Survey on 2D platformer game Procedural Content Generation

54345382 ZHAI Guanxun
54736608 CHEN Yang
54732173 GUAN Xinjia

I. Introduction

Ever since Rogue became popular in mid-1980s, the procedural level generation (also referred to as PCG) has been an interest in game design field. Another example is Dwarf Fortress, in which almost all the game content can be generated automatically. The rogue-like game has a deep and profound influence on video game industry, both for the indie game and commercial games. Nowadays many open-world games or text-based games tend to use PCG methods to generate dynamic contents in the game world.

As 2D platformer games has always been an important genre of games, by applying PCG methods on 2D platformer game level designing, we may be able to generate game contents in a more efficient and interesting way, such methods might also create band new gameplay. From the other perspective, PCG is also useful for game designers' testing and might give some inspiration to them. By these means, it will be a good incision point for PCG research. And we did some survey on 3 related papers in this area. The ideas proposed by these papers along with their outcomes have some common points and can be considered as recommendable research materials

II. Survey Cases

1. Polymorph Generation Model

If you want to a game become popular, the game must be designed to suit for a large number of people. But every player begins games with verified game playing habits, different skill levels and develop their skills at different rates. So there are some problems, like the game is too easy and make players feel bored or the game is too difficult and make players feel frustrated.

In fact, the normal method to solve this problem is Dynamic Difficulty Adjustment(DDA). DDA, a general category of approaches that alter games during play, in response to player performance. However, this technique also has some problem. Most of DDA techniques focuse on basic parameter tweaking which is designed by author. We can look it as the author intuition, and it will not adjust the level dynamically and appropriately. One game which is called Polymorph can solve above problem. Polymorph is a 2D platformer game that generates and adjusts its levels during play as a means of DDA. Specifically, Polymorph tackle the DDA problem by creating a machine-learned model of difficulty in 2D platformer levels along with a model of the player's current skill level. The design of Polymorph can show how a game can be designed to accommodate the skill and experience of every individual player by incorporating machine learning and dynamic level generation. Polymorph consists of a data collection tool, a level generator, a game engine, and a learned model of level difficulty. And the method will focus on the part of how to data collection and the design of the learned model of level difficulty.

The strategy of mass data collection and statistical machine learning can answer two questions about what makes a 2D platformer level difficult or easy and how to determine if a player need to more challenges. The strategy of data collection is that a tool asks the player to play a short level segment, and collects the data on the level and the player's behavior along the way before the game beginning. The level segments are generated by an adaptation of the action rhythm-based generator from Smith et al. (2009). But the generator was modified to create the shorter segments rather than full levels. The reason for using these short level segments is that they seem an ideal length for custom, player performance-based, generated level chunks. As the player progresses through a Polymorph level, each time they successfully pass through a segment of this length (or die), another segment of the same length is generated and placed in front of them, extending the level. Then we can get the data about player game skill level and knew which difficulty level suits for him. See table 1.

Feature	Correlation coefficient
Gap_Kill	0.7749
JumpUp	0.7095
Gap_Gap	0.6765
Gap_Avoid	0.6177
FlatGap	–0.5316
KillEnemy	–0.5222
Avoid	0.3818
JumpDownGap	0.3219
JumpUpGap	–0.0842
Avoid_Thwomp	0.0709

Table 1 – Ten most highly correlated features. Underscores indicate adjacent components.

The statistical model that Polymorph has learned from the collected data is a ranking of level segments according to their difficulty. And the data collection tool keeps track of features representing the player's behavior while they are playing. Given these features, the researchers of Polymorph have applied a Multilayer Perceptron Algorithm to rank all of the generated level segments as a model of difficulty. At first, Polymorph evaluates player features on the model trained from the collection data. Then, before the player progresses to the next segment of the level a new segment is chosen that is ranked either higher or lower as necessary for the player's behavior. We can see from Table 1 that several of the component combination features are significantly correlated with the difficulty of generated level segments. And we know that the difficulty of the level segment is based on some kinds of features. In this way, when the player plays the game more skillful, the level will become more difficult.

Figure 1 – Two examples of Polymorph's level segments

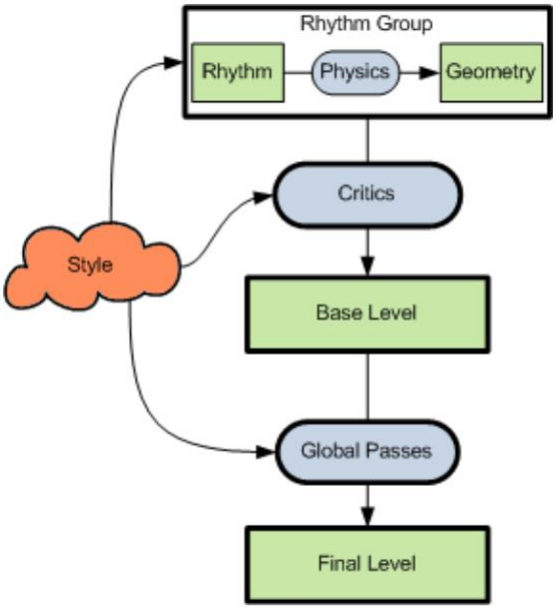
Looking at the Figure 1, there are two example segments. It shows two of the level segments that were created by Polymorph's level generator. The first one is based on the data of the easy level, while the second one is based on the difficult level.

The model of Polymorph is a good idea about dynamic level generation. We can focus on get the initial data more intelligent, and make use of the technique of machine learning to predict the player skill level.

2. Rhythm-based Level Generation for 2D Platformers

This is a research on Rhythm-based level generation, which means how to used tuned patterns of game experience, the "Rhythm", to generate game levels in a game. The rhythm here is a representation of the density of players' behaviors. Like in low density scenarios, players most likely act in a slow and low frequency way. This rhythm is generated in groups, where a defined length of time is considered as a complete rhythm group. And the number of behaviors in each group decide what the rhythm is like.

The way how the rhythms are translated to level designing is a 2-layer grammar-based geometry generation system. Shown as the following figure 2:



Basically, the 2-layer refers to Rhythm and Geometry here. Origin is the Rhythm generating. After somehow we can get a tuned rhythm, verbs like "move", "jump", "wait" will be generated based on the rhythms. Then the generated grammar patterns will be validated by the Physics constraints. And Geometry layer will generate a set of geometries based on the given patterns after physics validation. The output of Rhythms groups will be some coarsely connected patterns of geometries. The Critics are used for solving the over-

generation problem, which means the generated level is too big and have some undesirable levels. This problem is mainly because the grammars cannot be applied globally. Even local geometry patterns are playable, after connection, the global pieced level might have many discontinuous parts. After critics, the base level can be generated. Then Global Passes will be applied for a double validating. This phase applies two algorithms. The first is to tie the levels to the ground such that players have less possibility of unintentionally falling off the platform. The second is level decoration with coins (or gems). This is both useful for guiding the players and improving playing experience.

The researchers of this method consider PCG not merely as an approach to level designing. They consider PCG more like an assisting tool for human designers. As is evaluated, the outcome of this method is applicable while somehow not variable enough to create better game experience. By combining with the DDA as stated in previous part, the game experience might be improved. Even so there is still an absent research area, suggested by the researchers, that is the level analyzing. For now, it might be hard to decide if computer-generated levels can be good enough as human designers' work. Overall this method is considered to be extensive by adding new verbs and description, also some global adjustments.

3. Another approach: 4-layer hierarchy model of level generation

Another method of generating procedural level for platform game is four-layer hierarchy model. This model is an imitation of music composition. Like a beautiful melody composed of combination of basic notes, repeat rhythm, an interesting and complex level can be constructed by repeating and reshuffling basic elements in the game.

The first layer also the basic element of the model is Components. They are the basic constructive units of platform games. such as spikes, hills, vines and etc. A typical component includes both an obstacle and a resting spot which maybe a simple empty land that must be jumped over and then move forward on. Components' constructions have two dimensions, one is the space they have to fill in and the other is "tweaking" values.

The second layer of the mode is 'Patterns'. Only components can't generate complex rhythm for a platform game. So, some individual components are grouped or combined into a longer sequence. With this mechanism one can generate linear sequences, which help form challenging and fun levels to attract players. Current model defines 4 pattern types: basic, complex, compound, and composite. Among these four types, basic pattern is the simplest pattern which only consists of one component. In this pattern, components repeat several times without any change. A complex pattern is also a repetition of the same component. The difference is that constructions of components in complex patterns can be changed. Compound patterns and composite patterns are both made of two different types of components. The former one introduces the rhythmic mechanic while the latter one has more requirements in which two composite components are not only different but coordinate with each other as well.

With four types of patterns, one can only build linear sequences for simple level. To deal with the non-linear structure, the third layer "Cell" is constructed. A cell is an encapsulation of some patterns in which the specific characteristics are ignored. Above the cells is the fourth layer "Cell structures", giving players chances to choose different paths and reach different or same destinations. Kate Compton and Michael Mateas build a library of possible cell

structures. A few of the most common cell structures found in platform games are illustrated in Figure 1.[1]

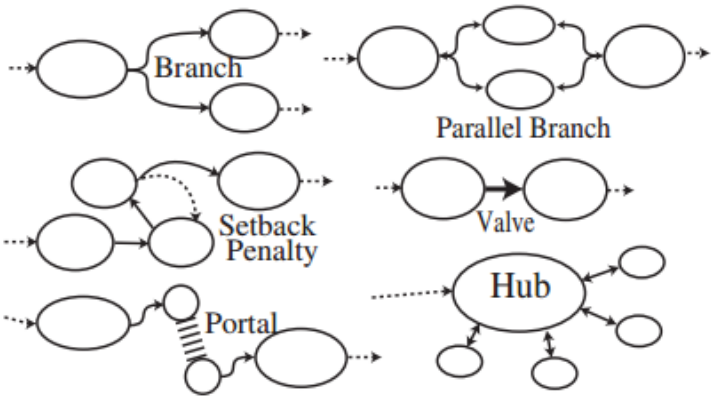


Figure 1: Cell structures create non-linear levels

With this four-layer model, we can generate procedural levels for platform games easily and flexibly. A level is started with a cell structure which can be broken down into more cell structures. Each cell maps to a specific pattern, which can be expanded into more patterns as well. And each pattern contains several components variations.

This four-layer model gives a high abstraction of platform games composition from the macroscopic angle. In the model, physical models can be established and algorithm for difficulty calculation can be applied on patterns. Hill-climbing search is used to find the optimal pattern for a given degree of difficulty.

III. Summary

As a special game designing approach, PCG is considered to be useful in both the designing and testing process. PCG has already proved its usefulness in text generation in many games. While for a more complex case like level generating, PCG methods still can't compete human designers' creativity and experience somehow. So the research work sill need to start with some simpler form of games. Here we investigate how the 2D platformer games can make use of PCG generation, considering that platformer game is a simple yet typical genre games that have very close relationship with level designing.

We did survey on 3 articles which are related to difficulty adjustment and pattern generation. These surveys all aims at 2D platformer games level generation.

Polymorph model suggests using a dynamic difficulty adjustment method for level generation. The level is not generated as a whole. Instead the level is based on player's score in the past game process. The main idea of this research is how different patterns contribute to players' experience. The data was collected and analyzed by the history data of players' performance in the game. And correlation coefficient was statistically calculated using machine learning techniques. As difficulty can be regarded as the core design factor of a level, this research contributes to the basis of the 2D platformer game level generation.

Associated with previous research, another 2-layer level generation method was proposed by the researchers who have worked on Polymorph model. This idea is about how to grammar to generate a level. This method uses keywords to generalize the patterns of a level by the tension of players' actions, which means how many behaviors players will have in a defined time length. If player need to do a sequence of "jumps" in a short time, this pattern will be identified as a "high-density" rhythm. The generating will inverse this

identification and use defined terms to generate the geometry of the level by piecing different rhythm groups. Then after physical validation, the level can be generated. However, considering automatically generation will have some unexpected results, the Critics and Global Passes steps are also required to generate a playable level.

Then there is another approach of similar topic with previous research, whereas the method is a 4-layer generation model. This idea is more about how to encapsulate contents in a hierarchy. The first layer are components of obstacles and resting paths. Then the components can be combined with 4 types of "patterns": basic, complex, compound, and composite. Then the patterns will be encapsulated into "Cell", which means a sequence of patterns will be recognized as a whole structure part of a level. And obviously the final layer of level can be applied. This 4-layer model is straightforward and has a clear hierarchy. It can be used for the level generation of a whole level.

IV. Conclusion

From what we learned from the research on 2D platformer level generation, we can find that some techniques are not considered to be a substitute to human designer. Another purpose of using PCG in 2D platformer generation is for testing and experiment on the factors to use in level designing. While considering the potential of machine learning techniques' developing, some automatic level generation tools might really be developed.

And considering the methods proposed, 2D platformer game levels' most basic factor is the difficulty patterns. By tuning and combining them with proper guidance, like rhythms in a song, such levels can create flow in players' playing experience in some degree. However, the actual results might still be unplayable due to lacking of proper validation. This might be a further research topic considering its potential.

V. Reference

[1] Jennings-Teats M, Smith G, Wardrip-Fruin N. Polymorph: A model for dynamic level generation[C]//Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. AAAI Press, 2010: 138-143.

[2] Smith G, Treanor M, Whitehead J, et al. Rhythm-based level generation for 2D platformers[C]//Proceedings of the 4th International Conference on Foundations of Digital Games. ACM, 2009: 175-182.

[3] Compton K, Mateas M. Procedural Level Design for Platform Games[C]//AIIDE. 2006: 109-111.

[4] Procedural Content Generation Wiki. http://pcg.wikidot.com/category-pcg-papers