# Semantic Descriptions for Logical Content Generation

Thomas Smith
Ninja Theory, Ltd.
Westbrook Centre,
Milton Road,
Cambridge, CB4 1YG, UK
t.a.e.smith@bath.ac.uk

Julian Padget
Centre for Digital
Entertainment,
University of Bath,
Bath, BA2 7AY, UK
masjap@bath.ac.uk

Andrew Vidler
Ninja Theory, Ltd.
Westbrook Centre,
Milton Road,
Cambridge, CB4 1YG, UK
andrew.vidler@ninjatheory.com

## ABSTRACT

Human designers understand a range of potential purposes behind objects and configurations when creating content, which are only partially addressed in typical procedural content generation techniques. This paper describes our research into the provision and use of semantic information to guide logical solver-based content generation, in order to feasibly generate meaningful and valid content.

Initial results show we can use answer set programming to generate basic roguelike dungeon layouts from a provided semantic knowledge base, and we intend to extend this to generate a range of other content types. By using semantic models as input for a content-agnostic generation system, we hope to provide more domain-general content generation.

## Categories and Subject Descriptors

I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Logic programming*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods; K.8.0 [**Personal Computing**]: General—*Games*

## General Terms

Design, Performance

## Keywords

Answer set programming, knowledge representation, procedural content generation, semantics

## 1. INTRODUCTION

Research into procedural content generation (PCG) for games is a broad topic, however many implemented systems are highly bespoke and tightly coupled to either the game or the content they were initially designed to produce. This makes it difficult for developers unfamiliar with the workings of a generator to alter its output or repurpose it for other games or content types, limiting opportunities for the reuse of code and systems and the benefits this can bring.


Proceedings of the 10th International Conference on the Foundations of Digital Games (FDG 2015), June 22-25, 2015, Pacific Grove, CA, USA.
ISBN 978-0-9913982-4-9. 

In this paper, we address one of the 'grand goals' of PCG research suggested in a recent overview paper [10] — specifically, the difficulty of developing *Multi-level, Multi-content PCG*. We approach this through one of the concrete research challenges listed: *General Content Generators*. We hope to provide a step towards the development of plug-and-play content generation, by allowing designers to specify the requirements for content within a game in order to be able to automatically generate content for it.

Current PCG systems are often highly bespoke and encode implicit assumptions about the desired content in the design or implementation of the generator, which can make it difficult to predict precisely their expressive capabilities. The behaviour and output of a generator can also be challenging to alter without an understanding of the intention of particular features. In a similar manner, human designers encode implicit meaning in the placement of objects within a level, and alteration of human-designed levels can lead to unintended consequences if the rationale behind the original layout is not fully understood. These designer intentions are not explicitly captured by traditional editing tools, complicating attempts to procedurally alter content or generate levels with a similar degree of conceptual sophistication.

By allowing formal specification of the desired relationships between entities and concepts within the desired generated content, it should be possible to capture some of these implicit intentions. It should then be feasible to use modular logic programs based on common repair actions or typical designer approaches to iteratively refine a model of the content, from a high-level abstraction right down to the concrete placement of meshes and effects within the game engine. This approach would also support correcting, or suggesting alternatives to, designer-generated content. In-editor possibilities include highlighting violations of the provided constraints, such as completion path feasibility or obstacle placement rhythms, with potential for automated repair.

The availability of a truly general, modular content generator would reduce the time spent developing bespoke solutions for particular applications. The same proven code base could be reused across multiple projects, and familiarity with the system would transferable into each new context. In addition, any improvements to the capabilities or the efficiency of the system could provide benefit across all projects that used it. This research intends to provide a step towards developing reusable, meaningful content generation.

### 1.1 When is a door not a door?

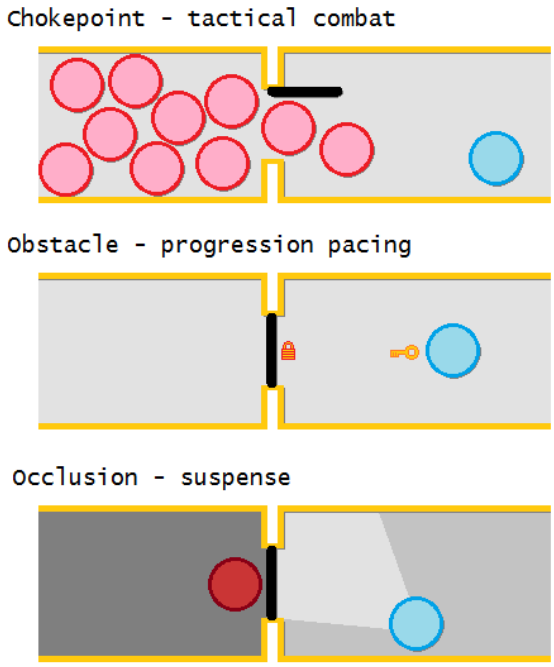


**Figure 1: Potential roles for a door in gameplay.**

As a concrete example, a door in a video-game level may serve multiple potential purposes. Depending on the game, it could be intended to provide a choke-point for oncoming enemies, an obstacle to progression until the correct key is found, a line-of-sight blocker intended to separate visible areas, all of the above or more.

Semantic annotation for these possible roles means that any or all of them could be taken into account during the generation process: in order to augment the combat experience, progression pacing, streaming efficiency, or a combination of these or other reasons. During the generation of a level, it may be desirable to use a door for reasons relating to the theme and appearance of the content — semantic annotation would allow a holistic approach to considering the impact that would have on other game aspects.

## 2. BACKGROUND

The proposed research draws upon advances detailed in related work in a number of areas: the development and evaluation of PCG systems, the use of answer set programming (ASP) for PCG, and the role of semantics in augmenting and guiding procedural generation. It also makes use of theories, knowledge representation formats and systems developed by researchers in the field of Semantic Web [3].

PCG is increasingly being considered as a tool to augment designers' creativity. A number of recent papers [6, 9] consider the concept of 'mixed initiative' generation, where the authorial burden is shared to some degree between the designer and the system. However, there are still issues with the close coupling between most generators and their games, and the resulting difficulties evaluating and comparing bespoke generator systems [4, 9].

Another active research area is the application of semantic data to improve consistency between appearance and affordance in virtual worlds [5]. For PCG, this would involve establishing of a layer of 'meaning' meta-data to capture and record designers' intentions explicitly. This could then be used to produce constraints on generation.

There is already a significant body of existing research on semantic knowledge representation in areas relating to the Semantic Web. There are a range of tools and formats which have been developed to specify ontologies describing classes, entities, and their relationships [1], however so far it appears that semantic data is almost exclusively used for the description of existing entities rather than the generation of new, valid information.

Answer set programming is a comparatively recent approach for PCG, which uses a logic solver to select valid outputs from a constraint-bound space of possible solutions [8]. Given an appropriate formalisation of the facts and rules associated with the search space, it can be guaranteed to select only instances that satisfy hard gameplay constraints such as connectivity and solution existence. This power comes at the cost of applying a range of domain-specific techniques to limit the potential combinatorial explosion in the answer set space [7].

One possible approach is to integrate a semantic model to guide successive, iterative generation steps; using a range of smaller ASP programs that gradually refine an abstract model of the generated content into the final output. Dormans [2] describes a similar, graph-based system using rewrite systems to iteratively refine a model of the desired content, prior to translation to a usable play space.

## 3. GOALS AND RESEARCH QUESTIONS

The overall goal of the research is to demonstrate the feasibility and efficiency of using logic-based solvers to generate a range of content from semantic descriptions of the desired generative space. As an initial target, we aim to show that general logic-based solver modules can be used with a suitable knowledge base of constraints and relationships to generate playable roguelike dungeon levels.

The proposed research will consist of two main tasks: the investigation of suitable formats and patterns for the provision of high-level semantic data about the desired properties of playable spaces, and the development and refinement of reusable generator components that can transform these semantic specifications into concrete playable worlds. Ultimately, it should be possible for designers to supplement a provided ontology of base concepts with refinements specifying classes and relationships within the desired content, and then use the solver system and this augmented ontology to automatically generate or verify and evaluate content that fits their specifications.

We can then demonstrate the versatility of our system by generating sample artefacts of a range of kinds and evaluating them against the output of other respected generators, as in [4]. There is also potential for live evaluation by professional game designers, who will be able to give feedback on the perceived usability of the system.

# 4. INITIAL STUDY AND PROPOSAL

The current proof-of-concept is based upon `dlvhex`, an ASP solver which supports plugins for external computation [3], and is able to read appropriate values from a supplied ontology and use these as input to an ASP layout solver for 2D roguelike dungeon levels. The use of ASP allows certainty that properly constructed constraints such as feasible paths from start to end will hold valid in all generated instances, whilst also potentially supporting more complex requirements such as a desired branching factor or the presence of loops and short-cuts within the game level.

## 4.1 `dlvhex`: ASP + computation

Though ASP provides useful guarantees regarding gameplay requirements, it is unsuited to some specific generation tasks such as solving geometric constraints. The `dlvhex` system supports plugins to provide external computation such as collision checking or path planning, and can also access external sources of data such as ontologies or other databases.

Both description logics (ontologies) and logic programming (ASP) provide reasoning-amenable formalisations of facts relating to a particular area, however they do so in different ways. One of the research challenges of this project will be developing ways to integrate the two approaches.

## 4.2 Further work

Further work currently includes defining a broader base ontology for a range of common cases, and improving solver support for these tasks. The system needs to support writing data back into the ontology for iterative refinement, and reading fixed facts for mixed-initiative generation. A transformation scheme would guide progression between tasks, and further consideration is needed for the method of translating solver output into playable content

Finally, system integration with an industry standard editor and an interface for editing semantic data and supporting mixed-initiative work would enable in-editor evaluation by game design professionals. It should be possible to annotate existing objects and compose or select design constraints with as little friction as possible in order to encourage experimentation and rapid iteration.

Development of a content-agnostic generator system as described would approach the goal of being able to produce 'plug-and-play' PCG middleware for a wide range of games and content types [10].

# 5. REFERENCES


[1] G. Antoniou and F. Van Harmelen. Web ontology language: OWL. In *Handbook on Ontologies*, pages 67–92. Springer, 2004.

[2] J. Dormans. Level design as model transformation: a strategy for automated content generation. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, page 2. ACM, 2011.

[3] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. dlvhex: A prover for semantic-web reasoning under the answer-set semantics. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 1073–1074. IEEE, 2006.

[4] B. Horn, S. Dahlskog, N. Shaker, G. Smith, and J. Togelius. A comparative evaluation of procedural level generators in the Mario AI framework. In *Proceedings of the 9th International Conference on Foundations of Digital Games*. Society for the Advancement of the Science of Digital Games, 2014.

[5] J. Kessing, T. Tutenel, and R. Bidarra. Designing semantic game worlds. In *Proceedings of the The Third Workshop on Procedural Content Generation in Games*, pages 2:1–2:9. ACM, 2012.

[6] A. Liapis, G. N. Yannakakis, and J. Togelius. Sentient sketchbook: Computer-aided game level authoring. In *Proceedings of the 8th Conference on the Foundations of Digital Games*, pages 213–220. ACM, 2013.

[7] A. J. Smith and J. J. Bryson. A logical approach to building dungeons: Answer set programming for hierarchical procedural content generation in roguelike games. In *Proceedings of the 50th Anniversary Convention of the AISB*, 2014.

[8] A. M. Smith and M. Mateas. Answer set programming for procedural content generation: A design space approach. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(3):187–200, 2011.

[9] G. Smith, J. Whitehead, and M. Mateas. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(3):201–215, 2011.

[10] J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. O. Stanley. Procedural Content Generation: Goals, Challenges and Actionable Steps. In *Artificial and Computational Intelligence in Games*, volume 6 of *Dagstuhl Follow-Ups*, pages 61–75. 2013.