Designer Modeling for Sentient Sketchbook

Antonios Liapis
Center for Computer Games Research
IT University of Copenhagen
Copenhagen, Denmark
Email: anli@itu.dk

Georgios N. Yannakakis
Institute of Digital Games
University of Malta
Msida, Malta
Email: georgios.yannakakis@um.edu.mt

Julian Togelius
Center for Computer Games Research
IT University of Copenhagen
Copenhagen, Denmark
Email: julian@togelius.com

***Abstract*—This paper documents the challenges in creating a computer-aided level design tool which incorporates computer-generated suggestions which appeal to the human user. Several steps are suggested in order to make the suggestions more appropriate to a specific user's overall style, current focus, and end-goals. Designer style is modeled via choice-based interactive evolution which adapts the impact of different dimensions of quality based on the designer's choice of certain suggestions over others. Modeling process is carried out similarly to style, but adapting to the current focus of the designer's actions. Goals are modeled by estimating the visual patterns of the designer's final artifact and changing the parameters of the algorithm to enforce such patterns on generated suggestions.**

I. Introduction

The increasing popularity of digital games attracts individuals who do not necessarily possess programming or game design experience. Such individuals often wish to contribute to existing games with user-generated content, such as custom levels, or with more extensive modifications which significantly change the gameplay (game mods). In an attempt to motivate such involvement, game companies often include easy-to-use game editors with their titles or release their in-house scripting language to the public. Such intuitive tools mitigate the programming skill required from end-users; however, the tools still assume that the prospective creators have enough ingenuity, drive and experience with digital games to see an idea from its conception to its completion.

Sentient Sketchbook was introduced in [1] in an attempt to impart some expert knowledge regarding level design to end-users as well as to foster their creativity. Sentient Sketchbook is a computer-aided level design tool; operating on sketches rather than detailed maps, it allows quick and effortless prototyping of level concepts and provides several evaluations of game level quality via heuristics and visual aids. The tool is enhanced by genetic algorithms in order to be able to autonomously create playable game levels; the game levels generated by the software are presented in real-time, while the human user interacts with the tool, as suggestions which can replace the user's design. The addition of a proactive computational designer contributing to the design process allows human and computer not only to co-create playable levels but also to be inspired by each other's creativity [2].

The suggestions generated by the computational designer are the most sophisticated component of Sentient Sketchbook in terms of computational intelligence. In order to motivate a creative dialogue between man and machine, these suggestions should appeal to the user of the tool so that they are selected more often. A previous study with expert users interacting with Sentient Sketchbook concluded that suggestions were selected when users wanted a cheap shortcut to creating a playable level as well as when fine-tuning an almost complete level. This same study, however, identified several cases where suggestions were not deemed useful. This paper implements several concepts of designer modeling [3] in order to address the majority of these cases and create personalized, contextually aware models of the user's goals, style and process which generate suggestions tailored to the task at hand. This paper tests a number of assumptions *in vitro*, in controlled experiments with artificial agents that possess an a priori defined style as well as by applying the proposed models of process and goals on previous design sessions of Sentient Sketchbook. Following the conclusions drawn in this paper, the proposed designer models will be integrated into Sentient Sketchbook and tested with human designers.

The paper is structured as follows: Section II provides a brief survey of procedural game content generation and clarifies the concept of designer modeling. Section III presents the Sentient Sketchbook tool, while Section IV presents the three designer models of style, process and symmetry goals implemented for Sentient Sketchbook. Section V tests the model of designer style via experiments with artificial agents, while Section VI uses past sessions with human designers to demonstrate the performance of the models of process and symmetry goals. The findings are discussed in Section VII, and the paper concludes with Section VIII.

II. Related Work

The computational generation of suggestions in Sentient Sketchbook falls under the larger topic of procedural content generation. The adaptation of these suggestions according to the designer's style, goals or process falls under designer modeling. A short overview of both concepts follows.

A. Procedural Content Generation

The algorithmic generation of game content such as dungeons in *Rogue* (Toy and Wichman 1980) or the gameworld in *Civilization IV* (Firaxis 2005) has a rich history in the game industry. While relatively recent, academic interest in procedural content generation (PCG) has seen a number of innovations in the algorithms used. Among such algorithms, genetic search has been a popular choice, leading to a family of search-based procedural content generators [4]. Genetic algorithms evolve content to optimize its similarity with desirable outcomes [5], its visual appearance [6], its performance in a simulation [7], or its appeal to a specific user [8]. Personalization of generated content can be based on explicit human choices as interactive evolution [9] or learned from gameplay data [10].

B. *Designer Modeling*

Designer modeling has been suggested in [3] as a method for capturing a designer's intentions and preference and accommodating them via a computer-aided design tool. Designer modeling is therefore envisioned as an instance of user modeling applied to computer-aided design. The term can incorporate any computational model which recognizes the goals, preferences or process of a human designer. Such a designer model can be useful for personalized, responsive computer-aided design tools in the game industry or elsewhere. While similar to player modeling [11], the designer model should be considered distinct as it needs to incorporate the designer's intentions to satisfy the player, and can be seen as second-order player modeling.

A successful designer model should recognize the preferences, process and goal of a designer interacting with the tool, although accomplishing each of these aspects may hinge on different algorithmic processes. Modeling the preference or overall style of a designer falls under the category of preference learning [12], and requires extensive information on a designer's choices, rankings or ratings among alternatives. Such adaptive models of taste have been trained based on a user's choice of one artifact over others [6] or from a user's rankings of artifacts in order of preference [13]. On the other hand, modeling the goals or intentions of a designer may be achievable via goal recognition [14], e.g. by suggesting possible next steps via a probabilistic model of cognitive associations based on past interactions [15]. However, such a method will likely only present previously seen (or created) concepts and thus stifle the designer's creativity. Finally, modeling the designer's process can be seen as a short-term plan recognition problem [16]. Although past interaction data can inform the model of trends in the design process, a process can be highly situational: a designer may focus on fine-tuning different properties of their creations at different stages of the process, without necessarily looking at the bigger picture until the very end. A model of process could therefore be more accurate if it learned solely from the designer's current actions rather than from a large pool of past interactions.

III. Sentient Sketchbook

Sentient Sketchbook is a computer-aided level design tool which incorporates computer-generated suggestions as alternatives to the user's creations. The tool allows both a human and a computational designer to create *map sketches*, which are high-level, low-resolution abstractions of game levels. Map sketches can be transformed into several types of game levels [17], although this paper will focus on strategy game levels. Sketches for strategy game levels consist of tiles representing bases (where players can build units), resources (collected by players as in-game currency for building units), empty tiles and impassable tiles (which allow and block unit movement respectively). The low-resolution sketches can be transformed via constructive algorithms such as cellular automata into a detailed level in 2D or 3D (see Fig. 2).

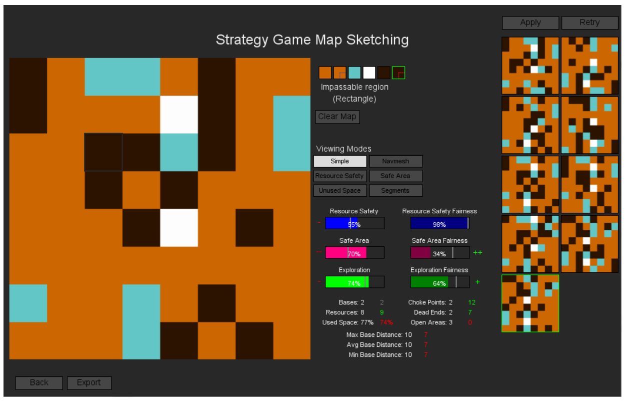


Fig. 1: The user interface of Sentient Sketchbook: the user draws a sketch on the canvas to the left, the computational suggestions are to the far right, and between them is the tile palette, alternate displays and evaluations of the user's sketch.

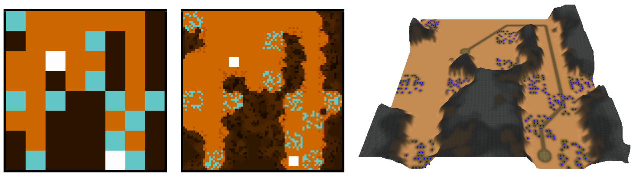

(a) Map Sketch (b) Final Level (2D) (c) Final Level (3D)

Fig. 2: A map sketch and the strategy game level it creates. White tiles are bases, cyan tiles are resources and dark tiles are impassable.

Map sketches for strategy games are evaluated via six fitness dimensions which evaluate properties affecting the pacing and tactics of gameplay. These evaluations are shown on the user interface (see Fig. 1), and are re-evaluated every time the user edits the map sketch. The six fitness dimensions are: f_{res}, which evaluates how safe (i.e. nearby) resources are to any base and b_{res} how balanced the distribution of safe resources is among bases; f_{saf}, which evaluates how many safe passable tiles are near any base and b_{saf} how balanced this distribution of safe passable tiles is among bases; f_{exp}, which evaluates how much exploration is needed to find all bases starting from all other bases and b_{exp} how balanced such exploration is when starting from different bases. Argumentation and mathematical definitions for these fitness dimensions are included in [17]. Additionally, map sketches are constrained to be playable, by having all bases and resources connected via passable paths.

The most important feature of Sentient Sketchbook, in terms of computational initiative in the creation process, is the generation of alternatives to the user's creation and their presentation as suggestions during the design process. At any time, the user can select one of the generated suggestions to replace their current sketch, and continue editing it. These suggestions are generated via a short evolutionary sprint of constrained search from an initial population consisting of mutations of the user's current sketch. Constrained optimization ensures that the shown suggestions are playable, i.e.

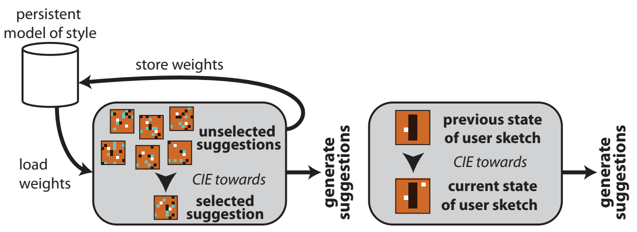
that bases can reach each other and all the resources via passable paths. In previous versions of Sentient Sketchbook [1], six suggestions are generated via six independent FI-2pop GAs [18] to maximize a fitness dimension (each suggestion optimizes for one dimension), while another six suggestions are generated via Feasible-Infeasible novelty search to create visually diverse map sketches [19].

IV. Designer Modeling for Sentient Sketchbook

During a previous set of user evaluations of Sentient Sketchbook reported in [1], computer-generated suggestions were not used in several cases. Based on user feedback and from a qualitative evaluation of their interaction data, suggestions were not selected in cases where (a) the user had preplanned a map layout before even starting the tool, (b) the number of bases in the generated suggestions were different than those of the user's sketch, (c) instances where suggestions appeared disconnected to what the user was focusing on at that time and (d) the generated suggestions lacked the visual symmetries found in most human-authored maps. Cases of type (a) could arguably never benefit from computer-generated suggestions, while cases of type (b) can be avoided by a simple algorithmic tweak that limits the number of bases to that of the user's sketch (or a minimum of two bases as a strategy game requires two players). However, cases of type (c) and (d) can not be countered trivially; this paper proposes additions to Sentient Sketchbook which, to an extent, address these cases. Cases of type (c) are addressed in two ways: by maintaining a persistent model of designer preference based on past choices among presented suggestions, as described in IV-A, and by adapting a model of process according to the latest map editing activity, as described in IV-B. Cases of type (d) are addressed by extrapolating the target visual symmetries of the user's final goal and adapting the mapping from genotype to phenotype to always exhibit such symmetries, as described in IV-C.

A. Acommodating a Designer's Style

A straightforward way of personalizing content to a designer is by biasing dimensions which are deemed important to the user based on their past choices of suggestions. This model of style is equivalent to choice-based interactive evolution [13] (CIE) and assumes that each user has an overarching style which favors certain strategic properties over others, e.g. levels with winding paths and scattered resources. Since Sentient Sketchbook presents multiple suggestions per step, the user's selection of one suggestion over others is assumed to contain information on their overall style. The model of style is represented as a weighted sum of all fitness dimensions, where the weights are adapted to capture those strategic qualities prevalent (either higher or lower) in the selected suggestion compared to unselected ones. These weights can be negative, in which case they drive objective-driven search towards maps with e.g. fewer safe resources. Weights are adjusted via the simple but straightforward weight update rule of eq. (1): if the fitness score f_n (in dimension n) of the selected suggestion is higher than the averaged fitness score in dimension n of the unselected suggestions, then the weight w_n increases; vice versa, if the fitness score of the selected suggestion is lower, then w_n decreases.



(a) Modeling Style (b) Modeling Process

Fig. 3: Modeling style and process, with one sequence of weight updates (via CIE) shown inside the gray box. The model of style only learns when a user selects a suggestion, and it stores the weights into a persistent database. The model of process learns every time users edit their sketches but does not store the updated weights; after every user action, the model always starts learning from weights of 0.

$$w_n^{t+1} = w_n^t + \alpha(f_{n_S} - \bar{f}_{n_U}) \tag{1}$$

where α is a weight update step (0.01 for this study), f_{n_S} is the selected suggestion's score for fitness n and $\bar{f}_{n_U}$ is the average score for fitness n across all unselected suggestions.

Weight updates are performed for a maximum number of epochs ($3 \cdot 10^5$ in this study) or until the selected map has a higher score (calculated as the weighted sum of fitness scores for all dimensions) than all other maps. Once adaptation is complete, the weights are normalized so that the sum of their absolute values is 1. The designer's style, as captured by the weighted sum, can be directly used as a fitness function for the feasible population of the FI-2pop GA which creates Sentient Sketchbook's suggestions.

B. Extracting the Designer's Process

Modeling a designer's overall style can be useful for long periods of interactions with extensive use of suggestions. Modeling the designer's process, on the other hand, is more situational and focuses on what the user is *currently* doing. The proposed model takes into account the designer's current point in the process (i.e. their current sketch) and their sketch prior to the last applied action; such an action could be painting a set of impassable tiles, adding or removing a base or resource, or replacing a sketch with a suggestion. The model assumes that the user implicitly prefers their current version of the sketch to the previous one; this is true for most cases since few user actions (such as erasing tiles) can be considered a "step backwards" in the design process.

As with the model of style, the model of process largely follows the paradigm of choice-based interactive evolution in the sense that a fitness score is derived from a weighted sum of strategic qualities, where the weights are adapted according to the user's last action (and its initial state and end state). The scores in different fitness dimensions of the current and previous sketch are used to adapt the weights of the model of process, using eq. (1) where $\bar{f}_{n_S}$ is the score of the current sketch in dimension n and $\bar{f}_{n_U}$ is the score of the previous sketch (since the score is singular, it is not averaged). However,

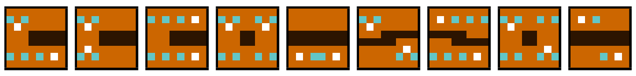

(a) Map (b) V_t (c) V_b (d) H_l (e) H_r (f) P_t (g) P_b (h) P_l (i) P_r

Fig. 4: The effects of different symmety types and map halves when creating symmetrical maps from the base sketch of Fig. 4a. Vertical reflection is shown as V_t (reflecting the top half) and V_b (reflecting the bottom half); horizontal reflection is shown as H_l (reflecting the left half) and H_r (reflecting the right half). Point symmetry centered on the map's midpoint is achieved by rotating by 180^o the top, bottom, left or right half of the map (P_t, P_b, P_l, P_r respectively).

the model adapts its weights via eq. (1) for a minimum of $5{\cdot}10^4$ epochs, even when the current sketch has a higher score than the previous; this ensures that a sufficient bias is placed on prominent features of the current sketch. The model of process can use weights adapted from the previous action to bias the model of the next action (as a form of "memory"), similarly to the model of style. However, this paper assumes that the highly situational nature of the creative process necessitates a *tabula rasa* approach by setting each step's initial weights at 0; in this fashion, the model has no memory of previous actions and is not biased by assumptions of strategic quality except those behind the user's latest action.

C. Fulfilling Goals of Symmetry

While models of style and process focus on the strategic qualities of the generated suggestions, a popular pattern in both AAA strategy games and in past interaction data of Sentient Sketchbook was the prevalence of symmetries in the designers' creations. Previous work [13] included symmetries as fitness functions and adapted a preference model towards more symmetrical levels. Instead, this paper assumes that symmetry is not an explicit designer preference but an intended goal of the design process. The model of symmetry goals, as it will be identified, assumes that once users introduce a certain symmetry to their level (even in early steps) they are expected to follow that symmetry throughout the process, even when focusing on different strategic qualities such as chokepoint or resource placement. As visual symmetry is assumed to be orthogonal to strategic quality (e.g. a map can be balanced even if it does not appear symmetrical), the search process can target fitness dimensions of strategic quality while the mapping from genotype to phenotype is constrained to create only symmetric maps. This potentially allows the search to be biased by models of style or process while the appearance of the artifacts is biased towards the target visual "feel".

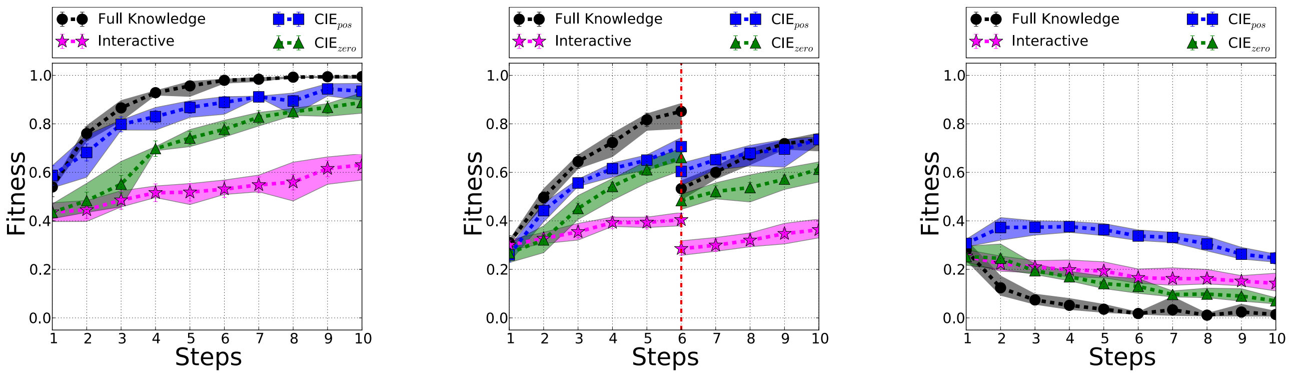
This paper introduces several restrictions to the mapping between genotype to phenotype, which ensure either reflective or point symmetry (see Fig. 4). Since these symmetries are along axes passing through the map's midpoint, the genotype contains data for the tiles of one half of the map sketch (top, bottom, left or right); the phenotype uses that data with the necessary inversions to create both halves of the map sketch. Choosing which mapping to use for suggestions depends on the current symmetries of the user's sketch, if any such symmetries exist at all. In order to find the best mapping for symmetric maps as well as which half of the user's sketch should seed the population of suggestions, eight symmetric versions of the base sketch are created (see Fig. 4) and evaluated on their similarity with the base sketch. The symmetry heuristic (sym) is calculated via eq. (2) and assigns equal importance to the symmetry of bases as it does to that of resources or impassable tiles, even though the latter are often more numerous than the former; this design choice is due to the importance of bases (as starting player positions) to the gameplay of a strategy game. If the best scoring symmetry is above a minimum threshold ($sym > 0.75$ in this study), the suggestions offered as alternatives to the current sketch will use that mapping while their initial population will consist of mutations of the half of the map designated by the symmetry type. If no symmetry is above the threshold, the genotype describes the entire map sketch as in the default setting of Sentient Sketchbook.

$$sym = \frac{1}{N}\sum_{n=1}^{N}\frac{n_{\cup}-n_{\cap}}{n_{\cup}} \tag{2}$$

where $n_{\cup}$ is the number of tiles of type n which are in either the base sketch or its symmetric version; $n_{\cap}$ is the number of tiles of type n which are at the same position in both the base sketch and its symmetric version; N is the subset of non-empty tile types which are present in the base sketch. For instance, if there are no impassable tiles in the base sketch, then $sym = \frac{1}{2}\left(\frac{B_{\cup}-B_{\cap}}{B_{\cup}} + \frac{R_{\cup}-R_{\cap}}{R_{\cup}}\right)$ with B and R being bases and resources respectively.

V. Experiments with Artificial Agents

In order to test how the model of style can (a) learn from past choices and (b) influence future shown suggestions, a number of simulated uses of Sentient Sketchbook were performed, using different models of style. The artificial agents that use the tool are assumed to create map sketches solely by selecting among generated suggestions in every step and replacing their user sketch with their favorite suggestion. Directly drawing tiles on the canvas is not allowed since the model of style does not learn from manual edits of the sketch. The simulation follows the process of Sentient Sketchbook closely: every time the user's sketch changes (in this case by being replaced by a generated suggestion), a new evolutionary sprint of 10 generations is performed starting from an initial population consisting of mutations of the user's current sketch. The algorithm has a population of 20 individuals (feasible or infeasible) and evolves via a FI-2pop GA using fitness-proportionate roulette wheel selection and two-point crossover with a 1% chance of mutation. Mutation can swap tiles with their adjacent neighbors or transform passable tiles to impassable and vice versa. Unlike previous versions of Sentient Sketchbook, the feasible population is guided by the adaptive model of designer style, represented as a weighted sum. After the short evolutionary sprint the 6 fittest map sketches are shown to the user (although the shown sketches may be fewer if there are not enough feasible individuals in the population). The choice of 6 suggestions shown to the user is made under the assumption that the adaptive model of style replaces the current 6 suggestions which are evolved towards specific strategic qualities. The remaining 6 of the 12 total suggestions are evolved via novelty search which does not make any



(a) Artificial agent choosing the highest f_{exp}. The Y axis is the score of f_{exp}.

(b) Artificial agent choosing the highest f_{saf} for steps 1-5 and the highest f_{res} for steps 6-10. The Y axis is f_{saf} before step 6 and f_{res} after it.

(c) Artificial agent choosing the lowest f_{res}. The Y axis is the score of f_{res}.

Fig. 5: The average fitness (dotted line) and the fitness range (filled area) of shown suggestions in consecutive steps of simulated runs. Values are averaged across 50 runs.

assumptions on designer taste or strategic quality and act as random stimulus which is expected to prompt diagrammatic lateral thinking on the visual level [2].

For the purposes of brevity, the model of style will be evaluated via three artificial agents attempting to create small strategy map sketches which consist of a 8 by 8 tile grid, two bases and 4 to 10 resources. To demonstrate as many different behaviors as possible, the artificial users select suggestions according to a single objective throughout the run, or change their mind halfway through the run by selecting according to a different objective. The previous experiment with expert users [1] demonstrated that selected suggestions are evolved primarily towards exploration (f_{exp}) and secondly towards safety of resources (f_{res}) and area control (f_{saf}). In light of this finding, one artificial agent chooses solely the suggestion with the highest f_{exp} (Fig. 5a), the second chooses the suggestion with the highest f_{saf} for the first half of the run, then choosing the suggestion with the highest f_{res} for the second half of the run (Fig. 5b). The above agents operate under the assumption that map sketches with high scores in strategic qualities are more desirable to human users; the third agent subverts this by choosing the suggestion with the lowest f_{res} (Fig. 5c). Since f_{res} has high scores when all resources are very close to bases and low scores when resources are equally far away from all bases, it may be the case that level designers prefer to place resources in contested areas; the third artificial agent simulates such level designers.

Figure 5 compares the design sessions of the three artificial agents based on the fitness of the shown suggestions. For each step (i.e. every time an artificial agent selects a suggestion to replace its current sketch), all shown suggestions are evaluated according to the fitness used by the artificial agent for choosing its preferred map sketch; the average of these fitness scores is shown as a dotted line, while the filled area represents the range of all suggestions' fitness scores (maximum to minimum). Values presented are averaged across 50 runs, with the average fitness score of suggestions displaying its standard error as error bars. Between steps, an evolutionary sprint of 10 generations is performed according to a specific (adapted or otherwise) measure of designer preference; for initialization, an empty map (without impassable tiles, bases or resources) is evolved for 10 generations to create the suggestions shown at step 1. The methods being evaluated include the adaptive model of style (CIE) with different initialized weights: CIE$_{pos}$ which initializes all weights with an equal positive amount adding up to 1 (in this case of 6 weights, all weights start from $\frac{1}{6}$), and CIE$_{zero}$ which initializes all weights at 0. The former model assumes that users prefer maps with high strategic qualities — and is in line with previous work on adaptive models of taste [6], [13] — while the latter model makes no assumptions whatsoever. The experiment includes two baselines for comparison: the *full knowledge* model which uses the same fitness function as the agent to evaluate and evolve its feasible individuals, and the *interactive* model which evaluates feasible individuals according to their genotypical distance from the previous selected suggestion (which is assumed to have optimal fitness of 1). The interactive model does not show content for selection in every generation, and follows Takagi's suggestions for faster convergence in interactive evolution via distance-based prediction of unseen artifacts' fitness values [9].

Observing Fig. 5a, the CIE$_{pos}$ model is quick to accommodate the user's taste and create maps which score high according to the user's fitness. Obviously, the full-knowledge model outperforms other methods as it has a single objective (compared to the multiple fitness dimensions of CIE methods) and does not need to learn from user choices. The CIE$_{zero}$ model needs more steps to learn from user choices, leading to a slow start which eventually however also reaches high values. Interactive evolution understandably performs poorly as it only rewards similarity with the previous selected individual, essentially limiting the maximum fitness it can reach to the fitness of the previous individual. If the previous selected individual is not particularly good according to the user's fitness, only the stochasticity of the genetic search can lead to improvements; the small stepwise fitness score increase shows that such improvements are minor and unlikely. It should be noted that CIE$_{pos}$ and the full knowledge methods start from a higher score at step 1, as the suggestions of step 1 are evolved according to a weighted sum with positive weights and a single objective (respectively). By comparison, the CIE$_{zero}$ and interactive methods reward all individuals with 0, as they have initial weights of zero and no previous selected individual respectively; the random search before step 1 results in suggestions with lower scores than the other two methods.

A similar behavior across all methods is observed in the first 5 steps of the second artificial agent in Fig. 5b, as the full knowledge model outperforms CIE models which in turn outperform interactive evolution. When the artificial agent changes its selection criteria at the 6th step, the full knowledge model is shown to have a relatively low f_{res} score (the criterion for steps 6-10) due to the fact that evolution previously focused entirely on optimizing f_{saf} to the detriment of other strategic qualities. By comparison, the CIE$_{pos}$ model, which has positive weights for f_{saf} as well as f_{res} (at least initially), seems to optimize towards both objectives during the first steps and thus has individuals with relatively higher scores in f_{res} on step 6. CIE$_{zero}$ is not initially biased towards maximizing all strategic qualities, and thus adaptation is less likely to result in positive weights for f_{res} in steps 1-5, leading to a lower fitness in f_{res} by step 6, and a slower adaptation of its weight in steps 6-10. Finally, interactive evolution performs poorly both in steps 1-5 for f_{saf} and in steps 6-10 for f_{res} and results in overall worse maps with low scores in both fitnesses.

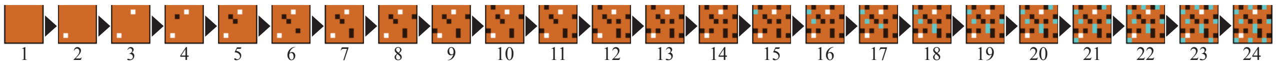
The most surprising behavior for CIE$_{pos}$ is demonstrated with the third artificial agent that chooses the suggestion with the lowest f_{res} score (Fig. 5c). The other methods have similar behaviors as in other experiments: the full knowledge model reaches the lowest (i.e. most desirable) scores, followed by CIE$_{zero}$ and interactive models in order of performance. CIE$_{pos}$ initially improves the f_{res} of suggestions, despite the fact that the user desires the opposite; this is likely due to the initial positive weights which are not adapted to a sufficient degree so that they become negative. Eventually, the f_{res} of suggestions begins to drop, although the process is very slow. It is likely that the remaining fitness dimensions in the weighted sum have positive weights (at least in the first few steps) which may drive evolution towards directions not reflected in the user's taste and thus not shown in the figure. The poor behavior of CIE$_{pos}$ can be traced back to the initial bias towards suggestions with high scores in the strategic qualities chosen; since this same bias worked well in cases where users selected according to the strategic qualities, there is no obvious answer regarding which version of CIE is more desirable.

VI. Experiments with Past Design Sessions

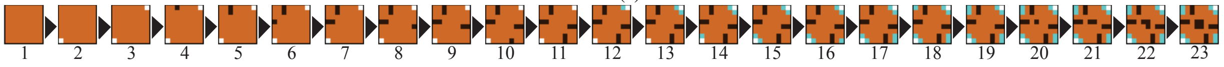
In order to test how the models of process and symmetry goals perform, simulated runs with artificial users would not provide any insights since the artificial agent does not draw directly on the canvas (which is used for modeling process) and does not exhibit visual symmetries (which is modeled as a designer goal). Instead, experiments for these two models will use previous interaction data of experts in game development who used Sentient Sketchbook to create game levels. Of the 24 design sessions reported in [1], this paper will focus on 3 sessions of different users while they created small maps, as the number of steps is more manageable. The entire process from empty canvas to final map sketch will be considered (see Fig. 6), with each step corresponding to a single designer action. In session 1, the designer started by placing bases, then adding impassable regions to make discovery of enemy bases more challenging and finally added resources; both the final map and the intermediate steps seem rather haphazard and have no obvious symmetries. In session 4, the designer starts by placing bases as far away from each other as possible (on two corners of the map), then places impassable tiles, resources and more impassable tiles in that order; starting from the symmetrically placed bases and in most steps of the creation path, the symmetrical nature of the map is rather obvious to a casual human observer. In session 7, the designer starts by placing resources at the top of the map followed by a nearby base; consecutively, another base at the bottom of the map is added followed by resources symmetrical to the ones at the top. After step 19, the user is assumed to run out of ideas and successively replaces their sketch with computer-generated suggestions; steps 20-23 are all computer-generated suggestions, which explains why they lack the symmetries of the human-authored sketch.

A. Adapting to Current Process

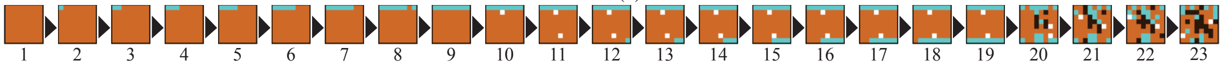
In terms of the model of process, the weights for all strategic qualities on each step of the creation path are shown in Fig. 7. For session 1, the weights of f_{exp}, b_{exp}, f_{saf}, b_{saf} change drastically in steps 3-15 while the user adds impassable tiles which increase the difficulty of reaching an enemy base (f_{exp}, b_{exp}) and create "pockets" of safe areas near each base (f_{saf}, b_{saf}). Since the weights of these four qualities change so drastically in steps 3-15, generated suggestions would evolve to maximize a strategic quality in one step only to attempt to minimize it in the next step (e.g. b_{saf} at step 12 and 13). Once the user begins placing resources after step 15, the weights of f_{res} and b_{res} are the only ones being adjusted (since remaining fitnesses do not depend on resources). These weights also change dramatically in steps 15-24, focusing from f_{res} to b_{res} and vice versa. For session 4, the model of process is much more stable; since the bases are initially placed as far away as possible, exploration is always at its optimal value and the weights of f_{exp} and b_{exp} never change throughout the process. Between steps 4 and 12, the user's addition of impassable tiles blocks off passable regions which are rendered safe for one of the bases, leading to the model of process focusing singularly (with a weight of 1) on f_{saf}. Steps 14 and 15 detect that the user places resources next to the bottom base balancing those of the top base, and increases the weight of b_{res} to 1. Once the designer places bases in the corners furthest from the bases, the model adjusts f_{res} to -1, detecting that the current focus is unsafe resources. Finally, when the designer adds impassable regions in steps 20-23, weights for f_{res}, b_{res}, f_{saf}, b_{saf} change drastically as the distances between bases change, in turn affecting all of those fitnesses. For session 7, the absence of two bases until step 11 makes the calculation of any fitness dimension impossible as they all hinge upon comparisons of distances between bases. While strategic qualities can not be evaluated, their weights are obviously not adjusted. Between steps 11 and 19 the user places resources and therefore the weights of f_{res} and b_{res} are adjusted: since the user adds resources near the base that didn't previously have any, the model detects that the user attempts to balance the map (b_{res}). Once the user begins making large changes to the sketch via suggestions at steps 20-23, the weights are adjusted drastically, although surprisingly the weights are rather close to each other



(a) Session 1

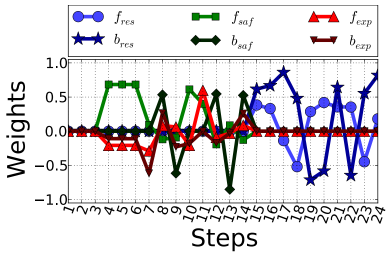


(b) Session 4

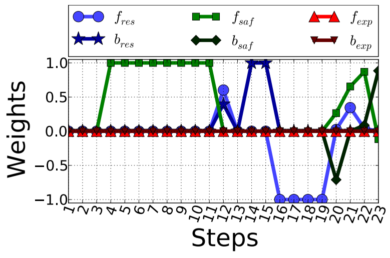


(c) Session 7

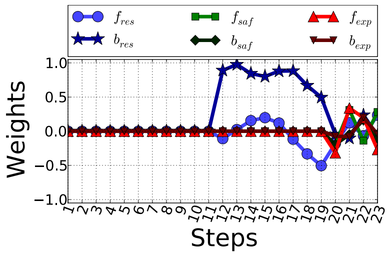
Fig. 6: A sample of the creation paths of design sessions on small maps, displaying all steps taken. Session names are from [2].



(a) Session 1

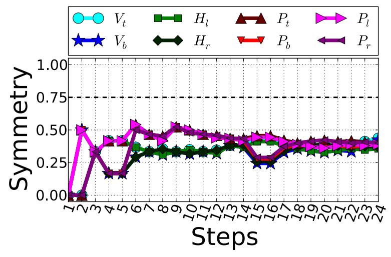


(b) Session 4

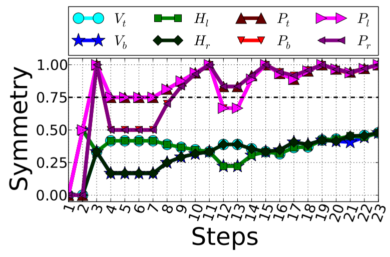


(c) Session 7

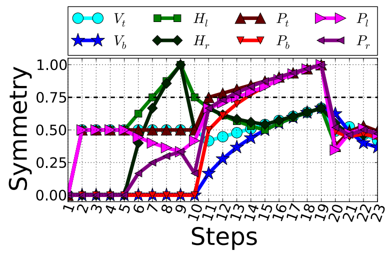
Fig. 7: Stepwise changes in the weights of the six fitness dimensions of strategic qualities for the creation paths of Fig. 6. The changes reflect the designer model of process which compares the sketch of the current step with that of the previous step.



(a) Session 1



(b) Session 4



(c) Session 7

Fig. 8: Stepwise changes of the sym score for different symmetry types for the creation paths of Fig. 6. The symmetry model with the highest score over the threshold of 0.75 (shown as a dotted line) would cause suggestions to have a perfect symmetry according to that model by changing the mapping from genotype to phenotype.

and none of them reaches very high (or very low) values.

B. Evaluating Symmetry Goals

Concerning symmetry, Fig. 8 displays the sym values of different symmetry types. According to Fig. 8a, session 1 has low sym values for all symmetry types from start to finish, which can be verified with a casual observation of all steps in Fig. 6a. According to Fig. 8b, the point symmetry of the two bases at step 3 immediately results in a value of 1 for all point symmetries (P_t, P_b, P_r, P_l). While impassable tiles are not added in a symmetrical fashion in steps 4-7, the symmetry of bases pushes the sym score of P_l above the threshold of 0.75 and thus all suggestions during these steps would have point symmetry. Consecutive additions of impassable tiles complete the symmetry, and later additions of resources do not affect the best symmetry as much (since two of three tile types have perfect symmetry). From step 3 until the end of the session, there is at least one point symmetry with a sym score above the 0.75 threshold, so all suggestions from step 3 onward would exhibit point symmetry. According to Fig. 8c, on the other hand, the model assumes a goal of horizontal symmetry during steps 7-10 due to resources on the left half being reflected on the right half. Once the user begins drawing the bottom row of resources at step 12, the model assumes a goal of point symmetry (or vertical symmetry to a lesser degree) as the bottom half begins to reflect the top half. While suggestions in steps 12-19 would exhibit point symmetry, once the user drastically changes their map via computer-generated alternatives at steps 20-23, no particular symmetries are found. This lack of symmetry in generated suggestions in the previous version of Sentient Sketchbook shown in Fig. 6 reveals the need for a designer model of symmetry goals, as the suggestions at steps 20-23 would have perfect point symmetry similar to the user's sketch at step 19.

VII. Discussion

Although not yet tested with human designers, the experiments described in this paper show several promising properties for each of the designer models proposed. The designer model of style can find the key strategic qualities favored by a designer, even when it starts without any prior bias (CIE$_{zero}$). However, this model of style assumes that designers normally create maps with high scores on the six fitness dimensions of Sentient Sketchbook; a subversive user has shown that adaptive models may underperform when that assumption is not met and, presumably, would fare worse when user preferences are completely orthogonal to the prescripted fitness dimensions. Moreover, this designer model requires that users choose suggestions in order to learn their style, which may be problematic if those suggestions are not appealing in the first place. The model of process avoids this caveat by using the designer's manual edits as well as any chosen suggestions in order to provide focused, situational feedback. A potential caveat for the model of process is its dependency on base tiles; as shown in session 7, the model is not adapted (and provides random suggestions) while the user draws map sketches which do not contain bases. Finally, the model of symmetry goals can identify the most common visual patterns in maps and dynamically switch between mappings from genotype to phenotype in order to ensure symmetries in generated suggestions. As the model of symmetry goals does not affect search, it can be used with any other designer model as well as with novelty search; its only possible limitation is when map sketches have symmetries not included in any mapping, in which case the model would fail to detect them.

While each model focuses on different aspects such as visual appearance or consistent preference, there is great potential in combining the different models in a meaningful way. Different symmetry values (sym) could be used as fitness dimensions for modeling style, along with (or instead of) strategic qualities; in that case, the model would discover a persistent visual style and bias search towards it. The current model of style, on the other hand, could inform the search for new suggestions during the periods when the model of process is not being adapted due to lack of base tiles on the user's map sketches. Finally, the model of style could incorporate the detection of new features and thus not limit itself to six strategic qualities; this is likely easier to accomplish with a database of choices from multiple designers, allowing for a sufficient data volume to facilitate learning.

VIII. Conclusion

This paper has argued that modeling a designer's goals, style and process can enhance computer-aided design; Sentient Sketchbook was used as an example of how such models can create more personalized feedback. Having identified the most challenging cases where suggestions did not appeal to past users of the tool, this paper proposed models which adapt the importance of different strategic qualities based on overall style or current process as well as different mappings from genotype to phenotype which can be turned on or off based on the estimated target visual patterns of the designer's final creations. These initial findings must be tested with human users, in order to ascertain which of the models are more potent at capturing and accommodating designer intentions.

Acknowledgment

The research is supported, in part, by the FP7 ICT project C2Learn (project no: 318480) and the FP7 Marie Curie CIG project AutoGameDesign (project no: 630665).

References

[1] A. Liapis, G. N. Yannakakis, and J. Togelius, "Sentient sketchbook: Computer-aided game level authoring," in *Proceedings of the 8th Conference on the Foundations of Digital Games*, 2013, pp. 213–220.

[2] G. N. Yannakakis, A. Liapis, and C. Alexopoulos, "Mixed-initiative co-creativity," in *Proceedings of the 9th Conference on the Foundations of Digital Games*, 2014.

[3] A. Liapis, G. N. Yannakakis, and J. Togelius, "Designer modeling for personalized game content creation tools," in *Proceedings of the AIIDE Workshop on Artificial Intelligence & Game Aesthetics*, 2013.

[4] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, no. 99, 2011.

[5] D. Ashlock and C. McGuinness, "Landscape automata for search based procedural content generation," in *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, 2013.

[6] A. Liapis, G. N. Yannakakis, and J. Togelius, "Adapting models of visual aesthetics for personalized content creation," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 3, pp. 213–228, 2012.

[7] J. Togelius, R. De Nardi, and S. Lucas, "Towards automatic personalised content creation for racing games," in *Proceedings of IEEE Symposium on Computational Intelligence and Games*. IEEE, 2007, pp. 252–259.

[8] G. N. Yannakakis and J. Togelius, "Experience-driven procedural content generation," *IEEE Transactions on Affective Computing*, vol. 99, 2011.

[9] H. Takagi, "Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296, 2001, (invited paper).

[10] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience for content creation," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 1, pp. 54–67, 2010.

[11] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. Andre, "Player modeling," *Dagstuhl Seminar on Game Artificial and Computational Intelligence*, 2013.

[12] J. Fürnkranz and E. Hüllermeier, *Preference Learning*. Springer-Verlag New York, Inc., 2010.

[13] A. Liapis, H. P. Martínez, J. Togelius, and G. N. Yannakakis, "Adaptive game level creation through rank-based interactive evolution," in *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, 2013.

[14] B. A. Goodman and D. J. Litman, "On the interaction between plan recognition and intelligent interfaces," *User Modeling and User-Adapted Interaction*, vol. 2, pp. 83–115, 1992.

[15] H.-C. Wang, "Modeling idea generation sequences using Hidden Markov Models," in *Proceedings of the 30th Annual Meeting of the Cognitive Science Society*, 2008.

[16] H. A. Kautz, "A formal theory of plan recognition," Ph.D. dissertation, Bell Laboratories, 1987.

[17] A. Liapis, G. N. Yannakakis, and J. Togelius, "Towards a generic method of evaluating game levels," in *Proceedings of the AAAI Artificial Intelligence for Interactive Digital Entertainment Conference*, 2013.

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

[19] A. Liapis, G. N. Yannakakis, and J. Togelius, "Enhancements to constrained novelty search: Two-population novelty search for generating game content," in *Proceedings of Genetic and Evolutionary Computation Conference*, 2013, pp. 343–350.