Identifying Catastrophic Failures in Offline Level Generation for Mario

Adeel Zafar, Hasan Mujtaba
FAST-NUCES Islamabad, Pakistan
Adeelzafar.pk@gmail.com

***Abstract*— Video games are pushing the boundaries of the creative medium to be more realistic. This realism demands the game content to be tailored to improve the gaming experience. Generating content is a challenging task and automated approaches based on Artificial Intelligence techniques can help the gaming industry with this problem.**
The focus of our research is to produce adaptive levels for action-adventure games. We present a technique to identify catastrophic failures in offline level generation of the popular game "Mario". Our approach produces levels that have high replay value and have limited catastrophic failures, thereby improving the quality of the levels generated. This paper also presents taxonomy of Procedural content generation and Search-based PCG techniques. That is to our best knowledge the first wide-ranging survey of both the approaches.

Keywords- Adaptation, Catastrophic failures, Procedural content generation, Search-based PCG

I. INTRODUCTION

Gaming industry has seen a huge expansion and has earned quite a bit in recent times. The USA Entertainment Software Association (ESA) reported that over 68% of the American population play games [1]. Video games are not only a source of entertainment but they also provide a controlled environment which is useful to test different theories and techniques for testing Artificial Intelligence (AI). The demand for more realism in video games has made content generation a challenging task for game developers. Developing content requires developers and this often leads to unrealistic deadlines and stress for the developers. This not only raises the development costs of the game but also affects the quality of the content being developed. Reusing content leads to the problem of repetitive game play. This decreases the re-playability of the game. Players get bored by playing the same level again and again and following a certain pattern of play.

This is where "Procedural Content Generation (PCG)" comes into play. PCG is the creation of content using evolutionary practice that results in a changeable range of possible game play scenarios. Theoretically, using PCG game designers can produce infinite levels and the overhead of developing massive content is reduced. PCG can be used to generate different types of content depending upon the gaming genre or the demand of the developers. For example, in action games it can generate different levels (or rounds) for the player, each time the game is played. This dynamically generated new content allows the gamer to have a different gaming experience every time they play, thereby increasing the replay value of the game.

PCG can be used in different aspects of game development for a number of genres i.e. developing rules for non-digital games, weapons for shooter games and levels for action adventure games.

Rogue was the first game that was developed using PCG. It had monsters, dungeons and the content generation process was pseudo random. Due to the fast paced nature of the genre, there is a special need for generating content for action adventure games. The approach presented in [3] is the most reliable approach studied so far. We have identified "catastrophic failures" in that approach and present a future technique. Catastrophic failures are issues that can make levels for a game unplayable or too simple. They damage the reliability of game and cannot be applied to commercial games, where reliability can determine the success or failure of a product.

PCG is a relatively new concept and proper testing mechanisms need to be designed that measure the quality of the generated content. In this paper we discuss these topics and present our work with PCG. In Section II we provide a literature survey of relevant techniques. Section III deals with identifying catastrophic failure for the Mario game. Section IV provides detail that why we selected MARIO and the game mechanics. Section V concludes the paper and suggests future directions of the research.

II. LITERATURE SURVEY

This section provides an introduction and an overview of relevant work done in Procedural content generation (PCG) and its sub-domain Search based procedural content generation (SPCG). There are different types of content that are generated using PCG. Some of them are listed below.

- Generating tracks for racing games.
- Mission for adaptable player experience.
- Levels for action games.
- Generating maps for tower defense games.
- Rules for non-digital games.

PCG can generate different content in a game. For example, manually designing trees that behave realistically requires considerable time and effort. The complexity of this problem increases if a forest has to be designed. There are thousands of trees in a forest. If we want to increase the aesthetic value of a game, the designer would have to sit down and will need to design and model different kinds of

978-0-7695-4927-9/12 $26.00 © 2012 IEEE
DOI 10.1109/FIT.2012.20

trees. This increases development time and costs money. Game designers can use Lindenmayer systems; it is recursive grammar rewriting frameworks as shown in "Fig.1". It can generate fractal patterns and iterate n times to get attractive shapes appearances [4]. Rocks have same problem as trees, they can also be generated using L-shape systems.

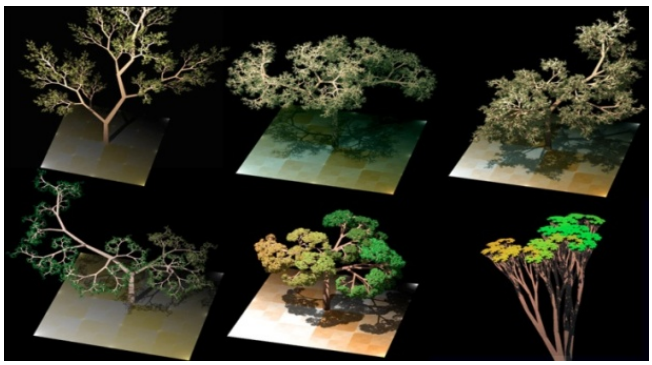

Figure 1. L-Systems for modeling tress

PCG can also generate weapons that player might require in a game. Different weapons with different effects and use can be generated. The game Borderlands, uses 65000 different weapons, designing these weapons manually is a daunting task. Another game Galactic Arms Race, overcomes this problem by using cgNeat algorithm to evolve different type of weapons (Fig. 2) [4].

Figure 2. Evolved weapons generated in galactic arm race

Similarly in Role Playing games "Quests" are generated, it is a mechanism for giving players tasks and goals [4]. Various quests can be generated procedurally. There are also other gaming content that can be generated procedurally and our focus of work is how levels can be generated in action adventure games. Procedural content generated algorithms can work simultaneously and developer can choose if the content is generated online or offline. Online content generation means that content is generated at runtime, while the user is playing the game and the user can observe what is happening in the environment of game. Secondly, the content can be generated based on the performance of player in the previous level. This technique is the offline generation of content and it comes in the definition of PCG [3]. Our focus in this paper will be on offline generation of levels for the Mario game.

Another important consideration is whether the content generated is necessary or optional in the game [3]. Necessary content is required to progress further in a game; it can include main quests or rules for a game. Whereas the optional content can be skipped from the game and it is not an essential part of the game e.g. sub-quests or spawning of enemies etc.

Commercially successful games, Left 4 Dead and its sequel Left 4 Dead 2, use a “director program” that handles the content of the game. The director of game places the enemies in different positions and numbers based upon each player's current situation, status, skill, and location [5]. There were also physical gestures that were mapped to the game, making the game more interesting and having more replay value. Other examples of games that use PCG or similar approach are, *ToeJam & Earl* (1991) which was the first action game that generated levels on the go. *The Elder Scrolls III: Morrowind* (2002) generated water effects on the fly. Similarly, *RoboBlitz* (2006) game for XBox 360 generated content randomly. *Borderlands* (2009) and *Terraria* (2011) generated weapons and 2D landscape procedurally [6].

PCG can generate a diverse type of content and hence there are varieties of algorithms. Some examples of algorithms are [7]

- Adaptive Difficulty
- Automatic Game Design
- Design Of Level Content
- Dynamic World Generation
- Map Generation
- Search Based Procedural Content Generation

A. Relevant work done in PCG

Joris Dormans and Sander Bakkes [8] described that the levels of action adventure games have double structure basically there are missions and spaces in an action adventure game, these games are different from other games as level design is more important in adventure games, So it is clever to break the level in to mission and spaces. A grammar is user to generate missions and it is very helpful as well, they can generate discrete structures which are useful as compared to linear structures. After generating a mission from grammar, a 2-D graph can be used to represent it. This process is interesting and cooperative, if recursion is used so that we can produce varied levels. The most important thing that can be used is to model the player according to his or her behavior; it can be done by

exploiting player models and than using generative grammar to adjust dynamic difficulty for player. The game changes are adaptive. Basically our focus of work is also on adjusting dynamic difficulty but it is very different from the idea used here and we cannot say that the existing technique is PCG or not. Online changes in game are not considered as PCG [3].

Julian Togelius has focused on content generation for the Mario game [3]. There are two versions of Mario that were generated using infinite Mario bros. In an offline version the player plays the game and each action of player is recorded, then the next level is generated according to the player's previous performance. In contrast, the online version changes the level when it detects any action has taken place. This allows the player to be challenged continuously and it will generate different levels for different players. However, these approaches had errors and lacked was reliability.

Erin Hastings, Ratan Guha, and Kenneth Stanley introduced a new algorithm in their paper which was the best paper in *CIG-2009* [9]. The *content-generating NeuroEvolution of Augmenting Topologies* (cgNEAT) algorithm, this algorithm was designed to fulfill the player's requirements as the content generated in most weapon games is static. The algorithm is introduced in Galactic Arms Race (GAR), in which player is a pilot which will fight enemies and acquire weapons. The weapons were generated according to players approach, a weapon that is favored by player is generated more often.

Gillian Smith [10] presented a novel technique that was based on both game designer and procedural generated content. The design tool is known as *TANAGRA*. The game designer can place constraints and the algorithm used can map the rest of the level.

Georgios Yannakakis, and Julian Togelius [11] presented a Human Computer Interface (HCI) approach in which the content is generated keeping in mind the experience of players.

B. Relevant work done in Search-Based PCG

Procedural content generation has been a feature of various games but there are some extremes that PCG cannot overcome therefore a new technique known as SPCG came on to the scene. SPCG is an extension of PCG; in SPCG the algorithm used is not *constructive* [2].It is *generate and test* algorithm. Basically, *generate and test* algorithm generates the content and then tests it according to some mechanism. If the content generated is of sufficient quality than it is added to the game otherwise it is rejected. Such an approach will insure that the content is of adequate quality and the game possessing such content can be added in commercial games.

In SPCG the test function used to evaluate the content not only accepts or rejects the content but marks it with some value, which is used for fitness calculation. The content generated next time will depend on its corresponding fitness value [2]. Togelius and Schmidhuber [12] presented a technique of SPCG. A grid game (Pac-man) was used for their experimentation. The rules were driven from the approach for Pac-man. The content included were game agents, the grid which had walls, enemies and awards of different colors.

Browne [13] developed an approach for necessary content of board games. Rules were represented as expression tress and then designed in game language. The rules generated were all used in the evolution process. No bad rule was removed straight away either it was given a fitness value that was very low. Their priorities were also low. So, in that way they were in the population but our solution had diversity. The measure used for testing was simulation based.

Togelius [14] conducted an experiment both online and offline generation of racing tracks. The generation process included both optional and necessary content and it was designed for a simple racing game. The tracks were represented as curves. There were different fitness functions used i.e. simulation based, personalized and static. The candidates having fitness function were evaluated according to the performance of neural network drive on the track. The neural network was first trained to drive in a specific human way. The fitness function calculated the maximum and average speed and accordingly fitness was assigned to that fitness function. The tracks proved to be of sufficient quality.

Isaac [15] presented an approach for generating rocks, these rocks were diverse and they can be used in virtual world as well.

We now present our experimentation with PCG and its results.

III. EXPERIMENTATION AND ANALYSIS

A. Identifying Catastrophic failures

In this section we have identified some faults from the previous technique and we have proposed a new future technique.

Generating levels for action-adventure games are really important as our previous section showed that there is very limited work done on generating procedural levels for action games. The first effort in this field was ToeJam & Earl [7].

The thing to keep in mind while designing content for game is adaptivety. The content that depends on the previous performance of players is PCG. For example, in first-person shooter game if the player has used a weapon more than once, the next level should have similar weapons from the one that are used. Similarly, in an action- adventure game if there are fewer obstacles and player has performed well than the next level should have more enemies. This can help in adjusting the difficulty of the game. In most games, the difficulty is mapped at the start (beginner, medium, expert) but this is not what PCG means. The algorithm used by PCG insures that content generation is not predictable, and it depends on player's previous performance. This will increase the entertainment and fun level of player, and that is the motive of PCG.

The importance of avoiding catastrophic failure is highlighted in [16]. Unplayable can destroy the gaming experience and could be devastating for the sales of a game. Therefore, first we identify catastrophic failures, and then we present a technique that minimizes these failures. The game developed in [3] is available online and the following failures were identified in it.

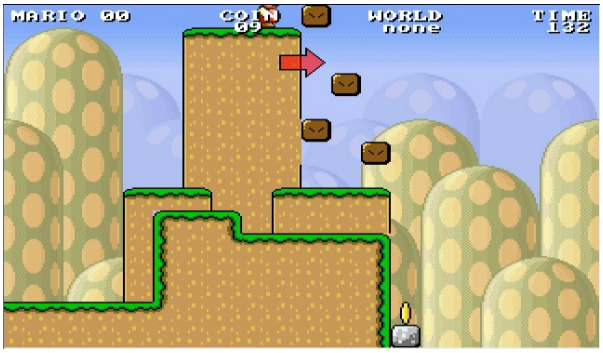

Figure 3. Height of ground

Here, we are only considering the offline version of PCG as we know that online version is not considered as PCG.

While playing the game we found out that after playing several levels in the offline version of Mario, the height of ground increases to a great deal and Mario goes out of the scene. This failure not only disturbs the game play but is also harmful to the aesthetic value of the game.

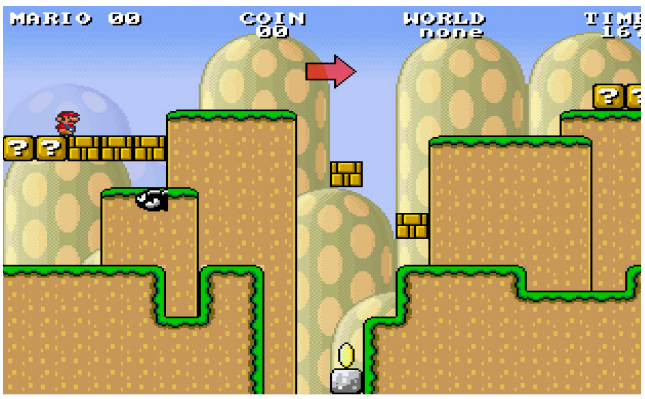

Figure 4. Height of ground and optional content

Due to the increase in height and certain other metrics, the level generated next time has the optional content out of the range of Mario.

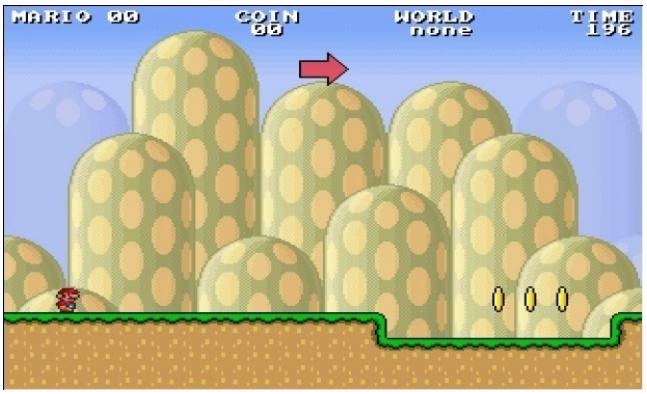

Figure 5. Simple level generation

In offline level generation, when the player dies too early the keys recorded are very limited and the algorithm doesn't know what to do next. Therefore, it generates very simple and straight-forward levels. This is a very unreliable approach and it can harm the commercial success of games.

Figure 6. Unplayable levels

The rising of ground, placing of optional content like blocks, flowers and cannons can make the game unplayable. This to us is the most important catastrophic failures found as we can see from the "Fig.6" that the Mario will be stuck in a place where there is no way for Mario and the game will be in an unplayable scenario.

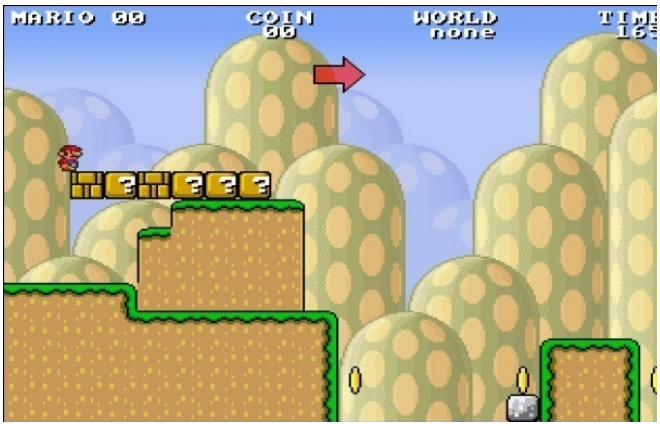

Figure 7. Placing of Mario

In the offline version of game when a new level starts, Mario is placed above the optional content (blocks). This will give the game a very unreal look and will harm the aesthetic value of game. The game will be less reliable.We can see one more thing that there are some scenarios when the gaps are more than the jump strength of Mario.

These all are the catastrophic failures that were found during the analysis of this game. Some of the failures were also highlighted by the author in [3] but there were no detail analysis and the consequences of these failures in the game play were also missing. We have identified these failures and proposed a future approach that will generate levels that have fewer catastrophic failures and it will strength the way of PCG in the commercial market.

The pseudo-code identifies the catastrophic failures and represents a way to handle them. Our future technique that is still in prototype will remove all these faults and it will be more reliable than the existing technique.

Satisfying constraints and Handling catastrophic failures.

If height is greater than the defined height

Reduce height

After certain level redraw height

Generated Key is less than the defined size

Generate random levels from default key

Placing of Mario

Place Mario on ground, checking position function and dimensions of Mario

Remove un-playability

Place canons, blocks and flowers from three dimensions only

This part of the algorithm will check whether the generated levels have catastrophic failure or not. Only such levels will be provided to the user which satisfies this part of the algorithm. This will remove all the catastrophic failures discussed before.

The height of the platform will be according to constraints specified, therefore the new levels generated will not have the problem that Mario is out of the view and will improve the aesthetics of game. The Mario will be placed below the optional content so that catastrophic failure will also be removed. The technique is still under prototype and it needs more work to be done.

IV. SELECTING MARIO

Mario is a fictional character in the Japanese created game. The character is a short plumber who rescues his princess. During the game play Mario collects coins, disturbs blocks and fights against enemies. Mario was first appeared in platform games as our previous chapter focused on how Mario evolves from genre to genre. It is the best selling game of its times. The following table shows some Mario history [17].

TABLE I. HISTORY OF MARIO

Series	First game	Created by	Date
Mario franchise	Donkey Kong	Shigeru Miyamoto	(1981)

Now we will see all the aspects of Mario and why it was selected. First and far most is the game-play mechanics. Mario has the most simple game-play mechanics. What is game-play mechanics? When there are certain characters in the environment and we add some rules for those characters to explain how they will play it becomes a game. These rules are the *game-play mechanics*. The mechanics of Mario are when Mario will be hit by an enemy it will die, when it jumps on that enemy, Mario is awarded some points. Then there are other mechanics such as coin collection and so on. These mechanics define the actions of the player what it can perform [18].In Mario the levels are represented as straight forward grids of blocks. The size of each block is 8 * 8 pixels.

After the mechanics, the second most important thing was the previous experiments that were done on Mario. Mario has been used for a series of experiments there are different controllers that are used for AI competitions. Usually there are controllers for board games like chess, Go etc but there are very limited controllers for platform games and action adventure games. In [19], Julian arranged a competition that was making the best controller for Mario. The controllers that were based on certain AI performed much better than the controllers of other learning algorithms. There were large amount of submissions and a best controller was chosen at the end. Therefore, when we are targeting the commercial market, a game like Mario could be a good example because we have the controllers and other learning algorithms that are already available. Mario also has the best look and feel of a platform game and the users are mostly familiar with the game.

VI. CONCLUSION AND FUTURE WORK

The catastrophic failures for offline level generation of Mario were identified and a novel improved approach is proposed. The identification of these failures was important

because if we are proposing our own technique it should be more reliable than the existing technique and it should have fewer failures. Our approach removes the catastrophic failures and produces levels that are more accurate, reliable and playable. The previous approaches studied in literature had very limited scope. Most of them focused on random level generation and producing a single content in game that was based on PCG. Only one approach produced the whole levels of action adventure game keeping in mind the fun and entertainment value. Our approach will improve level generation, levels will be random, that take care of user's entertainment and that are catastrophic failure free.

The future of PCG is search based procedural content generation. We can make an SPCG approach along with some other computational technique. Search based PCG is making a mark and in the near future it will take place of all other content generation techniques. However there are some constraints which need some academic research.

ACKNOWLEDGMENT

The authors would like to thank Julian Togelius for his support.

REFERENCES

[1] ESA, "Industrial Facts", http://www.theesa.com/facts/index.asp, 2011

[2] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, Cameron Browne, " Search-based Procedural Content Generation ", IEEE Trans. Comput. Intellig. and AI in Games , 172-186, 2011

[3] Julian Togelius, Emil Kastbjerg, David Schedl and Georgios N. Yannakakis, "What is Procedural Content Generation? Mario on the borderline", Proceedings of the 2nd International Workshop on Procedural Content Generation in Games, Article No. 3, 2011

[4] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup, "A Procedural Content Generation for Games: A Survey", IEEE Transactions on Computational Intelligence and AI in Games, Volume: 3, Issue: 3, Publisher: IEEE, Pages: 172-186, 2011

[5] Wikia, "The Director", http://left4dead.wikia.com/wiki/The_Director, 2011

[6] 3D computer graphics, "Procedural generation" http://en.wikipedia.org/wiki/Procedural_generation, 2012

[7] Wikidot, "Procedural content generation", http://pcg.wikidot.com/, 2011

[8] Joris Dormans and Sander Bakkes, "Generating Missions and Spaces for Adaptable Play Experiences", IEEE Transactions on Computational Intelligence and AI in Games, Volume: 3, Issue: 3, Publisher: IEEE, Pages: 216-228, 2011

[9] Erin J. Hastings, Ratan K. Guha, and Kenneth O. Stanley, "Evolving Content in the Galactic Arms Race Video Game", CIG'09 Proceedings of the 5th international conference on Computational Intelligence and Games Pages 241-248, 2009

[10] Gillian Smith , Jim Whitehead, Michael Mateas, "Tanagra: Reactive Planning and Constraint Solving for Mixed-Initiative Level Design", Special Issue on Procedural Content Generation, vol 3. iss. 3, September 2011.

[11] Georgios N. Yannakakis and Julian Togelius, "Experience-Driven Procedural Content Generation", IEEE Transactions on Affective Computing (in press).

[12] Togelius, J., Schmidhuber, "An Experiment in Automatic Game Design". In:Proceedings of the IEEE Symposium on Computational Intelligence and Games,Perth, Australia, pp 252-259, 2008

[13] Browne, C.: Automatic generation and evaluation of recombination games. PhD thesis, Queensland University of Technology 2008

[14] Togelius, J., De Nardi, R., Lucas, S.M.: Towards automatic personalised content creation in racing games. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games ,2007

[15] Isaac M. Dart, Gabriele De Rossi and Julian Togelius, "SpeedRock:procedural rocks through grammars and evolution", Proceedings of the 2nd International Workshop on Procedural Content Generation in Games Article No. 8 , 2011

[16] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, Cameron Browne, "Search-based Procedural Content Generation: A Taxonomy and Survey", pp- 172-186, 2011

[17] Games, "Platform games", http://en.wikipedia.org/wiki/Mario, 2012

[18] Muurtegel, "Game Mechanics" http://muurtegel.com/blog/2011/03/18/gameplay-mechanics/, 2011

[19] Julian Togelius, Sergey Karakovskiy and Robin Baumgarten, "The 2009 Mario AI Competition", IEEE Trans. Comput. Intellig. and AI in Games, pp- 55-67, 2012.