Towards a Generic Method of Evaluating Game Levels

Antonios Liapis[1], **Georgios N. Yannakakis**[1,2], **Julian Togelius**[1]
1: Center for Computer Games Research, IT University of Copenhagen, Copenhagen, Denmark
2: Institute of Digital Games, University of Malta, Msida, Malta
Mail: anli@itu.dk, georgios.yannakakis@um.edu.mt, julian@togelius.com

Abstract

This paper addresses the problem of evaluating the quality of game levels across different games and even genres, which is of key importance for making procedural content generation and assisted game design tools more generally applicable. Three game design patterns are identified for having high generality while being easily quantifiable: area control, exploration and balance. Formulas for measuring the extent to which a level includes these concepts are proposed, and evaluation functions are derived for levels in two different game genres: multiplayer strategy game maps and single-player roguelike dungeons. To illustrate the impact of these evaluation functions, and the similarity of impact across domains, sets of levels for each function are generated using a constrained genetic algorithm. The proposed measures can easily be extended to other game genres.

Introduction

Automatically evaluating the quality of game levels, and other types of game content, is useful for several purposes. One might want to pre-screen or classify levels generated by users, or give rapid feedback to in-house level designers. Most importantly, many procedural content generation algorithms rely on being able to evaluate the quality of the content they are generating. In particular, this goes for all algorithms of the search-based variety, which use some kind of search or optimization algorithm to find good levels or other game content (Togelius et al. 2011). The literature contains ample examples of evaluation functions that attempt to capture specific qualities of specific content types for specific games, based either on qualitative reasoning and experience of game-play, or on crowd-sourced player experience data (Pedersen, Togelius, and Yannakakis 2010). Almost none of the proposed evaluation functions can be meaningfully applied to content outside their own narrow domain. A function that evaluates the playability or interestingness of levels for *Super Mario Bros* (Nintendo 1985) will not work on maps for *Starcraft* (Blizzard 1998) or quests for *Skyrim* (Bethesda 2011). In order to be able to produce general procedural content generation methods that can be plugged into unseen games with minimal adjustments, evaluation functions with some degree of generality need to be devised. This paper contends that the best way to achieve this is to abstract away from game-specific features towards more high-level concepts, while retaining applicability to a particular game. Fortunately, game design literature contains a number of qualitative concepts which we can build on to construct relevant quantitative measures.

This paper introduces a method for evaluating game levels consisting of six formulas evaluating area control, exploration and balance. These formulas, inspired by game design patterns, are quite generic and can be applied to a broad range of levels for different game genres; in this paper, this is demonstrated with maps for multiplayer strategy games and with dungeons for single-player roguelike games. The methods suggested for evaluating game levels sacrifice accuracy for speed, allowing for realtime feedback to the designer while they edit a level. Moreover, this paper demonstrates that using these evaluations or a combination thereof as a measure of quality for a content generator can result in high-quality game levels which satisfy one or more of the followed design patterns.

Related Work

The lack of a common language to describe games and their components has given rise to the notion of *game design patterns*. Introduced by Björk and Holopainen (2004), game design patterns are touted as "a tool for understanding and creating games", with each pattern describing "a part of the interaction possible in a game". Although patterns described by Björk and Holopainen focus on gameplay, they include patterns of game levels (e.g. Spawn Points, Safe Havens) and identify gameplay patterns pertaining to spatial navigation, such as Exploration. More domain-specific patterns have emerged for describing quests and levels in role-playing games (Smith et al. 2011) as well as levels for first-person shooters (Hullett and Whitehead 2010). According to the current literature, these design patterns facilitate the study of existing games, the discussion between designers as well as for teaching game design or level design. This paper contends that such patterns, or a sufficiently descriptive subset thereof, can be used for evaluating games and levels, providing tangible feedback to human designers or for driving the automated generation of content.

The automated generation of content has a long history in the game industry; the first ventures included the generation of dungeons in *Rogue* (Toy and Wichman 1980) and of the gameworld in *Elite* (Acornsoft 1984). Perhaps due to the success of these early titles, world or level generation has been a popular domain for procedural generation in titles such as *Torchlight* (Runic 2009) and *Civilization IV* (Firaxis 2005). Recent academic interest in procedural game content generation has also been directed towards the creation of game levels for existing games such as Starcraft (Togelius et al. 2010) or TORCS (Cardamone, Loiacono, and Lanzi 2011). Many of these applications use search-based processes (Togelius et al. 2011) such as genetic algorithms, which require an evaluation function in order to select the most appropriate content for reproduction; these evaluations can be provided directly or indirectly by humans (Hastings, Guha, and Stanley 2009; Cardamone et al. 2011), or can be formulated mathematically according to predicted entertainment (Pedersen, Togelius, and Yannakakis 2010) or concepts popularized by successful games such as chokepoints (Togelius et al. 2010). Sorenson and Pasquier (2010) propose a generic framework for level generation using a Feasible-Infeasible two-population Genetic Algorithm, and show its potential in platformer game levels and 2D adventure games. Their work shares the goal of a domain-independent framework and the method of constrained evolutionary optimization with this paper; however, the presented encoding of levels as sketches allows for a faster optimization while the presented evaluation methods are more extensive, do not require a single path from a start to a goal position, and use the notion of challenge only implicitly when choosing evaluations.

Methodology

This paper presents quantifiable metrics which can be used to evaluate game levels. While applicable to any type of game level, these metrics are used to evaluate *map sketches*, i.e. coarse abstractions of game levels. Such simple sketches are easy to design, fast to evaluate, and can be optimized automatically via a genetic algorithm. This section presents the concept of map sketches, the formulas used to evaluate them, and the evolutionary algorithm used to optimize them.

Map Sketches

The notion of a map sketch stems from the need of abstracting complex game levels into their basic building blocks. Map sketches were introduced by Liapis, Yannakakis, and Togelius (2013a), and are in many ways the game level equivalent to *procedural caricatures* (Smith and Mateas 2011) or *abstract game mechanics* (Nelson and Mateas 2007), which isolate the essential mechanics of a game from thematic context and details. The abstract visualization and compact size allows both a human designer and an algorithm (an *artificial designer*) to process and edit map sketches with less effort. A map sketch consists of a grid layout with a small number of tiles; the basic building blocks of a sketch are *empty* tiles and *impassable* tiles, which allow and block movement respectively, as well as *special tiles* which are domain-dependent (e.g. bases, traps, spawn points, checkpoints). Special tiles can be further divided into different sets of tile types, and are usually the definitive factors of the map's gameplay. In this paper, special tiles are considered to allow movement. An additional level of domain-dependent interpretation, which can be either stochastic or deterministic, allows the abstract map sketches to be converted into fully detailed maps for use in different game genres.

Evaluating Level Patterns

In order to inform the formulas which can evaluate game levels in a domain-independent (or at least domain-minimal) fashion, the closest analogue is the study of Björk and Holopainen (2004) in general game design patterns. Although this study includes hundreds of patterns, some of the more high-level patterns which translate well to level design are those of *symmetry*, *area control* and *exploration*. Symmetry is "a common feature in games to ensure that players have equal opportunities" and instantiates patterns such as player balance or team balance[1]. Area control "can give access to otherwise unavailable actions and can make the use of actions and tactics easier" and usually incorporates control over the game's resources. Finally, exploration is the "goal of learning the layout of the game world, or locating specific parts or objects in it" and assumes that some information about the level is initially imperfect or uncertain.

In order to be meaningful, area control and exploration require a set of two or more tiles identified as *reference tiles* (S_N); reference tiles usually have a special purpose in the game, such as player bases. For control measures, each reference tile can "own" a number of nearby tiles, if it is closer to these tiles than all other reference tiles. The safety of any tile t to a reference tile i is measured by eq. (1); the closest reference tile i has a positive safety value, while remaining reference tiles have a safety value of zero. For exploration measures, it is assumed that if a large part of the map must be covered in order to discover one reference tile when starting from another reference tile, then the exploration value is high. This map coverage is simulated by a flood fill algorithm, and the exploration required from a reference tile i to all other reference tiles is shown in eq. (2).

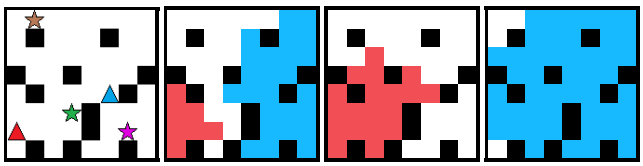
$$s_{t,i}(S_N) = \min_{\substack{1\leq j\leq N \\ j\neq i}} \left\{ max\left\{0, \frac{d_{t,j} - d_{t,i}}{d_{t,j} + d_{t,i}}\right\}\right\} \quad (1)$$

$$E_i(S_N) = \frac{1}{N-1} \sum_{\substack{j=1 \\ j\neq i}}^{N} \frac{E_{i\rightarrow j}}{P} \quad (2)$$

where N is the number of elements in set S_N; $d_{t,i}$ is the distance from tile t to element i; P is the number of passable tiles on the map; and $E_{i\rightarrow j}$ is the map coverage when a four-direction flood fill is applied starting from element i and stopping once element j has been found (see Fig. 1).

According to Björk and Holopainen (2004), control encompasses both control of areas and control of strategic resources: the former includes all passable tiles and the latter

[1]To avoid confusion between Björk and Holopainen's gameplay symmetry and visual symmetry, this paper uses the term *balance* instead.



(a) (b) (c) (d)

Figure 1: Sample metrics for the map sketch in Fig. 1a with S_M (stars), S_N (triangles) and impassable tiles (black). Measuring safety of M tiles with regards to N tiles, the purple star is much closer to the blue triangle than to the red triangle, and thus has a high safety value. The remaining stars have small safety values since they are equally close (green star) or equally faraway (brown star) from both triangles. Measuring the area control of N tiles, Fig. 1b displays safe tiles for the red triangle (A_1 in red) and for the blue triangle (A_2 in blue). Measuring the exploration of N tiles, Fig. 1d displays exploration from the red triangle to the blue triangle ($E_{1\to2}$) and Fig. 1d displays exploration from the red triangle to the blue triangle ($E_{2\to1}$).

includes only tiles which provide bonuses or serve a particular function (usually special tiles). While both evaluations of control require the definition of reference tiles, the safety of strategic resources also requires the definition of tiles constituting strategic resources (named *target tiles* or S_M). The mathematical formulation of strategic resource control (f_s), area control (f_a) and exploration (f_e) are shown below:

$$f_s(S_N, S_M) = \frac{1}{M}\sum_{k=1}^{M}\max_{1\le i\le N}\{s_{k,i}\}$$

$$f_a(S_N) = \frac{1}{P}\sum_{i=1}^{N} A_i$$

$$f_e(S_N) = \frac{1}{N}\sum_{i=1}^{N} E_i$$

where M and N is the number of elements in sets S_M and S_N respectively; $s_{k,i}$ is the safety metric of element k of S_M to element i of S_N, i.e. $s_{k,i}(S_N)$ from eq. (1); A_i is the map coverage of safe tiles for element i (see Fig. 1b); P is the number of passable tiles in the map and E_i is the exploration metric from eq. (2) for element i. A tile t is safe for element i if $s_{t,i} < C_s$; the constant $C_s = 0.35$ throughout this paper, as it creates a good ratio of contested areas in most maps.

The evaluations of f_s, f_a and f_e either average or aggregate among reference tiles. To ensure that reference tiles are symmetrical in terms of these evaluations, e.g. that the exploration value of one reference tile is not much larger than the others', the evaluations of *balance* are introduced. The mathematical formulation of strategic resource control balance (b_s), area control balance (b_a) and exploration balance (b_e) are shown below:

$$b_s(S_N, S_M) = 1 - \frac{1}{MN(N-1)}\sum_{k=1}^{M}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}|s_{k,i} - s_{k,j}|$$

$$b_a(S_N) = 1 - \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\frac{|A_i - A_j|}{max\{A_i, A_j\}}$$

$$b_e(S_N) = 1 - \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\frac{|E_i - E_j|}{max\{E_i, E_j\}}$$

Evolutionary Optimization

Map sketches are optimized towards the measures of quality presented above via artificial evolution. A map sketch is directly encoded in a genotype as an array of integers with an integer denoting whether the tile is impassable, empty or a specific type of special tile. The direct encoding used has the least representational bias, while the small size of map sketches does not hinder its evolutionary optimization.

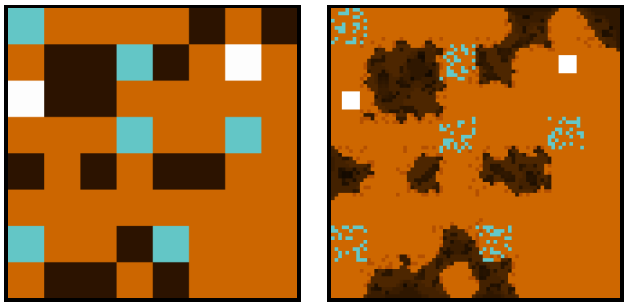
While map sketches are evaluated on area control, exploration or balance, there are some minimal criteria which must be met for the map to be playable. In this paper, a map is playable only if all special tiles (S_I) are connected with each other via a passable path. Playability is secured through constrained optimization carried out by a Feasible-Infeasible two-population Genetic Algorithm (Kimbrough et al. 2008) which maintains a population of infeasible individuals along the population of feasible individuals. All offspring which fail to satisfy the playability constraints are transferred to the infeasible population, regardless of their parents' origin. The infeasible population evolves towards minimizing the distance from feasibility; this process increases the likelihood of infeasible parents creating feasible offspring. The distance from feasibility d_{inf} for a set S_I, which normally includes all special tiles on a map, is:

$$d_{inf}(S_I) = \frac{2u_c}{I(I-1)}$$

where I the number of elements in set S_I and u_c the number of disconnected pairs of elements in set S_I.

In addition to the constraint of connectivity among all special tiles, the number of each type of special tile can be constrained to a range of values. Maps failing these additional constraints are repaired automatically by removing excess special tiles and adding missing special tiles. This repair mechanism is stochastic, with excess special tiles chosen at random and removed while a missing special tile is placed on a randomly chosen empty tile.

Each population (whether feasible or infeasible) optimizes its individuals through fitness-proportionate roulette-wheel selection, where each individual may be selected more than once. The best individual of the generation is transferred unchanged to the next generation. Recombination is performed through two-point crossover. An offspring has a 1% chance of being mutated; during mutation, a portion of the map's tiles (between 5% and 20%) are transformed. Tile transformation follows two strategies, each



(a) Sketch (b) Final Strategy Map

Figure 2: An example strategy game map sketch and the final map it creates via randomization and cellular automata; dark areas are impassable, white tiles are player bases and blue tiles are resources.

having an equal chance of being applied on each tile: a tile may be swapped with a random neighboring tile, or an empty tile may be changed to impassable and vice versa. All results presented in this paper are created from 100 generations of evolution on a total population of 100 maps.

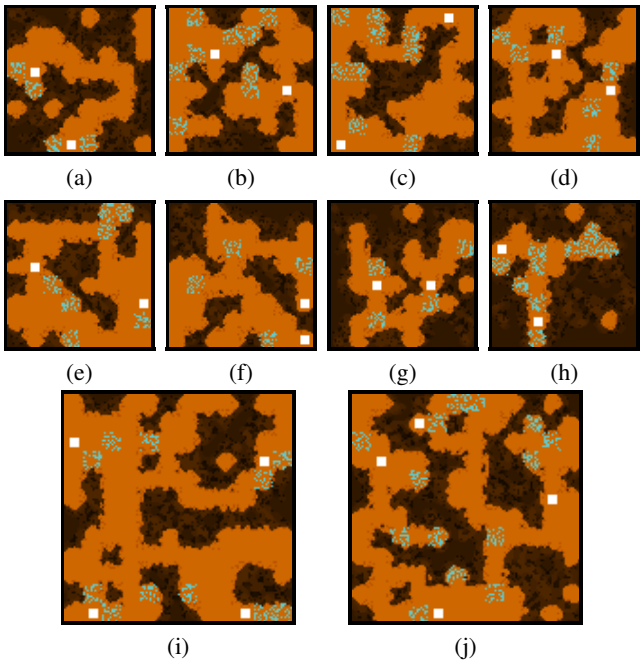
Case Study: Strategy Game Levels

While strategy games encompass a broad range of playstyles, their levels' most universal features consist of resources (R) and players' starting locations (bases or B): in the presented abstraction of strategy games only one resource type is considered (see Fig. 2). Following typical strategic gameplay, each player is expected to start at a base tile (chosen at random) and collect resources in order to build units; such units must reach the enemy players' bases and destroy them.

Evaluations & Results

Some of the most competitive multiplayer games are strategy games such as *Starcraft* (Blizzard 1998); the high degree of antagonism places player balance as the most significant property of high-quality maps. In addition, easily controlled space near each player largely determines the difficulty of the map, while readily accessible resources allow players to build up their forces before attacking their enemies. When resources are predominantly in contested areas, players must rush to secure them with the few forces they can initially muster, creating a faster and more aggressive gameplay. Finally, the location of the players' bases also affects the pace and challenge of the map since enemy bases which are difficult to find can afford the enemy player time to build up their forces without being harassed.

Translating these desirable properties of strategy game maps to evaluation functions is straightforward. Area control around bases amounts to $f_a(S_B)$, with its corresponding evaluation of balance $b_a(S_B)$. Easy access to resources from bases amounts to $f_s(S_B, S_R)$, with its corresponding evaluation of balance $b_s(S_B, S_R)$. Discovery of enemy bases is measured by $f_e(S_B)$ and its corresponding evaluation of balance $b_e(S_B)$. The evaluation functions presented above



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j)

Figure 3: Strategy game maps optimized according to different evaluation schemes. Results displayed are the best individuals among 20 runs, after 100 generations of evolution on a population of 100 individuals.

were deemed highly useful by expert game developers and level designers in a user study in which user-authored strategy maps were being evaluated (Liapis, Yannakakis, and Togelius 2013b). Combining all these evaluations results in high-quality maps favoring slow, defensive gameplay; alternatives to these evaluations are also included in Fig. 3.

Figure 3 displays strategy game maps evolved towards maximizing different combinations of the evaluations presented above and variations thereof. These maps constitute the best among 20 runs of artificial evolution, with the evaluation functions being maximized in each strategy game map shown below; combined evaluations (denoted with a $\cup$) are aggregated into a single fitness as the sum of all scores:

Fig. 3a: $F_s = f_s(S_B, S_R) \cup b_s(S_B, S_R)$

Fig. 3b: $F_a = f_a(S_B) \cup b_a(S_B)$

Fig. 3c: $F_e = f_e(S_B) \cup b_e(S_B)$

Fig. 3d: $F_s \cup F_a \cup F_e$

Fig. 3e: $\neg f_s(S_B, S_R) \cup b_s(S_B, S_R) \cup F_a \cup F_e$

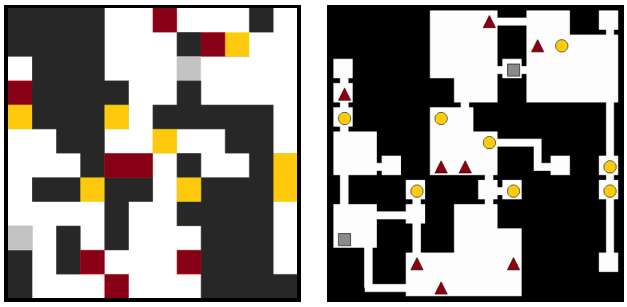
Fig. 3f: $b_s(S_B, S_R) \cup F_e \cup f_a(S_R)$

Fig. 3g: $F_s \cup f_a(S_R) \cup b_a(S_R)$

Fig. 3h: $b_s(S_B, S_R) \cup f_a(S_R \cup S_B) \cup b_a(S_R \cup S_B)$

Fig. 3i: $F_s \cup F_e$

Fig. 3j: $F_s \cup F_a \cup F_e$



(a) Sketch (b) Final Dungeon

Figure 4: An example dungeon sketch and the dungeon level it creates; black areas are impassable, red tiles (triangles) are enemies, yellow tiles (circles) are rewards and gray tiles (squares) are exits.

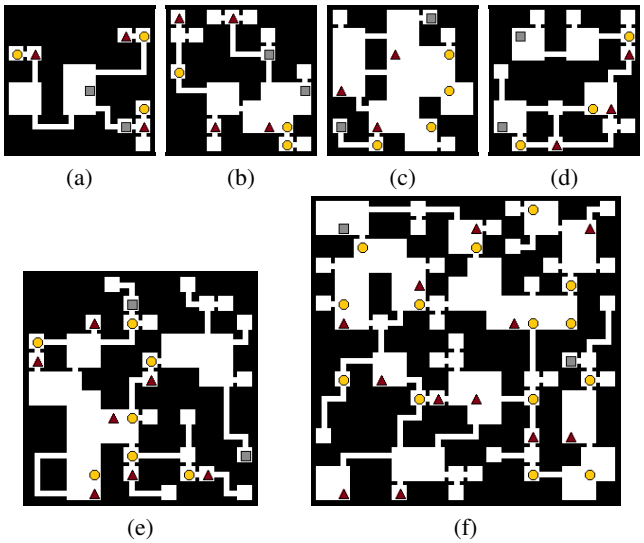
Case Study: Roguelike Dungeons

Following the success of *Rogue* (Toy and Wichman 1980), dungeons for single-player games (roguelikes) are a popular domain of procedural content generation. The special tiles in roguelike dungeons consist of dungeon exits (X), enemies (E) and treasures (T). Following the roguelike play logic, a player changes dungeon levels once they step on an exit tile; it is assumed that the player starts the current level at one of the exit tiles. Within this dungeon level, a player encounters and battles monsters at enemy tiles and collects rewards (weapons, spells, or gold) on treasure tiles.

Evaluations & Results

At its most basic, the purpose of a roguelike dungeon is for the player to travel from the entrance to the exit; placing the exit tile in a hard to reach location extends the gameplay time of the level. While traveling to the exit, the player encounters monsters and collects treasures; in order to make gameplay challenging, treasure is usually offered as a reward for vanquishing monsters. Treasures are therefore usually near a monster acting as its "guardian". Assuming that all treasures and monsters are equally valuable or powerful respectively, it would be useful for each monster to provide equal rewards in terms of treasure. A popular design paradigm for dungeons, exemplified in *Diablo* (Blizzard 1996), is the absence of monsters near the entrance of each level, allowing some breathing room for a player to prepare for the coming challenges. Finally, assuming that monsters are powerful enough to challenge a player, having the player battle two monsters simultaneously is over the intended challenge level of the dungeon. Evenly distributing the monsters throughout the dungeon, with no monster being close to another, averts the danger of a player having to fight two monsters at once.

These desirable properties of roguelike dungeons can be transformed into evaluation functions with minor reinterpretations. Since entrance and exit are both represented as exit tiles (X), a winding path from the entrance to the exit can be evaluated by $f_e(S_X)$. The placement of treasures close to monsters can be evaluated by $f_s(S_E, S_T)$, while the placement of monsters away from each other and from the entrance can be achieved by $f_a(S_X \cup S_E)$. The evaluations of balance (b_s, b_a, b_e) are not as significant in the single-player dungeons as in the competitive multiplayer strategy maps. However, having a balanced distribution of treasures near each monster as per $b_s(S_E, S_T)$ couples challenge and reward more closely, while balance in the space "allotted" to each monster and exit tile $b_a(S_X \cup S_E)$ better distributes these special tiles on the map. Finally, the balance in the exploration from the entrance to the exit and vice versa as per $b_e(S_X)$ seems the least useful of the proposed evaluations; however, since the entrance can be in any tile within S_X, preemptively ensuring that all of them are equally difficult to find guarantees a winding path from entrance to exit.



(a) (b) (c) (d) (e) (f)

Figure 5: Dungeons optimized according to different evaluation schemes. Results displayed are the best individuals among 20 runs, after 100 generations of evolution on a population of 100 individuals.

Figure 5 displays dungeons evolved towards maximizing different combinations of the evaluations presented above. These dungeons constitute the best among 20 runs, with the evaluation functions being maximized in each dungeon shown below; combined evaluations (denoted with a $\cup$) are aggregated into a single fitness as the sum of all scores:

Fig. 5a: $F_s = f_s(S_E, S_T) \cup b_s(S_E, S_T)$

Fig. 5b: $F_a = f_a(S_X \cup S_E) \cup b_a(S_X \cup S_E)$

Fig. 5c: $F_e = f_e(S_X) \cup b_e(S_X)$

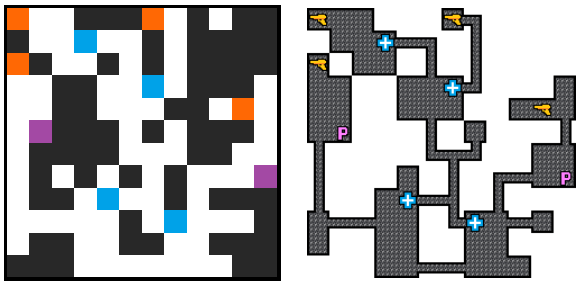
Fig. 5d: $F_s \cup F_a \cup F_e$

Fig. 5e: $F_s \cup F_a \cup F_e$

Fig. 5f: $F_s \cup F_a \cup F_e$

Extensions and Future Work

This paper demonstrated how generic and parametrizable formulas of game level quality can be used to quantitatively evaluate two disparate types of game levels: multiplayer strategy games and single-player roguelike dungeons.



(a) Sketch (b) Final FPS Map

Figure 6: A sample FPS map sketch and the final level it creates, with weapons (W) in orange, healing packs (H) in blue, and team spawn points (P) in purple. The map shown is evolved, as with other presented experiments, to optimize: $f_a(S_P \cup S_H) \cup b_a(S_P \cup S_H) \cup f_e(S_P \cup S_W) \cup b_e(S_P \cup S_W)$.

.

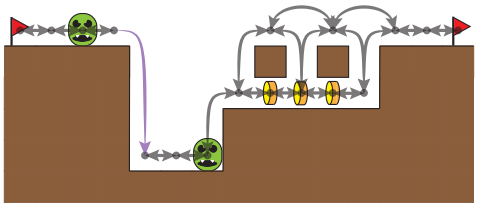


Figure 7: An example of how a platformer level can be evaluated with the current formulas. The level starts and ends at the flags, enemies are in green and rewards as yellow coins; the player is assumed to jump up to 2 blocks. Unlike the current method, the platformer needs a directed graph, i.e. the first monster (top) can reach the second (bottom) but not vice versa since the purple arrow is one-directional; the player can't jump back to the ledge. The coin area is an example of high exploration, as there are multiple paths to the finish.

Moreover, these measures of quality can be used to drive an optimization method — in this case a constrained genetic algorithm — which can automatically generate high-quality game levels. While this paper focused on two adequately different game genres to demonstrate how general the formulas are, there is a broad number of game levels which can also be evaluated by the same formulas.

Competitive multiplayer first-person shooters (FPS) such as *Team Fortress 2* (Valve 2007) can be evaluated as strategy maps (replacing resources with weapons and ammo); cooperative first-person shooters such as *Left 4 Dead* (Valve 2008) can be evaluated as dungeons (replacing monsters with bosses and rewards with ammo and healing). An example map sketch created for a competitive FPS level is shown in Fig. 6. For shooters, a useful addition to the current evaluations would include the line of sight of each tile.

The current model of evaluating game levels implicitly assumes an undirected graph for navigation; for games where backtracking is not part of the intended experience, such as platformers, a directed graph would better represent the level progression (e.g. a cliff which an avatar can jump off but not climb up again). Such a directed graph can easily be integrated with strategic resource safety and area control, while exploration can be simulated as breadth-first-search. Figure 7 shows an example platformer level and how it can be evaluated with a directed graph. Similarly to platformer levels, adventure games — especially those featuring lock and key puzzles — can be evaluated via a directed graph to ensure that the key is reachable before the locked door is opened.

It should be noted that the evaluation methods presented could be further enhanced. Additional metrics based on the affordances and constraints of other game genres (such as shooters and platformers) have already been suggested; the existing formulas could also be refined. Specifically, the safety metric is problematic when more than two reference tiles are included in the set; in such cases, unsafe tiles can be equally far from two of these elements, while remaining elements may be much farther way. Finally, the current formulas evaluate the initial topology of the level but do not take into account the game's progression. Further enhancements to these evaluations could include simulations of actual gameplay; unlike the proposed metrics, however, such simulations would likely be particularly domain-dependent.

The maps and dungeons included in this paper were generated by a constrained genetic algorithm able to create high-quality game content while ensuring that certain playability and designer-specified criteria are met. Due to the small size of the map sketches and the fast evaluation methods presented in this paper, the evolutionary process is computationally lightweight and fast compared to experiments where a complete Starcraft map was optimized (Togelius et al. 2010). As the size of the maps and the number of special tiles increase, however, the number of feasible individuals decreases and optimization is slower. The aggregation of different fitness dimensions into a weighted sum is designed to speed up evolution, but may be problematic when the combined fitness dimensions are conflicting; this behavior could be avoided via multi-objective evolutionary methods.

Conclusion

In this paper, we have introduced three generic metrics for game levels: area control, exploration and balance. We have also shown that it is straightforward to derive specific evaluation functions for particular games in different genres based on these metrics, and that using the derived evaluation functions in search-based game level generation creates maps with the intended patterns. Moreover, we have suggested a number of ways in which this work can be expanded: generalizing the proposed metrics to more game genres and establishing new metrics. Following this path, we hope to arrive at an agreed-upon collection of game level metrics that can be reused in a plug-and-play manner in procedural content generation and assisted game design tools, which would substantially benefit game development and game research.

Acknowledgments

This research was supported, in part, by the FP7 ICT project SIREN (project no: 258453) and by the FP7 ICT project C2Learn (project no: 318480).

References

Björk, S., and Holopainen, J. 2004. *Patterns in Game Design*. Charles River Media.

Cardamone, L.; Yannakakis, G. N.; Togelius, J.; and Lanzi, P. L. 2011. Evolving interesting maps for a first person shooter. In *EvoApplications (1)*, 63–72.

Cardamone, L.; Loiacono, D.; and Lanzi, P. L. 2011. Interactive evolution for the procedural generation of tracks in a high-end racing game. *Interface* 395–402.

Hastings, E. J.; Guha, R. K.; and Stanley, K. O. 2009. Automatic content generation in the galactic arms race video game. *IEEE Transactions on Computational Intelligence and AI in Games* 1(4):245–263.

Hullett, K., and Whitehead, J. 2010. Design patterns in fps levels. In *Proceedings of Foundations of Digital Games*.

Kimbrough, S. O.; Koehler, G. J.; Lu, M.; and Wood, D. H. 2008. On a feasible-infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. *European Journal of Operational Research* 190(2):310–327.

Liapis, A.; Yannakakis, G.; and Togelius, J. 2013a. Generating map sketches for strategy games. In *Proceedings of Applications of Evolutionary Computation*.

Liapis, A.; Yannakakis, G.; and Togelius, J. 2013b. Sentient sketchbook: Computer-aided game level authoring. In *Proceedings of ACM Conference on Foundations of Digital Games*.

Nelson, M., and Mateas, M. 2007. Towards automated game design. In *Procedings of the 10th Congress of the Italian Association for Artificial Intelligence*.

Pedersen, C.; Togelius, J.; and Yannakakis, G. N. 2010. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games* 2(1):54–67.

Smith, A. M., and Mateas, M. 2011. Computational caricatures: Probing the game design process with ai. In *Proceedings of AIIDE Workshop on Artificial Intelligence in the Game Design Process*.

Smith, G.; Anderson, R.; Kopleck, B.; Lindblad, Z.; Scott, L.; Wardell, A.; Whitehead, J.; and Mateas, M. 2011. Situating quests: Design patterns for quest and level design in role-playing games. In *Proceedings of the 4th International Conference on Interactive Digital Storytelling*.

Sorenson, N., and Pasquier, P. 2010. Towards a generic framework for automated video game level creation. In *Proceedings of the 2010 international conference on Applications of Evolutionary Computation - Volume Part I*, 131–140.

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

Togelius, J.; Yannakakis, G.; Stanley, K.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* (99).