Design Patterns and General Video Game Level Generation

Mudassar Sharif, Adeel Zafar, Uzair Muhammad
Faculty of Computing
Riphah International University Islamabad, Pakistan

***Abstract*—Design patterns have become a vital solution for a number of problems in software engineering. In this paper, we have performed rhythmic analysis of General Video Game Level Generation (GVG-LG) framework and have discerned 23 common design patterns. In addition, we have segregated the identified patterns into four unique classes. The categorization is based on the usage of identified patterns in game levels. Our future aim is to employ these patterns as an input for a search based level generator.**

***Keywords*—General video game level generation; rhythmic analysis; procedural content generation; design pattern; search based level generator**

I. Introduction

With the passage of time, digital games have become a large industry. In 2014, the gaming industry generated more than 47 billion dollars worldwide [1]. However, with expansion, this industry is also facing a number of problems. The most important aspect in this regard is the total cost and budget that is being utilized for game development. Game content upholds a significant chunk of game development and with technical improvement in devices like smartphones, the content is becoming more complex and demanding. Therefore, the rapid development of game content is vital [10]. Procedural Content Generation (PCG) is the algorithmic creation of game content with less human intervention. Procedural content generators capture game rules as an input and then generate essential content for a game. PCG has been used frequently by indie game developers to generate diverse content including characters [3], terrains [3], [9], dungeons [4] and levels [5]–[8], [11], [18].

Level Generation has been the most significant and old problem in PCG domain. Yet, most of the level generation work has been done for specific games [5]–[7], [18]. Generating content for a suitable single type of game is important but it undermines the capability and reusability of a generator. On the other hand, a level generator that can generate levels for multiple games can possess considerable challenges. In this regard, an important step has been made by introducing a General Video Game Level Generation (GVG-LG) framework [8]. This framework generates levels for multiple games, unlike other level generators. The GVG-LG framework is comprised of Random, Constructive and Search-Based Level Generator (SB-LG). The initial effort was to identify design patterns from the GVG-LG framework and to employ them as objectives for SB-LG.

In this study, we have performed rhythmic group analysis for the identification of design patterns. After the analysis of each game presented in the GVG-LG framework, 23 unique design patterns were identified. These patterns were further classified into four different categories. The central aim of this research was to utilize these design patterns as objectives for the SB-LG in the GVG-LG framework. Our effort for pattern identification is inspired by the work done for Super Mario Bros (SMB) [6].

The paper is further divided into five sections. The second section explains the existing knowledge about PCG, the importance of design patterns in level generation and level generation for general video games. The third section of the paper presents the GVG-LG competition and analysis for identification and classification of design patterns. Lastly, we argue about the application of design patterns and their usage as objectives for SB-LG.

II. Existing Work and Background

A. Procedural Content Generation

PCG is the algorithmic creation of game content with limited or indirect user input [2]. Content includes assets of a game, i.e. maps, quests, textures, characters, rules, terrains, dungeons, levels, and sprites, etc. PCG is not a new domain and had been used in 80's for generating hundreds of stars in Elite [17].

Most of the algorithms that are used for generation of content are constructive and generate-and-test algorithms [17]. Constructive algorithms generate the content once and do not iterate upon it for further content improvement. On the other hand, generate-and-test algorithms first generate the content and then iterate upon it to make it of sufficient quality. In literature, these algorithms are referred to as search-based algorithms.

Along with advantages, PCG also has some limitations, for example creating a generator for each game may require more time and cost as compared to the manual creation of the content [11]. The main reason for creating a general level generator is to overcome such disadvantages. In addition, if we want to create content with ultimate control and with specific details, the best choice is to create the content manually [12]. Therefore, the control and evaluation of content in PCG poses challenges.

B. Level Generation

Level generation is the oldest and complex task in PCG domain. It requires the understanding of all the elements of a game and how to fit them into a level. The procedural

generation of levels has witnessed notable attention and various studies have been conducted in this regard. Most of the work has been focused on the generation of levels for specific games [7], [11], [18].

C. Design Patterns and Game Levels

Alexander initially developed patterns for problem-solving. It consists of two components: problems and their solution. The problem refers to a common and recurring design element in object-oriented development [13]. In a software application, the design patterns give insight to designers about architectural knowledge and provide a template for many situations [20].

In games, patterns are the problems created by designers for players to solve [5]. There is a collection of possible design choices in a game that can provide architectural knowledge to a designer. In other words, these design choices are architectural chunks for a game design which can automate game development.

Design patterns have been used previously for the generation of levels for specific games. Hullett et al. used design patterns for generating levels in the first-person shooter game [18]. Similarly, Dahlskog et al. [6] identified patterns of enemies, gaps, valleys, multiple paths, and stairs to generate levels for the game of SMB. Initially, the author proposed a straightforward way of combining the discovered design patterns into a game level [5]. In addition, the author used vertical slices of existing levels as design patterns and generated levels of sufficient quality [6].

In a recent study [7], a multi-level generator was also proposed. In this approach, three layers of abstraction for design patterns (meso, micro, and macro) were proposed and game levels were generated by using SB-LG. The literature review gives a clear indication of the usage of design patterns for generating game levels.

D. General Video Games Level Generation

To the best of our knowledge, most of the level generation work has been done for specific games like SMB [5]–[7], [14] and Rogue [15]. These generators possessed sufficient advantages. However, the problem is in the re-usability, development time and cost of such generators.

Preferably, the grand goal of Artificial Intelligence is to model general solutions that can be applied to a particular set of problems. For video games, this can only be done when we have a method to describe the games. Video Game Description Language [9] was developed originally for the Stanford General Video Game Playing. This language has mostly been used to tackle the problem of general games.

For the general video game level generation problem, an important step was identified in [11], where a video game descriptive language was used to generate multiple levels for general games (Sokoban, Lava, Block Faker, Gem and Destroy Game). Though the generator possessed notable advantages, it had no framework to compare other generators. Similarly, Neufeld et al. [16] introduced a general video game level generator by using Description Language and Answer Set Programming. The generator was tested against three different games and generated levels had a structure similar to many of the existing levels.

In this regard, a significant step has been taken by introducing the GVG-LG framework. The framework is based on GVG-AI framework and allows users to create and test their own level generators against a variety of games [8]. Three distinct generators: Random, Constructive and SB-LG were introduced within this framework. After detailed experimentation, SB-LG proved to be the best out of three. The SB-LG is based on an evolutionary algorithm, which takes an array of tiles as input and generates a level for the game.

III. Identifying Patterns for GVG-LG Framework

A. Rhythmic Groups

Rhythmic groups are short and non-overlapping sets of components that unfold an area of challenge. This approach assists to recognize challenging areas within a game level and provides a way to discover the complication behind such areas [19]. Rhythmic groups are quite modular, therefore provide assistance in patterns identification and their re-usability in a game level. In this research, rhythmic analysis was applied on a set of games to investigate design patterns inside a level. For this purpose, a game level is divided into cells. The cell is a section of game-play that ends, where the player can choose a new path. Cells inside a level design helps to analyze the structure and to provide a catalog of several paths through a level. The path may be of diverse difficulties, depending on the structure and dimensions of the cell.

B. Search for Patterns

The GVG-LG framework is built upon the GVG-AI framework. It consists of 92 different games with 5 levels of each game. Level of each game is divided into small groups to identify the challenging areas through rhythmic group analysis. By analyzing the GVG-LG framework, it is founded that most of the games had common design structure with most common elements. Therefore, primarily focus is based on the underlying structure of game levels for identification of design patterns. Design patterns are categorized into four classes on the basis of their rationale in the level:

- **Solid Sprites:** Blocks the movement of the player.
- **Collectible Sprites:** Can be destroyed by the player on interaction.
- **Harmful Sprites:** Are harmful and can kill the player on interaction.
- **Enemies:** Agents having ammunition and are harmful to player.

1) Analysis of Existing Games for Solid Sprites: In this section, rhythmic group analysis is applied on the GVG-LG games to identify design patterns for solid sprites. In Fig. 1, five cells are highlighted for the recognition of patterns. Cell 1 consists of a squared shaped solid block or sprite. Whenever player meets a solid sprite in his way, he moves up or down and provides a transition to a different path. Cells 2 and 3 are in rectangular shape. These two cells have the same purpose of creating a wall but, here player requires more effort to pass

through. The structure of cell 2 shows that it can be obtained by connecting two or more solid sprites vertically, and similarly cell 3 can be obtained by connecting them horizontally. In a similar way, cell 4 represents the boundary of the level which can be established by assembling solid sprites vertically and horizontally without blocking any internal space of the level. Boundary sprites make a dashboard and allow a player to play inside a specified area.

Fig. 1. Analysis of an existing level for solid sprites.

To completely block an area or to form a room inside a level, these sprites can be connected in a two-dimensional way. Cell 5 consist of a movable sprite. The Player can use a key to unlock such type of sprite to find a path. The analysis of existing level shows some interesting aspects of level design for games. The structure of existing cells can be obtained by assembling solid sprite by using different patterns. Table 1 shows some common design patterns for placement of solid sprites.

TABLE I. SOLID SPRITES FOR GVG-LG

Solid sprites	
Single	Single solid sprite at a free space.
Boundary	Collection of solid sprites to form game dashboard.
Wall	Two/multiple sprites connected vertically or horizontally to block a path.
Room	Vertically and horizontally connected sprites to surround an area.
Movable	Sprites that can move after unlocking it by key.

2) Analysis of Existing Games for Collectible Sprites: Mostly all 2D platform games have collectible sprites in the form of rewards. Collectible sprites are objects in a level that can be destroyed by the player on interaction and provides a reward, such as points, coins or weapons [19]. In Fig. 2, five cells are identified for the collectible sprites. Cell 1 consist of a single sprite at a free space, the player requires little effort to deal with it. Cells 2 and 3 shows sprites in a group form, where a player needs more effort for interaction. If the player wants to acquire maximum points in less time, he may choose the path where sprites are in grouped form. Similarly, cell 4 consist of a collectible sprite along with enemy and cell 5 has a collectible sprite between harmful sprites in a hidden form, which creates a challenging environment for the player.

Moreover, these sprites may move in a single or multiple lines. Each line of sprites may move in same direction or in opposite direction. Table 2 shows collectible sprites which can

Fig. 2. Analysis of an existing level for collectible sprites.

be destroyed on player interaction but, the player may require different skills for each.

TABLE II. COLLECTIBLE SPRITES FOR GVG-LG

Collectible sprites	
Single	Single sprite at a free space.
Group	Two or more sprites together.
Single line and moving	Multiple sprites in a line and moving in same direction.
Multi-line and moving	Multiple lines of sprites and each line moves in opposite direction to its nearer line.
Risk and reward	Collectible sprite with an enemy together at a place.
Hidden	Collectible sprites surrounded by other types of sprites.

3) Analysis of Existing Games for Harmful Sprites: In Fig. 3 cell 1 presents single sprite at a place and cell 2 shows multiple sprites together, while cell 3 consists of two different types of harmful sprites. In addition, a hole presented in cell 4, may also be harmful and if designed using multiple patterns; will pose a challenge for the player.

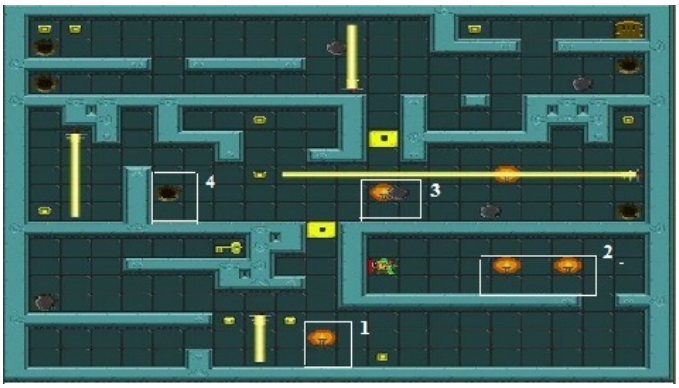

Fig. 3. Analysis of an existing level for harmful sprites.

TABLE III. HARMFUL SPRITES FOR GVG-LG

Harmful sprites	
Single	Single sprite at a place.
Group	Two or more harmful sprites at a place.
Multi-type	Harmful sprites of multiple types together at a place.
Hole	Single or multiple holes together.

Table 3 shows patterns of harmful sprites for the GVG-LG

framework. By increasing numbers and types of these sprites, the player may face a difficult environment to play. Levels where goals are surrounded by a group of harmful sprites like in fire game, the player cannot reach his goal without defeating these harmful sprites.

4) Analysis of Existing Games for Enemies: Enemies patterns presentation in Table 4 may give a meaningful difference in the game-play. For example, 2-enemies together at a place can block the player path in an effective way.

TABLE IV. ENEMY SPRITES FOR GVG-LG

Enemies	
Single	Single enemy at a free space.
Two	Two similar enemies together at a place.
Single line and moving	More than two similar enemies moving in single line and in same direction.
Multi-line and moving	Multiple lines of similar enemies and each line moves in opposite direction to its nearer line.
Randomly moving	Enemy/group of similar enemies moving randomly at an area inside a level.
Multi-type	Enemies of multiple types together at a place.
Multi-type and moving randomly	Enemies of multiple types moving randomly at an area inside a level.
Hidden	Enemy/enemies behind collectible sprite.

Similarly, enemies moving in multiple lines and in multiple directions can give a hard-hitting to the player than enemies moving in single line. Player requires different skills to defeat enemies of each type, therefore if enemies of multi-types in multiple lines are placed in a level than the game-play becomes more enhanced to proceed in next level.

IV. APPLICATION OF IDENTIFIED PATTERNS

A. Design Pattern-Based Level Generator

The suggested 23 design patterns will give a new experience to players by providing a better and enhanced gameplay. In the proposed technique, it is suggested that these identified patterns would be provided as an input to SB-LG and then it will generate a level of a game by using specified constraints about any game. In PCG, Search-based content generation is a special case of the generate-and-test approach [17]. In such type of generation, an evaluation function is used to assign a fitness value to the generated content. Similarly, assigning fitness value to newly generated content depends upon previously generated content. A defined population of content instances is placed in system memory. For each generation, these contents are evaluated and assigned a fitness value. In SMB, SB-LG takes input slices from the first level and that first level is generated by using constructive approach. Though in this case, the SB-LG will take patterns from the available array and will create levels by connecting and rearranging these patterns.

To construct a level generator effectively, a developer must understand these two major ideas: firstly, selection of design patterns that make up the level for a game, and secondly the way they fit together to create an entire level that will be playable and well-balanced. Here, it is suggested that a probability value must be assigned to each design pattern on the basis of occurrence in existing games. A comparison between occurrence of design patterns and a set of GVG-LG games is shown in Fig. 4. Similarly, there should be a defined sequence for the selection of design patterns. For example, boundary pattern will be selected first and after its implementation other patterns from same class or distinct class will be placed inside it. Because boundary provides a layout for a level to encompass all other sprites.

Fig. 4 shows the occurrence of identified common design patterns in the given set of games. Game play-ability can be changed by increasing the quantity of these patterns inside a level. For this purpose, SB-LG will assign a fitness value to each design pattern. Games such as Aliens and Rogue have a high probability for the presence of enemies. Therefore, changing the fitness value of enemies pattern will enhance the play-ability of the game level. Similarly from the Fig. 4, it is found that to create a level layout boundary patterns must be selected first such that, other sprites can be placed inside it. This approach may give significantly better output by placing the variations of patterns and increasing the length of the game platform.

In this section, two patterns are discussed in detail to find the impact of patterns on enhancement of level design.

TABLE V. DESCRIPTION OF MULTI-TYPE AND RANDOMLY MOVING ENEMIES PATTERN

Multi-type and randomly moving	
Problem	The player can defeat or jumps over the single or two enemies of same type. Similarly, enemies of same type can be handled by similar attacks.
Solution	This new environment does not allow player to take long jump. By placing enemies of multiple type that moves randomly, player needs different type of attacks to deal with them.
Using the pattern	Use this pattern several time in layout to give a hard-hitting to player.
Comments	Provide reward or coins to increase attacking power and to balance the playability of that level.

Description of multi-type and randomly moving enemies is given in Table 5. To make a level difficult for the player enemies of multiple types are placed in such a way that they move randomly across their position, which does not allow a player to go through a long jump.

TABLE VI. DESCRIPTION OF GROUPED HARMFUL SPRITES

Harmful Sprites in grouped form	
Problem	The player can protect itself from a single harmful sprite easily.
Solution	By placing two or more harmful sprites in a grouped shape player cannot pass through them easily.
Using the pattern	Placing rewards and goals inside group creates a great challenge for player.
Comments	Use group patterns in a way that level remains playable.

Similarly, description of grouped harmful sprites is given in Table 6. By placing multi-type harmful sprites in different places, the player needs good decision-making power to protect himself from them. If a player successfully solves a pattern then he may face next challenge from the same group. A group of multiple harmful sprites can give difficult game-play to a player for survival in a level as compared with single harmful sprite. On the other hand, if the number of sprites in a group are increased for each level then it may provide a sequential play to proceed in next level.

Games	Solid Sprites					Collectible Sprites						Harmful Sprites				Enemies							
Design Pattern	Single	Boundary	Wall	Room	Movable	Single	Group	Single line & moving	Multi-line & moving	Risk & reward	Hidden	Single	Group	Multi-type	Hole	Single	Two	Single line & moving	Multi-line & moving	Randomly moving	Multi-type	Multi-type & moving randomly	Hidden
Bait	✓	✓	✓		✓	✓									✓								
Black smoke	✓	✓	✓	✓	✓	✓							✓										
Bolo adventure	✓	✓	✓	✓	✓					✓					✓								
Bomber man	✓	✓	✓	✓	✓							✓	✓	✓									
Dig dug	✓	✓	✓	✓		✓	✓			✓	✓	✓	✓	✓	✓								
Boulder chase	✓	✓	✓			✓	✓			✓	✓	✓	✓	✓									
Chips challenge	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓									
Aliens						✓	✓	✓	✓							✓	✓	✓	✓		✓		
Rogue like																✓	✓	✓		✓	✓	✓	✓
Defender		✓					✓					✓	✓					✓	✓	✓			
Road fighter		✓																✓	✓		✓		
Wild gun man		✓	✓			✓										✓	✓	✓	✓	✓	✓	✓	

Fig. 4. Comparison of design patterns and a set of GVG-LG games. (Presence of each design pattern in a game is shown by tick mark. Occurrence of Solid sprites and Harmful sprites is high in first 7 games, where as Enemies have high occurrence in last 5 games.)

V. Conclusion and Future Work

In this paper, we have discussed the ongoing work on design pattern-based level generator. This paper highlights the importance of design patterns and how design patterns can play a significant role in the level generation for general video games. Rhythmic group analysis was applied on a given framework to identify some common design patterns. The level of each game was divided into small challenging areas called cells. This approach assists to identify patterns within a level. For the initial experimentation, 23 distinct design patterns were proposed. Afterwards, these design patterns were divided into four classes: solid sprites, collectible sprites, harmful sprites, and enemies. Each game level has a design chunk composed of above-mentioned sprites. We claim that by the arrangement of these design patterns in a sequence of difficulties and using as an objective for the SB-LG, it will give a new experience to the player. In this proposed method, the SB-LG will take these patterns from the available array and will create levels for a game. It is suggested that selection of the design pattern should be in a sequential way and on the basis of its probability value in existing game level. This technique may give significantly better output by placing the variations of patterns and increasing the length of the game platform. Finally, it is concluded that these design patterns provide a useful and tangible way to generate levels for general video games.

Acknowledgment

We acknowledge Riphah International University Islamabad for support of this research.

References

[1] Statista, Global PC and console games revenue in 2014 and 2019, [www.statista.com/statistics/237187/global-videogames-revenue/], December 2015.

[2] Togelius, J., Kastbjerg, E., Schedl, D., Yannakakis, G.N.: What is procedural content generation?: Mario on the borderline. In: Proceedings of the 2nd Workshop on Procedural Content Generation in Games (2011)

[3] RM Smelik, T Tutenel, KJ de Kraker, R Bidarra, A proposal for a procedural terrain modelling framework. Euro graphics Association, 2008.

[4] Vander Linden, Roland, Ricardo Lopes, and Rafael Bidarra. Procedural generation of dungeons. IEEE Transactions on Computational Intelligence and AI in Games 6.1 (2014): 78-89.

[5] Steve, Dahlskog. Patterns And Procedural Content Generation. (2016).

[6] Dahlskog, Steve, and Julian Togelius. Patterns as objectives for level generation. (2013).

[7] Dahlskog, Steve, and Julian Togelius. A multi-level level generator.2014 IEEE Conference on Computational Intelligence and Games. IEEE,2014.

[8] A Khalifa, D Perez-Liebana, SM Lucas, General video game level generation. Proceedings of the 2016 on Genetic and Evolutionary Computation Conference.ACM, 2016.

[9] Schaul, Tom. "A video game description language for model-based or interactive learning." Computational Intelligence in Games (CIG), 2013 IEEE Conference on. IEEE, 2013.

[10] www:gamasutra:com=view=feature=174311=proceduralcontentgeneration:php

[11] Khalifa, Ahmed, and Magda Fayek. Automatic puzzle level generation: A general approach using a description language. Computational Creativity and Games Workshop 2015.

[12] Yannakakis, Georgios N., and Julian Togelius. Experience-driven procedural content generation. IEEE Transactions on Affective Computing2.3 (2011): 147-161.

[13] Alexander C, Ishikawa S, Silverstein M, i Rami JR, Jacobson M, Fiksdahl-King I. A pattern language. Gustavo Gili; 1977.

[14] Shi, Peizhi, and Ke Chen. Learning Constructive Primitives for Online Level Generation and Real-time Content Adaptation in Super Mario Bros. arXiv preprint arXiv:1510.07889 (2015).

[15] Dormans, Joris. Adventures in level design: generating missions and spaces for action adventure games. Proceedings of the 2010 workshopon procedural content generation in games. ACM, 2010..

[16] Neufeld, Xenija, Sanaz Mostaghim, and Diego Perez-Liebana. Procedural level generation with answer set programming for general videogame playing. Computer Science and Electronic Engineering Conference(CEEC), 2015 7th. IEEE, 2015.

[17] Togelius, Julian, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. "Search-based procedural content generation: A taxonomy and survey." IEEE Transactions on Computational Intelligence and AI in Games 3, no. 3 (2011): 172-186.

[18] Hullett, Kenneth, and Jim Whitehead. Design patterns in FPS levels. proceedings of the Fifth International Conference on the Foundations of Digital Games. ACM, 2010.

[19] Smith, Gillian, Mee Cha, and Jim Whitehead. A framework for analysis of 2D platformer levels. Proceedings of the 2008 ACM SIGGRAPH symposium on Video games. ACM, 2008.

[20] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, U.S.A., 1994.