Patterns, Dungeons and Generators

Steve Dahlskog
Malmö University
Faculty of Technology and Society
Malmö, Sweden
steve.dahlskog@mah.se

Staffan Björk
Göteborg University
Department of Applied Information Technology
Göteborg, Sweden
staffan.bjork@gu.se

Julian Togelius
New York University
Department of Computer Science and Engineering
New York, NY, U.S.A
julian@togelius.com

ABSTRACT

This paper analyses dungeons, of the varieties commonly found in role-playing games, into several sets of design patterns at different levels of abstraction. The analysis focuses on mechanical patterns that could be either straightforwardly instantiated or recognized by a well-defined process. At the most concrete level a set of fundamental components were identified, followed by a long list of micro-patterns which can be directly instantiated. Shorter lists of meso- and macro-patterns, which can be identified mechanically, are also identified. The direct motivation for this analysis is to find building blocks and objectives for a search-based procedural dungeon generator, however we believe the analysis can be useful for understanding this class of game artifacts in general. In particular, the constraints on patterns being instantiable or recognizable leads to a stricter pattern analysis than many other attempts at analyzing game design.

Keywords

design patterns, dungeons, procedural content generation

1. INTRODUCTION

Design patterns have become an important tool for analyzing and reasoning about game design. They provide a relatively formal way of talking about game design, which is appealing particularly for those who want to automate the analysis and/or generation of game content. In recent research, a method has been devised for procedurally generating platform game levels based on design patterns [19, 21]. This method is based on the idea that design patterns can be ordered into different levels of abstraction, from smaller and more concrete patterns to larger and more abstract patterns. The larger patterns can be instantiated in multiple ways through different combinations of smaller patterns. Levels can then be generated through searching for combinations of smaller patterns that yield certain larger patterns.

This paper addresses the domain of dungeons, the type of levels with a spatial puzzle quality first introduced in *Dungeons & Dragons* [31] and typically found in "roguelikes" such as *Rogue* [59], *Moria* [40], and *Hack* [24], and computer role-playing games (RPGs) such as *Bard's Tale* [37] and *Ultima* [28]. We identify a relatively large set of design patterns at different sizes and levels of abstraction – micro-, meso- and macro-patterns. The motivation for carrying out this analysis is to find patterns that can be used for pattern-based or search-based dungeon generation and for this reason the granularity is finer and the level of detail of the pattern collection is higher than what is typical for pattern collections. We believe the pattern analysis carried out here has value for other purposes as well, such as understanding the design space and design affordances of dungeons as a game artifact.

2. RELATED WORK

Due to the approach chosen in this paper the related work is connected to several different research areas related to games. In the following section an overview of these will be presented together with examples of relevant games.

2.1 Game Spaces and Dungeons

Exploration or movement through spaces are common gameplay features in games. In line with this, Aarseth calls spatiality a defining element of games and argues for a possible use in classifying games according to how the space is implemented in the game [2]. Exploring the concept of Game spaces, Nitsche introduces three concepts relevant to this paper in the form of labyrinths (linear or unicursal), mazes (branching or multicursal) and arenas (open structures with areas of free movement bounded by surrounding enclosement) [50]. Similarly, Aarseth call games where an avatar has to be moved from a starting to end position "place-oriented" and identifies hubs, open landscapes, and uni- or multicursal structures as components of quest-based games [1].

A common game space in fantasy RPGs is the dungeon. The dungeon has been present in fantasy RPGs more or less from its introduction in *Dungeons & Dragons* [31] in 1974 until today. The following description is provided in a later edition of the game: "A Dungeon is a group of rooms and corridors in which monsters and treasures can be found." [32]. This definition is arguably rather open and could include other types of game spaces from other types of games, like bunkers, castles and other buildings in, for example First-Person Shooters. For the purposes of this paper, we accept this definition and the consequence that it might include similar spaces in other games, but we base our analysis only on games that are commonly agreed to fall into the RPG genre.

The ubiquity of dungeons in RPGs indicate that they solve recurring problems in game design, and provide a key part of the player experience. From a player perspective, it is likely that the rather confined space of the dungeon provides interestingness by allowing exploration of a non-trivial layout of space, and excitement due to incorporated components such as enemies, traps, and treasures. The constraint on player movement can create additional tension and exploring dungeons may in many cases also impose a level of resource management (e.g. considering how much provisions and consumables should be brought or when attempt to resupply should be made). Similarly, it is likely from a designer's viewpoint that explicitly limiting the player's available choices and access to information helps structure the order in which players gains access to the game (and thus the story). By providing access to certain areas only in a specific order, it becomes easier to combine a storyline with the relatively free exploration of non-dungeon parts ("overworld") of many RPGs.

2.2 Design Patterns

Design patterns is the idea that certain design solutions can be described on an abstracted level so they can both be re-used in similar contexts and casual relations between them can be identified. They were originally developed for architecture by Alexander *et al.* to help end users take part in design processes [3] as part of a large movement focusing on understanding design methods (*cf.* [39]). The use of design patterns for understanding games was first introduced by Kreimeier [41] and then followed by Björk & Holopainen who developed a collection of approximately 300 design patterns [13]. Similar approaches include the 400 rules project [9] and the game ontology project [62]. This paper follows a convention to mark patterns through the use of SMALL CAPS.

While the collection developed by Björk & Holopainen include some patterns appropriate for analyzing the design of dungeons in RPGs, other more specific collections related to this have been developed. Hullett & Whitehead [34] explored the design of levels in First-Person Shooters and created a collection of 10 design patterns which have later been incorporated in the collection initiated by Björk & Holopainen [12]. Smith *et al.* analysed 20 games and the resulting design patterns were grouped into level and quest patterns in RPGs [56][1].

Gameplay design patterns have been combined with the Mechanics-Dynamics-Aesthetics framework [35] to explain how patterns dealing with concrete rules or game elements can cause dynamic behaviors and through this aesthetical experience. This has been used to both explore camaraderie [10] and pottering [44] in games and to compare similarities and difference between the game *X-COM: UFO Defense* [46] and its remake *XCOM: Enemy Unknown* [26, 16].

2.3 Procedural Content Generation

Procedural content generation (PCG) in games refers to methods for algorithmically creating game content (e.g. games, rules, worlds, levels, items, etc.). PCG might be used independently or to assist a human designer, with human design objectives and partial designs as input. While many early digital games featured some kind of PCG, it has only become an active area of research in academic settings within the last few years [54]. PCG has been used in many games of different genres, but is perhaps most central to games relying on runtime generation dungeons like *Diablo* [15], *Rogue* and *Ultima I*. Even as early as in 1979, methods for generating dungeons were present in both pen-and-paper RPGs (*Advanced Dungeons & Dragons* [30]) as well as in digital games (*Akalabeth: World of Doom* [52]).

A recent paper surveyed procedurally generated dungeons showing previous work from a technical perspective, amount of control over the generative process, the output and results. In the paper, the authors argue for researching dungeons due to its close relation to successful games [61]. Procedural dungeon generation has also been surveyed in the textbook on PCG in games [54]. A wide variety of different methods have been applied to generating dungeons. These include evolutionary algorithms [5, 4, 45, 60, 6], grammar expansion [22, 23], cellular automata [38], constraint solving [33, 55], and various ad-hoc methods such as dungeon diggers and binary space partitioning [54].

2.4 Design Patterns used in PCG

Given that design patterns are formalizations of game and level design into relatively simple and independent components, it stands to reason that they would be useful in content generators. A content generator is after all exactly a generative formal design theory. Thus, several recent experiments in procedural content generation use the language of design patterns to describe parts of the content generator or the artifacts it produces [25, 51].

Dahlskog *et al.* have taken the metaphor further and developed a search-based level generator based on design patterns on different levels of abstraction [19, 20]. The prototype implementation, which generates levels for *Super Mario Bros* (SMB) [47], uses patterns on the micro-, meso- and macro-levels. Micro-patterns are simple vertical slices of *SMB* levels, meso-patterns are larger features such as enemy hordes or "pipe valleys", and macro-patterns are sequences of meso-patterns. The patterns at each level of abstraction are composed of multiple patterns of a lower level of abstraction. This configurations allows levels to be created through searching the space of sequences of micro-patterns for occurrences of macro-patterns. In effect, micro-patterns are used as building blocks and meso- and macro-patterns as objectives.

3. CLASSIFICATION OF DUNGEONS

In order to argue for what a dungeon looks like we surveyed a large number of dungeons in a set of RPGs and gathered empirical data by brute force incremental analysis similar to [14]. The approach focused on identifying patterns based on game space and game mechanics related to dungeons in each game. The approach included actual game playing, primarily on emulators, supported with strategy guides, maps and Youtube clips to minimize playtime outside the actual dungeons (i.e. minimal time was committed to exploring story or solving puzzles). We picked games from an exhaustive source [8] and correlated this source with the game magazine *Computer Gaming World*'s lists

[1]http://rpgpatterns.soe.ucsc.edu/

Table 1: Fundamental Components

Tile	The basic unit of space in a dungeon. Individual Tiles have Boolean attributes associated with them: *Passable* and *Seethru.*
Level	A rectangular space of Tile.
Wall Tile	The base classification of Tiles in a Level. Tiles belonging to this category are not *Passable* and not *Seethru.* The Base Content of a Tile in *Rogue* is "Rock".
Ground Tile	A classification for Tiles that are *Passable* and *Seethru.*
Item	A game object that can be in a Tile but which can also be picked up, carried, and dropped in other Tiles.
Agent	A game object that can perform actions, e.g. moving and attacking, and is located on a specific Tile. The player's avatar is an Agent as are monsters. They typically hinder other from entering the Tile they are in, i.e. they temporary remove the attribute *Passable* from the Tile they are in.
Line-of-Sight(A,B)	A Boolean function returning if all Tiles on a straight line between point A and B in a Level are *Seethru.*
Traversable(A,B)	A Boolean function returning if a player can move between point A and B in a Level. A Route is a traversal solution and the length of different Routes may be needed for some design patterns.
Sequenced(A,B)	A Boolean function returning if point A must be visited before point B in a Level.

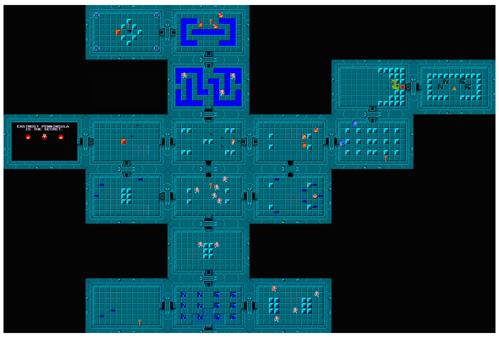

Figure 1: A dungeon in The Legend of Zelda (*Connected Rooms*).

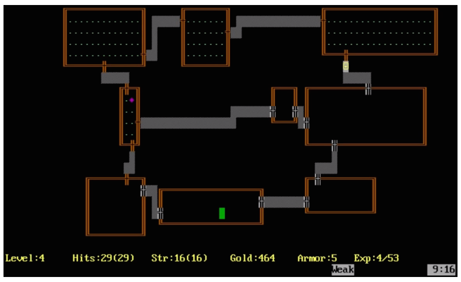

Figure 2: A dungeon in Rogue (*Rooms & Corridors*).

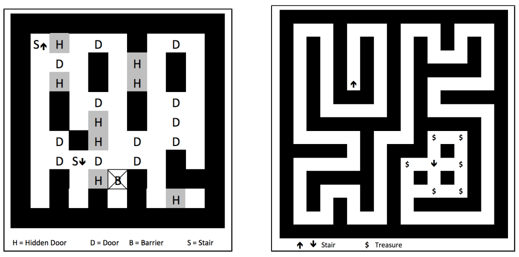

Figure 3: Dungeons in Ultima I (*Maze*) and Ultima II (*Labyrinth*).

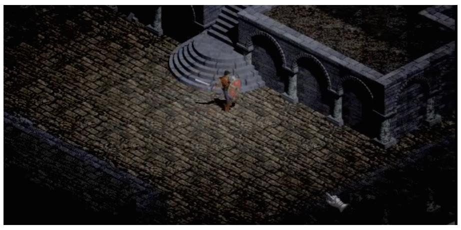

Figure 4: A dungeon in Diablo (*Open area*).

of RPGs [53] to have a rich data set. All in all, 91 games released between 1975-1993 were analyzed in detail but several more modern games will be used as examples in the text. The motivation for focusing on our analysis on the period between 1975 and 1993 is due to resource limitations as well as these early games have a stronger focus on actual dungeons whereas modern games have the possibility of adding open world-like areas with the effect that the player spend more time there than in dungeons (*c.f. The Elder Scrolls V: Skyrim* [11]). We intend to add more games to the list in the future as we extend the project with content generation.

In essence, the surveyed games showed similarities on the account of topology and the different dungeons could be classified and grouped into five different types. Commonly the topology could be described as scarce or dense depending on how much traversable game space the dungeon layout contains (dense dungeons have more traversable space).

1. *Connected Rooms* is a type of dungeon that is most common in classic text-based adventure games. The player moves from interesting sites (rooms) without explicit corridors, pathways or tunnels to the next site. Games like *Colossal Cave Adventure* [18] and *Zork* [43, 36] but also *The Legend of Zelda* [49] (see Fig. 1) have such dungeons; they are efficiently mapped with graphs.

2. A *Rooms & Corridors* dungeon is often scarce with a small set of rooms connected with non-branching corridors. Typical examples of this type of dungeons are

present in *Rogue* (see Fig. 2) and roguelike games from the 1980-ies. Corridors are functional game space and events (combat) can take place there. If the game allows the player to dig through walls, graphs will not be sufficient to map the game space and 2D-matrices are needed.

3. *Labyrinths* are unicursal structures with a single path leading through the dungeon. Earlier games like *Ultima II* [29] (see Fig. 3). These dungeons are often dense and demand 2D-matrices for mapping.

4. *Mazes* are multicursal layouts with multiple paths leading through the dungeon. Both earlier and later games have this kind of topology. It is often dense and games like *Akalabeth: World of Doom* and *Ultima I* have these kinds of dungeons (see Fig. 3).

5. *Open area*-dungeons consists of extremely open space (for a dungeon) with obstacles (e.g. thin walls) that hinders free maneuvering, but in comparison with the other types of dungeons, the tactical maneuvering have greater importance. Corridors are uncommon and if one considers the traversable space of these dungeons they are very dense. Games like *Telengard* [42] and *Diablo* (see Fig. 4) are typical for this kind of dungeons.

4. PATTERNS

After surveying the 91 games, the next step was to identify design patterns within these related to the level, or dungeon, design. However, the domain of the study needed to be delimited to a manageable size and an emphasis was placed on the *Rooms & Corridors*, *Labyrinths*, and *Mazes* classifications. More specifically, we decided on studying dungeons as they can appear in two-dimensional games where movement primarily takes place horizontally, and is characterized by exploration in constrained spaces and progression. Further, we restricted ourselves to games where the dungeons can be specified as combinations of multiple tiles, though movement between the tiles may or may not be pseudo-continuous. This definition includes dungeons in classic CRPGs such as early games in the *Legend of Zelda* series and *Final Fantasy* [57] series as well as roguelikes like *Rogue*, *Nethack* [58] and *Diablo*. We are excluding games with a large amount of vertical moment, in particular platformers like *Super Mario Bros*, and games which prominently feature three-dimensional exploration like *Tomb Raider* [17]. Furthermore, we are looking at patterns that exist in multiple games, not just one or a few. Even so, all the 91 games analyzed continued to be used as sources when possible.

Design patterns provide abstract descriptions of solutions to common problems. Although this abstraction can easily be contextualized by human designers given a specific design problem, the same does not apply to systems that procedurally generate solutions. The strategy we applied to bridge this issue was to introduce several layers of interrelated patterns where the lowest levels have a low degree of abstraction and therefore can be easily instantiated or recognized by an algorithm, making the patterns useful as part of a procedural dungeon generator. Previously Dahlskog & Togelius [19, 20] and Ferreira & Toledo [25] have done so in other domains. As a result, the pattern analysis is here in some senses more rigorous and formal than similar pattern analyses found elsewhere, but also more limited. In relation to the different levels of the Mechanics-Dynamics-Aesthetics (MDA) framework [35], the focus was squarely on mechanical patterns, omitting dynamical or aesthetic ones.

As the immediate motivation for this study is to find a set of patterns that can be used for content generation, we needed to place constraints on the patterns we found. A first constraint we set up was to use a set of fundamental components and that patterns should be described in terms of these components and other patterns. This implied a hierarchy of patterns which lead to a classification of patterns as either micro-patterns, meso-patterns or macro-patterns similar to what was done earlier by Dahlskog *et al.* [19]. The relation between these is that meso-patterns can be built out of configurations of multiple micro-patterns, and macro-patterns can be built out of combinations of micro- and meso-patterns. Micro-patterns have the further constraint that they should be *mechanically instantiable*: it should be possible for a constructive algorithm to instantiate any of those patterns at any given position in a tile by simply "dropping it in" without analyzing more than the immediately neighboring tiles.

If should be noted that the collection presented here is non-exhaustive. The patterns presented are however sufficient to create the dungeons with all the main features found in RPG dungeons. When reading the tables, note that most entities are classified as being more specific versions of other entities and this is indicated by listing the more general entity in parenthesis after an entity name.

4.1 Fundamental Components

For the context of this paper, two Boolean attributes are relevant for TILES: *Passable* and *Seethru*. *Passable* TILES can be both entered and left while *Passable* are those that can be seen through as well as allow ranged attacked to pass through them. These and several other basic concepts can be found in table 1.

4.2 Micro-patterns

The Micro-patterns introduced here make use of *types* that describes categories of Micro-patterns. For example, all patterns classified as *Space* create areas where players can move and individual Micro-Patterns make use of this type to describe how they can be combined with other patterns to create larger areas where players can move around. A list of identified micro-patters can be found in table 2.

4.3 Meso-patterns

While Meso-patterns are also mechanical in the sense that they can be observed from static instances of a game in progress (and thus identified by a static evaluation function), they are abstract in the sense that their existence relies on combinations of Micro-patterns. Many of the patterns mentioned in this section (and the next) have already been documented as patterns earlier (*cf.* [12, 56, 34]), although in some cases under other names. However, those descriptions were not specific enough to be the basis for an intended dungeon generator so alternative versions are presented here. Our list of identified meso-patterns can be found in table 3.

4.4 Macro-patterns

The highest level of abstraction used in this collection

Table 2: Micro-patterns.

SPACE	A SPACE is a group of connected TILES that share the same attributes and optionally other design features such as topological properties or gameplay functionality. The simplest SPACE is simply one TILE.
CORRIDOR (Space)	A CORRIDOR is a series of horizontal or vertical GROUND TILE. The end points of a CORRIDOR can be connected to other *Spaces*. In *Rogue* CORRIDORS connect ROOMS but in other games like *Ultima I* (see Fig. 3) CORRIDORS is the main spatial component and ROOMS are missing.
CONNECTOR (Space)	A CONNECTOR is a 1 GROUND TILE long CORRIDOR that is used to let passages in dungeons turn or allow intersections by being connected to other *Spaces*.
ROOM (Space)	A ROOM consists of several GROUND TILES but wider than a CORRIDOR and allow for more freedom in movement.
DOOR (Space)	The DOOR is a barrier that has two states; "open" or "closed". Open doors are *Passable* and *Seethru* while closed door are neither. Connection to other *Spaces* are typically in a north-sound or west-east direction. In *Diablo* ENEMIES does not notice the player character through DOORS. Some DOORS have a connected key that allow for a third state "locked". Some games have "breakable" DOORS which puts the door in a constant "open" state.
HIDDEN DOOR (Space)	HIDDEN DOOR is a DOOR that functions like a WALL until it has been revealed.
KEY (Item) [56]	A *Key* is a *Item* allowing the ability of altering any or a specific DOOR'S state regarding being "locked".
PROPS (Space)	PROPS are GROUND TILES with decorations but no extra functionality. Doors in *Rogue* are PROPS!
OBSTACLES (Item)	OBSTACLES occupy TILES and hinder AGENTS from entering them. They can be modeled as ITEMS that cannot be picked up and make TILES temporary lose any *Passable* attribute like AGENTS but can also be modeled as AGENTS that cannot perform actions. They may be destroyable or movable. Examples include boulders in *Rogue*.
INSTALLATIONS (Props)	INSTALLATIONS are PROPS (or OBSTACLES) that allow actions, typically by the players' avatars when they are next to them, but INSTALLATIONS can also provide actions to other AGENTS or be activated by the game system. Example includes fountains in roguelikes and mana pools in *Diablo*.
CONTAINERS (Installation) [56]	CONTAINERS allow caches of ITEMS to be accessed from the TILES they are placed in. While they typically are INSTALLATIONS they can also be ITEMS. Like DOORS, CONTAINERS may be locked so possession of KEYS or abilities to pick locks may be need to have access to the ITEMS inside CONTAINERS. *The Legend of Zelda: A Link to the Past* [48] has *Chests* that need KEYS to be unlocked while *Nethack* provides a multitude of ways to open locked *Chests*, none which involve KEYS.
STAIRS (Installation)	STAIRS are INSTALLATIONS that allow movement away from the current LEVEL. This is typically to another LEVEL which requires the position of a entry point on that LEVEL. When movement back is possible the natural design solution is to have STAIRS on the entry point which leads back to the original STAIRS TILE. If this is not possible, a PROP indicating the entry point can provide diegetic consistency [12]. In several games this vertical movement affects the difficulty or strength of the opposing ENEMY.
IMPASSABLE SPACE (Space)	An IMPASSABLE SPACE a group of connected TILES that are not *Passable* but *Seethru* and thereby allows LINE-OF-SIGHT through them. In *Diablo* the lava lakes works as IMPASSABLE TERRAIN.
TRAPS (Installations) [56]	TRAPS are typically hidden and perform a one-time attack on AGENTS entering the TILE they are placed in. Traps are typically hidden until activated or revealed, and may be disarmed when revealed. They are typically modeled as INSTALLATIONS with automatic activation but could also be created as AGENTS that cannot move. Examples of TRAPS include bear traps in *Rogue* and pits in *Nethack*.
ENEMIES (Agents)	ENEMIES are simply AGENTS whose primary behavior is to attack players' avatars. A main dichotomy regarding ENEMIES are if they can move or not.
SPAWN POINTS (Installation) [56]	SPAWN POINTS are INSTALLATIONS from which ENEMIES appear. Generators in *Gauntlet* [7] are SPAWN POINTS.
PORTALS (Installations) [56]	PORTALS allow instantaneous movement between to spatially separated points in a level. They can be modeled as STAIRS which may be either PROPS or OBSTACLES and can either work only in direction or both directions. The *Bard's Tale* games include PORTALS in their dungeon design.

Table 3: Meso-Patterns

CHOKE POINTS (Space) [12, 34, 56]	CHOKE POINTS are TILES that are the only connections between two different parts of a LEVEL, or in other words: a TILE C is a CHOKE POINT if there exists TILES A and B so that TRAVERSABLE(A,B) is true but all ROUTES require passing through C. CHOKE POINTS can be created from DOORS or PORTALS and allow sequences of SEQUENCED relations to be constructed; a CHOKE POINTS C between A and B enforce SEQUENCED(A,C) and SEQUENCED(C,B). CHOKE POINTS larger than TILES can be constructed through placing SPACES such as ROOMS behind other CHOKE POINTS.
SPECIAL ROOMS (Room)	These are ROOM created together with specific content such as ITEMS or ENEMIES in them as well as possibly having restrictions on access to them. *Shops* in *Nethack* is an example of a SPECIAL ROOM; it is only accessible through a CHOKE POINT, includes a *Shopkeeper* and have INSTALLATIONS on all TILES to handle buying and selling ITEMS. *Beehives* is another example from the same game which populate a ROOM with ENEMIES in the form of *Killer Bees* and *Queen Bees* together with *Royal Jelly* ITEMS.
DEAD ENDS (Space)	DEAD ENDS are locations from which players must move through previously explored areas. The simplest form of DEAD END is constructed by placing a CHOKE POINT between a GROUND TILE and the rest of a LEVEL. However, other SPACES such as CORRIDORS or ROOMS can also be the DEAD ENDS after a CHOKE POINT. This pattern is subjective to the amount of gameplay that is provided behind the CHOKE POINT, no or little gameplay can make a larger SPACE into DEAD ENDS while a small SPACE with rich gameplay is less of a DEAD END (cf. the *Shop* in SPECIAL ROOMS pattern.)
CONDITIONAL PASSAGEWAYS (Choke Point) [12, 56]	CONDITIONAL PASSAGEWAYS are TILES that are only *Passable* when a character has a special skill or *Item*. They need to be CHOKE POINTS to avoid functionally becoming equivalent to OBSTACLES. The possibility of CONDITIONAL PASSAGEWAYS necessitate the consideration of a CONDITIONALLY TRAVERSABLE(A,B) function. The presence of CONDITIONAL PASSAGEWAYS can make CHOKE POINTS directional in that they only are CHOKE POINTS when moving from A to B but not when moving from B and A. CONDITIONAL PASSAGEWAYS can be created through the use of DOORS and KEYS but *Pokémon* [27] provides an example of another solution: a Bicycle is needed to go on "Cycling Road" (Kanto and Sinnoh regions) and Seaside Cycling Road.
ONE-WAY TRAVEL (Route) [12]	This is design solution which makes a ROUTE between A and B so that TRAVERSABLE(A,B) is true while TRAVERSABLE(B,A) is false. This is typically done through PORTALS or STAIRS but the introduction of one-way DOORS is another possibility.
FLANKING ROUTES (Route) [12, 34]	These ROUTES offer alternatives to what is perceived as the most direct ROUTE between SPACE A and B. They can be created through first creating a ROUTE that is TRAVERSABLE(A,B) and then create another ROUTE which is less obvious. This latter feature can be achieved by making the ROUTE longer, hiding it through the use of SECRET DOORS, or making it CONDITIONALLY TRAVERSABLE(A,B) through the use of CONDITIONAL PASSAGEWAYS.

are Macro-Patterns. These are pattern defined through the use of the fundamental components and lower-level patterns, typically focusing on longer periods of gameplay or — somewhat paradoxically — specific gameplay aspects that depend on a combination of circumstances. Like in the case of Meso-Patterns, they have already been identified in a variety of other games. Table 4 contains our list of macro-patterns.

5. DISCUSSION AND CONCLUSIONS

The patterns presented above are only a sample of the possible patterns for procedurally generating dungeons. Many more, e.g. ARENAS, BOSS MONSTER DUNGEON, MULTI-LEVEL DUNGEON, and SECRET AREAS, could have been included, though that would require a longer paper format. However, one can ask what the benefits of the presented framework here provides given that several of the games examined already create procedurally generated dungeons. However, the pattern-based content generation approach allows us to build generators that respect design constraints and implement patterns on different levels, unlike the relatively unstructured output of many dungeon generators.

The pattern collection provides an abstract model of level design in "dungeon crawl" RPGs. These patterns were constructed after reviewing 91 games using dungeon so they reflect actual design practice of game designers. As such they present a model usable for both designers and researchers, and can support either manually, procedurally generated dungeon designs or a mixed-method approach. Although the pattern collection does not yet reach the level of abstraction used by Aarseth or Nitsche (i.e. [1] and [50]), we believe further work can define labyrinths, mazes, and hubs as patterns and potentially as part of systems for PCG dungeons. A first indication of how such patterns would look can be gleamed from the LABYRINTH and HUB-AND-SPOKE patterns by Smith *et al.* [56].

While the meso- and macro-level patterns have been described so that implementing them should be fairly unproblematic, in several cases alternatives have been provided. This shows how patterns can be implemented in different ways to achieve the same gameplay functionality and offers designers choices to use the most appropriate solutions.

One of the pattern presented, SNIPER LOCATIONS, was actually not found in the examined games. It was included to show how the pattern collection can be extended to support additional types of gameplay (which in this case already exists in other genres) through introducing patterns building on already existing patterns. While this can also be done through adding new fundamental components or micro-patterns, this has a larger risk of fundamentally changing the design so resulting gameplay no longer is seen as being part of the genre.

On a more theoretical level, the model of using Micro-, Meso-, and Macro-patterns show how game design practice can be analyzed in greater detail and be described to a level where different levels of abstract on the "mechanical" level of design patterns can be implemented in code for dungeon RPGs. In future work we aim at showing how Meso-patterns can be given the requirement to be *mechanically recognizable*, *i.e.* it should be possible for an algorithm to recognize all instances of any given meso-pattern in a dungeon through a direct evaluation function. We hope that, despite the limited scope, the added rigor might be useful outside of procedural content generation as well since it shows how the knowledge contained in many previously identified patterns can be rephrased to be usuable more directly in implementation.

Concluding, the collection of patterns presented here provide an overview of level design for dungeons to a level of granularity that supports the design of PCG dungeon systems. While implementation of such a system is the next step in our work, we believe this collection has a value in describing dungeon level design through a tiered model that can also support manual construction of dungeons as well as provide a tool for further analysis of dungeon level designs.

6. REFERENCES

[1] E. Aarseth. From Hunt the Wumpus to Everquest: Introduction to Quest Theory. In *Proceedings of the 4th International Conference on Entertainment Computing*, ICEC'05, pages 496–506, Berlin, Heidelberg, 2005. Springer-Verlag.

[2] E. Aarseth. Allegories of space: The question of spatiality in computer games. In F. von Borries, S. P. Walz, and M. Böttger, editors, *Space Time Play: Synergies Between Computer Games, Architecture and Urbanism: the Next Level*, pages 44–55. BirkHäuser, 2007.

[3] C. Alexander, S. Ishikawa, and M. Silverstein. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, New York, August 1977.

[4] D. Ashlock, C. Lee, and C. McGuinness. Search-based procedural generation of maze-like levels. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(3):260–273, 2011.

[5] D. Ashlock, C. Lee, and C. McGuinness. Simultaneous Dual Level Creation for Games. *Computational Intelligence Magazine, IEEE*, 6(2):26–37, May 2011.

[6] D. Ashlock and C. McGuinness. Automatic generation of fantasy role-playing modules. In *Proceedings of the 2014 IEEE Conference on Computational Intelligence and Games*. IEEE, August 2014.

[7] Atari Games. Gauntlet. [Digital game], 1985.

[8] M. Barton. *Dungeons and Desktops: The History of Computer Role-playing Games.* A K Peters Ltd, 2008.

[9] H. Barwood and N. Falstein. 400 Rules Project. Web page, February 2015.

[10] K. Bergström, S. Björk, and S. Lundgren. Exploring aesthetical gameplay design patterns: camaraderie in four games. In A. Lugmayr, H. Franssila, O. Sotamaa, C. Safran, and T. Aaltonen, editors, *MindTrek*, pages 17–24. ACM, 2010.

[11] Bethesda Game Studios. The Elder Scrolls V: Skyrim. [Digital game], 2013.

[12] S. Björk. Gameplay Design Patterns 2.0. Web page, February 2015.

[13] S. Björk and J. Holopainen. *Patterns in Game Design.* Charles River Media game development series. Charles River Media, 2005.

[14] S. Björk, S. Lundgren, and J. Holopainen. Game Design Patterns. In *Proceedings of the 2003 DiGRA International Conference: Level Up*, 2003.

[15] Blizzard North. Diablo. [Digital game], December 1996.

[16] A. Canossa, S. Björk, and M. J. Nelson. X-COM: UFO Defense vs. XCOM: Enemy Unknown— using

Table 4: Macro-Patterns

QUICK RETURNS [12]	QUICK RETURNS intend to let players explore a part of a LEVEL but offer a quick way of returning to previously explored parts after reaching a certain point. This can be modeled by designing so that TILES A and B are TRAVERSABLE(A,B) (or CONDITIONALLY TRAVERSABLE(A,B)) with a certain minimum length but the solution for TRAVERSABLE(B,A)) is shorter. This is typically achieved through constructing TRAVERSABLE(A,B) through a number of CHOKE POINTS but providing ONE-WAY TRAVEL from B to A or a CONDITIONAL PASSAGEWAY at B that activates a shorter CONDITIONALLY TRAVERSABLE(B,A) than A had to B. The Portals to the town in *Diablo* and the Castle of Ordeals are examples of QUICK RETURNS.
BACKTRACKING LEVELS [12]	Somewhat misnamed as the pattern can be applied to parts of LEVELS, BACKTRACKING LEVELS denote design solutions where players need to move from TILE A to B and then return following basically the same ROUTE. BACKTRACKING LEVELS can be constructed from inserting a chain of CHOKE POINTS between A to B and making B part of a DEAD END. BACKTRACKING LEVELS can also be applied to chains of LEVELS, the goal of *Rogue* is to descend through LEVELS until one finds the amulet of Yendor and then ascend back up to the starting point.
SNIPER LOCATIONS [12, 34]	Places advantageous to making ranged attacks against ENEMIES classify as SNIPER LOCATIONS. These can most easily be created by having two SPACES connected by an IMPASSABLE SPACE. However, SNIPER LOCATIONS should be relatively safe also in that ENEMIES cannot quickly reach them. The use of SECRET DOORS, CONDITIONAL PASSAGEWAY or ROUTES of a certain minimum length between the two SPACES can achieve this.

gameplay design patterns to understand game remakes. In *Proceedings of the Ninth International Conference on the Foundations of Digital Games*, 2014.

[17] Core Design. Tomb Raider. [Digital game], 1996.

[18] W. Crowther. Colossal Cave Adventure. [Digital game], 1976.

[19] S. Dahlskog and J. Togelius. Patterns and Procedural Content Generation: Revisiting Mario in World 1 Level 1. In *Proceedings of the First Workshop on Design Patterns in Games*, pages 1:1–1:8, New York, NY, USA, 2012. ACM.

[20] S. Dahlskog and J. Togelius. Patterns as Objectives for Level Generation. In *Proceedings of the Second Workshop on Design Patterns in Games*, May 2013.

[21] S. Dahlskog and J. Togelius. A Multi-level Level Generator. In *Proceedings of the 2014 IEEE Conference on Computational Intelligence and Games*, pages 389–396. IEEE, August 2014.

[22] J. Dormans. Adventures in Level Design: Generating Missions and Spaces for Action Adventure Games. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, pages 1:1–1:8, New York, NY, USA, 2010. ACM.

[23] J. Dormans and S. Leijnen. Combinatorial and exploratory creativity in procedural content generation. In *Proceedings of the 2013 Workshop on Procedural Content Generation in Games*, 2013.

[24] J. Fenlason, K. Woodland, M. Thome, J. Payne, A. Brouwer, and D. Kneller. Hack. [Digital game], 1982-1985.

[25] L. Ferreira and C. Toledo. A search-based approach for generating angry birds levels. In *Proceedings of the 9th IEEE International Conference on Computational Intelligence in Games*, 2014.

[26] Firaxis Games. XCOM: Enemy Unknown. [Digital game], 2012.

[27] Game Freak. Pokémon Red/Blue Version. [Digital game], 1996.

[28] R. Garriott. Ultima. [Digital game], 1981.

[29] R. Garriott. Ultima II: The Revenge of the Enchantress. [Digital game], 1982.

[30] Gary Gygax. Dungeon Masters Guide (sic!). [Role-playing game], 1979.

[31] Gary Gygax and Dave Arneson. Dungeons & Dragons. [Role-playing game], 1974.

[32] Gary Gygax and Dave Arneson and Frank Mentzer. Dungeons & Dragons Set 1: Basic Rules. [Role-playing game], 1983.

[33] K. Hartsook, A. Zook, S. Das, and M. Riedl. Toward supporting stories with procedurally generated game worlds. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 297–304, Aug 2011.

[34] K. Hullett and J. Whitehead. Design Patterns in FPS Levels. In *FDG '10: Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pages 78–85, New York, NY, USA, 2010. ACM.

[35] R. Hunicke, M. Leblanc, and R. Zubek. Mda: A formal approach to game design and game research. In *In Proceedings of the Challenges in Games AI Workshop, Nineteenth National Conference of Artificial Intelligence*, pages 1–5. Press, 2004.

[36] Infocom. Zork 1. [Digital game], 1980.

[37] Interplay Productions. Tales of the Unknown, Volume I: The Bard's Tale. [Digital game], 1985.

[38] L. Johnson, G. N. Yannakakis, and J. Togelius. Cellular automata for real-time generation of infinite cave levels. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, pages 10:1–10:4, New York, NY, USA, 2010. ACM.

[39] J. Jones. *Design Methods*. Architecture Series. Wiley, 1992.

[40] R. A. Koeneke and J. W. Todd. Moria. [Digital game], 1994.

[41] B. Kreimeier. The case for game design patterns. 2002.

[42] D. Lawrence. Telengard. [Digital game], 1982.

[43] P. Lebling, M. Blank, and T. Anderson. Special Feature Zork: A Computerized Fantasy Simulation Game. *Computer*, 12(4):51–59, April 1979.

[44] S. Lundgren and S. Björk. Neither playing nor gaming: pottering in games. In M. S. El-Nasr, M. Consalvo, and S. K. Feiner, editors, *FDG*, pages 113–120. ACM, 2012.

[45] C. McGuinness and D. Ashlock. Decomposing the level generation problem with tiles. In *IEEE Congress on Evolutionary Computation*, pages 849–856. IEEE, 2011.

[46] Mythos Games. UFO: Enemy Unknown (marketed as X-COM: UFO Defense in NA). [Digital game], 1994.

[47] Nintendo. Super Mario Bros. [Digital game], 1985.

[48] Nintendo EAD. The Legend of Zelda: A Link to the Past. [Digital game], 1991.

[49] Nintendo R&D4. The Legend of Zelda. [Digital game], 1986.

[50] M. Nitsche. *Video Game Spaces: Image, Play, and Structure in 3D Worlds.* Game studies. MIT Press, Cambridge, MA, U.S.A., 2009.

[51] A. Pantaleev. In search of patterns: Disrupting rpg classes through procedural content generation. In *Proceedings of the 2012 Workshop on Procedural Content Generation in Games*, pages 57–61, May 2012.

[52] Richard Garriott. Akalabeth: World of Doom. [Digital game], 1979.

[53] Scorpia. Scorpia's Role-Playing Game Survey. *Computer Gaming World*, 87:16–27, 107–109, 1991.

[54] N. Shaker, J. Togelius, and M. J. Nelson. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research.* Springer, 2014.

[55] A. M. Smith and M. Mateas. Answer Set Programming for Procedural Content Generation: A Design Space Approach. *IEEE Trans. Comput. Intellig. and AI in Games*, 3(3):187–200, 2011.

[56] G. Smith, R. Anderson, B. Kopleck, Z. Lindblad, L. Scott, A. Wardell, J. Whitehead, and M. Mateas. Situating quests: Design patterns for quest and level design in role-playing games. In M. Si, D. Thue, E. André, J. C. Lester, J. Tanenbaum, and V. Zammitto, editors, *ICIDS*, volume 7069 of *LNCS*, pages 326–329, Berlin / Heidelberg, 2011. Springer.

[57] Square. Final fantasy. [Digital game], 1987.

[58] The NetHack DevTeam. NetHack. [Digital game], 1987.

[59] M. Toy, G. Wichman, K. Arnold, and J. Lane. Rogue. [Digital game], 1980.

[60] V. Valtchanov and J. A. Brown. Evolving Dungeon Crawler Levels with Relative Placement. In *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering*, C3S2E '12, pages 27–35, New York, NY, USA, 2012. ACM.

[61] R. van der Linden, R. Lopes, and R. Bidarra. Procedural generation of dungeons. *Computational Intelligence and AI in Games, IEEE Transactions on*, 6(1):78–89, March 2014.

[62] J. P. Zagal, M. Mateas, C. Fernández-vara, B. Hochhalter, and N. Lichti. Towards an ontological language for game analysis. In *in Proceedings of International DiGRA Conference*, pages 3–14, 2005.