Probabilistic Foundations for Procedural Level Generation

Sam Snodgrass
Drexel University, Department of Computer Science
Philadelphia, PA, USA
Sam.PSnodgrass@gmail.com

Abstract

Procedural content generation (PCG) has become a popular research topic in recent years, but not much work has been done in terms of generalized content generators, that is, methods that can generate content for a wide variety of games without requiring hand-tuning. Probabilistic approaches are a promising avenue for creating more general content generators, and specificially map generators. I am interested in exploring probabilistic techniques that could lead to generalized procedural level generators.

Introduction

Games have been gaining a wider and wider audience in recent years. With the rise of independent developers, there are more people creating games than ever before. However, not all developers have the time or money to create the enormous games that the big companies can provide. Procedural content generation (PCG) allows for the creation of large amounts of content, while avoiding the overhead of manual authoring. PCG is the automatic generation of content through the use of algorithms (Togelius et al. 2011). Procedural content generation provides replayability to games, can save developers time and money, by allowing them to partially rely on generated content instead of on hand authored content, can allow the creation of new game genres, and can also contribute to the design of adaptive games.

I want to explore probabilistic approaches in order to develop generalized PCG techniques, or PCG techniques that are applicable to a wide range of domains. There has been a lot of work done in the field of PCG (Togelius et al. 2011). However, many of the procedural content generation techniques have been developed with specific domains (particular genres or games) in mind. I would like to develop a generator that is able to create levels or maps for any genre of game (e.g., puzzle, strategy, shooter, etc.). I believe that probabilistic approaches, such as Markov chains (Markov 1971), open a path towards generalized map generators.

Procedural Level Generation

Procedural level generation can be used to add replayability and unpredictability to a game, or to avoid hand authoring a large amount of content. There is a large corpus of work in the field of procedural map generation. In particular, the platformer game genre has received a lot of attention from researchers, to the point that there is even a map generation competition which uses a platformer game as its testbed (Shaker et al. 2011). It is not within the scope of this paper to outline all the map generation techniques developed, but I will cover two different examples. First, an interesting technique, related to platformer games, is proposed by Smith et al. (Smith et al. 2009). Smith's approach generates platformer levels that enforce a certain rhythm for the player. This approach relies on the set of available player actions as well as a set of designer defined metrics in order to achieve the desired rhythm for the map. A different approach, from the 2010 Mario AI Championship: Level Generation Track (Shaker et al. 2011), Takahashi and Smith generate maps for *Super Mario Bros.* that are tailored to player's playstyle and skill. The levels are generated stochastically, using hand tuned probabilities based on the player's style and skill. In addition to generators for platformer games, generation techniques for real-time stategy games have been developed (Togelius et al. 2010) (Uriarte and Ontanón 2013), as well as for dungeon crawlers (e.g., *Spelunky* and *Diablo 3*). Those are just a few examples. For a more broad view of PCG the reader is referred to surveys by Hendrikx et al. (Hendrikx et al. 2013) and Togelius et al. (Togelius et al. 2011). Moreover, but for a few exceptions (Sorenson and Pasquier 2010; Sorenson, Pasquier, and DiPaola 2011), there has not been much work towards generalized content generators. As part of my research I plan to explore the possibility of using probabilistic approaches and machine learning for this purpose.

Evaluation

There are two major groups of evaluation techniques: human-based techniques, and metric-based techniques (Hoeft and Nieznańska). Human-based techniques typically refer to play testing, but also include techniques such as forced rank evaluations (Shaker, Smith, and Yannakakis 2014) and physiological measures during gameplay (Mandryk and Inkpen 2004; Drachen et al. 2010). These techniques need to be done offline, because the content needs to already be generated in order for a player to give feedback on it. Intuitively, humans are good judges of enjoyable content, but special care must be taken when using

self reporting techniques as biases may be introduced into the data, depending on how the content is presented to the testers, the size of the group of people used for evaluation, etc. Metric-based techniques refer to automatic methods that evaluate content based on a set of given parameters. Metric-based techniques can be online or offline. For instance, evolutionary algorithms rely on a fitness function (online) (Togelius et al. 2011), which measures the quality of a generated member of the population in order to guide the algorithm towards a better solution. Alternatively, Shaker et al. (Shaker, Smith, and Yannakakis 2014) propose using a metric-based technique involving heatmaps in order to measure the expressive range of the content generator (offline). Care must be taken when using metric-based approaches as well. Using metrics that are similar to the input parameters of the generator can lead to skewed results, and also cannot provide much insight into the quality of the generator. At best, the evaluation would reveal that the generator behaves as intended, but would not pick up on unexpected behavior (Shaker, Smith, and Yannakakis 2014).

The aforementioned evaluation techniques evaluate the content generated by a system. However, it is important to evaluate the quality of the generator as well. For instance, two systems may be able to generate the same content, but one system may be preferable over the other, based on other metrics (e.g., running time, memory usage, etc.). I would like to devise evaluation techniques for the generators themselves, instead of the content being generated, and see what kind of guarantees different systems can offer.

Research Plan

I have applied simple machine learning techniques to the problem of procedural map generation (Snodgrass and Ontañón 2014). I believe that machine learning techniques are a promising approach to a generalized procedural map generator. In order to achieve the goal of a generalized generator, I need to accomplish the following:

1. **Identification of Techniques**: I need to determine which techniques (in addition to Markov chains) are applicable to PCG, and techniques which are ill suited to PCG. Additionally, I need to decide which of the applicable techniques will further my goal of generalization.
2. **Testing**: Once I have decided on a set of technique I need to apply the techniques to a few different domains, in order to ascertain whether my belief in the applicability and generality of the technique is well founded.
3. **Abstraction**: Maps from different genres (or even different games within the same genre) can have radically different representations. For example, *Spelunky* and *Diablo III* are both dungeon exploration games, but Spelunky uses a two-dimensional, retro-style, side view map using (relatively low resolution) squares as the primary building blocks, whereas Diablo III uses a 3-dimensional, semi-realistic, top-down view map with a high resolution. In order to learn from and generate maps in these two games, I will need to be able to automatically abstract the maps into some uniform representation. As an alternative to abstracting the maps, the system itself could be abstracted so that it learns and generates at a very high level or is able to automatically determine the important qualities of a given set of maps.
4. **Evaluation**: If I am able to abstract the input, then the system can generate the maps in the same abstracted format. Therefore, a generalized evaluation technique would be a technique that is able to accurately and reliably determine the quality of the generated abstracted maps. However, if instead, I abstract the generation technique and not the data, I would need to devise a set of measurements that are general enough to apply to any kind of map, or a way to automatically determine which type of map is being evaluated, and evaluate that map based on a set of metrics specific to that type of map. In addition, I will need to devise evaluation techniques for the generators themselves, independent of the content being generated. Note that some work has been done in creating more general content evaluation techniques. For example, Liapis et al. (Liapis, Yannakakis, and Togelius 2013) propose a general map evaluator based on the properties of balance, area control, and exploration.

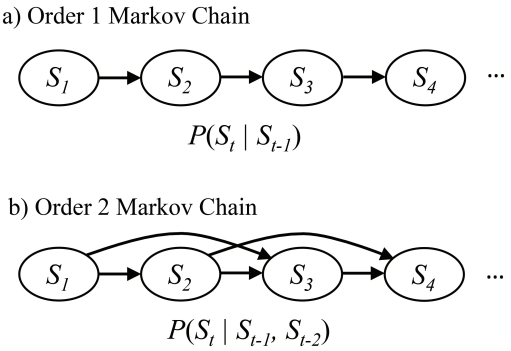

Figure 1: Visual representation of a standard Markov chain (top) and an order two Markov chain (bottom).

Regardless of the path taken, abstraction is going to be an important part of this process.

Progess

In my current research I have made progress in the *Identification*, *Testing*, and *Abstraction* steps outlined in the previous section. I have not explored the *Evaluation* step much yet.

Identification

I have identified Markov chains (Markov 1971) as a viable approach to learning map structures. Markov chains are a method of modeling probabilistic transitions between different states. A Markov chain consists of a set of states $S = (s_1, s_2, ..., s_3)$, and a conditional probability distribution (CPD) $P(S_t|S_{t-1})$. This corresponds to the probability of moving to the state represented by S_t given that the previous state was equal to the value of S_{t-1}. Additionally, more previous states can be accounted for in a higher-order Markov chain. In this case, the set of states is the same but

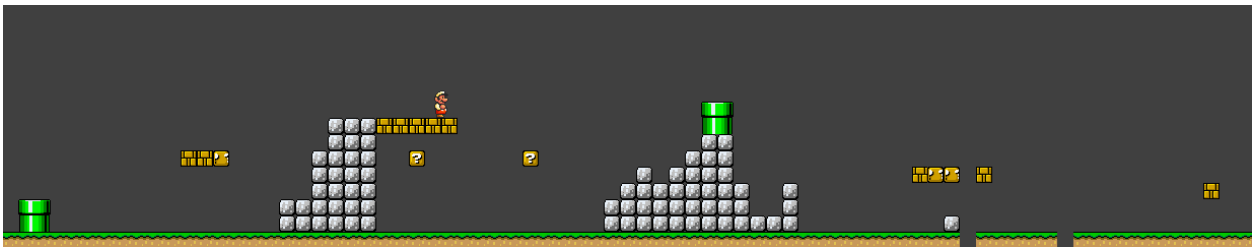

Figure 3: A section of a map generated using our Markov chain technique.

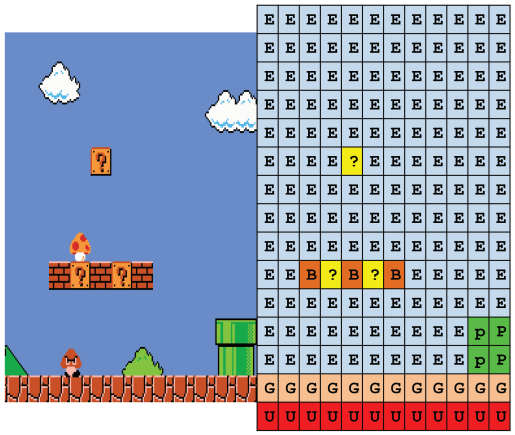

Figure 2: A sample encoding of a section from a *Super Mario Bros.* map. Color has been added for clarity.

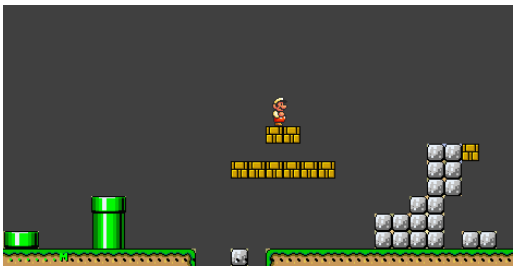

Figure 4: A section of a map generated using our hierarchical Markov chain technique.

the CPD is defined as $P(S_t|S_{t-1}, S_{t-2}, ..., S_{t-k})$, where k is the number of previous states taken into account. Figure 1 shows a visual representation of a standard and an order two Markov chain. With relation to learning maps, if the maps are generated as a set of tiles, then each different tile would correspond to a state, and we would learn the probability of generating some tile given the previous tile. This approach is promising because all maps contain some form of dependencies between local sections. These dependencies could be learned using Markov chains. Though my current work has been with Markov chains, I am still looking for other probabilistic approaches to apply and test.

Testing

I decided to use *Super Mario Bros.* as the inital testbed for my technique. Maps in Super Mario Bros. are easily encoded as tiles, because it is a 16-bit game. Figure 2 shows a sample encoding of a section from a Super Mario Bros. map. We tested our Markov chain technique using a set of maps from the original Super Mario Bros. and have had some success (Snodgrass and Ontañón 2014). Figure 3 shows a section of a map generated using our Markov chain approach. In the future, I plan on applying my method (and future methods) on games with varying levels of complexity, and varying gameplay styles, such as puzzles games, strategy games, etc.

Abstraction

We have expanded on our Markov chain technique by implemented a hierarchical Markov chain approach. That is, after encoding the maps as a set of tiles, we further encode the tilized maps as a set of high-level tiles corresponding to different structures within the maps (e.g., pipes, slopes, platforms, etc.). This abstraction technique should be applicable to a large class of maps. Though the paper I am presenting at this conference shows a technique using hand coded abstraction rules, I am working on an extension that will automatically detect the high-level structures by employing unsupervised machine learning techniques, further automating and adding to the generalizability of the system. Figure 4 shows a section of a level generated using this hierarchical technique (from our paper being presented at this conference).

Evaluation

In my previous work, I have used metric based approaches to evaluate the quality, and I have used character agents in order to test playability of the maps generated by our system. However, the metrics I used have been tailored to the domain (Super Mario Bros.). In my future work, I intend to apply my system to different domains, such as, *Lode Runner* and *Rainbow Island*, as well as different genres of games, such as puzzle and strategy games, which will force me to devise more general evaluation metrics and techniques.

Conclusion

Procedural content generation techniques can be useful to any game company, but small game companies can benefit the most, as they typically have much smaller budgets. PCG can save developers hours spent on content creation, and can add replayability and unpredictablity to their games.

More generalized methods are particularly beneficial, because generalized methods will be applicable between different games and genres. Therefore, the studio can spend some time creating a strong, generalized content generator, and then use that generator for a plethora of new games, in a similar way to how game engines have been used.

References

Drachen, A.; Nacke, L. E.; Yannakakis, G.; and Pedersen, A. L. 2010. Correlation between heart rate, electrodermal activity and player experience in first-person shooter games. In *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games*, 49–54. ACM.

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

Hoeft, R., and Nieznańska, A. Empirical evaluation of procedural level generators for 2d platform games.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2013. Towards a generic method of evaluating game levels. In *AIIDE*.

Mandryk, R. L., and Inkpen, K. M. 2004. Physiological indicators for the evaluation of co-located collaborative play. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 102–111. ACM.

Markov, A. 1971. Extension of the limit theorems of probability theory to a sum of variables connected in a chain.

Shaker, N.; Togelius, J.; Yannakakis, G. N.; Weber, B.; Shimizu, T.; Hashiyama, T.; Sorenson, N.; Pasquier, P.; Mawhorter, P.; Takahashi, G.; et al. 2011. The 2010 mario AI championship: Level generation track. *TCIAIG, IEEE Transactions on* 3(4):332–347.

Shaker, N.; Smith, G.; and Yannakakis, G. N. 2014. Evaluating content generators (draft).

Smith, G.; Treanor, M.; Whitehead, J.; and Mateas, M. 2009. Rhythm-based level generation for 2D platformers. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, 175–182. ACM.

Snodgrass, S., and Ontañón, S. 2014. Experiments in map generation using markov chains.

Sorenson, N., and Pasquier, P. 2010. Towards a generic framework for automated video game level creation. In *Applications of Evolutionary Computation*. Springer. 131–140.

Sorenson, N.; Pasquier, P.; and DiPaola, S. 2011. A generic approach to challenge modeling for the procedural creation of video game levels. *Computational Intelligence and AI in Games, IEEE Transactions on* 3(3):229–244.

Togelius, J.; Preuss, M.; Beume, N.; Wessing, S.; Hagelback, J.; and Yannakakis, G. N. 2010. Multiobjective exploration of the starcraft map space. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, 265–272. IEEE.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *Computational Intelligence and AI in Games, IEEE Transactions on* 3(3):172–186.

Uriarte, A., and Ontanón, S. 2013. Psmage: Balanced map generation for starcraft. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, 1–8. IEEE.