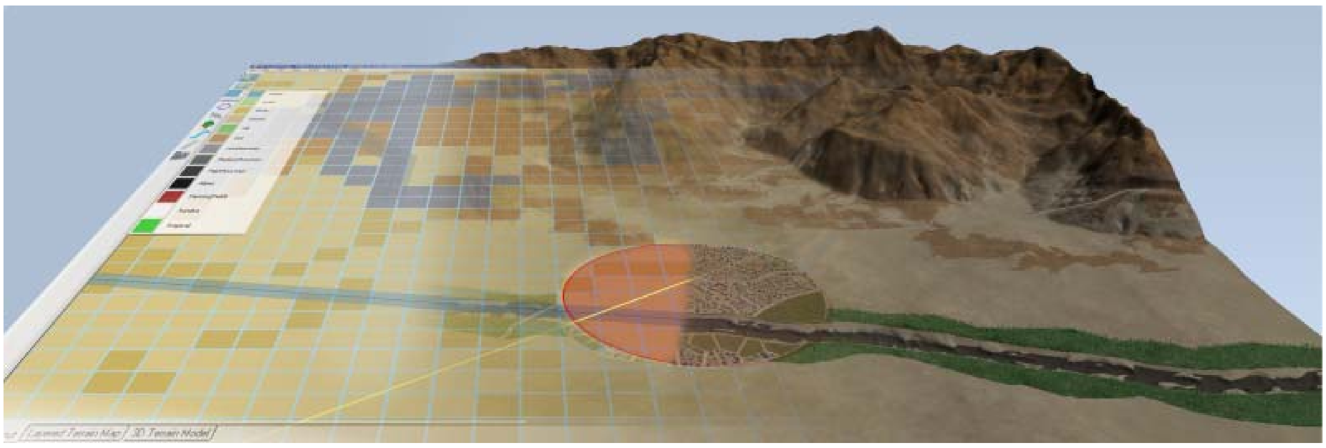
Integrating semantics and procedural generation: key enabling factors for declarative modeling of virtual worlds

R. Bidarra, K.J. de Kraker, R.M. Smelik, T. Tutenel



Abstract— **Manual content creation for virtual worlds can no longer satisfy the increasing demand arising from areas as entertainment and serious games, simulations, movies, etc. Furthermore, currently deployed modeling tools basically do not scale up: while they become more and more specialized and complex, they strikingly fail to assist designers in concentrating on their creative tasks.**

This paper presents the main concepts of what we designate *declarative modeling of virtual worlds*, a novel modeling approach that allows designers of virtual worlds to concentrate on stating what they want to create instead of on describing how they should model it. We discuss the two main characteristics of this approach – *semantics* and *procedural generation* – and describe how they mutually reinforce, support and amplify designers' efforts, thus empowering their creative skills.

We conclude that this declarative approach provides designers with the productivity gain of procedural generation techniques, while still allowing for abundant control and flexibility. In addition, it significantly reduces the complexity of virtual world modeling, making it accessible to whole new groups of users and applications.

Index Terms—**declarative modeling, procedural methods, semantics, virtual worlds.**

[*]R. Bidarra and T. Tutenel are with the Computer Graphics Group, Delft University of Technology, The Netherlands ([*]corresponding author: +31 15 278 4564; e-mail: R.Bidarra@tudelft.nl).

K.J. de Kraker and R.M. Smelik are with the Modelling, Simulation & Gaming Department, TNO Defence, Security and Safety, The Netherlands.

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).

I. INTRODUCTION

DESPITE all technological developments and so many advanced tools and techniques, current content creation for virtual worlds still keeps much of the handcraft character of traditional creative work. Nowadays, this fact is becoming more and more critical, because of the increasing demand for and complexity of such labor-intensive content. For this reason, the potential of procedural methods as an alternative to manual content creation is attracting increasing attention and research efforts from both industry and academia.

Most currently available procedurally-based content creation methods and tools do partially address these demands by offering semi-automated procedural solutions that help generate parts of virtual worlds (e.g. terrains, roads, cities or buildings). However, for most designers, they present several fundamental drawbacks, e.g. they are mostly technically complex, run at non-interactive rates, are often unintuitive in use, yield just one specific type of content, and hardly offer proper control on the output, which in turn cannot easily be integrated into a complete and consistent virtual world. Table I briefly summarizes the strengths and weaknesses of most manual 3D modeling and traditional procedural modeling approaches used so far in designing virtual worlds [1], [2].

To date, no research method or commercial tool has been proposed that provides an integrated solution to the problem of procedurally creating a complete virtual world, ranging from a mountainous landscape to man-made structures. We believe that overcoming this challenge will be instrumental in

Table I Strengths and weaknesses of both manual and traditional procedural modeling approaches

	Manual modeling	Procedural modeling
Productivity	×	✓
Adaptability	×	✓
Variation	×	✓
Intuitiveness	✓	×
User control	✓	×
Interactivity	✓	×
Versatility	✓	×
Completeness	✓	×
Simplicity	×	×

the much desired acceptance of procedural methods in mainstream virtual world development.

We developed one such integrated approach – *declarative modeling of virtual worlds* –, which combines the strengths of both manual and procedural modeling (see Table I), and thus provides a less complex and more productive workflow to model virtual worlds. In this paper we describe the main concepts of this novel modeling approach which, in essence, enables designers of virtual worlds to concentrate on stating *what* they want to create instead of on describing *how* they should model it.

After a short overview (Section 2), we describe the two main characteristics of this approach – *semantics* (Section 3) and *procedural generation* (Section 4) – and finally discuss how they mutually reinforce, support and amplify designers' efforts, thus empowering their creative skills (Section 5).

II. DECLARATIVE MODELING OF VIRTUAL WORLDS

From the analysis of Table I above, it follows that a hybrid solution combining the strengths of both approaches should be sought. We argue that in order to achieve this synthesis, the missing key element in current systems is a resolute and consistent deployment of the high-level modeling vocabulary actually used by designers in their creative process. In other words, the tools provided to designers should support and capture their intent, by understanding and operating at the level of semantics they are used to. For example, virtual world designers might prefer to simply state:

(a) that this hilltop should have some sort of visibility, while they are sketching a mountainous terrain; or

(b) that they want a medieval-typed city along these river banks, which centre should concentrate around three bridges at the indicated points; or

(c) that the interior of this house should be richly laid out and furnished in an art nouveau style, whereas for that office, a clumsy, decayed fashion is desirable; or

(d) that a given warehouse should be crammed with old, dusty crates and rusty barrels in a mess.

The above are very clear examples of *declarative modeling*: statements that express and describe *what* you want (your intent), without assumptions or remarks on *how* you should bring it about. The novel approach we bring forward here is based on this paradigm, and we therefore properly designate it *declarative modeling of virtual worlds.* In essence, this approach proposes to apply a variety of semantics-based techniques to overcome each of the weaknesses of procedural modeling methods identified in Table I. In Fig. 1 we schematically describe the generic terms of its main workflow.

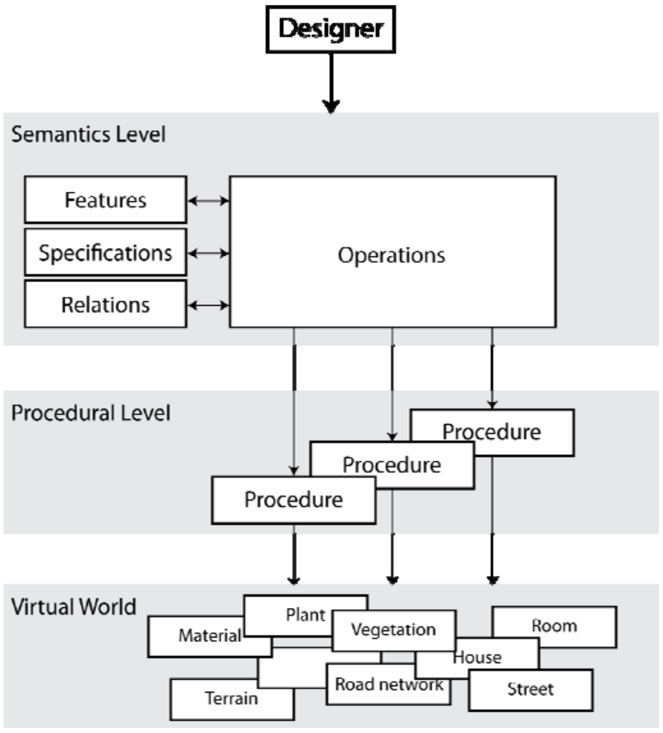


Fig. 1 Generic approach of declarative modeling of virtual worlds

Basically, this scheme differs from conventional procedurally-based modeling, sporadically used by designers and technical artists, in that it incorporates a semantics layer between the designer and the procedural techniques. This *semantic level*, provides designers with a powerful front-end that generates and steers the underlying *procedural level*, while encapsulating the complexity of the latter.

At the semantics level, designers have at their disposal high-level tools and methods which, so to say, 'talk their language'. Here we distinguish three generic categories:

features, in a very broad sense, constitute the basic vocabulary to express their creative design intent. One can think of all sorts of objects and abstract classes (*nouns* like tree, hill, road, chair, knife, etc.), but also of many actions that somehow are meaningful for the gameplay (*verbs* like catch, open, fill, destroy, etc.);

relations assist designers in further specifying their intent for some (groups of) objects or for an aspect or part of the virtual world. Typical examples of relations are not only geometric constraints between objects (distance, adjacency, etc.), but also logical roles and functional conditions among them (e.g. 'freezer requires power', 'water lessens thirst', 'bridge connects riverbanks');

specifications are the closest designers might wish to come to an actual procedure, providing them with a sort of 'language' for specifying the 'vision of the scene or world' they have in mind. For example, they may describe *how* *many* of *which* features should occur *under which circumstances* in the virtual world; or *how* strict certain predicates or relations among *which* features should be enforced. Specifications may also typically include object attributes, properties or modifiers (*adjectives* like large, full, old, dirty, etc.) which characterize under which appearance, state or presentation features should be created or laid out in the world, etc.

Two advantages of this approach become apparent from this scheme. First, the above elements of designers' intent provide the necessary 'intelligence' to be able to select and configure the appropriate types of procedure methods, or even generate new procedures that suit that intent. Typically, as pointed out in the Introduction, each procedural method yields just one type of content. However, we now possess a higher level insight on the desired global results. Therefore, we are able to select, trigger and control each of those methods and their partial output. Together, they combine to bring in the variety and richness devised by the designer for the virtual world.

The second advantage of this approach is that during a declarative modeling session, we are not just interpreting user commands and passing them on to drive the procedural methods below. Instead, we are incrementally gathering all expressions of designer's intent in a structured manner, so that a consistent high-level model of the virtual world, in terms of its meaningful features, can be maintained throughout the iterative design process. Consequently, each new modeling action can more easily be faced with that model, in order to assess to what extent it conforms to all previous designer intent expressed so far. Whenever any conflicts are detected, their solution will naturally be much more complete and adequate if sought at this same semantics level, because it is much closer to the way of thinking of designers (and also because their intervention, at some point, might even be indispensable). For example, if a designer decreases a room's dimension such that its specified furniture no longer fits in it, he might be asked to revise those furniture specs; or if a town was built along a road, and this road is subsequently rerouted within the virtual world, the designer might have to decide whether the town is also meant to move together with it.

We have implemented a declarative framework, called *SketchaWorld*, which demonstrates the feasibility of this approach, and its suitability for modeling complete virtual worlds. In the next two sections, we briefly discuss several aspects and facilities of this framework, illustrating with some examples how it integrates semantics and procedural methods, respectively.

III. Semantics

As depicted in Fig. 1, the SketchaWorld framework deals with all high-level information relating to virtual world objects at the semantics level. In this context, we have defined object semantics as all information, beyond its 3D geometric model, that helps conveying the *meaning* and the *role* of the object in the virtual world [3]. This semantics includes *features, relations* between features and further *specifications*.

One of our main schemes for specifying and representing this knowledge is the *semantic library*. It is partly based upon WordNet [4], a database that includes a complete dictionary of the English language, with descriptions, disambiguation between multiple meanings of the same word and, most importantly, among other things, parent-child relationships between different concepts. Every class in the semantic library is a WordNet concept derived from the *physical object* concept. Among other entities, the semantic library contains generic descriptions of classes of features (e.g. seat, window, bridge or road), including attributes, properties, roles, etc. The semantic library can be easily edited and expanded with new classes, e.g. by means of the usual inheritance and specialization mechanisms, using an intuitive interface.

In the semantic library, we extended the WordNet database with the ability to define various relations between features, including e.g. geometric constraints. With them, you can define, for example, that a vase should be placed on the windowsill, or that a street requires lampposts to be placed every 50m at the sidewalk, or that a certain type of city 'prefers' a commercial district somewhere near its center. By defining such generic properties and relationships at the class level, they automatically hold for every concrete instance of a class (e.g. all seats) and of all its subclasses (e.g. chairs, banks, sofas) in the virtual world. Logically, whenever desired the designer can override or extend this semantics for specific instances. Relationships like the geometric constraints above, defined in classes, are very useful in both manual and procedural modeling settings. In the former, for example, objects being manually placed in a virtual world can automatically be snapped to a valid position within their environment. The use of relationships in combination with procedural modeling techniques is dealt with in the next section. A more detailed overview of the semantic library can be found in [5].

In the SketchaWorld framework, the features from the semantic library provide the high-level vocabulary for the various forms of interaction, by means of which designers declare their intent for specific scenes and virtual worlds. For example, designers can define a natural landscape from its top view by brushing a grid with *ecotopes* (an area of homogeneous terrain and features). These ecotopes encompass both elevation information (elevation ranges, terrain roughness) and soil material information (sand, grass, rock, etc.). Designers can also sketch terrain features as roads, rivers or complete cities, by interactively indicating where and relative to what they are to be placed (see cover Fig). And a designer can also describe a desired scene or environment, by means of its semantic specification (from which various procedures are to be generated or invoked), indicating e.g. which objects should be placed inside this particular type of office, which ambiance this should be given, or what are the main conditions for its layout. In the next section we briefly describe how this declarative input of the designer is used to procedurally create a concrete and matching instantiation of the virtual world

One last aspect worth mentioning is that, in addition to its usefulness during the modeling process, semantically rich content can also play a very important role at runtime in many applications of virtual worlds, e.g. entertainment games, training simulators, etc. For instance, during a game or simulation, an artificial intelligence component can more easily reason over the virtual world using much of the virtual world semantics, defined in the modeling process, instead of relying only on separate, usually ad-hoc data structures, for planning paths and actions of virtual characters. Another example of applying object semantics at runtime is to improve the behavior semantics of objects, when user or virtual character interacts with it. For this purpose, we introduced the concept of *services* in our semantic library [6]. These services are specified in the available classes, again relying on its parent-child relationships and class attributes to make their definitions inherited and reusable. For instance, every physical object that is heavy enough and not too big, can be used to e.g. break open a window in a game world. Instead of having to annotate every game object suitable for this purpose, in our semantic library this 'service' needs to be defined solely for the main *physical object* class, applying only to classes that comply to stated weight and size constraints. The concept of services enables designers to easily create virtual world objects that are *aware* of each other's services, and *behave* as one would reasonably expect; as such, it is an important step towards improving object interaction in a virtual world.

IV. Procedural generation

SketchaWorld allows one to create complete, highly detailed virtual worlds by simply stating one’s whishes. The combination of semantics and integrated procedural methods allows it to automatically generate a realistic virtual world that fully matches with the high-level declaration by the designer.

One of the innovative declarative input methods introduced by SketchaWorld is designated as *procedural sketching* [7], a novel paradigm which significantly increases the usability of procedural modeling techniques, and provides a fast and intuitive way to model virtual worlds. Procedural sketching allows designers to quickly specify terrain features and directly see the effects of the resulting procedural modeling operations. It lets designers interactively sketch their virtual world in terms of high-level terrain features; for instance, one can coarsely sketch out an area for a city or forest. These sketched features are then procedurally expanded by a variety of integrated procedural methods and properly fit into the virtual world, adhering to the feature’s semantics and relationships with surrounding features. Design of a virtual world is a creative and iterative process, therefore procedural sketching provides a short feedback loop between each sketch operation and the visualization of generated results. This, combined with unlimited undo and redo facilities, strongly encourages designers to experiment and quickly check the effects of their modeling operations.

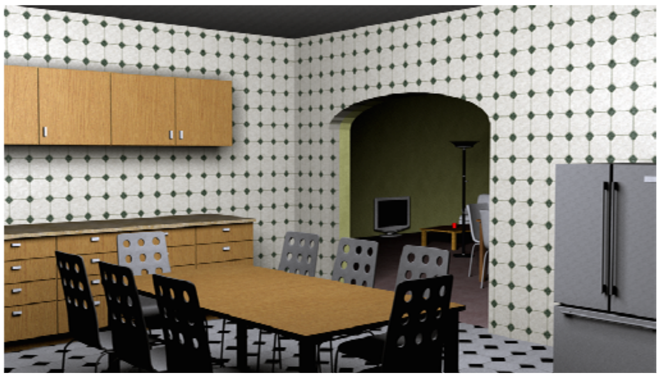


Fig. 2 The placement of the furniture in these rooms was handled by our layout solving approach based on the geometric relationships between the different feature classes defined in the semantic library.

On the procedural level (see Fig. 1), SketchaWorld integrates many procedural modeling techniques to automatically create a variety of content that match with designer’s intent, stated using either procedural sketching or any of the other forms of high-level scene descriptions mentioned above. We can distinguish three categories of procedural modeling techniques used: feature creation, feature placement and feature fitting.

Some of the features that are placed in the world include fixed geometric models. For many other features, however, their exact form and appearance are created with procedural generation techniques in order to better fit them in their context. For example, specific techniques are used to shape the landscape and rivers, to generate the facades of buildings or to alter a bridge to match requirements as length, supported weight, number of road lanes, etc.

In addition, to place heterogeneous groups of features or objects in the virtual world, SketchaWorld uses a *layout solving* approach [8]. As explained in the previous section, features typically contain geometric relationships that are used by the layout solver to find a valid location according to their semantics: where to place furniture inside a room, or where to place forests in a landscape. This layout solving approach was used to create the furniture layout of the kitchen and living room shown in Fig. 2. Designers can steer the placement of features, e.g. by providing a coarse layout with procedural sketching, by describing placement hints or by configuring other preferences. In any case, after automatic placement, designers can always intervene, where desired, by modifying the placement of individual features.

Traditional procedural generation techniques only generate one specific part of a virtual world [9]-[12]. Fitting all these generated features to form a consistent and lifelike virtual world is not a trivial matter. The influence of features on their surroundings goes both ways: a new feature adapts to fit in with its surroundings, but at the same time affects nearby existing features. For example, when a road feature is introduced into a virtual world, the road path needs to adapt to the landscape, e.g. to avoid steep slopes and impassible terrain; reversely, the local elevation profile of the landscape also needs to be adjusted in order to create a smooth road surface and embankment; and, in case it crosses a river, a bridge feature needs to be inserted and embanked too. Using the object relationships described in the previous section, our approach automatically fits all generated terrain features with their surroundings, and detects and solves conflicts arising among features. See [13] for more details on automatic virtual world consistency maintenance.

The consistency among all features in the virtual world model is automatically maintained not only throughout the execution of procedural modeling operations, but also when the designer chooses to manually edit parts of the world. Normally, such operations may have consequences for other surrounding features. For example: when moving a road, all adjacent lots and buildings typically need to move with them, and if the road becomes longer, new building features might need to be generated and fitted; or when a table is removed from a room scene, the plates and cutlery lying on it should likely be removed as well. Therefore, relationships among features expressing design intent need to be validated also after such manual edits, possibly involving a partial regeneration of an affected area of the virtual world.

By integrating both new and existing procedural generation techniques within the SketchaWorld framework, and transparently steering them through high-level operations on the virtual world model at the semantic level, the realism of the integrated output of those techniques is significantly improved. But above all, it elevates procedural modeling to the level of a real *declarative modeling* approach, effectively providing intuitive semantic concepts close to the level of designer’s intent.

V. Conclusions

Increasingly detailed and complex virtual worlds are nowadays commonly used in most entertainment and serious games, simulations, etc. We identified the main challenges and increasing demands currently faced by designers of virtual game worlds, and concluded that procedural modeling techniques only partially address them.

We proposed *declarative modeling of virtual worlds* as a novel approach, which more thoroughly and successfully solves these challenges. It combines the integrated use of various procedural modeling techniques with a semantics-driven model that effectively allows capturing designer's intent. As a result, designers of virtual worlds, using an intuitive vocabulary and familiar interaction, can concentrate on stating *what* they want to create, instead of on describing *how* they should model it. Several features of SketchaWorld, our prototype system that implements this approach, have also been discussed and illustrated.

Salient feature of this declarative approach is that it retains in the model of the virtual world much more information than simply its geometric data. On the one hand, this semantically-rich content facilitates providing designers with a declarative vocabulary that is more intuitive and much closer to their creative way of thought. On the other hand, once available for in-game use, such semantically-rich content can dramatically improve the interaction among objects, therefore improving gameplay as well.

SketchaWorld demonstrates that our declarative modeling approach provides designers with the productivity gain of procedural generation techniques, while still allowing for abundant control and flexibility. In addition, its framework provides us with a flexible base to integrate numerous techniques and algorithms from procedural modeling research. Finally, its interactive workflow significantly reduces the complexity of virtual world modeling, making it accessible to whole new groups of users and applications.

References

[1] R. M. Smelik, K. J. de Kraker, T. Tutenel and R. Bidarra, "A Survey of Procedural Methods for Terrain Modelling," *Proceedings of the CASA Workshop on 3D Advanced Media in Gaming and Simulation (3AMIGAS)* Amsterdam, The Netherlands, 2009, p. 15-24.

[2] G. Kelly and H. McCabe, "Interactive City Generation Methods," *SIGGRAPH '07: ACM SIGGRAPH 2007 Posters* San Diego, CA, USA, 2007.

[3] T. Tutenel, R. Bidarra, R. M. Smelik and K. J. de Kraker, "The Role of Semantics in Games and Simulations," ACM *Computers in Entertainment*, vol. 6, no. 4, pp. 1-35, December 2008.

[4] G. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39-41, November 1995.

[5] T. Tutenel, R. Bidarra, R. M. Smelik and K. J. de Kraker, "Using Semantics to Improve the Design of Game Worlds," *Proceedings of the fifth Artificial Intelligence and Interactive Digital Entertainment International Conference* Stanford, CA, USA, 2009, p. 100-105.

[6] J. Kessing, T. Tutenel and R. Bidarra, "Services in Game Worlds: A Semantic Approach to Improve Object Interaction," *Proceedings of the 8th International Conference on Entertainment Computing (ICEC)* Paris, France, 2009, p. 276-281.

[7] R. M. Smelik, T. Tutenel, K. J. de Kraker and R. Bidarra, "Interactive Creation of Virtual Worlds Using Procedural Sketching," TBP in: Proceedings of Eurographics 2010 - Short papers, 3-7 May, Norrköping, Sweden, 2010.

[8] T. Tutenel, R. Bidarra, R. M. Smelik and K. J. de Kraker, "Rule-based Layout Solving and its Application to Procedural Interior Generation," *Proceedings of the CASA Workshop on 3D Advanced Media in Gaming and Simulation (3AMIGAS)* Amsterdam, The Netherlands, 2009, p. 15-24.

[9] J. Gain, P. Marais and W. Strasser, "Terrain Sketching," *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games (I3D '09)* Boston, MA, USA, 2009, p. 31-38.

[10] M. de Villiers and N. Naicker, "A Sketching Interface for Procedural City Generation," *Technical Report, Department of Computer Science, University of Cape Town*, 2006.

[11] E. Bruneton and F. Neyret, "Real-time Rendering and Editing of Vector-based Terrains," *Computer Graphics Forum: Eurographics 2008 Proceedings*, vol. 27, no. 2, pp. 311-320, April 2008.

[12] O. Deussen, P. Hanrahan, B. Lintermann, R. Mech, M. Pharr and P. Prusinkiewicz, "Realistic Modeling and Rendering of Plant Ecosystems," *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* Orlando, Florida, USA, 1998, p. 275-286.

[13] R. M. Smelik, T. Tutenel, K. J. de Kraker and R. Bidarra, "SketchaWorld: A Declarative Framework for Procedural Modeling of Virtual Worlds," submitted for publication.