Generative spaces: assisting the level designer in creating world maps

Abstract

Creating two-dimensional tile maps by hand is a long and tedious process. Tile maps contain an inherent structure and repetition that we should be able to exploit. We provide a way of generating these maps automatically using machine learning techniques while still allowing the level designer to specify the high level spatial structure of the world. With these set of tools level designers will be able to create worlds that are much larger in scope.

1 Introduction

The development of a video game requires a great variety of assets. A typical game combines music, art and code to form a coherent whole.

This need for resources is one of the reasons why games are so costly to develop. Procedural content generation—the automatic creation of content-has emerged as an answer to this problem and is currently widely used, particularly in independent studios with lower budgets. For example, *No Man's Sky (2016)* is a space exploration game that uses procedural generation to generate a whole universe of over 18 quintillion planets $1.8 \cdot 10^{19}$, suffice to say that you would need many lifetimes to explore them all.

Most approaches to procedural generation are rule based, meaning that they rely on the definition of explicit rules. L-systems are an example of such a rule based system. they allow the creation of tree-like structures. TODO: Reference to an image of an L-system. The limitations of rule based systems lies on the fact that they cannot learn from already existing examples and reproduce them. We would not be able to show an L-system images of oaks for example and expect it to reproduce trees in the same style. We would need to define exactly the rules that represent "oakness", the attributes that make an oak an oak. TODO: citation L-systems

Data-driven techniques for procedural generation are still relatively rare; in this paper we try to see whether it is possible to reproduce the spatial style of a given game using data-driven techniques.

- We show how 2d Markov chains can be applied to the generation of 2d tile-maps. Section 2 provides a few background information on what tile-maps are and their current uses while Section 4.1 describes the technical side of their implementation.
- We argue that the generated maps keep the spatial style of the originals by comparing compare how different input maps influence the generated maps in Section 4.2.
- We come up with a backoff smoothing method appropriate for 2d chains. First we explain why it can be useful in Section 3 and then we give the technical details of its implementation in Section 4.3.
- We provide a C# implementation of a text based 2d Markov chain which is accessible at the following link.[1]

2 Background

- What exactly is a tile map
- Image of a tilemap
- Explain why it is time consuming to create manually
- Show how it is still relevant today by showing all the indie games that use them today.

Tile maps are a staple of many genres of games, they are formed by individual tiles of predefined size and assembled into a grid. Traditionally they are built manually: the level designer chooses one tile for each cell of the grid. Given that the grids can be very large, we can easily imagine that this is a time-consuming process. It puts a human limit into the size of the maps that can be used in a game. Being able to generate these maps procedurally would lift a burden off the shoulders of the level designers and allow the generation of much larger worlds.

3 The idea

describe the idea in broad terms - take a set of maps as an example - generate a Markov chain based on the examples - explain why it should retain the spatial structure of the game -> analogy with Markov chains for text generation. - explain the lack of data problem - explain how backoff smoothing helps

[1] https://github.com/stonecauldron/markov2d

4 Technical details

4.1 2D Markov chains

- kernel explanation
- choice of kernel

4.2 Spatial style conservation

4.3 Backoff smoothing in 2d

5 Related work

Game companies generally do not publish their procedural content generation methods so it is hard to evaluate what is common in the industry. In academia, researchers have looked into various approaches for automatic level generation. The game *Super Mario Bros.* in particular has received much attention.

Shaker et al. (2012) uses generative grammars combined with an evolutionary algorithm to create *Mario* levels that adapt themselves to the player's experience. In another insightful paper, Sorenson and Pasquier (2010) also uses an evolutionary algorithm to create a generic framework for level creation. Their approach allows the level designer to enforce multiple constraints on the generated levels making it adaptable to multiple genres of games.

Rather than being evolutionary, our method draws from machine learning techniques. The designers do not have to explicitly articulate rules about their designs but simply need to provide the system with examples of what they want to create.

Papers using machine learning methods to generate content are still a minority, bu the trend seems to be on the rise. Our use of Markov chains is directly inspired by the work of Snodgrass and Ontañón (2013) and Dahlskog, Togelius, and Nelson (2014). They both use Markov chains to create *Mario* levels with very convincing results. The difference with our approach lies in the fact that *Mario* levels have a linear nature. The maps we focus on are non-linear; the player can move through both dimensions.

Adam J Summerville and Mateas (2015) explore non-linear level generation in the context of dungeons for the game *The Legend of Zelda.* Their use of a graph to represent the game space comes from Dormans (2010). Both papers gave us the idea of using a graph to define the high level structure of the world. Our focus is more on world maps rather than dungeons. Dungeons have a clear compartmentalization of individual rooms while world maps have a more organic unfolding of space. Furthermore, dungeons have elements such as locked doors that restrict the order in which the player can traverse the rooms. World typically do not have these types of constraints.

Finally, the video game level corpus provided by Adam James Summerville et al. (2016) encouraged us to contribute to the corpus with our own extracted data and gave us a set of guidelines to follow when formatting the data.

6 Conclusion and further work

- conditional random fields
- limits of procedural generation

Can we automate the holistic feel of a level? How does one recreate themes in design works successfully? Can machine learning capture the essence of something?

References

Dahlskog, Steve, Julian Togelius, and Mark J. Nelson. 2014. "Linear levels through n-grams." *Proceedings of the 18th International Academic MindTrek Conference on Media Business, Management, Content & Services - AcademicMindTrek '14*, 200–206. doi:10.1145/2676467.2676506.

Dormans, Joris. 2010. "Adventures in level design: generating missions and spaces for action adventure games." *… Workshop on Procedural Content Generation in Games*, 1–8. doi:10.1145/1814256.1814257.

Shaker, Noor, Miguel Nicolau, Georgios N. Yannakakis, Julian Togelius, and Michael O'Neill. 2012. "Evolving levels for Super Mario Bros using grammatical evolution." *2012 IEEE Conference on Computational Intelligence and Games, CIG 2012*, no. 1: 304–11. doi:10.1109/CIG.2012.6374170.

Snodgrass, Sam, and Santiago Ontañón. 2013. "Generating Maps Using Markov Chains." *Aiide*, 25–28. http://www.aaai.org/ocs/index.php/AIIDE/AIIDE13/paper/viewPDFInterstitial/7447/7632.

Sorenson, Nathan, and Philippe Pasquier. 2010. "Towards a Generic Framework for Automated Video Game Level Creation Applications of Evolutionary Computation" 6024: 131–40 ST–Towards a Generic Framework for Auto. doi:10.1007/978-3-642-12239-2_14.

Summerville, Adam J, and Michael Mateas. 2015. "The Learning of Zelda : Data-Driven Learning of Level Topology," no. Fdg.

Summerville, Adam James, Sam Snodgrass, Michael Mateas, and Santiago Ontañón. 2016. "The VGLC: The Video Game Level Corpus." *Arxiv*. http:

//arxiv.org/abs/1606.07487.