defense scenarios.

2.1 Military training case: Tactical Air Defense

In a tactical air defense scenario, a platoon commander has the challenging task of setting up a ground-based defense system against threats from the air. He is responsible for protecting a zone of terrain and probably some high-value objects, for instance a city, an airfield, or an oil refinery, within that zone. At his disposal are a number of mobile anti-air teams, such as Stinger teams. A Stinger team consists of infantry units with a shoulder launched Stinger missile system or, in most cases, mounted on a light or armored vehicle (see Figure 1a). The commander plans the deployment of each Stinger team in his zone. He has to consider many variables, factors, and uncertainties, but in all considerations the terrain and its features play a major role.

To defend against the air threat, the commander first estimates possible approach routes the enemy aircraft can take. Although he may have some additional intelligence information (e.g. "four jet fighters are flying in from the south"), the estimation is, for the most part, based on a map of the terrain. The presence of terrain features such as valleys, rivers, roads, forests and villages may give clues about the approach route the enemy pilots will take, as these features can provide cover for the attacking aircraft or can be used by pilots for visual orientation.

Once the commander has a clear estimation of the route the enemy will most likely take, he sets up his air defense system accordingly, while considering a number of criteria:

1. The covered *depth* of the air defense. A Stinger team can fire twice before needing a long reload operation (approximately 10 minutes). If some of the aircraft made it through the first line of defense, ideally there should be other lines taking over.
2. The *sight* a Stinger team has at its position, which is often limited by terrain features such as hills or forests.
3. The effective *range* of the anti-air weapon. A Stinger has a maximum firing range in which the missile is effective, but it also has a minimum distance the missile has to travel before it can lock on the target.
4. The physical *reachability* of the position by the team, which depends on the type of vehicle and the accessibility of the terrain, and there may be time constraints as well.
5. Considerations for the *safety* of the Stinger team, such as cover, camouflage and visibility from the air.
6. The *overlap* in cover of the Stinger teams. Defended areas should ideally be covered by more than one team.
7. The *strength* of the cover will vary in the zone; high-value areas should be defended better than other areas.
8. Whether the Stinger team will be in range of the *radar* support network it needs to track targets.

In practice, an optimal solution cannot be found for all these variables, so a commander has to prioritize areas in his zone and the above criteria. His decision is based on experience and practice.

Commanding an air defense platoon requires specialized training. To be able to set up an effective air defense, an aspirant commander needs to acquire tactical insights, experience and flexibility in coping with varying threats and circumstances. A Job Oriented Training (JOT) approach has proven to be very successful here. JOT promotes active learning; knowledge comes

as a result of doing the job in a training environment followed by extensive peer evaluation of the performance, mistakes, lessons learned and alternative solutions [5].

Training game

To support the JOT curriculum, the Netherlands Organisation for Applied Scientific Research TNO has developed a serious game for air defense. The game focus is on learning to set up an air defense for different circumstances, terrain, and types of threats, i.e. how to position Stinger teams in such a way that together they form an effective defense against possible air threats.

A typical training session starts with a realistic briefing by a military instructor, which includes the zone of terrain the trainees are to defend, situation and intelligence reports and the identification of high-value targets. Trainees work in pairs and start to examine the situation and terrain on a paper map. They assess probable approach routes the enemy aircraft might take and the intent of the enemy (e.g. destroy a high-value building). The trainees use the game to evaluate and adjust their ideas based on the 3D view of the terrain and discuss and compare their opinions with peers.

In the next phase of the training session, each team deploys an air defense setup using the game. The game adopts the paradigm of learning by doing; once a trainee places a Stinger team on the terrain map, a diagram of the lines of sight from that position is immediately displayed. This sight diagram is determined on the basis of a realistic Stinger model, and takes the terrain and its features into account. The sight diagrams allow the trainees to discuss the strengths and weaknesses of their deployment and adjust it accordingly (see Figure 1b). With their defense set up, the game can show how well the team performed, by simulating the enemy air attack in the 3D simulation. The Stinger teams automatically fire on enemy aircraft within their effective range and results of weapon interactions are calculated based on the Stinger and aircraft models. At the end of a session, the teams present their air defense solution to the group and discuss alternatives.

Scenario creation

The instructor creates scenarios by selecting and configuring the terrain zone, the air defense goal, air threat type and approach route, and the available Stinger teams. The scenarios vary with terrain complexity and threat level.

Currently, the game offers a fixed set of large terrain models. This includes both geo-specific (based on real-world GIS data) and geo-typical (lifelike but fictive) terrain models. Geo-typical terrain has the advantage that it can be tailored to exactly meet the training goals. For each scenario a suitable terrain can composed. For example, a geo-typical terrain can combine highly varying terrain types in one model (e.g. a mountain range with a forest, a flat desert, a valley with a village and a river) or it can have a height profile that matches the exact line-of-sight requirements of the scenario. Geo-specific terrain allows the trainees to do a live training in the same field as they did their virtual training, which can teach them how their decisions work out in the real world.

2.2 The importance of terrain

In Tactical Air Defense and in other games for training military personnel, the terrain in which a scenario is executed plays a key role in the training. On a strategic level, securing particular areas or features of a terrain (e.g. a hill overlooking a city, a bridge across a river) can be an objective of a military scenario. Large terrain features can affect the performance of sensors, such as the mobile radar systems used for air defense; therefore to place these sensors, one should consider hindering terrain features such as hills and mountains within the sensor's range.

On a smaller scale, the terrain largely defines the tactical situation. Together with environmental conditions (season, weather, time-of-day) it determines visibility and lines-of-sight. These are key factors in, for instance, first-person infantry trainers such as Virtual Battlespace 2. Natural terrain features such as rocks can also provide cover for infantry.

Wells [6] argues that terrain features not only have strategic and tactical value, but are also essential for trainees to be able to orientate, navigate and immerse themselves in a virtual environment. Features of well-known size such as trees and buildings give important cues for sizes and distances in the virtual world. Recognizable natural or man-made landmarks such as rivers or a church help trainees to navigate. A terrain model that is detailed and rich in natural and man-made features is perceived as more lifelike, thus increasing the immersion of trainees into the virtual world.

3. CURRENT TERRAIN MODELING METHODS

In Tactical Air Defense, the choice of terrain turns out to be an important part of creating a training scenario. Scenario creation usually starts with obtaining a suitable terrain model. However, the instructor currently has to choose from the fixed set of terrain models that were shipped with the game. Although they can cover many possible settings, they do significantly limit the number of potential training scenarios. The instructor cannot create new terrain, nor can he easily make changes to existing terrain models, e.g. displace terrain features to change the line of sight.

In contrast to, for instance, mission rehearsal scenarios, training scenarios often take place on geo-typical terrain. It would be very helpful for instructors if they themselves were able to design new geo-typical terrain models or modify existing ones. This would allow them to create variations of terrain models with increasing complexity (e.g. by adding mountain ranges or forests, thereby limiting lines of sight). A more complex terrain entails a more difficult scenario, as a good air defense solution will be less obvious. Increasing the variety in terrain will also prevent the trainees from becoming too familiar with the specifics of a particular terrain model. Therefore, in this section we analyze existing manual and automatic methods for modeling geo-typical terrain, assessing the extent to which they are suitable for use by training instructors.

3.1 Manual modeling methods

Manual design of geo-typical terrain is comparable to game level design for commercial computer games. These fictional terrain models have evolved from primitive to highly detailed and, at least visually, very advanced. However, the workflow, tools, and techniques used to create these models have not advanced that much. Game levels are designed almost entirely by hand using complex tools and primitive constructs, e.g. manual creation and placement of geometry. Creating terrain models is currently thus both a complicated and tedious task, which requires specialized 3D modeling skills. It can take a skilled designer many months to complete a level, making it a costly and lengthy process.

This becomes problematic when games are used for training and instruction. For commercial games, the terrain model and game scenarios are largely predefined by the game developer, but for training games, it is typically the end user, in our case a military training instructor, who defines the scenarios. Military instructors usually lack the time, budget and, most importantly, the required 3D modeling skills for creating terrain models. Although they will have a clear picture of what kind of training scenario they want to create, and of the features the terrain should have, current terrain modeling tools, which often have been developed with expert game level designers in mind, simply do not support their way of thinking.

3.2 Automatic generation methods

Because of the disadvantages of modeling terrain by hand, automated terrain generation would seem a more feasible solution for terrain modeling by instructors, being a fast and easy way to acquire terrain models while not requiring 3D modeling skills. We evaluate both procedural methods in scientific literature as well as three relevant commercial tools for automatic terrain generation.

Research on procedural content generation

Procedural methods generate content, such as textures, models or even art, through algorithms steered by random numbers. The main advantage of these methods is their ability to automatically generate a large amount of content from a limited set of input parameters. Procedural methods are often applied to generate terrain or its features.

Height-maps (i.e. regular grids of elevation points) are often used as the basis of a terrain model. There are many procedural algorithms for generating height-maps, often based on fractal noise generators, such as Perlin noise [7, 8], 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 the height-map results in natural, mountainous-like structures (for a textbook on fractal noise and height-map generation, see Ebert et al. [9]). These height-maps can be transformed further based on simulations of physical phenomena, for instance erosion. Thermal erosion levels 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 can be simulated using, for example, cellular automata, where the amount of water and dissolved material that flows out to other cells is calculated based on the local slope of the terrain surface. Musgrave treats both types of erosion in his PhD thesis [10] and Olsen discusses several optimizations [11].

Basic noise-based height-map generation delivers results that are fairly random; user control is only on a global level, often using unintuitive parameters. Several researchers have addressed this issue. Frade et al. [12] introduce an evolutionary approach to develop Terrain Programmes (TPs), which are combinations of functions that are applied to a terrain grid to generate a terrain model. Starting from an initial population of basic TPs, each new generation of TPs is created based on mutations of one or two selected TPs in the current population. The terrain designer selects these TPs based on example terrain models generated by these TPs. Stachniak and Stuerzlinger [13] propose a method that integrates constraints (expressed as mask images) into the terrain generation process. They employ a search algorithm that finds an acceptable set of deformation operations to apply to a random terrain in order to obtain a terrain that (approximately) adheres to these constraints. Schneider et al. [14] introduce an editing environment in which the user edits the terrain by interactively modifying the base functions of the noise generator (by replacing the Perlin noise grid with a set of user-drawn gray-scale images), while viewing the results in 3D. Zhou et al. [15] describe a technique that generates terrain based on example input height-map and a user line drawing that defines the occurrence of large-scale curved line features, such as a mountain ridge. Features are extracted from the example height-map and matched to the sketched curves and seamed together in the resulting height-map. De Carpentier and Bidarra [16] introduce procedural brushes: users paint height-mapped terrain directly in 3D by applying simple terrain raising brushes but also brushes that generate several types of noise in real-time.

Based on terrain types, elevation and slope data, vegetation can be distributed automatically. Some approaches use quite complex ecosystem simulations that take into account soil information as well as the competition for space and sunlight between plants [17]. 3D models of plants are also an ideal candidate for procedural generation. Although plants of the same species all have a unique form, their basic structure is very similar. Because of this, plant models can, for instance, be generated using rule-based systems known as L-systems. These L-systems contain a starting symbol and a set of rewriting rules that transform the starting symbol into a more complex one. Here these transformations are mostly geometric transformations, e.g. they define the translation of a shape, the addition of new shapes relative to an existing structure, or the change from one shape to another one. When considering an L-system to create trees, the starting rule could define the tree trunk and other rules could add branches to an existing branch, or add leaves to smaller branches (see e.g. [18]).

Finally, procedural methods are also being applied to road network generation and urban modeling, for an overview of approaches, see e.g. Watson et al. [19] or Kelly and McCabe [20].

Most of these procedural content generation algorithms have several drawbacks that can hinder their use. The main challenge of procedural modeling is to find a good balance between automation and control. For application in training scenarios the generated results are often too random; the user has too little control over the outcome. For instance, one fractal height-map will often differ quite a bit from another that was generated with exactly the same input parameters. This is because procedural algorithms use random numbers a great deal. The algorithm's input parameters influence the result only at a very global level. In the end, the exact values of the random numbers determine, for instance, whether there is a steep mountain range or a valley at a specific location in the terrain, while the parameters only influence the changes of mountain ranges to occur. Furthermore, the parameters often require an in-depth knowledge of the algorithm (e.g. the number of noise octaves or the persistence value) to estimate the effect of a parameter on the outcome. Combined, these drawbacks typically give users little control over the generation process and force them to use a time consuming trial and error approach, as was noted in e.g. Stachniak and Stuerzlinger [13]. In our application domain, this is not an acceptable working method.

Besides the issues identified for using an individual method, it is also far from trivial to tune procedural methods to work well together to generate a fully featured terrain model. As we will see below, most commercial tools focus on a specific aspect of terrain modeling (e.g. elevation data). To our knowledge, there is currently no tool or integrating framework that combines these various algorithms in a usable way.

Commercial tools

We review three commercial automatic terrain generation tools that have been around for several years and have a substantial user base: TerraGen, GeoControl and L3DT. There are numerous other tools available, but these three deliver, in our

view, the most impressive results and have more advanced editing capabilities.

TerraGen 2 [21] uses an elaborate network of nodes (nodes generate noise or apply filters and even mathematical functions to intermediate results) to generate elevation data. Users control this process by placing the nodes, setting their parameters and connecting them in a specific order. During generation, this network is traversed and outputs of nodes are blended resulting in a height-map. The resulting terrain can also automatically be populated with external objects (for instance, tree models).

TerraGen delivers very impressive visuals, which have been used in several movies. However, to be able to use this tool effectively, background knowledge on mathematics and noise generation is necessary and extensive experimentation with the tool is needed. The tool has a steep learning curve, but is very powerful once mastered. Therefore, we conclude that this tool is much more usable for computer artists, who focus on creating aesthetically pleasing textured height-maps, than for training instructors, who focus on the functional requirements of a terrain.

GeoControl 2 [22] is a height-map editor that iteratively generates elevation data using its "Dynamic Level Generation" algorithm. The process starts with a 2 x 2 grid of height pixels and subdivides this grid using a fractal noise algorithm until the desired terrain dimensions are reached. A user can define the noise characteristics to be used in each subdivision. Additionally, filters can be applied to this basic noise algorithm, for instance erosion or smoothing filters. Like TerraGen, GeoControl generates high-quality height-maps.

One feature of GeoControl is the isoline. Users define an isoline by setting the elevation value along the line and the noise characteristics of the transition area around the line. As they draw an isoline on a height-map, a mountain ridge with these properties is generated along this line that blends in nicely with the existing map. After this process is finished, users can apply a procedural method to automatically generate rivers and lakes on the height-map. Besides this, it is also possible to modify the terrain afterwards by drawing a vector line and define a flattening operation along this line. This can, for instance, be used to manually create the embankment of a road in the terrain.

GeoControl's isolines can, with some practice, be used to draw height profiles that adequately match the user's wishes. Still, the complete modeling process of this tool can be quite complex and the quality of the results depend on knowledge of the effect of parameters and the dependencies between generation steps.

L3DT [23] allows a user to design a height-map by drawing on a design map using a brush. This brush is actually a set of generation parameters that are set by the user. These include the elevation, the amount of erosion, the roughness of the terrain, whether this cell is a source of water, and a climate profile. Each grid cell in the design map is automatically expanded to 64 x 64 points in the resulting height-map by applying noise, erosion and water flooding algorithms.

Climate profiles are used for generating a large terrain texture that is draped on the height-map, for each type of material (e.g. grass, rock) the conditions under which it can occur (e.g. elevation range, slope range, water level) are specified. After the height-map is generated, a scoring mechanism determines the placement of materials based on the climate profile. The resulting terrain texture is very convincing, resembling a satellite image.

From the tools evaluated, L3DT's design map is in our view the most suitable working method for non-expert terrain modelers. The tool is however limited to generating height-maps and corresponding terrain textures. To our knowledge there is no tool that can generate a fully featured 3D terrain model, and which is still suitable for non-expert use.

3.3 Suitability of current modeling methods

It is our belief that none of the methods and tools discussed above provide an ideal and complete solution for terrain modeling by training instructors. Manual terrain modeling methods are time-consuming and require 3D modeling skills, and automated methods are either too complex in use, lack user control or (as in the case of TerraGen, GeoControl and L3DT) focus mainly on one aspect of a terrain model, namely elevation data and corresponding terrain textures. Therefore, in practice, instructors are forced to reuse the terrain models that were shipped with the game or, alternatively, hire an external party to create custom terrain. The predefined terrain models most likely do not always adequately match their training requirements, and therefore restrict the training scenarios they can create. Hiring a third party to create new game terrain is mostly too expensive or involves a significant delay. The end result is that the training scenario is adjusted to match the available terrain, instead

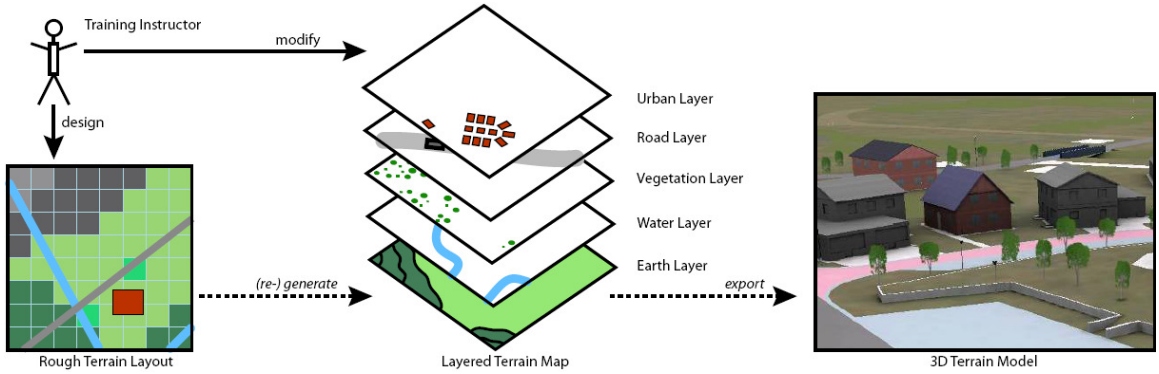


FIGURE 2: The workflow of the declarative terrain modeling framework.

of the other way round, which clearly has a negative impact on the overall training effectiveness.

4. A NOVEL INTEGRATED TERRAIN MODELING APPROACH

To remedy the situation discussed above, 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 an expeditious 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 fundamentally different from the current constructive approach (focusing on "how do I model it?"). It is ideally suited for serious games, in which more often than not the terrain model designers are end users, such as instructors, and not artists.

We are developing a modeling framework to support this declarative approach. Earlier, we have identified key requirements for this framework [24] 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 creates the terrain model accordingly. The framework thus lets instructors focus on declaring the terrain they need, without bothering with low-level 3D modeling tasks or difficult parameter tuning. Our framework provides automated modeling by integrating a variety of procedural content generation methods in a usable way.

The typical modeling workflow in our framework is as follows (see Figure 2). Users (in this case, training instructors) compose a digital sketch of the rough layout of the terrain. They declare the location of important terrain features, such as forests, mountains, cities and villages. Once they are satisfied with the rough terrain layout, the framework generates a high-resolution terrain map that complies to the specified features at large, but has, on a small scale, a high level of detail and variations in elevation, vegetation etc. Instructors can view the terrain in 3D and can manually edit the terrain map or modify the rough layout where desired. The modeling process is thus iterative: users can go back and forth between the rough layout and the detailed map. When they are satisfied with the results, the terrain map can be automatically exported to a 3D terrain model that can be used in the training game.

The terrain map is a layered data structure, see Figure 2. 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:

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

Although the layers are kept separately in the editing phase, they obviously have interdependencies (see Figure 3). To generate a consistent terrain, the generation process of the layers is ordered in such a way that a layer can be based on 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 Earth layer. The major roads generation method for the Road layer will have the sketched road lines 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.

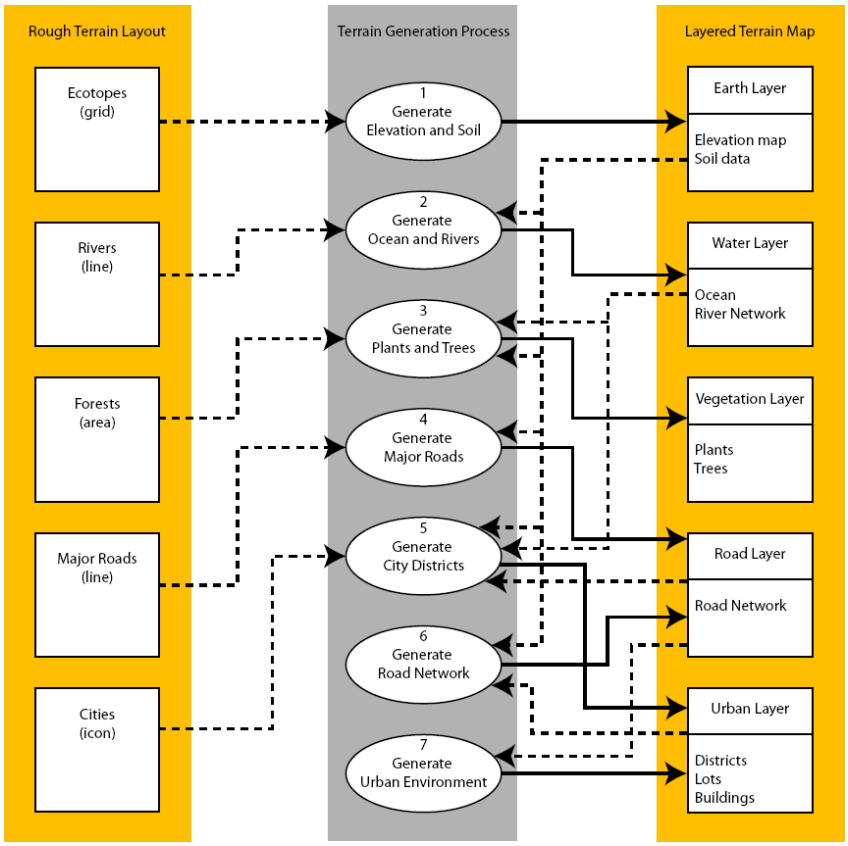


FIGURE 3: Our terrain generation process, dashed lines indicate inputs and solid lines outputs of the generation steps.

Still, to obtain a fully consistent and valid terrain, a merging phase is necessary after the generation process of Figure 3. 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. when a highway is modified so that it crosses a river on the Water layer, this road can be modified to include a bridge or it can be rerouted). The framework is responsible for merging 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 can be 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. Furthermore, we deploy semantically rich mechanisms to maintain consistency between terrain layers. In the next section, we discuss the generation method of two of the five layers: the Earth and Vegetation layer.

5. SKETCH-BASED TERRAIN GENERATION METHOD

We explain how terrain layers are generated based on the rough terrain layout using a training scenario from Tactical Air Defense. The first training session for aspiring platoon commanders is aimed at familiarization with the Stinger missile system and its parameters (e.g. its range, rate of fire, etc.). The trainees do not have any previous experience with the system. Following the JOT philosophy, they are simply told to place an air threat, a Stinger team and a radar system on the terrain, and to experiment with them. They are to discover and explore, by doing, the constraints of the missile system, as well as the influence of external factors, including:

- the influence of terrain and its features on the sight diagram of the Stinger team;
- the influence of the aircraft's approach route, angle and altitude; and

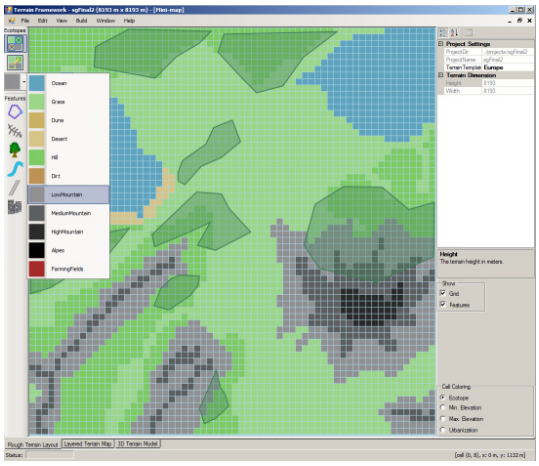


FIGURE 4: A rough terrain layout for the example scenario.

- the relation between the position of the Stinger team and the mobile radar and radar coverage for the team.

By moving the team and radar around the terrain, while examining the sight diagrams, and by running the simulated engagements with different setups, the trainees will get a good understanding of the basic principles in half a day of training.

An instructor preparing this scenario will wish to expose his trainees to a large variety of terrain types, ranging from flat grasslands to steep mountain ranges. A geo-typical terrain works best here: it can contain lots of varying terrain types while keeping the model at a reasonable size. In a geo-specific terrain on the other hand, such level of variation would usually be found only in ranges of hundreds of kilometers.

Using our framework, an instructor can declare such a terrain. He chooses a size for his terrain and sketches the rough layout of the terrain (see Figure 4). The base of the terrain is declared by specifying where in the terrain which ecotopes occur. An ecotope describes both the type of terrain, e.g. a specific type of desert, hills or mountains, and the associated range of elevation. The instructor's sketch is performed on an ecotope grid, a 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. Thus the drawing of the grid is quite similar to painting a pixel bitmap ([25] introduced the idea of procedurally expanding a bitmap with terrain colors to a height-map). On top of this grid, the instructor can place major terrain features, such as a forest. These are drawn as polygonal shapes (similar to the point, line and area shapes in common GIS vector files). In this way the instructor is able to declare the basic training terrain, with a large variety of terrain combinations e.g. ocean and beaches, sparsely vegetated grasslands, thick forests, hills, and a chasm between two mountain ranges.

5.1 Earth layer generation method

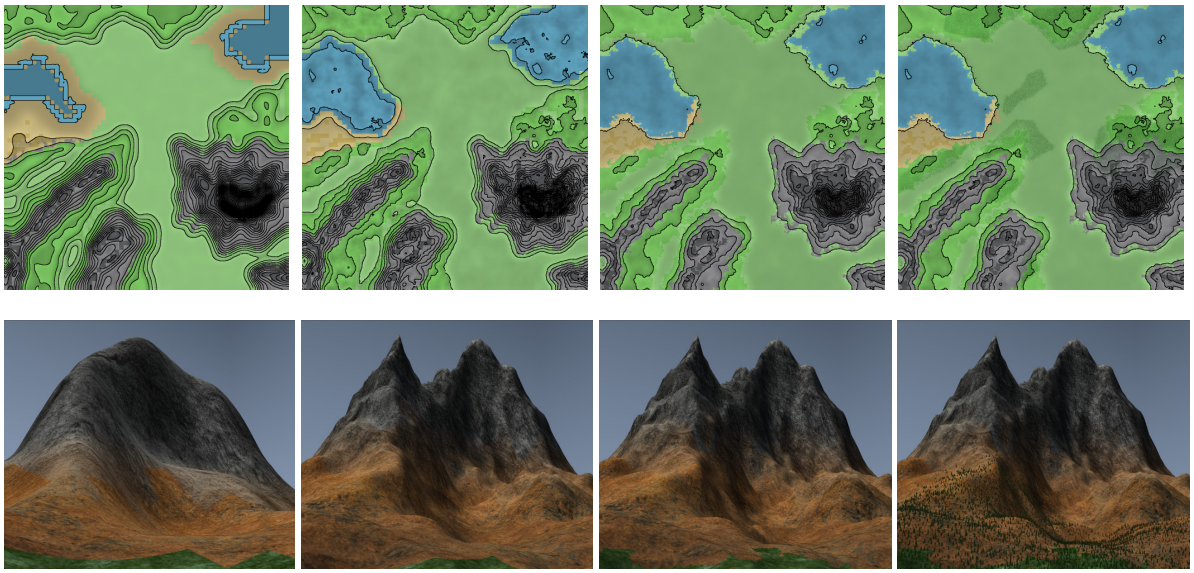
Based on this rough terrain layout the terrain layers are generated. The first layer to be generated is the Earth layer, which primarily contains elevation and ecotope data. The Earth layer generation is based only on the ecotope grid of the rough terrain layout; the terrain features are used for the other layers. The generation process consists of a number of steps, in which the rough terrain layout is amplified by interpolation of the coarse data to a full size terrain, combined with several noise input maps and filters to make the terrain more plausible and life-like.

The complete Earth layer generation proceeds as follows. We start by creating a temporary data grid with the same dimensions as the ecotope grid of the rough terrain layout, called an ecomap, which contains the following elevation information on each grid cell:

- Base Elevation: the base value in meters of the elevation at the center of this cell;
- Elevation Variation: the range in meters of the small scale variation in elevation in the cell; and
- Terrain Roughness: a factor describing the variation in elevation in this cell, lower values resulting in a smoother terrain.

These three data values are computed for each cell on the basis of the user-specified ecotope grid of the rough terrain layout. The definition of each ecotope includes value ranges for base elevation, elevation variation and roughness. For each grid cell in the ecomap, the ranges defined in this cell's and neighboring cells' ecotope are weighted using a Gaussian smoothing kernel. From the resulting ranges, a random value is chosen for the three data values, which on average will be at the middle of the range.

Once we have a grid with elevation information at the cell centers, we can amplify this ecomap into the Earth layer. This is where an interpolation method and fractal noise comes in. For each point (x, y) in the Earth layer, we interpolate the Base Elevation at nearby centers of the ecomap using Catmull-Rom interpolation [26]. This results in a very smooth terrain shown in Figure 5a.



a) b) c) d)

FIGURE 5: 2D contour-maps and 3D views of the Earth layer: a) interpolation only, b) interpolation and scaling by noise, c) final Earth layer, d) Earth and Vegetation layers combined.

We make the elevation profile more realistic by scaling the elevation by a weighted combination of two multi-fractal Perlin noise fields: the first field is generated with its parameters set to result in noise with sharp ridges, while the second field parameters are set to produce smooth, rolling noise, see e.g. [9] for more details. The weight factor for the two fields is a mask image that is based on the ecotope grid: where the ecotope is defined to be mountainous, the mask image is white (i.e. the factor is 1.0), elsewhere it is black (i.e. 0.0), with a smooth transition between the regions. This is because the ridged multi-fractal noise is well-suited for mountains, while the smoother noise is better suited for e.g. rolling hills. For some additional, small-scale (several meters) variation, we add to each point some Perlin noise in an interpolated range of the Elevation Variation. Figure 5b shows this intermediate Earth layer; the elevation profile has changed and is no longer unnaturally smooth.

While this creates a realistic profile in the z-direction, in the (x, y) field the transitions between e.g. ecotopes are still grid-like, see the blocky ecotope-color pattern in Figure 5a and 5b. To remedy this, we perturbate the landscape, by swapping each point with a random point in the field within a certain range. The (x, y) coordinates of each random point are determined based on a noise vector multiplied by the perturbation range. The resulting difference in the ecotope transitions is clearly visible when comparing Figure 5b and 5c.

The Earth layer is now completely filled, but for some final tuning, we apply two filters that improve the visual realism of the terrain: a simple erosion filter followed by a smoothing filter using a Gaussian smoothing kernel. The Earth layer resulting from the complete procedure is shown in Figure 5c. Note that it matches the rough terrain layout, but has features and variation of its own. If we would run the method again, another, similar but different Earth layer would result.

5.2 Vegetation layer generation method

Next, both the rough terrain map and the freshly generated Earth layer are used for generating the distribution of plants and trees in the Vegetation layer. Our implementation builds on the algorithm proposed by Deussen et al. [17]. Their algorithm simulates competition between plants for space. They abstract plants to circles, which represent the ecological neighborhoods of the plants, and start with a random distribution of plants of a field. In each iteration new plants are added, plants that are either too old or dominated by other plants are removed. Domination of plants occurs when two ecological neighborhoods intersect; the plant with a higher competitive ability dominates the other. The competitive ability depends on the water concentration of the

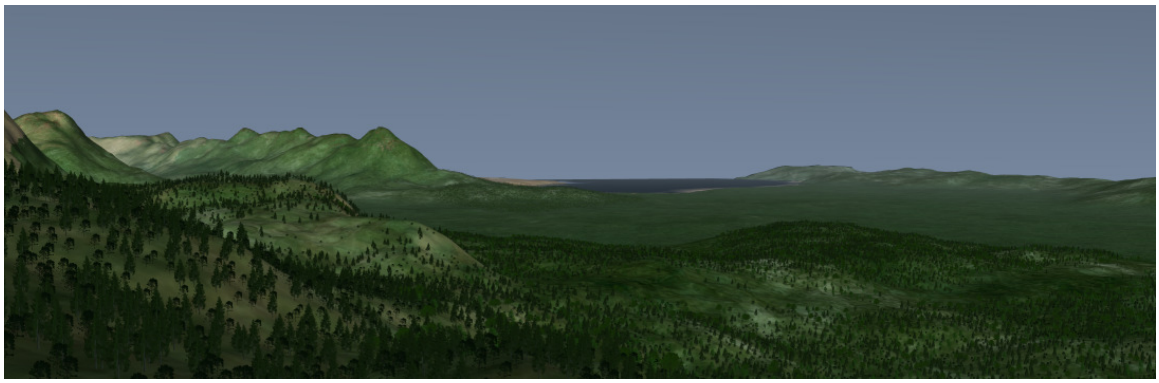


Figure 6: A wide view on the 3D terrain model automatically generated by our framework.

location combined with the plant's preference for wet or dry areas and the plant's relative size.

We use the same simulation of competition, but in our method the distribution of plants is based on the terrain features; vegetation density is high in forests and low in other areas. Per species, the number of plants in an area depends on the extent to which the Earth layer's ecotope supports that species, e.g. the desert ecotope only supports species that have a low need for water. The exact placement of a plant depends also on the Earth layer's local elevation and slope, e.g. some trees such as spruces and pines can grow on relatively steep mountainous terrain, while most others cannot. Figure 5d shows the distribution of trees of different species for this example scenario, the colors of the circles indicate the plant's species.

Figure 6 shows another view on the final 3D terrain model of the Earth and Vegetation layer in this scenario. The terrain model matches the rough terrain layout, shown in Figure 4, but clearly has a lot of variety of its own. It takes instructors very little time to sketch a rough terrain layout and, for this scenario, further manual refinements are not necessary. With this terrain model, trainees can quickly get a grasp on the effect of natural terrain features on the effectiveness of their air defense deployment.

5.3 Technical aspects

Our framework is implemented in C# / C++ .NET, with 3D visualization done in OpenSceneGraph [27]. To support an iterative approach, it is important that the generation processes are reasonably fast. Fortunately, several parts of the terrain generation and merging process are very well-suited for parallel processing. For instance, for the earth layer generation, the interpolation, noise scaling and perturbation steps are combined for each point, as it is possible to determine the definitive elevation of each point without knowing the values at the neighboring points. An emerging trend in parallel programming is to use a Graphics Processing Unit (GPU) as a general computation device, because it has a larger number of floating point processors available. A large part of our generation and merging process is performed in parallel on the GPU using NVIDIA's CUDA [28], a C-like programming language for performing all sorts of computations on GPU's. Our CUDA implementation is about 20 times faster than our original CPU implementation.

6. CONCLUSION

Due to the advancements in realism and immersion, computer games have an increasingly high potential for military training purposes. Training instructors are very competent at designing complex training scenarios, closely matching their stated learning objectives. However, they typically lack the technical skills that currently available tools require to build adequate terrain models. As a practical consequence, they often end up using predefined models, thus limiting training effectiveness.

We presented a novel declarative modeling approach consisting of a terrain modeling process and supporting framework, aimed at making terrain modeling tasks much more effective and accessible. The modeling framework developed supports non-specialists throughout the terrain modeling process, enabling e.g. training instructors to easily declare and generate a terrain that suits their training objectives. The careful deployment of procedural methods has been instrumental in this goal.

We evaluated our declarative modeling approach with a number of military training instructors and other training experts. They were very supportive of our approach, seeing the

framework as a valuable tool for modeling terrain for training scenarios. Most importantly, they realize they are no longer limited by the availability of terrain models, as they can create a new terrain model for each desired scenario by intuitively declaring its layout and features. They can even do this during a training session, e.g. to highlight a certain tactical aspect that came up in previous group discussions. For a large number of training scenarios, such as the Tactical Air Defense introductory scenario described in this paper, automatic generation based on the declared rough terrain layout yields a complete and fitting terrain model, without any further involvement of the instructor.

We are currently focusing on a number of research issues of our declarative terrain modeling framework.

Firstly, we are working on integrating city generation capabilities in the Urban layer of the framework, a key feature for many complex training scenarios. For this, we have developed novel mechanisms to create a believable layout, including positions, connections and dependencies, of the different types of districts in a city, e.g. upper class residential, industrial, etc (see [29]). Results for informally structured villages have also been presented in [30].

Secondly, we want to enable instructors to manually perform small scale adjustments in order to fine-tune the terrain to their scenario. For instance, in the tactical air defense case, instructors would like to be able to manually insert a special building as an objective or target. For this, we need a set of easy-to-use manual editing tools for each of the terrain layers.

Thirdly, we want to explore a variety of useful constraints a user could impose on a rough terrain layout to constrain the generation process. For instance, in an air defense scenario it might be helpful to specify line-of-sight constraints for a specific area of terrain.

Lastly, an ongoing challenge involves the consistency management of interacting terrain features, typically lying on different terrain layers. For many such interactions, constraint solving methods will likely be necessary to automatically readjust actual terrain features in a coherent and plausible manner.

Producing appropriate terrain models is crucial for the effectiveness of scenarios for military training, but it is seriously hindered by the complexity of current terrain modeling tools and methods. In order to realize the full potential of games for military training, it is essential to support and enhance the modeling process. We believe that this requires a shift from the conventional paradigm of terrain construction towards declarative terrain modeling. The approach discussed here is a firm step in this direction.

ACKNOWLEDGEMENTS

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). We thank the researchers from the TNO departments Modeling & Simulation and Training & Instruction for their comments and feedback. Finally, we thank the tactical air defense instructors of the Royal Netherlands Army for their constructive input.

REFERENCES

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

[2] Steel Beasts, eSim Games (2009). Available from http://www.steelbeasts.com.

[3] Tactical Iraqi Language & Culture, Alelo inc. (2009). Available from http://www.tacticallanguage.com/.

[4] America's Army, US Army (2009). Available from http://www.americasarmy.com.

[5] Hulst, van der, A., Muller, T., Besselink, S., Coetsier, D., Roos, C. (2008). Bloody Serious Gaming: Experiences with Job Oriented Training. *Proceedings of the Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*, Orlando, USA, 375-385.

[6] Wells, W. D. (2005). Generating Enhanced Natural Environments and Terrain for Interactive Combat Simulations (GENETICS). *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, New York, NY, USA, ACM Press, 184–191.

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

[8] Perlin, K. (2002). Improving noise. *SIGGRAPH 2002: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, 681-682.

[9] Ebert, D. S., Worley, S., Musgrave, F. K., Peachey, D., Perlin, K. (2003). *Texturing &*

Modeling, a Procedural Approach. Third edition, Elsevier.

[10] Musgrave, F.K. (1993). *Methods for Realistic Landscape Imaging*. Unpublished doctoral thesis, Yale University.

[11] Olsen, J. (2004). Realtime procedural terrain generation. Unpublished technical report, University of Southern Denmark. Available from http://oddlabs.com/download/terrain_generation.pdf.

[12] Frade, M., Fernandez de Vega, F., Cotta, C. (2009). Breeding Terrains with Genetic Terrain Programming: The Evolution of Terrain Generators. *International Journal of Computer Games Technology*, vol. 2009, Hindawi Publishing Corporation.

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

[14] Schneider, J., Boldte, T., Westermann, R. (2006, November). *Real-Time Editing, Synthesis, and Rendering of Infinite Landscapes on GPUs*. Paper presented at the Conference on Vision, Modeling and Visualization, Aachen, Germany.

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

[16] De Carpentier, G. and Bidarra, R. (2009). Interactive GPU-based Procedural Heightfield Brushes. In *Proceedings of the 4th International Conference on the Foundation of Digital Games*, Florida, USA.

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

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

[19] Watson, B., Müller, P., Veryovka, O., Fuller, A., Wonka, P., Sexton, C. (2008). Procedural Urban Modeling in Practice. *IEEE Computer Graphics and Applications*, 28(3), 18–26.

[20] Kelly, G., McCabe, H. (2006). A Survey of Procedural Techniques for City Generation. *Institute of Technology Blanchardstown Journal*, Dublin, Ireland, (14), 87-130. Available from http://www.gamesitb.com/SurveyProcedural.pdf.

[21] Terragen 2, PlanetSide Software (2009). Available from http://www.planetside.co.uk/terragen/tg2/.

[22] Geocontrol 2, Rosenberg, J. (2009). Available from http://www.geocontrol2.com/e_index.htm.

[23] L3DT, Torpy, A. (2009). Available from http://www.bundysoft.com/L3DT/.

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

[25] Roden, T., Parberry, I. (2004). From Artistry to Automation: A Structured Methodology for Procedural Content Creation. *Proceedings of the 3rd International Conference on Entertainment Computing*, Eindhoven, The Netherlands, 151–156.

[26] Catmull, E.E., Rom, R.J. (1974). A Class of Local Interpolating Splines. In Barnhill, R.E., Riesenfeld, R.F. (Eds.) *Computer Aided Geometric Design* (pp. 317-326). Academic Press, New York.

[27] OpenSceneGraph, Osfield, R., Burns, D. (2009). Available from http://www.openscenegraph.org.

[28] CUDA, NVIDIA Corporation (2009). Available from http://www.nvidia.com/object/cuda_home.html.

[29] 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 Alliez, P. and Magnor, M., editors, *Proceedings of Eurographics 2009: Short Papers*, pages 45-48, Munich, Germany. Eurographics Association.

[30] Smelik, R.M., Tutenel, T., de Kraker, K.J., Bidarra, R. (2009). A Case Study on Procedural Modeling of Southern Afghanistan Terrain. *Proceedings of the IMAGE 2009 Conference*, Saint Louis, Missouri, USA, 329-337.