Graph Grammars for Super Mario Bros * Levels

Santiago Londoño
University of Bonn
londono@cs.uni-bonn.de

Olana Missura
University of Bonn
olana.missura@uni-bonn.de

ABSTRACT

In side-scrolling games such as *Super Mario Bros*, the player interacts with a simplified, two-dimensional world made up of elements from a finite set. Conventionally, specialized designers build the levels of the game, applying their creativity and experience to produce content exhibiting structural correctness (the player must be able to traverse the level from start to end), interestingness and balanced difficulty, among other quality features.

Professional designers are able to consistently construct high quality game levels. In other words, their expert knowledge gets embedded in the resulting levels. We aim to create a learning mechanism capable of extracting design concepts from a set of human-authored levels and to use the acquired knowledge to algorithmically generate new ones. To that purpose we propose to use stochastic graph grammars in their both qualities, descriptive and generative. That is, first, given a set of levels, we will learn a grammar that encodes and abstracts structural properties presented in these levels together with their respective probabilities of occurrence. Next, the inferred grammar will be used to generate new levels possessing the same properties.

Diverse PCG techniques have been used to automatically generate game content since the early eighties [4]. The specific case of level generation for *Super Mario Bros* has also been explored. For instance, in [8] levels were represented as matrices and Markov Chains were employed to randomly generate new content. Although the matrix representation implicitly contains sound information about the elements of a level and their relations, it does so in a way that is too complex and expensive to analyze. This is a significant hindrance, as these relations are major determinants of the structure and quality of the level.

*Specifically, our application is aimed at generating levels for the framework *Infinite Mario Bros*, a clone of the classic *Super Mario World* originally published by Nintendo [9].

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.

From an analytical point of view, *Super Mario Bros* levels are composed of elements that are not isolated, but relate to each other in different ways. Moreover, an explicit and succinct representation of the levels is fundamental for the development of effective algorithms for learning and procedurally generating content. Therefore, we represent levels as *directed graphs*, where nodes are the elements making up the level and edges encode their semantic relations.

We classify all level elements of *Super Mario Bros* in various types, according to their interaction with the player. In the current iteration of the project, we have defined two types, yet more may be devised after we get our first results.

Platforms: are groups of solid sprites (e.g. brick blocks, rocks, pipes) that are consecutive and lie on the same row of the level matrix. They form solid positions upon which the character can stand and whence he can reach other elements. In a level graph, platforms are represented by edges, labeled as *platform*.

Item clusters: represent conglomerates of sprites that are not solid, but still interact with the player (e.g. coins, power-ups, enemies). If the cluster contains more than one sprite, each one of them has at least one neighbor. That is, another sprite in the same cluster located at a distance lower than an established threshold. Item clusters are represented within level graphs, by sets of nodes labeled as non-solid sprites. Each of these nodes is connected with its closest neighbor, through an edge labeled as *cluster*.

The concept of *reachability*, fundamental to our approach, is also described in terms of a relationship between nodes. In the graph abstraction, reachability between a platform and other platforms or item clusters, is represented as an edge with label *reach*.

In order to obtain the graph representation of the example levels we will learn from, we implemented an algorithm to transform a level in matrix form (i. e. grid), into a structurally equivalent graph and vice-versa.

We hypothesize that human-authored, high-quality levels contain design paradigms reflecting the knowledge and experience of their designers. Moreover, we assume that this knowledge is encoded in a way representable by graphs. Having this representation of levels, graph data mining techniques can be applied to extract the knowledge embedded

in a set of human-authored examples.

String grammars have been previously used to procedurally generate game levels. Two notable examples are presented by Dormans [3] and Shaker et al. [7]. Dormans [3] builds level layouts for adventure games over mission scripts. First, a human-authored, context-free grammar is used to generate the mission as a series of actions and thereafter, a concordant level is built by means of a space grammar. Shaker et al. generate *Super Mario Bros* levels. The authors use genetic programming to search the space of levels derivable from a predefined grammar in a combined approach called Grammatical Evolution [7].

Compton and Mateas [1] model platformer levels by means of context-free string grammars. The grammars define how a set of patterns of varied complexity, can be put together to give form to a complete level. These patterns represent optimal sequences of basic game components, automatically built through a hill-climbing algorithm. The PCG process is graph-grammar driven and in principle similar to our approach, as an initial graph is recursively expanded to form a new level, following the grammar rules. However, their approach assumes that the grammars are provided and no mechanisms to automatically learn them are discussed.

In this work we go a couple of steps further by using graph grammars to encode the relationships between various elements of a level and by learning these grammars instead of requiring them to be predefined. Specifically, we implemented the *SubdueGL* algorithm [5]. SubdueGL is based on the *Subdue* algorithm originally proposed by Cook and Holder [2] for the discovery of frequent substructures in graphs, where a *substructure* is a subgraph that can occur one or more times in the input graph. Given a set of graphs, SubdueGL learns a Node Label Controlled (NLC), context-free graph grammar, striving to achieve an optimal balance between the size of the substructures and their frequency of occurrence. This criterion is called the *MDL principle* [2]. The induced graph grammars are expected to define patterns frequently observed in high-quality levels.

To accurately represent the occurrences of found substructures, we extend SubdueGL to learn stochastic NLC graph grammars. This is achieved by introducing some additional computations into the learning process, to get a maximum-likelihood estimate for the "probability of applying" each derived production. We base our extension on the approach described by Oates et al. [6].

The level generation algorithm creates a new graph using the learned grammar: It expands an initial graph (e.g. a single non-terminal node representing the start of the level) by iteratively applying the productions of the stochastic NLC graph grammar. On each iteration, the algorithm replaces a non-terminal node with a subgraph. To do so, it selects a suitable production of the graph grammar, in accordance with their probabilities. In the next step, the so generated graph is transformed into a level grid and additional details such as background sprites and decorations are rendered onto it.

The PCG system we described is still work-in-progress. We have already developed the representation of levels as graphs and the graph grammar learning mechanism, but the content generation module is still under development. We plan to perform an evaluation to verify whether the generated levels preserve the quality features of the examples and at the same time, are novel enough to amuse human players. To do so, we intend to use metrics proposed by Shaker et al. in [7] to compare the levels used as training instances with the newly generated levels.

Furthermore, we would design an experiment involving human players to qualitatively evaluate how enjoyable are our results and to compare our system against existing level generators for *Infinite Mario Bros*, namely (*i*) the *default generator* included as part of the framework, as developed by Markus Persson; (*ii*) the *GE generator* by Shaker et al. [7], based on grammar evolution; (*iii*) the *winner of the Mario AI Championship 2012* [10], an approach driven by the player's score, as presented by Chen et al.

1. REFERENCES

[1] K. Compton and M. Mateas. Procedural level design for platform games. In *AIIDE*, pages 109–111, 2006.

[2] D. J. Cook and L. B. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, pages 231–255, 1994.

[3] J. Dormans. Adventures in level design: generating missions and spaces for action adventure games. In *Proceedings of the 2010 workshop on procedural content generation in games*, page 1. ACM, 2010.

[4] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 9(1):1, 2013.

[5] I. Jonyer, L. B. Holder, and D. J. Cook. Mdl-based context-free graph grammar induction and applications. *International Journal on Artificial Intelligence Tools*, 13(01):65–79, 2004.

[6] T. Oates, S. Doshi, and F. Huang. Estimating maximum likelihood parameters for stochastic context-free graph grammars. In *Inductive Logic Programming*, pages 281–298. Springer, 2003.

[7] N. Shaker, M. Nicolau, G. N. Yannakakis, J. Togelius, and M. O'Neill. Evolving levels for super mario bros using grammatical evolution. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 304–311. IEEE, 2012.

[8] S. Snodgrass and S. Ontañón. Experiments in map generation using markov chains. In *Proceedings of the 9th International Conference on Foundations of Digital Games, ser. FDG*, volume 14, 2014.

[9] J. Togelius, E. Kastbjerg, D. Schedl, and G. N. Yannakakis. What is procedural content generation?: Mario on the borderline. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, page 3. ACM, 2011.

[10] J. Togelius, N. Shaker, S. Karakovskiy, and G. N. Yannakakis. The mario ai championship 2009-2012. *AI Magazine*, 34(3):89–92, 2013.