Semantic constraints for procedural generation of virtual worlds

Ruben Smelik
TNO
The Netherlands

Krzysztof Galka
Wroclaw University of Technology
Poland

Klaas Jan de Kraker
TNO
The Netherlands

Frido Kuijper
TNO
The Netherlands

Rafael Bidarra
Delft University of Technology
The Netherlands

ABSTRACT

Procedural generation of virtual worlds is a promising alternative to classical manual modelling approaches, which usually require a large amount of effort and expertise. However, it suffers from a number of issues; most importantly, the lack of user control over the generation process and its outcome. Because of this, the result of a procedural method is highly unpredictable, rendering it almost unusable for virtual world designers.

This paper focuses on providing user control to deliver an outcome consistent with designer's intent. For this, we introduce *semantic constraints*, a flexible concept to express high-level designer's intent in intuitive terms as e.g. line of sight. Our constraint evaluation method is capable of detecting the context in which such a constraint is specified, automatically adapting to surrounding features of the virtual world. From experiments performed within our prototype modelling system, we can conclude that semantic constraints are another step forward in making procedural generation of virtual worlds more controllable and accessible to non-specialist designers.

Categories and Subject Descriptors

I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism; I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction techniques*

1. INTRODUCTION

Virtual worlds are featured in more and more areas of modern multimedia technology, such as entertainment and serious games, simulations, and movies. The quality of a virtual world model poses a direct impact on the user's immersion. Because of the ever-increasing cost of creating these worlds by hand, semi-automated procedural methods for content creation are being explored. Starting with fractal height-maps (e.g. [9]), the research on procedural modelling is constantly growing in potential, now involving almost all features of virtual worlds, for instance vegetation [10], roads [6] and urban environments [8].

The main issue of the procedural approach lies in its core: the unpredictability of the outcome. The process is typically driven by a minimal amount of user input and offers very limited ways of influencing and controlling the generated results [11]. Noticeable work to address this issue concentrated on controlling the generation of elevation profiles, e.g. using 2D user drawn imagery [17, 2, 5], height-map elevation constraints [15, 7, 1], or agents [4]. Furthermore, for L-systems, bounding volumes have been applied to constrain the growth of plants [10]. More recently, a more general approach introduces guides to constrain procedural structures based on L-systems [3]. The main drawback of these solutions is that they are either still somewhat lacking in user control, or require investing significant time in modelling and fine tuning to achieve the desired result. Furthermore, they are not easily extensible to other features of the virtual world [12].

In this paper, we focus on capturing high level designer's intent. An example of such intent is to have a clear line of sight between two locations in a virtual world. In the design of entertainment game worlds as Assassin's Creed or Oblivion, this intent could for instance be to have a *vista point*, where the player has an impressive view on the city he or she is going to visit next. In serious games for e.g. military training, it could be to have a suitable overwatch position on a hill to support a friendly unit on patrol in the valley.

In current manual modelling systems, such intent cannot be made explicit [16]. As a result, a designer has to manually preserve this intent throughout the modelling session, which makes experimentation or exploration of alternatives more cumbersome. In the context of procedural generation of virtual worlds, this kind of intent is not only to be translated into procedure parameters, but it also needs to be automatically *maintained* throughout the modelling process.

A natural way to manage this is to map intent to *constraints* imposed on the generation procedures. Maintaining intent by means of constraints is not a novelty; for instance, it has already been successfully applied to 2D game level

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PCGames 2011, June 28, Bordeaux, France

Copyright 2011 ACM 978-1-4503-0804-5/11/06 ...$10.00.

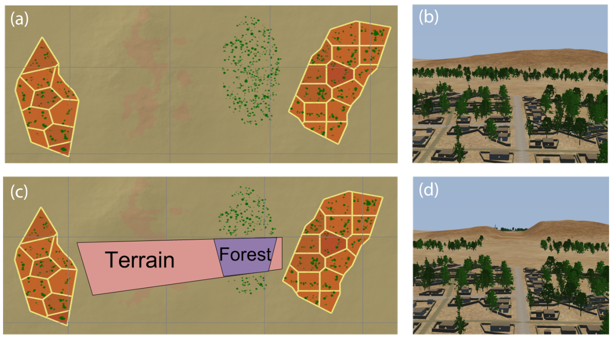

Figure 1: A *line of sight* constraint: two villages with inter-visibility obstructed by hills and forest. 2D (a) and 3D (b) overview of situation before, (c) *line of sight* constraint composed of two feature constraints, (d) 3D view after constraint evaluation, resulting in a clear view between the two cities.

generation, e.g. the platform level generator of Smith et al. [14]. The main limitation of existing approaches focussed on 3D virtual worlds is that they usually address a single feature (height-maps), and are often not efficient enough to function in an interactive modelling framework.

Therefore, there is a need for a more flexible and efficient mechanism capable of expressing and automatically maintaining high-level intent over a specified area of the virtual world and all terrain features within this area. This paper introduces *semantic constraints*, a novel concept for high-level user control over procedural generation of virtual worlds. Examples resulting from the application of semantic constraints to virtual worlds include tight mountainous passageways forming choke points, lookout spots with an unobstructed line of sight over a designated area or valleys with limited access, all of which can have great impact on e.g. game-play or training value. Furthermore, by supporting flexible constraint composition and context awareness, semantic constraints enable designers to express their intent in an *accessible* way.

2. SEMANTIC CONSTRAINTS

In our approach, a semantic constraint is a control mechanism imposed on the generation process in order to satisfy explicit designer's intent over a specific area. We denote this area of the virtual world as the constraint's *extent*. A semantic constraint adapts to the current context of its extent, i.e. the local terrain and nearby features. The constraint is re-evaluated when the terrain is modified or whenever a new feature is introduced within the constraint's extent. Because of this, the virtual world remains plausible and consistent with the designer's intent. As a result, designers can start with specifying the high-level features of their world and provide additional detail later on.

To manage this behaviour, a semantic constraint can be composed of several sub-constraints, called *feature constraints*. Semantic constraints are abstract, high level constructs, which convey the vocabulary that is directly used by designers to express their intent. Depending on the context of the extent, semantic constraints automatically apply a subset of their feature constraints.

Feature constraints are specialized to operate on a single type of feature such as a forest. They are mapped to low-level operations to achieve a specific result, like limiting the height of vegetation within a designated area. The feature constraints of a semantic constraint are independent of each other, but, together, are configured to fulfil the common goal.

An example of composition of constraints is a *line of sight* constraint set in a complex virtual world. The evaluation of this semantic constraint can affect terrain, vegetation and urban features generation processes. An example of this is shown in Figure 1, where the *line of sight* constraint is composed of two feature constraints that affect the generation of the terrain and the forest.

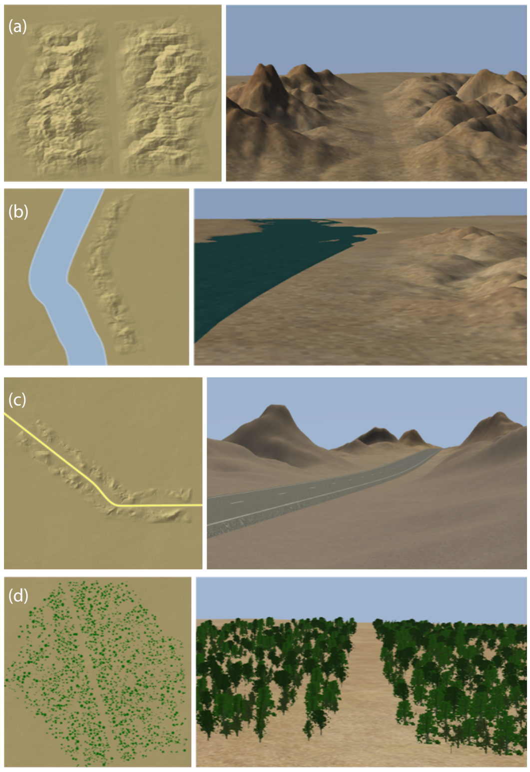

Figure 2: Results of different application schemes of a *choke point* in the context of: (a) bare terrain, (b) a river, (c) a road, and (d) a forest.

3. CONSTRAINT EVALUATION METHOD

Context detection is the analysis of a constraint's extent to derive a specific application scheme. An *application scheme* is a set of instances of feature constraints, suitable for the context (see Figure 2). As a semantic constraint is aware of what feature constraints it is composed, context detection embeds the process of deriving an application scheme in the semantic constraint. Each instance of a feature constraint in such a scheme is linked to a single terrain feature. This enables the feature to inform the constraint regarding changes to its state.

An *association relationship* is a relation between a semantic constraint and features that are not necessarily in its extent. This relation links features that are not the object of constraint application, yet provide context. As an example we describe a *route* constraint, which ensures a route

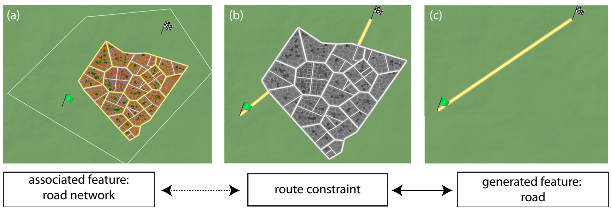

Figure 3: Constraint association of a *route constraint:* (a) *route definition* between two locations (flags), (b) association with existing network, (c) removing this network result in a direct connection.

between two locations in the virtual world. For this, it takes the existing road network into account (see Figure 3). In case there is no connection in the area, or the local road network provides only part of the route, the constraint evaluation has to introduce as many new roads as needed to connect the two locations. Therefore, the existing road network is in association with the route constraint. Using this association relationship, the constraint is notified in the event of removal of any existing roads, resulting in a re-evaluation of the context and proper handling of the situation.

The evaluation of a semantic constraint can result in an application scheme consisting of numerous feature constraints. As a result, for a given terrain feature, several semantic constraints can impose multiple low-level demands. To manage all these different feature constraints, a feature maintains a stack of applicable constraints (see Figure 4). By mapping the constraints in this stack to corresponding operations, we obtain a sequential list of operations that have to be performed during the generation of the feature to satisfy the semantic constraints. Through a sequential analysis of a feature's attached constraints, we check their consistency and handle any conflicting demands of different constraints. This analysis can result in changing a feature constraint's parameters or canceling its current application.

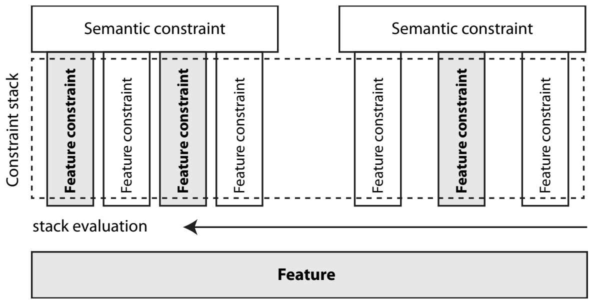

Figure 4: Schematic view of the creation and evaluation of a constraint stack.

An important aspect of the consistency analysis is handling interactions between semantic constraints. We based the mechanism for this on our generic method for interactions, described in [13], which detects and handles interactions between features, resulting in connections (e.g. a bridge of a road over a river) and conflicts (e.g. the removal of trees obstructing a road running through a forest). Each constraint in a feature's stack issues a *claim* for its extent in order to reserve it for exclusive use. The interaction resolution method considers the extents of other constraints to determine whether the claim is partially or fully granted, resulting in a modification of the constraint's extent. This mechanism is vital to be able to specify overlapping constraints without these constraints conflicting with each other.

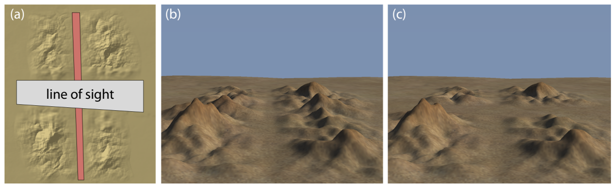

Figure 5: Constraint priorities: (a) 2D view of areas granted to a *choke point* and *line of sight* constraint, (b) before and (c) after specification of the *line of sight* constraint on top of the *choke point* constraint.

For the interaction handling process, *constraint priorities* have been defined. In case a constraint issues a claim for extent that overlaps with an existing constraint's extent, the constraint's type priority value determines whether the claim is granted. An example of this mechanism can be seen in Figure 5, where a *line of sight* constraint is placed over an existing *choke point* constraint. The *line of sight* constraint claims an extent overlapping with the *choke point*'s extent. As it has a higher priority than the *choke point*, the claim is granted. The *choke point* constraint adapts to this, resulting in a consistent coexistence of the two.

4. IMPLEMENTATION

By composingseveral specialized feature constraints, we were able to implement interesting types of complex semantic constraints, such as *line of sight*, *choke point*, *route*, *concealment area* (an area with cover from a specific threat). All of the mentioned constraints featured several context-dependent application schemes. This proved to be essential to handle the variety of possible situations and for achieving a smooth and natural integration in the virtual world. Because of the modular approach, creating new semantic constraints is relatively easy and fast, as typically it can be composed of existing feature constraints.

As an example of an integrated constraint, we give an outline of the implementation of the *line of sight* constraint. Based on the input observer and observation locations, we calculate a *view plane*. This view plane consists of all required lines of sight starting at the observer that have a view on the observation area, essentially providing a threshold height for each location within the constraint's extent. To enforce the line of sight, we need to modify the height of both the terrain and all features in it. For the terrain, we calculate a scale factor s as $min(\frac{H(x,y)}{h(x,y)})$, where $H(x,y)$ is the treshold height value of the view plane and h(x,y) is the original elevation value at that point. Scaling the elevation in such a way induces unnatural transition artifacts; we use a blending approach to create a more smooth transition. The blended result is defined as $H'(x,y) = lerp(s*h(x,y), h(x,y), d(x,y))$, where $d(x,y)$ is a linear interpolation factor based on the distance from the direct line between observer and observant.

The prototype was integrated into SketchaWorld, our virtual world procedural modelling framework [13]. We used CUDA for efficient processing of computationally expensive constraints in order to provide feedback at interactive rates.

The integration of semantic constraints within our virtual world modelling framework is *loosely coupled*. The framework signals, by means of events, the (re-)generation of a particular terrain feature. By subscribing to these events, we extend the generation process by processing the stack of attached constraints.

5. CONCLUSIONS

Procedural methods have the potential to provide a significant increase in productivity for virtual world modelling. However, they often lack the level of user control required for typical modelling scenarios, which limits their practical application.

This paper introduced *semantic constraints* for providing high level control over the procedural generation process of complex virtual worlds. Our constraint evaluation method allows for flexible composition and extension of semantic constraints, which then in turn adapt to their context and are automatically and consistently maintained. Integrated in our prototype system SketchaWorld, designers can use semantic constraints in an interactive manner.

The currently implemented examples of semantic constraints provide a starting point for future work in this direction. To increase the richness of the designer intent that can be specified, we will foremost focus on introducing new and more elaborated semantic and feature constraints. We would also like to incorporate more constraint application schemes to cover a wider range of possible contexts and terrain features, e.g. a *choke point* in an urban environment.

For relatively complex designer intent, expressed through constraints, we are able to maintain consistent results throughout a virtual world modelling session, which significantly alleviates the effort for designers to preserve their intent. Therefore, we conclude that *semantic constraints* are another step forward in making procedural generation more controllable and accessible to designers.

Acknowledgements

This research is supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO).

6. REFERENCES

[1] F. Belhadj. Terrain Modeling: a Constrained Fractal Model. In *AFRIGRAPH '07: Proceedings of the* 5^{th} *International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 197–204, New York, NY, USA, 2007. ACM.

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

[3] B. Beneš, O. Šťava, R. Měch, and G. Miller. Guided Procedural Modeling. In *Computer Graphics Forum: Proceedings of Eurographics 2011*, pages 325–334, Llandudno, UK, 2011. Eurographics Association.

[4] J. Doran and I. Parberry. Controlled Procedural Terrain Generation Using Software Agents. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):111–119, June 2010.

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

[6] E. Galin, A. Peytavie, N. Marchal, and E. Gurin. Procedural Generation of Roads. In *Computer Graphics Forum: Proceedings of Eurographics 2010*, volume 29, pages 429–438, Norrköping, Sweden, May 2010. Eurographics Association.

[7] K. R. Kamal and Y. S. Uddin. Parametrically Controlled Terrain Generation. In *GRAPHITE '07: Proceedings of the* 5^{th} *International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, pages 17–23, New York, NY, USA, 2007. ACM.

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

[9] F. K. Musgrave, C. E. Kolb, and R. S. Mace. The Synthesis and Rendering of Eroded Fractal Terrains. In *SIGGRAPH '89: Proceedings of the* 16^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 41–50, New York, NY, USA, 1989. ACM.

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

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

[12] R. M. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra. Integrating Procedural Generation and Manual Editing of Virtual Worlds. In *PCGames '10: Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, pages 1–8, New York, NY, USA, 2010. ACM.

[13] R. M. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra. A Declarative Approach to Procedural Modeling of Virtual Worlds. *Computers & Graphics*, 35(2):352–363, April 2011.

[14] G. Smith, J. Whitehead, and M. Mateas. Tanagra: a Mixed-Initiative Level Design Tool. In *FDG '10: Proceedings of the* 5^{th} *International Conference on the Foundations of Digital Games*, page 209–216, New York, NY, USA, 2010. ACM.

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

[16] T. Tutenel, R. Bidarra, R. M. Smelik, and K. J. de Kraker. The Role of Sematics in Games and Simulations. *ACM Computers in Entertainment*, 6:1–35, 2008.

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