Mapping Chess Aesthetics onto Procedurally Generated Chess-like Games

Jakub Kowalski[1*], Antonios Liapis[2], and Łukasz Żarczyński[3]

[1] Institute of Computer Science, University of Wrocław, `jko@cs.uni.wroc.pl`
[2] Institute of Digital Games, University of Malta, `antonios.liapis@um.edu.mt`
[3] Institute of Computer Science, University of Wrocław, `luk.zarczynski@gmail.com`

Abstract. Variants of chess have been generated in many forms and for several reasons, such as testbeds for artificial intelligence research in general game playing. This paper uses the visual properties of chess pieces as inspiration to generate new shapes for other chess-like games, targeting specific visual properties which allude to the pieces' in-game function. The proposed method uses similarity measures in terms of pieces' strategic role and movement in a game to identify the new pieces' closest representatives in chess. Evolution then attempts to minimize the distance from chess pieces' visual properties, resulting in new shapes which combine one or more chess pieces' visual identities. While experiments in this paper focus on two chess-like games from previous publications, the method can be used for broader generation of game visuals based on functional similarities of components to known, popular games.

Keywords: Procedural Content Generation, Chess Variants, Digital Aesthetics, Evolutionary Algorithms, Simplified Boardgames

1 Introduction

For over a decade, digital games have been the domain of choice for research in computational and artificial intelligence, culminating in several handbooks on the topic [1, 2]. The vast majority of the research output on this domain has been treating digital games as *systems*, focusing on their functional aspects. Specifically, research in artificial agents for playing the game usually focuses on their efficiency, using the game score attained as a benchmark of their success [3]. On the other hand, research in procedural content generation (PCG) often focuses on functional components of games, such as rules and levels, and assesses them based on solvability in the sense that the game rules allow an end-state to be attained [4] and a level's goal to be reached [5].

An example of functional concerns of game research can be found in the General Game Playing (GGP) domain for general-purpose agents [6, 7]. However, recent research on General Video Game AI [3] broadens this scope towards

---

level generation [8, 9] and game rules generation [10]. This paper is inspired by early GGP research on chess-like games [11, 12], approaching it using Simplified Boardgames [13]. As Pell's generator for METAGAME was able to produce game rules using randomized choices without any automatic evaluation [12], the rule generator for Simplified Boardgames uses artificial evolution combined with agent-based heuristics to ensure the strategic depth of generated games [14].

However, games are also aesthetic experiences that capture players' attention and allow for user interaction not only based on the combination of their rules or the spatial arrangement of their levels. Digital games elicit players' emotions through a combination of audio, visual and gameplay stimuli, and motivate exploration of the game's world by spreading visually stunning and unique vistas in different locations. While game art has been generated algorithmically in a number of commercial games and research projects, it is not clear how the functional components can be mapped to a specific audiovisual look and feel. As an example, the *Sonancia* system [15] generates levels and then uses the components within these levels (e.g. the presence or absence of monsters) to allocate background sounds in each room of the level. In the case of Sonancia, the mapping is made on design assumptions that the presence of monsters leads to a more tense experience; however, it is possible that such a mapping between different elements of games can be learnt [16].

Even board games, which have a more limited physical medium, use visuals to convey important gameplay affordances, e.g. the symbols in card games such as *Uno* (Mattel, 1992) or the shape, size and color of house tokens in *Monopoly* (Parker Bros., 1935). A systems-heavy board game such as chess also relies on the shape and size of its pieces to denote their importance and function: the size of the queen and king show that they are powerful pieces that should not be placed in harm's way (compared to the smaller, simpler pawn pieces).

Procedurally generated game rules thus require assets tailored to this particular game, so that they allow easy distinction between games created by the same system, and better fit to this game's style. For chess and chess-like games, multiple human-made piece shapes already exist. These pieces can be used as an *inspiration set* towards which new game pieces can be generated based on a mapping between the visual appearance and the function of chess pieces.

In this paper, we present an evolutionary-based method of generating shapes for any chess-like game, given its rules in Simplified Boardgames language. The goal for the generator is two-fold. First, the shapes evolved for one game should look similar, so that they are easily identified as parts of a whole. Second, we would like the shapes of the pieces to correspond to their role and importance in the game. Chess pieces are used as inspiration and their mapping between visual and strategic properties is used to create the visuals of pieces in generated games. This is done by finding similarities in the functional properties of chess pieces and new pieces, and targeting the visual dimensions of the closest chess pieces for evolving the shape of the new piece. Experiments in this paper focus on evolving pieces for two procedurally generated games introduced in [14, 17].

2 Background Work

This section highlights relevant work on procedural content generation for games and provides a brief description of the Simplified Boardgames language.

2.1 Procedural Content Generation

Digital games have used algorithms to generate content since the early 1980s with games such as Rogue (Toy and Wichman, 1980) and ELITE (Acornsoft, 1984). Generating content procedurally has been primarily motivated within commercial game development to increase replayability with nigh-endless variations of games and to decrease development time and cost. The game industry has traditionally focused on generating levels such as the star systems of *Stellaris* (Paradox, 2016), the gameworld of *Minecraft* (Mojang, 2011) or the dungeons of *Diablo* (Blizzard, 1996). There has been a strong academic interest in procedural content generation (PCG) in the last decade, focused primarily in level generation [18, 19, 5]. Contrary to the carefully scripted algorithms traditionally used in commercial games, PCG research regularly uses complex artificial and computational intelligence methods such as machine learning [19], declarative programming [4] and artificial evolution [20].

While level generation has been the most popular domain for PCG in academia and in commercial games, other facets of games such as visuals, audio, and game rules have also been explored [21]. Relevant examples for this study include the evolution of rulesets for colliding objects and scoring [22] or the evolution of mechanics based on direct code modification [23]. For board games, board layouts and rules have been evolved based on a broad range of metrics in the *Ludi* system [24]. In our work, we are using the games generated by the evolutionary system described in [14] for Simplified Boardgames. The system extends and formalizes the idea of Relative Algorithm Performance Profiles (RAPP) [25] and produces fully symmetrical games with one royal piece, and an initial row of pawn-like pieces. The evaluation function uses a number of algorithms (player profiles) with various degrees of intelligence. To assess the strategic properties of a generated game, different AI algorithms are simulated against each other and results are compared with the results obtained on human-made chess-like games. Based on the RAPP assumption, we expect that all games that behave similarly to human-made games will also be good. Results show that playable and balanced games of good quality can be obtained in this fashion. However, such games' rules might not necessary be intuitive and easy to learn for human players. For this reason, additional human-readability measures and generated natural language descriptions of the game rules have been presented in [17].

Game visuals have often been evolved for different domains and with different purposes. Game shaders have been evolved towards a designer-specified prevalent color [26]. The color and trajectory of particle effects representing players' weapons have been evolved based on how often the weapon was fired compared to others [27, 28]. Colorful flowers have been evolved collaboratively in a Facebook game [29] based on the principles of interactive evolution [30]. In terms of

evolving shapes for use as game sprites, spaceships' outlines have been evolved towards breaking patterns found in previous evolutionary steps [31] as well as to portray specific gameplay properties visually [32]. Inspired by cognitive psychology, several fitness dimensions of shapes were defined in [33] and used to evolve symmetrical shapes of spaceships' hulls. Spaceships could be evolved based on a weighted sum of these dimensions; the weights could be adapted to a user's choice among spaceships [33] or specified by a designer to create visual styles for different alien races [34]. The visual metrics used in this paper are largely inspired by the dimensions of [33], although in this case the goal is to minimize distance with known chess shapes or combinations thereof.

2.2 Simplified Boardgames

Simplified Boardgames is the class of fairy chess-like games introduced by Björnsson [13]. The language describes turn-based, two player, zero-sum chess-like games on a rectangular board with piece movements described in regular language and independent of move history. The language can describe many of the fairy chess variants in a concise way, including games with asymmetry and position-dependent moves. The usage of finite automata for describing pieces' rules, and thus for move generation, allows fast and efficient computation of all legal moves given a board setup. However, it has some important limitations, as it cannot express actions like castling, en-passant, or promotions.

Here we follow formal specifications from [35] to provide a shortened necessary introduction. A chess-like game is played between a *black* and *white* player; the white player always moves first. During a single turn, a player has to make a move using one of their pieces, changing its position according to the specified movement rule for this piece. At any time, at most one piece can occupy a square: finishing the move on a square containing a piece (regardless of the owner) results in removing it (capturing). No piece addition is possible. After performing a move, the player gives control to the opponent.

For a given piece, the set of its legal moves is defined as the set of words described by a regular expression over an alphabet Σ containing triplets (Δx, Δy, on), where Δx and Δy are relative column/row distances, and on describes the content type of the destination square, which can be empty, occupied by an opponent piece, or occupied by an own piece.

Consider a piece and a word $w \in \Sigma^*$ that belongs to the language described by the regular expression in the movement rule for this piece. Let $w = a_1 a_2 \ldots a_k$, where each $a_i = (\Delta x_i, \Delta y_i, on_i)$, and suppose that the piece stands on a square $\langle x, y \rangle$. Then, w describes a move of the piece, which is applicable in the current board position if and only if, for every i such that $1 \leq i \leq k$, the content condition on_i is fulfilled by the content of the square $\langle x + \sum_{j=1}^{i} \Delta x_j, y + \sum_{j=1}^{i} \Delta y_j \rangle$.

The game may end in a tie, when a preset turn limit is reached. The player can win by moving a certain piece to a fixed set of squares (positional win), by capturing a fixed amount of the opponent's pieces of a certain type (capturing win), or by bringing the opponent into a state with has no legal moves. The terminal conditions may be asymmetric.

3 Methodology

The main task of our system is to read rules of an arbitrary chess-like game, and produce the shape for each piece defined in this game. We decided to ground our method on chess: the most famous boardgame in Western culture, and the game where both the rules and the shapes are commonly recognized.

This section describes our system's worflow, summarized in Figure 1. First, the strategic and visual metrics for chess are computued; then, the strategic metrics of a generated game are computed and pieces of this new game are mapped onto chess pieces based on their similarity in strategic metrics. We use that mapping to obtain target visual scores as an evolutionary objective. Two-step evolution first generates a common base shape and then individually evolves each piece starting from this base shape towards that shapes target visual scores.

To describe this full workflow, Section 3.1 first describes the strategic properties used and how we compute metrics for the most important aspects of pieces' behavior. Similarly, Section 3.2 introduces visual metrics that capture the visual style of a piece's shape. Section 3.3 describes how we compare pieces in a new game with those of chess, and how we find a mapping for their desired visual metrics. Finally, we can evolve the shape of each piece in a new game. The proposed algorithm operates in *steps*: first, a general shape is evolved based on the average of all pieces' visual metrics; then, the general shape is evolved further to closely match each piece's target visual metrics. Section 3.4 describes the genetic encoding of a piece's shape, while details of the evolutionary algorithm and alternative approaches for choosing the final shapes are described in Section 3.5.

Apart from chess, experiments in this paper use two procedurally generated games as a case study: *The Legacy of Ibis* described in [14], and the game presented in [17], which we refer to as *The Weather Chess*.

3.1 Strategic metrics

We identify several *strategic metrics* to describe a piece in terms of its role in the strategic gameplay of chess: its importance, its movement (e.g. agile, bulky), its usefulness in attack/defense, etc. This paper uses the following strategic metrics:

s_{po} The fraction of the piece occurrences in the game's initial state.

s_{ec} The fraction of piece movements that end with a capture.

s_{pw} This value is 1 if the piece can be used for a positional win, 0 otherwise.

s_{cw} This value is 1 if piece can be used for a capturing win, 0 otherwise.

s_{ba} The average ratio of the board area that can be covered by a piece from its initial position(s).

s_{mr} The number of moves required to reach the most distant square from its initial position(s).

s_{lm} The average number of legal moves for each reachable position on the board.

s_{sd} The average shift distance for one letter, i.e. $\sqrt{\Delta x^2 + \Delta y^2}$.

s_{dd} The average displacement distance for a word, i.e. $\sqrt{(\sum \Delta x)^2 + (\sum \Delta y)^2}$.
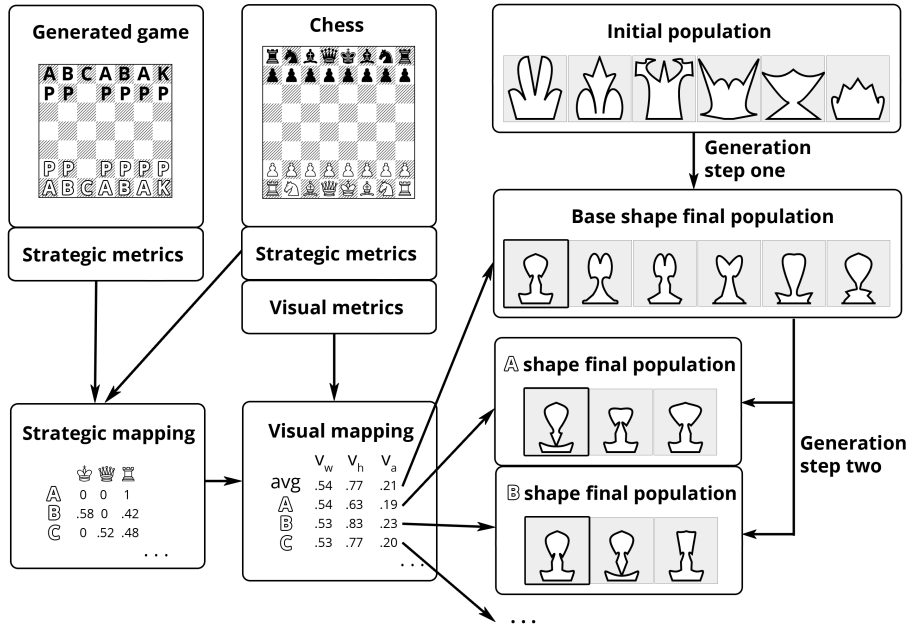
Fig. 1. Workflow of our system. After calculating strategic and visual metrics for chess and a generated game, we map pieces of this game onto chess pieces based on their strategic properties. Then, we use that mapping to obtain target visual scores as an objective. Two-step evolution first generates a common base shape and then individually evolves each piece starting from this base shape.

These measures are chosen because: s_{po} estimates the rarity of a piece; s_{ec} assesses its aggressiveness; s_{pw} and s_{cw} define a piece's importance in terms of winning the game; s_{ba}, s_{mr} and s_{lm} describe a piece's mobility; while s_{sd} and s_{dd} are indicators of the piece's movement style. Several other metrics were considered, but were omitted due to redundancies with existing metrics which would bias the similarity assessment for new games' pieces.

Table 1 lists values of strategic metrics for all considered games. When encoding chess as a Simplified Boardgame, we use a slightly modified version: there is no initial double pawn move, and —to compensate the lack of promotions— a player can win by reaching the opponent's back-rank with a pawn.

3.2 Visual metrics

We have defined four aspects that should be represented in visual metrics to capture the most important aspects of piece shape: piece size, style of its enclosed area (regardless of the size), style of its lines; and style of its angles. Inspired by [34] and [33], we use the following metrics:

v_w The ratio of the piece's width to the width of the drawing area.

Table 1. Values of strategic metrics for each piece in considered games.

metric	Chess						The Legacy of Ibis [14]					The Weather Chess [17]					
s_{po}	0.5	0.13	0.13	0.13	0.06	0.06	0.46	0.23	0.15	0.08	0.08	0.46	0.13	0.13	0.07	0.13	0.07
s_{ec}	0.66	0.5	0.5	0.5	0.5	0.5	0	0.25	0.5	0.6	0.5	0.5	0.75	0.5	0.5	0.75	0.66
s_{pw}	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
s_{cw}	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
s_{ba}	0.55	1	1	1	1	1	0.1	0.50	0.7	0.82	1	0.13	1	1	0.13	1	0.5
s_{mr}	6	5	2	2	2	7	2	4	6	4	7	6	5.5	10	4	8	7
s_{lm}	2.48	5.25	8.75	14	22.7	6.56	1.16	3.92	2.49	3.95	3.55	1.75	4.63	2.84	2.5	4.81	3.06
s_{sd}	1.28	2.24	1.41	1	1.21	1.21	2.36	1.31	1.28	1.74	1.72	1	1.91	1.83	1.33	1.31	1.41
s_{dd}	1.28	2.24	5.66	4	4.8	1.21	2.36	2.27	1.28	2.31	1.72	1	1.91	1.83	1.33	1.31	1.41

v_h The ratio of the piece's height to the height of the drawing area.

v_a The ratio of the piece's area to the drawing area.

v_{ta} The ratio of the piece's top $1/3$ area to the piece's area.

v_{ma} The ratio of the piece's middle $1/3$ area (on the x axis) to the piece's area.

v_s The intersection to union ratio between piece's left half and the (mirrored) right half; symmetrical shapes score 1 in this metric.

v_{my} The ratio of the piece's middle half on the y axis to the piece's area.

v_{tr} The area intersecting the piece and an upward-pointing triangle shape, over the piece's area.

v_p The ratio of the piece's perimeter to double its bounding box perimeter.

v_{sl} The ratio of the length of straight lines to the piece's perimeter.

v_{sa} The ratio of sharp angles ($0°$–$60°$) to all angles between lines.

v_{ga} The ratio of gentle angles ($120°$–$180°$) to all angles between lines.

Values in all metrics are bound to $[0, 1]$. Table 2 lists the visual metrics for games in this paper. Values for chess were extracted from the set of shapes of Fig. 2. Values for the other games were computed using the method described in Section 3.3.

3.3   Mapping from general games to chess

As noted above, chess pieces have certain visual properties that denote their function in game. New games, on the other hand, can be evaluated only in terms of their function; finding what visuals these new pieces should have is
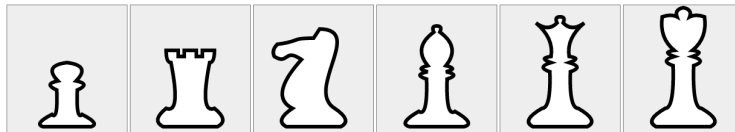

Fig. 2. The set of chess piece shapes used as a base for further computations.

Table 2. Values of visual metrics for each piece in considered games. Values for chess are computed from the shapes in Figure 2. Values for the generated games are obtained by our algorithm.

metric	Chess						The Legacy of Ibis [14]					The Weather Chess [17]					
v_w	0.45	0.59	0.52	0.54	0.52	0.52	0.55	0.54	0.53	0.53	0.55	0.54	0.53	0.53	0.53	0.52	0.45
v_h	0.52	0.8	0.82	0.63	0.90	0.98	0.81	0.63	0.83	0.77	0.83	0.63	0.76	0.75	0.78	0.98	0.53
v_a	0.12	0.29	0.18	0.19	0.20	0.26	0.23	0.19	0.23	0.20	0.24	0.19	0.20	0.19	0.20	0.26	0.12
v_{ta}	0.29	0.31	0.26	0.46	0.24	0.35	0.28	0.46	0.40	0.34	0.34	0.46	0.35	0.34	0.33	0.35	0.29
v_{ma}	0.28	0.31	0.29	0.18	0.29	0.25	0.30	0.18	0.22	0.24	0.26	0.18	0.23	0.25	0.24	0.25	0.29
v_s	1	0.77	1	1	1	1	0.91	1	1	1	0.94	1	1	1	1	1	1
v_{my}	0.84	0.73	0.88	0.73	0.85	0.83	0.82	0.73	0.79	0.79	0.78	0.73	0.79	0.82	0.80	0.83	0.85
v_{tr}	0.73	0.58	0.79	0.50	0.81	0.69	0.70	0.50	0.61	0.66	0.64	0.50	0.65	0.68	0.68	0.69	0.73
v_p	0.41	0.43	0.43	0.45	0.49	0.47	0.44	0.45	0.46	0.47	0.46	0.45	0.47	0.44	0.47	0.47	0.41
v_{sl}	0	0.04	0	0.29	0	0	0.02	0.29	0.12	0.14	0.08	0.29	0.15	0.11	0.12	0	0
v_{sa}	0.22	0	0.15	0	0.27	0.40	0.09	0	0.23	0.14	0.16	0	0.13	0.09	0.15	0.40	0.22
v_{ga}	0.33	0.6	0.38	0.29	0.33	0.33	0.50	0.29	0.32	0.31	0.42	0.29	0.31	0.35	0.32	0.33	0.33

not straightforward. In this paper, we compare the functional properties of chess pieces and new generated pieces, and approximate the visuals of these new pieces based on the visuals of their closest chess pieces. Even with this basic premise, a number of questions arise: (a) how is the functional similarity of pieces in different games with different rulesets assessed? (b) can a generated piece be similar to more than one chess pieces, and how is that handled? The following paragraphs elaborate on the decisions taken to address these questions.

A broad range of functional properties have been described both qualitatively and in terms of heuristics for calculating them in Section 3.1. The most straightforward way of assessing how closely a new generated piece matches another is through an Euclidean distance treating the nine strategic features as a 9-dimensional vector. Comparing pieces in this fashion ignores the fact that the pieces originate from different games, and are thus sensitive to the value ranges of the other pieces in the same game. For example, a game in which most pieces move to one or two adjacent spaces would classify all of its pieces as chess pawns or kings, ignoring the fact that some pieces may be more mobile (e.g. moving two spaces) than others (e.g. that move one space). Moreover, from a practical perspective not all strategic metrics are in the same value range (e.g. s_{mr} and s_{sd}) nor do their values deviate from one piece to the other in the same way. To address these concerns, pieces from both games are first standardized to their z-scores, which processes a raw value x as $z = (x - \bar{x})/\sigma_x$ where $\bar{x}$ is the mean value of all pieces in the same game and σ_x is the standard deviation of those values. This standardization ensures all metrics are in the same value range and clearly denotes outliers. In this way, more mobile pieces of one game will have similar scores to more mobile pieces in the other game (compared to the average mobility of each game) and thus would be mapped closer together.
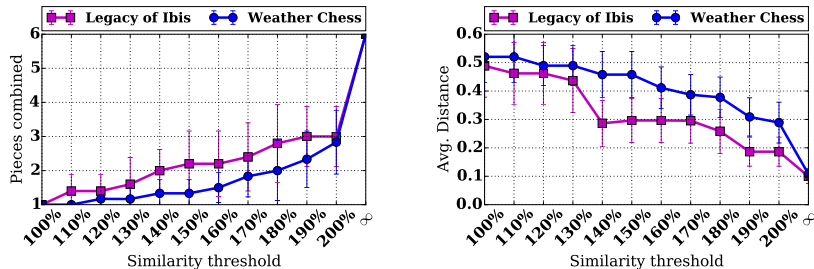
Fig. 3. Sensitivity of the chess piece mapping to different thresholds. The threshold is the highest distance ratio of the closest chess piece that is considered. Results are on average number of shapes combined and average pairwise visual distance; error bars show the 95% confidence interval for 5 shapes and 10 distances in Legacy of Ibis, and 6 shapes and 15 distances in Weather Chess. Infinity uses all chess pieces.

Once the distance in terms of the nine standardized strategic metrics between all pieces in the new game and all chess pieces is calculated, the closest chess piece to each new piece is identified and its distance is compared to the distance between other chess pieces and the new piece. Choosing the closest chess piece and its visuals as the target of evolution is an option, but in practice two pieces of a new game were often mapped to the same chess piece. Using all chess pieces' visuals based on a weighted sum where weights are proportional to the chess piece's similarity to the new piece is another option. In practice, however, the resulting target visual metrics were very similar for all pieces in the new game.An intermediate solution was devised instead, considering only chess pieces which are relatively close to the closest chess piece. This is done by dividing the distance between the new piece and a chess piece with the distance to the closest chess piece: the resulting metric W_i has a value range of $(0, 1]$: values close to 1 show that a chess piece is almost as similar to the new piece as the closest chess piece. Choosing an appropriate threshold, above which a piece is considered similar enough, is an ad-hoc design decision which can affect the results. Based on a preliminary sensitivity analysis (see Fig. 3), we chose to consider only pieces with a strategic distance at most 160% that of the closest chess piece ($W_i \geq 0.625$). Each visual metric for a new piece amounts to a weighted sum of visual metrics of all considered chess pieces, where the weight of chess piece i is a normalized version of W_i so that all normalized weights sum up to 1. These normalized weights for the two tested games are shown in Table 3.

3.4 Representation

Generated shapes are encoded in Scalable Vector Graphics (SVG) format, which makes them directly useful for real applications. The genotype of every piece is a series of lines (encoded in an array) that may contain straight lines, quadratic Bézier curves and cubic Bézier curves. The starting point of every line (except

Table 3. Weight of chess pieces in terms of strategic similarity with pieces for the Legacy of Ibis game (left) and The Weather Chess (right).

	♙	♘	♗	♖	♕	♔
🕊	0.42	0.58				
🦅			1.00			
↪			0.42		0.58	
🦩				0.48	0.52	
🐦	0.29			0.24	0.20	0.28

	♙	♘	♗	♖	♕	♔
☁				1.00		
🌧				0.52	0.48	
🌦			0.61	0.39		
🌦				0.42	0.58	
☁						1.00
☀	1.00					

the first) is the end point of its predecessor, so it is omitted. The drawing area has been set to 200×200 units. The first point of every shape should be placed on the lowest horizontal line. The first point is automatically connected to the last point, closing the shape and making the piece's basis. The line array is interpreted differently for symmetric and asymmetric pieces. When a piece is flagged as symmetric, we assume the array represents only the right half of the piece, while the left half is mirrored. For asymmetric pieces, its genotype explicitly contains all parts of the shape.

3.5 Evolution and its variants

A standard evolutionary algorithm scheme is used for evolving shapes. Tournament selection chooses $n/2$ pairs of parents to crossover from the current generation containing n individuals. Crossover produces two children, and each of them can be additionally mutated. This may produce inconsistent individuals, which are removed from the population. The next generation is created by choosing the best n shapes from the joint set of parents and children.

To encourage novelty, during selection any shapes similar to already selected shapes are omitted. Similarity is computed as a fraction of the area of intersection and the area of union of both pieces. Crossover cuts the array of parents' lines in half and joins both halves to create new shapes. If only one of the parents is symmetric, its representation is temporarily changed to asymmetric.

The mutation operator is more complex: one possibility is that the piece is converted to asymmetric. The other possibility is that one of its lines is chosen at random and one of two operations is applied: (a) the line type is modified: straight to arc, arc to double arc, double arc to straight, or a straight line to be split in half into two straight lines; (b) the line's points are transformed by a random vector (including control points of Bézier curves which can affect the shape without modifying the start and end points).

This paper proposes a two-step evolutionary algorithm: the first step evolves deeply and widely a population towards the average of all pieces' target values. All pieces' target values are averaged together on the same dimension; evolution targets similarity with those target (average) values as its fitness based on

the distance in a 12-dimensional vector for the 12 metrics of Section 3.2. Once evolution is complete, the fittest individual is chosen and evolution is carried out in a second step for each piece individually. The initial population for these runs uses copies of the fittest individual of the first evolutionary step. Each run targets similarity with the target values of that shape specifically as its fitness, and evolves for a few generations. The fittest individual of each run in the second step is chosen for that piece. To prevent shapes from being too similar, an additional similarity check is made when choosing the best shape for each piece.

The second step can be radically simplified by choosing shapes diectly from the final population of the first step described above. This makes the process slightly faster, yet reduces the likelihood that chosen pieces will be visually consistent with the desired ones. Additionally, in this case a similarity check is required to prevent the same shape from being chosen for different pieces.

The last variant evolves each piece independently from the beginning, using the similarity with that piece's target values (see Table 2) as its fitness. This method is the slowest and is likely to produce visually inconsistent shapes, but is used for comparison with the other two approaches.

4 Results

To test how the three variants of our algorithm compare, we use each of them to generate shapes for chess, The Legacy of Ibis, and The Weather Chess. In the first case, we aim to recreate the exact chess aesthetic. For the remaining two games, we create novel visuals based on our mapping method in Section 3.3.

In an initial exploration of the types of shapes produced by each method, we used randomized parameter sets. The first evolutionary step was run in two uniformly distributed settings: deep (200–400 generations, population size 40–100) or wide (50–200 generations, population size 200–500). For two-step evolution, the second step ran for 1–10 generations. We have used three ways to initialize the population: using copies of a triangle shape; using copies of chess pawns; using random shapes. The best shape representatives of these runs were chosen manually and presented in Fig. 4, covering most of the tested combinations of variants and initial shapes for all games.

Based on the findings of the initial exploratory phase, most parameters of the algorithms were chosen and finalized. To perform statistical comparisons between approaches, games and initial shapes, however, 20 independent evolutionary runs were performed on the same combination; their results are averaged across methods and shapes or games. All runs evolve a population of 100 individuals over a total of 100 generations; for two-step evolution, these generations were split between first and second step in several configurations, with 10, 20, or 50 generations dedicated to the second step (the first step respectively evolves for 90, 80 and 50 generations). Fig. 5 shows the average values of the final chosen pieces and the 95% confidence interval. Two relevant performance metrics are tested: (a) as the objective of evolution, the Euclidean visual distance from the target values of each piece (in Table 2); (b) as an indication of the visual

Exploratory results for Chess

init	avg.	♟	♞	♝	♜	♛	♚	avg.	♟	♞	♝	♜	♛	♚	♟	♞	♝	♜	♛	♚
	Two-step evolution variant							One-step evolution and pick							Independent evolution variant					
△	1.06	1.07	1.08	1.09	1.03	1.06	1.11	1.06	1.07	1.05	1.06	1.04	1.07	1.06	0.54	0.06	0.45	0.55	0.52	0.59
♟	0.43	0.60	0.24	0.50	0.48	0.54	0.63	0.39	0.52	0.35	0.44	0.46	0.54	0.42	0.60	0.03	0.50	0.60	0.56	0.66
RANDOM	0.46	0.55	0.32	0.48	0.61	0.52	0.62	0.39	0.46	0.37	0.48	0.41	0.47	0.50	0.61	0.24	0.47	0.58	0.56	0.67

Exploratory results for Legacy of Ibis

init	avg.						avg.										
	Two-step evolution variant						One-step evolution and pick						Independent evolution variant				
△	0.14	0.16	0.41	0.18	0.18	0.11	0.12	0.15	0.19	0.15	0.17	0.12	0.16	0.41	0.16	0.20	0.09
♟	0.12	0.23	0.19	0.15	0.22	0.12	0.16	0.12	0.23	0.14	0.17	0.10	0.19	0.41	0.15	0.17	0.07
RANDOM	0.12	0.18	0.33	0.15	0.14	0.14	0.30	0.24	0.24	0.19	0.18	0.20	0.16	0.46	0.16	0.17	0.11

Exploratory results for The Weather Chess

init	avg.							avg.												
	Two-step evolution variant							One-step evolution and pick							Independent evolution variant					
△	0.31	0.45	0.39	0.30	0.0.33	0.49	0.42	1.03	1.05	1.05	1.05	1.03	1.04	1.03	0.53	0.31	0.30	0.31	0.52	0.09
♟	0.29	0.47	0.33	0.38	0.30	0.44	0.20	0.30	0.31	0.30	0.33	0.28	0.33	0.26	0.52	0.35	0.34	0.34	0.53	0.04
RANDOM	0.29	0.44	0.34	0.26	0.32	0.36	0.23	0.29	0.38	0.34	0.31	0.30	0.32	0.23	0.53	0.32	0.36	0.33	0.56	0.09

Fig. 4. Example sets of evolved shapes for each game, initial shape (triangle, pawn, random) and algorithm variant. Fitness values are presented below the shapes.

consistency of shapes of the same game, the average Euclidean visual distance among all pairs in the same evolved set. With 6 pieces for chess and Weather chess, and 5 pieces for Legacy of Ibis, each method had a total of 340 pieces (in 20 runs) for the same initial shape.

It is clear from Fig. 5 that each method behaves as expected: independent evolution more closely matches the desired visuals of each piece but the resulting pieces are very different from each other, while the opposite is true for one-step evolution where shapes are picked from a final population of average shapes. Two-step evolution finds itself in-between the two extremes, with more generations dedicated to the second step leading to shapes closer to the desired visuals. Even with a few generations for the second step, however, the divergence among pieces in the same set is much higher (often double) than that of one-step evolution.
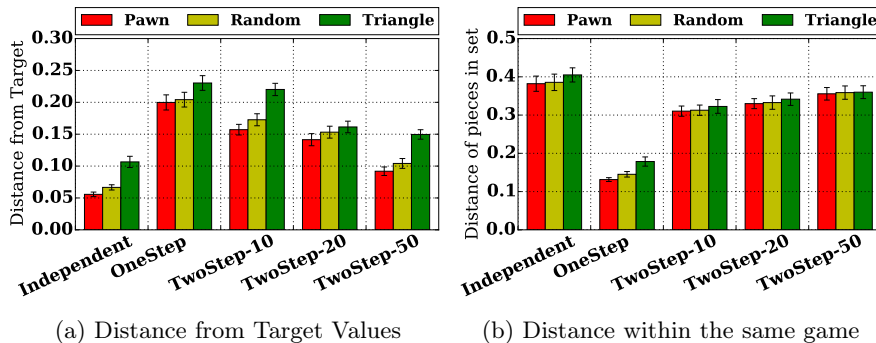
(a) Distance from Target Values   (b) Distance within the same game

Fig. 5. Sensitivity of the chess piece mapping to different thresholds. The threshold is the highest distance ratio of the closest chess piece that is considered. Results are collected from 20 runs per game, initial shape, and method; error bars show the 95% confidence interval for 340 individuals for Fig. 5a and 60 runs for Fig. 5b.

In terms of initial shapes, clearly pawns are better starting points since they are both more complex than triangle shapes and closer to the visual features targeted by any chess variant than random shapes. Comparing across games, it is surprising that chess shapes are more challenging to evolve for (in terms of final distance to target values) than those of the two new games despite the fact that its target values are derived directly from chess shapes.

5 Discussion

This paper describes a method for evolving shapes of chess-like pieces that belong to a number of previously generated games. Our approach maps the visual aesthetics of chess pieces to their in-game function. This allows us to estimate the intended visual aesthetics of new games' pieces based on their role for this particular game. A number of processing steps allow the system to identify pieces that, while in raw values may e.g. move slower than some chess piece, they serve a similar role in a game where generally all pieces move slower, for example. Generated shapes are encoded in SVG, a popular vector graphics format, which makes them easily scalable and useful for real applications.

The defined goal in this paper is to produce sets of shapes: a) which can be distinguished as belonging to the same game, and b) where shapes of individual pieces intuitively correspond to their role and importance in the game.

To satisfy the first goal, we use a two-step variant of the evolutionary algorithm, as well as a variant one-step algorithm which picks pieces from a general population. However, we observed that generated piece sets may contain too similar pieces. Although our similarity tests remove the most obvious cases, many cases may be sufficiently different in terms of area coverage but are still easily mistaken by human players due to common visual details. The third algorithm

variant runs evolution independently for each piece and usually produces pieces that are easier to distinguish and remember. Combined together, however, the pieces do not look like parts of the same game and can be too easily swapped with any other piece generated for other games. The balance between insufficient variability and too much variability is subjective, and results of the three algorithms often explore the limits in one or the other end of the spectrum.

The degree to which the second goal is fulfilled is more disputable. Although the chess-based grounding achieves its stated aim in some cases (e.g. often pieces needed to win the game are bigger), it is hard to unambiguously decide what style should characterize a piece given its rules. In the presented approach, the mappings to chess pieces based on strategic similarities gives results that can be justified. However, the process of evolution itself usually blurs all differences that are not close to extremes. As with the first goal, further experimentation on different similarity thresholds (e.g. in Fig. 3) can assess the tension between having too many pieces map to the same chess piece versus combining too many chess shapes in a way that makes the results indistinguishable.

This initial attempt at evolving shapes for games with procedurally generated rules focuses on making them visually pleasing and playable. While currently not all generated sets can be used immediately by players, a decent piece set can always be chosen by a human judge. Future work will analyze and improve on the strategic and visual metrics for a better mapping and a more concrete visual identity, respectively. In terms of evolution, we intend to explore using novelty search [36] —similarly to [34]— instead of the diversity preservation mechanisms used in this paper. Finally, we plan to address the problem of identifying which pieces are visibly consistent while being distinguishable. Towards that end, a deep learning approach [31] could be useful for visual similarity assessment rather than purely pixel-by-pixel operations currently used for diversity preservation.

This work, combined with past achievements in chess-like rule generation and generated natural language descriptions of such rules [14, 17], can lead to a system that can generate, on demand, games together with their rule explanations and their full visuals. Playing games offered by such a system will be a human equivalent of the General Game Playing challenge.

6   Conclusion

This paper proposed a way of generating the shapes of pieces in unseen, generated games based on their functional similarities with pieces in known, popular games. Focusing on several generated chess-like games and the original chess, experiments explored whether generated shapes could achieve a consistent look (as pieces of the same game) while allowing players to distinguish between pieces for actual play. Results indicate that depending on whether pieces were evolved as a whole (for one game) or independently, the balance between these two goals can be skewed towards one or the other. This first generative attempt can be strengthened algorithmically in order to better achieve both goals at the same

time. Future work could also explore the use of similar techniques for other games, including other board games or even digital games.

References

1. Shaker, N., Togelius, J., Nelson, M.J.: Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer (2016)
2. Yannakakis, G.N., Togelius, J.: Artificial Intelligence and Games. Springer (2018), http://gameaibook.org
3. Perez, D., Samothrakis, S., Togelius, J., Schaul, T., Lucas, S., Couëtoux, A., Lee, J., Lim, C., Thompson, T.: The 2014 General Video Game Playing Competition. IEEE Transactions on Computational Intelligence and AI in Games 8(3), 229–243 (2015)
4. Smith, A.M., Mateas, M.: Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games. In: IEEE Conference on Computational Intelligence and Games (2010)
5. Smith, G., Whitehead, J., Mateas, M.: Tanagra: Reactive planning and constraint solving for mixed-initiative level design. IEEE Transactions on Computational Intelligence and AI in Games 3(3) (2011)
6. Genesereth, M., Love, N., Pell, B.: General Game Playing: Overview of the AAAI Competition. AI Magazine 26, 62–72 (2005)
7. Genesereth, M., Björnsson, Y.: The International General Game Playing Competition. AI Magazine 34(2), 107–111 (2013)
8. Nielsen, T.S., Barros, G.A.B., Togelius, J., Nelson, M.J.: Towards generating arcade game rules with VGDL. In: IEEE Conference on Computational Intelligence and Games. pp. 185–192 (2015)
9. Khalifa, A., Perez, D., Lucas, S., Togelius, J.: General Video Game Level Generation. In: Genetic and Evolutionary Computation Conference. pp. 253–259 (2016)
10. Khalifa, A., Green, M., Perez, D., Togelius, J.: General Video Game Rule Generation. In: IEEE Conference on Computational Intelligence and Games (2017)
11. Pitrat, J.: Realization of a general game-playing program. In: IFIP Congress. pp. 1570–1574 (1968)
12. Pell, B.: METAGAME in Symmetric Chess-Like Games. In: Heuristic Programming in Artificial Intelligence: The Third Computer Olympiad. (1992)
13. Björnsson, Y.: Learning Rules of Simplified Boardgames by Observing. In: European Conference on Artificial Intelligence, FAIA, vol. 242, pp. 175–180 (2012)
14. Kowalski, J., Szykuła, M.: Evolving Chesslike Games Using Relative Algorithm Performance Profiles. In: Applications of Evolutionary Computation, LNCS, vol. 9597, pp. 574–589 (2016)
15. Lopes, P., Liapis, A., Yannakakis, G.N.: Targeting horror via level and soundscape generation. In: AAAI Artificial Intelligence for Interactive Digital Entertainment Conference (2015)
16. Karavolos, D., Liapis, A., Yannakakis, G.N.: Learning the patterns of balance in a multi-player shooter game. In: FDG workshop on Procedural Content Generation in Games (2017)
17. Kowalski, J., Żarczyński, Ł., Kisielewicz, A.: Evaluating Chess-like Games Using Generated Natural Language Descriptions. In: ACG 2017: Advances in Computer Games, LNCS, vol. 10664, pp. 127–139 (2017)

18. Liapis, A., Yannakakis, G.N., Togelius, J.: Towards a generic method of evaluating game levels. In: AAAI Artificial Intelligence for Interactive Digital Entertainment Conference (2013)
19. Summerville, A.J., Mateas, M.: Sampling hyrule: Multi-technique probabilistic level generation for action role playing games. In: AIIDE Workshop on Experimental AI in Games (2015)
20. Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games 3(3) (2011)
21. Liapis, A., Yannakakis, G.N., Togelius, J.: Computational game creativity. In: International Conference on Computational Creativity (2014)
22. Togelius, J., Schmidhuber, J.: An experiment in automatic game design. In: IEEE Symposium on Computational Intelligence and Games (2008)
23. Cook, M., Colton, S., Raad, A., Gow, J.: Mechanic miner: Reflection-driven game mechanic discovery and level design. In: Applications of Evolutionary Computation. vol. 7835, LNCS (2012)
24. Browne, C., Maire, F.: Evolutionary game design. IEEE Transactions on Computational Intelligence and AI in Games 2(1), 1–16 (2010)
25. Nielsen, T.S., Barros, G.A.B., Togelius, J., Nelson, M.J.: General Video Game Evaluation Using Relative Algorithm Performance Profiles. In: Applications of Evolutionary Computation, LNCS, vol. 9028, pp. 369–380 (2015)
26. Howlett, A., Colton, S., Browne, C.: Evolving pixel shaders for the prototype video game subversion. In: Proceedings of AISB'10 (2010)
27. Hastings, E.J., Guha, R.K., Stanley, K.O.: Automatic content generation in the galactic arms race video game. IEEE Transactions on Computational Intelligence and AI in Games 1(4) (2009)
28. Hoover, A.K., Cachia, W., Liapis, A., Yannakakis, G.N.: Audioinspace: Exploring the creative fusion of generative audio, visuals and gameplay. In: Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMusArt), vol. 9027, LNCS. Springer (2015)
29. Risi, S., Lehman, J., D'Ambrosio, D., Hall, R., Stanley, K.: Petalz: Search-based procedural content generation for the casual gamer. IEEE Transactions on Computational Intelligence in Games 8(3) (2015)
30. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. Proceedings of the IEEE (9), 1275–1296 (2001)
31. Liapis, A., Martínez, H.P., Togelius, J., Yannakakis, G.N.: Transforming exploratory creativity with DeLeNoX. In: International Conference on Computational Creativity (2013)
32. Soule, T., Robison, B.D., Heck, S., Haynes, T.E., Street, D., Wood, N.: Darwin's demons: Does evolution improve the game? In: Applications of Evolutionary Computation (2017)
33. Liapis, A., Yannakakis, G.N., Togelius, J.: Adapting models of visual aesthetics for personalized content creation. IEEE Transactions on Computational Intelligence and AI in Games 4(3), 213–228 (2012)
34. Liapis, A.: Exploring the visual styles of arcade game assets. In: Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMusArt). Springer (2016)
35. Kowalski, J., Sutowicz, J., Szykuła, M.: Simplified Boardgames (2016), arXiv:1606.02645 [cs.AI]
36. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search for novelty alone. Evolutionary Computation 19(2) (2011)