Integrated system for automatic platform game level creation with difficulty and content adaptation

Fausto Mourato, Manuel Próspero dos Santos and Fernando Birra

Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa, Portugal
p21748@campus.fct.unl.pt, {ps, fpb}@di.fct.unl.pt

Abstract. This article presents an overview over our system for the creation of platform game levels. It consists of a framework with a generic and flexible approach that integrates most of the concepts that can be found in this type of games. In addition, some procedural techniques are employed allowing automatic level generation, dynamic difficulty adjustment, optional content creation and item gathering or triggering related challenges. The system can be extended by adding new plugins to support other games or generation algorithms.

1 Introduction

In the last few years, *Procedural Content Generation* attracted the attention of researchers, in particular under the context of videogame development. In this document we will focus content generation for platform videogames like *Super Mario Bros.* and *Sonic – The Hedgehog*. Generating levels for a platform game raises some complex aspects such as level viability, challenge suitability and aesthetic coherence. An interesting ongoing commercial work based on these topics is the game *Cloudberry Kingdom* [2], with focus on the generation of valid and challenging gaming environments. Our particular system, hereby presented, directs attention to the inclusion of optional content and contextualized challenges. For the scope of this article, we will focus the following contributions:

- Generic framework that supports level representation for multiple existing games.
- Configurable rule-based structure for graph generation and path estimation.
- Multi-step algorithm featuring a novel approach to complement level content.
- Plugin based system that allows the integration of further generation techniques.

2 Related Work

Primal work related with platform games focused the identification of the main features in this type of games and how the global context could be characterized, in works presented, respectively, by Compton and Mateas [1] and Smith et al. [8].

Following these ideas, first efforts on creating prototypes that could create platform game levels appeared. Two different approaches are particularly noteworthy, namely

rhythm and chunk based. Rhythm based generation consists on creating levels to follow an adequate input or movement pattern [9]. For other studies related to this idea we refer to [10]. The chunk based generation consists on procedurally combining pre-authored small level parts, designated as chunks [4].

Personalization in this field has also been tried, in particular to perceive user feeling [6] and match user skills to inherent level difficulty [3].

3 System Overview

3.1 Level representation

For the purpose of this work we will concentrate on PC versions, considering also some console inspired implementations. In particular, our system is suitable for the following games:

- *Infinite Mario Bros.*, an open source adaptation of the classic videogame *Super Mario Bros.*, frequently used as a research tool [7].
- The original videogame *Prince of Persia*, which allows interesting considerations in trigger related challenges.
- *XRick*, a remake of the original videogame *Rick Dangerous*.
- *Open Sonic*, an adaption of the game saga *Sonic – The Hedgehog*.

In its basis, a level is represented as a two-dimensional matrix. This discretized approach brings some advantages on later processes such as the analysis of possible paths. Each cell can be filled with a certain *Block Type* from a hierarchy defined by the game designer. The system allows defining *Groups* to refer particular cells that only make sense when used together, such as pipes in *Infinite Mario Bros.* (Fig. 1). To allow representing additional specific features, such as the ramps and loops of *Open Sonic*, it is also possible to define objects that fill a range of cells, which have been designated as *Contraptions*. In addition, *Categories* might be defined representing a set of blocks calculated using logical operators. For instance, in *XRick*, one can define a category named "support" with the expression "solid block ∨ platform".

3.2 Analysis

A path related level analysis tool is provided by the system. The designer defines a set of rules for the creation of a graph representing a network of available paths for the avatar. A rule is defined with a cell pattern and the corresponding graph entries. For instance, in *XRick*, one can define the basic horizontal movement with the pattern presented in Fig. 2. The lower two cells represent the *category* support previously referred. In the whole level, for every occurrence of each pattern, the corresponding entries are added to the level graph. In the end it is possible to observe the final graph as shown on Fig. 3 in the context of the game *Prince of Persia*.

3.3 Automatic Level Generation and Content Adaptation

The system includes the possibility of incorporating algorithms for automatic level generation, based on a plugin architecture, to ease the development of further techniques. Currently we have tested simple heuristic generators and the genetic algorithm implementation presented in [5].

In order to improve level content we have also been developing some algorithms to adapt a basic level structure into a more complex gaming environment. Currently, the system analyses the produced graph and detects which nodes are mandatory and optional in the avatar path from the beginning to the end of the level. Also, dead end segments are detected and are used to provide power ups, collectibles or specific items that will be needed in the main path, improving content richness. Finally, difficulty is estimated based on graph transitions in a similar approach as proposed on [10], allowing an iterative process to search for a desired difficulty value.

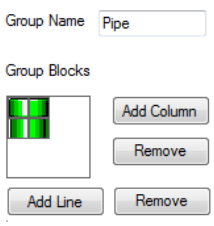

Fig. 1. Pipe in *Infinite Mario Bros.*, defined by a group of 2x2 cells

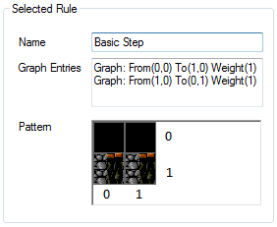

Fig. 2. Example rule defining possible bidirectional horizontal movement in *XRick*.

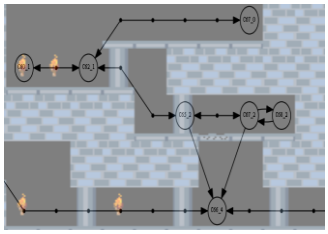

Fig. 3. Graph generated for the first level of the game *Prince of Persia*

4 Usage Examples

In Fig. 4 we show a screenshot of the system's main window depicting a level for the game *Infinite Mario Bros.*. That level was generated with a plugin that implements a simple random algorithm. Fig. 5 presents again the main window, now displaying a level from the game *XRick*, extracted from the original game with a level importer plugin. Finally, Fig. 6 shows a level for the game *Prince of Persia*, created with a genetic algorithm (left part) and followed by the content adaptation process presented in the previous section (right part).

5 Conclusions and Future Work

We presented a system that features a flexible and adaptable representation scheme that is compatible with different platform games. The architecture is extensible to allow the creation of additional plugins for different games or to implement new procedural content generation techniques. Some of the included algorithms are still under development but, however, the results are motivating. On the next steps we aim to extend the set of compatible games and also to expand the generation algorithms to include personalization, namely integrating user data obtained from gaming sessions.

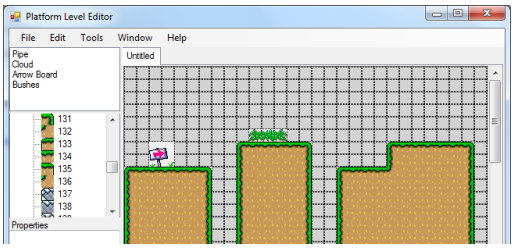

Fig. 4. Sample of an *Infinite Mario Bros.* level created with our system

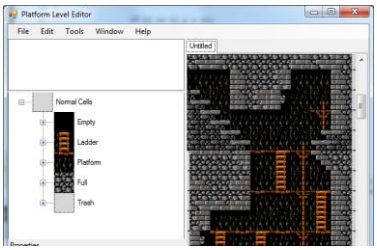

Fig. 5. Editing a level for the game *XRick*

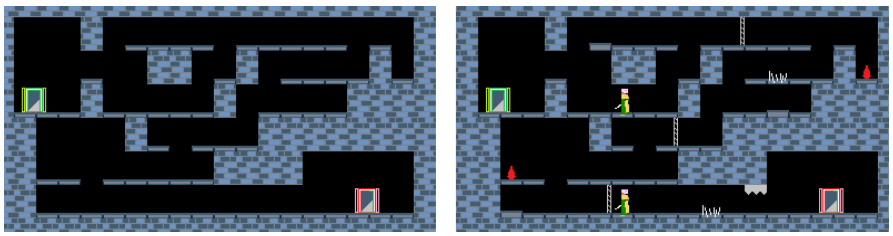

Fig. 6. Two steps on a composed generation process for the game *Prince of Persia*.

6 References

1. K. Compton, M. Mateas. Procedural level design for platform games, *Proc. of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2006.
2. J. Fisher, How to Make Insane, Procedural Platformer Levels, *www.gamasutra.com*
3. M. Jennings-Teats, G. Smith, N. Wardrip-Fruin. Polymorph: Dynamic Difficulty Adjustment through Level Generation. *Proc. of the Workshop on PCG in Games*, 2010.
4. P. Mawhorter, M. Mateas. Procedural Level Generation Using Occupancy-Regulated Extension. *CIG-2010 - IEEE Conference on Computational Intelligence and Games*, 2010.
5. F. Mourato, M. Próspero dos Santos, F. Birra. Automatic level generation for platform videogames using Genetic Algorithms. *ACE 2011, 8th international conference on advances in computer entertainment technology*, 2011.
6. N. Nygren, J. Denzinger, B. Stephenson, J. Aycock. User-preference-based automated level generation for platform games. *IEEE Symposium on Comp. Intelligence and Games*, 2011.
7. N. Shaker, J. Togelius, G. N. Yannakakis, B. Weber, T. Shimizu, T. Hashiyama, N, Soreson, P. Pasquier, P. Mawhorter, G. Takahashi, G. Smith, and R. Baumgarten. The 2010 Mario AI Championship: Level Generation Track, *Special Issue of IEEE Transactions on Procedural Content Generation*, 2010
8. G. Smith, M. Cha, J. Whitehead. A Framework for Analysis of 2D Platformer Levels, *Proceedings of the 2008 ACM SIGGRAPH symposium on video games*, 2008.
9. G. Smith, M. Mateas, J. Whitehead, M. Treanor. Rhythm-based level generation for 2D platformers, *Proc.s of the 4th International Conference on Foundations of Digital Game*, 2009.
10. G. Smith, J. Whitehead, M. Mateas, M. Treanor, J. March, M. Cha. Launchpad: A Rhythm-Based Level Generator for 2D Platformers. IEEE Transactions on Computational Intelligence and AI in Games (TCIAIG), vol. 3, iss. 1, March 2011.