Sentient Sketchbook: Computer-Assisted Game Level Authoring

ABSTRACT

This paper introduces Sentient Sketchbook, a tool which supports a designer in the creation of game levels. Using map sketches to alleviate designer effort, the tool automates playability checks and evaluations and visualizes significant gameplay properties. Most importantly, this paper introduces constrained novelty search via a two-population paradigm for generating, in real-time, alternatives to the author's design and evaluates its potential against current approaches. The paper concludes with a small-scale user survey during which industry experts interact with Sentient Sketchbook to design game levels. Results demonstrate the tool's potential and provide directions for its improvement.

1. INTRODUCTION

Over the last twenty years, computer games have grown from a niche market targeting young adults to an important player in the global economy [3], engaging millions of people from different cultural backgrounds. As both the number and the size of computer games continue to rise, game companies handle increasing demand by expanding their cadre, compressing development cycles and reusing code or assets. To limit development time and reduce the cost of content creation, commercial game engines and procedural content generation are popular shortcuts among game companies. Low-level engine features such as physics simulation and game-independent content such as vegetation are usually handled by third-party middleware such as *Havok Physics* (Havok, 2000) and *SpeedTree* (IDV, 2002) respectively. For content more contingent on a game's theme, mechanics and quests, most commercial game engines such as the *Unreal Development Kit* (Epic Games, 2009) assist in the fast and intuitive creation of game levels and scripts via game editors which automate mundane tasks such as pathfinding.

This paper presents a tool, named Sentient Sketchbook, which supports the design of map sketches via an intuitive interface, via real-time feedback regarding the map's playability or balance, and via the suggestion of alternative map designs. The tool assists the level designer as it automatically tests maps for playability constraints, calculates and displays navigable paths, evaluates the map on gameplay properties and adds details to the coarse map sketch. Additionally, the tool aims to enhance the designer's creativity through the suggestion of map alternatives generated by genetic search, which is novel as it enhances the constrained optimization capabilities of novelty search via the feasible-infeasible two-population paradigm. We assume that designers are driven by certain objectives pertaining to strategic gameplay (such as game pace and player balance) in their attempt to design a game level. However, it is likely that (some) designers are not following design patterns encapsulated by the set of objectives proposed — we argue that, in such cases, the principles of novelty search could be useful in providing alternative suggestions. These hypotheses will be tested with the presented version of Sentient Sketchbook.

The Sentient Sketchbook tool addresses certain limitations of mixed-initiative design by relying on simple map sketches to counter user fatigue and designer biases reported in the literature [9]. This paper extends previous work [14] which presented objective functions for evaluating strategy game levels and explored their optimization behavior. This paper focuses on the user interactions afforded by the tool and on the automatic generation of maps via feasible-infeasible novelty search. The former is evaluated via a small-scale survey with industry experts and the latter is evaluated via controlled experiments with no human interaction.

2. RELATED WORK

The Sentient Sketchbook tool combines computer-assisted design with the suggestion of alternative game levels generated procedurally via novelty search. A short overview of related work in these domains is presented below.

Computer-Assisted Design

Since its early stages of development, the computer was expected to assist in solving engineering problems by being involved in the creative design process and by automating tedious tasks. Computer-assisted design tools have often been identified by their dual role as "the designer's slave" [15] — performing simulations, analysis and constraint satisfaction tests — and as advisors when certain requirements are not met. As computers are becoming efficient at performing the former role, more researchers focus on the latter: Lubart [12] identifies that computers can assist in "idea generation and realization" via different roles: a) as a *nanny*, by facilitating a user to jot down and store abstract ideas, b) as a *penpal*, by spurring collaboration between multiple human designers, c) as a *coach*, by recommending sources of inspiration and d) as a *colleague*. The colleague role, which Lubart argues is the most ambitious, should contribute equally to the design discourse but could also incite creativity via "semi-random search mechanisms to generate novel, unconventional ideas".

In commercial games and game-like applications, computer-assisted design speeds up the development process in the form of game editors. Game editors use an intuitive graphic interface, allowing designers with little programming experience to script behaviors and create content, usually as part of a game level. Many of these tools ship with the final game, allowing end-users to generate content which increases replayability and fan loyalty. *Modding* via the provided game editors has often transcended the original game's concept and mechanics, leading to new game titles such as *Counter-Strike* (Valve, 2000) or new subgenres such as Multiplayer Online Battle Arenas. Over the years, game editors have become very sophisticated, driven by a desire to support the modding community or to reuse code across products. As an example, the *Unreal Development Kit* supports landscape sculpting, asset organization, scaling rendering accuracies and code-free visual scripting. On the other hand, game-specific editors such as the Creation Kit of *Skyrim* (Bethesda Softworks, 2011) allow less customization but offer game-tailored easy-to-use automations such as leveled item lists, navigation path generation and quest scripting.

Procedural Game Content Generation

The game industry has often relied on the procedural generation of game content during playtime to enhance the unexpectedness of the player's experience and increase the game's replayability value. From early games such as *Rogue* (M. Toy and G. Wichman, 1980) and *Elite* (Acornsoft, 1984) to contemporary titles such as *Torchlight 2* (Runic, 2012) and *Civilization V* (Firaxis, 2010), gameworld and level creation has been the principal application of procedural content generation (PCG); other applications include the creation of enemies as in *Darkspore* (Maxis, 2011) and weapons as in *Borderlands* (Gearbox, 2009). Although academic interest in PCG is relatively new [21], the majority of PCG researchers challenge the mostly random generative algorithms used in the game industry. Whether generating platformer levels [18], mazes [1], board games [2], racing tracks [19], weapons [4] or spaceships [11], most projects within academia attempt to control the algorithms' stochastic processes via constraints [10], objective functions [20] and predicted or reported player experience [22].

Novelty Search and Constrained Optimization

Genetic algorithms traditionally optimize a population of individuals by selecting the fittest according to a *fitness function* [5]. The solutions discovered by a genetic algorithm are sensitive to the design of the fitness function, especially if this function is ill-defined, difficult to quantify, or subjective. Novelty search [8] is a recent approach to genetic search, replacing optimization towards a quantifiable objective with optimization towards the population's diversity; it argues that by stimulating exploration of the search space, unforeseen high-quality solutions can be discovered. To better control exploration, the notion of *minimal criteria novelty search* was introduced in [7], where solutions which do not satisfy certain criteria are assigned a fitness of 0, severely limiting their chances of reproducing. In constrained evolutionary optimization, it is argued that infeasible solutions, especially those on the border of feasibility, contain valuable information which can guide search towards the discovery of feasible individuals [16, 13]. The feasible-infeasible two-population (FI-2pop) paradigm [6] optimizes infeasible individuals, stored in a different population than feasible ones, towards the border of feasibility in order to increase their likelihood of generating feasible offspring. While PCG projects use FI-2pop to optimize the population of feasible individuals according to an objective function [18, 10, 14], this paper contends that novelty search can be applied to the feasible population, generating diverse solutions which satisfy the subjective, unpredictable desires of the tool's users.

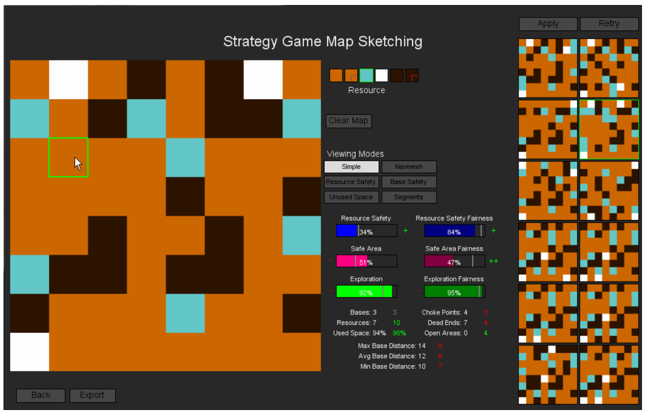


Figure 1: The Sentient Sketchbook tool during a design session. To the left is the sketch editor, far to the right are the automatically generated map suggestions; between these elements is the tile palette, the map display menu and an overview of map's fitness dimensions and metrics.

3. SENTIENT SKETCHBOOK

Sentient Sketchbook is a tool which allows a designer to create low-resolution map sketches. In this experiment the map sketches are abstractions of levels used in successful strategy games such as *Starcraft* (Blizzard, 1998), although other types of games can easily be represented in a similar sketch format. A map sketch consists of a small number of tiles (see Figure 1); tiles can be passable, impassable, player bases or resources. The map layout assumes that each player starts at a base and gathers resources to produce units which move through passable tiles to attack the opponent's base.

Sentient Sketchbook assists the design process as it tests maps for playability constraints, evaluates the map on gameplay properties, supports informative map displays and adds details to the coarse map sketch; these functionalities are presented in Section 3.1. Additionally, the tool aims to incite human creativity by generating map suggestions for the user; the generative algorithms are presented in Section 3.2.

3.1 Map Editor

A graphical interface has been developed to allow a user to manually edit a map sketch (see Figure 1). The map editor allows the designer to place tiles on the map; while the designer edits the map, the tool tests it for *playability*. A map is playable (or feasible) if all resources and bases are reachable from any other base or resource.

Playable maps are evaluated on quantifiable measures of quality regarding game pace and player balance. Game pace is affected by the location of a player's base: slow, defensive play is afforded if the base has many nearby resources, is surrounded by a large easily controllable area, and cannot be easily reached by other players. Player balance is tested on the same affordances across players. These *fitness dimensions* are presented below; more details on their inspiration and their mathematical formulation can be found in [14]:

Resource safety which evaluates how close every resource on the map is to any player base. Safe resources are close to only one base, while unsafe (contested) resources are at similar distances from two or more bases.

Safe area which evaluates how much area close to all player bases is considered safe; tiles around a player's base are safe if they are close only to that base.

Exploration which simulates how difficult every base is to find starting from all enemy bases. Exploration is simulated via a flood fill algorithm, which stops when an enemy base is discovered; the filled area is compared with the number of passable tiles on the map.

Resource safety balance which has high scores if all resources are equally safe (or unsafe) for all bases.

Safe area balance which has high scores if all bases have a similar number of safe tiles in their vicinity.

Exploration balance which has high scores if all bases are equally difficult to find from every enemy base.

These fitness dimensions are re-evaluated with every user interaction on the map and are displayed visually as progress bars. If a map has less than two bases or unreachable bases, all fitness dimensions display "N/A" to indicate infeasibility; if resources are missing or unreachable, only the resource safety and resource safety fairness dimensions display "N/A".

In addition to these fitness dimensions, a number of other *metrics* are collected and displayed via the map editor, mostly pertaining to map navigation. The displayed metrics include the number of bases and resources on the map, the shortest paths' length between bases, the percentage of used space[1] and the number of chokepoints, dead ends and open areas[2].

The editor allows the designer to switch between different map displays, which visualize navigational information and other properties (see Figure 2). At any time, the designer can switch to the final map view which displays the complete map on which a strategy game can be played (see Figure 3). The visualizations of the final map show how map sketches can be interpreted in different ways and how they can be applied to other game genres. These final maps (excluding Dungeon) are constructed via random processes and cellular automata to create an organic-looking map which retains all the properties (chokepoints, paths) of the coarse map sketch. Manual editing, evaluation and evolution are all done on the sketch level, making computations such as pathfinding easier and reducing the human effort required to design a complete map.

[1] *Used Space* consists of passable tiles which are on a shortest path between any two bases or any base and any resource.
[2] *Chokepoints* are tiles with two neighboring (passable) tiles, *dead ends* are tiles with one neighboring tile and *open areas* are tiles with the maximum number of neighboring tiles.

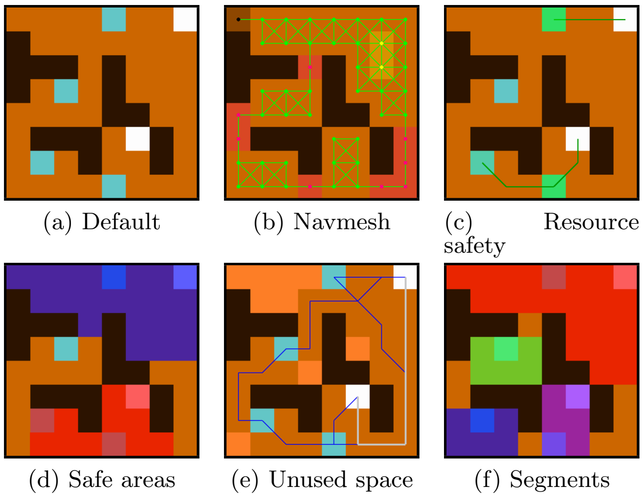


Figure 2: Different map displays on a small two-player map sketch: *Default* displays passable tiles (light brown), impassable tiles (dark brown), resources (cyan) and player bases (white). *Navmesh* displays the passable pathways on the map and the location of choke points (red), dead ends (black) and open areas (yellow); *Resource safety* displays the safety value of each resource (in shades of green) and connects bases to their safe resources. *Safe areas* shows the tiles close to each base (here in red and blue) which are considered safe. *Unused space* shows all shortest paths between bases and resources, and highlights leftover unused space (orange). *Segments* shows the passable areas (in different colors) which are surrounded solely by chokepoints.

3.2 Map Suggestions

In order for the tool to stimulate the user's creativity rather than to simply support it, it needs to provide new and unexpected alternatives to the user's current design. Through these *suggestions*, the tool intends to challenge the user's current design focus and provide unforeseen alternatives to achieving the designer's goals. The map suggestions are updated while the user edits the map, using the current map's appearance as their starting point. Currently map suggestions are generated via genetic algorithms (GAs) performing constrained optimization to maximize either the maps' scores in the fitness dimensions described in Section 3.1 or to maximize the maps' novelty compared to the designer's current sketch. Six genetic algorithms, running on separate threads, use the feasible-infeasible two-population paradigm [6] (FI-2pop GA) to optimize maps in a single fitness dimension each; the best evolved individual of each GA is included in the map suggestions. A seventh genetic algorithm uses the FI-2pop paradigm but evolves the feasible population via novelty search; the six most different evolved maps in this GA are also included in the map suggestions. Every GA runs for 10 generations, and optimizes a total of 10 individuals which initially consist of mutations of the authored map sketch. The 12 map suggestions (six evolved via objective-based search and six evolved via novelty search) are displayed, as thumbnails, on the edge of the tool's window; maps identical with the user's map or with each other are omitted. The user can at any time select a map suggestion and replace their current sketch with it. While a suggestion is selected, its scores in the fitness dimensions

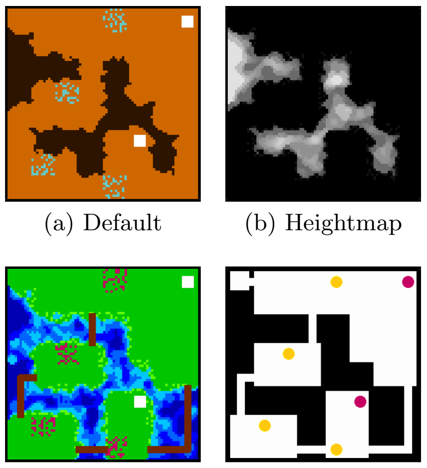

(a) Default (b) Heightmap (c) Waterways (d) Dungeon

Figure 3: Visualizations of the final map, which offers a higher-detail view of the map sketch in Figure 2. The *Default* visualization adds detail to resource tiles and impassable tiles, while *Heightmap* elaborates the impassable regions of *Default*. *Waterways* treats impassable tiles and chokepoints as water and replaces chokepoints with bridges, while *Dungeon* divides the passable segments and dead ends into rooms and treats chokepoints as corridors.

and its other metrics are displayed along with the current sketches' fitness scores and metrics respectively. There is a visual indication (via different colors) whether the suggested map increases or decreases the current map's score in that fitness dimension or metric. This extra feedback mechanism is expected to reduce the designer's cognitive load of closely inspecting each of the 12 map suggestions.

In order to optimize the map suggestions via evolution, each map is encoded as an array of integers: each integer represents a tile's type (passable, impassable, base or resource). All maps are tested for feasibility constraints: feasible maps must have an appropriate number of bases and resources and all resources and bases must be reachable from any other base or resource. The map's parameters are adjusted via a FI-2pop GA which evolves two populations, one with feasible maps and the other with infeasible maps. The infeasible population optimizes its members towards minimizing the distance from feasibility, which in this case is the number of unreachable bases and resources and the number of excess (or missing) bases or resources. As the infeasible population converges to the border of feasibility, the likelihood of discovering new feasible individuals increases. Feasible offspring of infeasible parents are transferred to the feasible population, boosting its diversity (and vice versa for infeasible offspring). The feasible population can be optimized to maximize different objective functions or to perform novelty search. Optimizing feasible individuals to maximize the fitness dimensions presented in Section 3.1 has been explored in [14]; the novelty in this paper is the evolution of the feasible population via novelty search. Following the novelty search paradigm [8], evolution optimizes feasible maps towards maximizing a function ρ corresponding to the average distance of the k most similar maps in the population and in an archive of novel maps. The archive stores the l highest scoring individuals in the population (in terms of ρ) and is reset at the start of every run of the evolutionary algorithm. Throughout this study, $k = 20$ and $l = 5$. The fitness score $\rho(i)$ for individual i is calculated as:

$$\rho(i) = \frac{1}{k}\sum_{j=1}^{k} dist(i, \mu_j) \qquad (1)$$

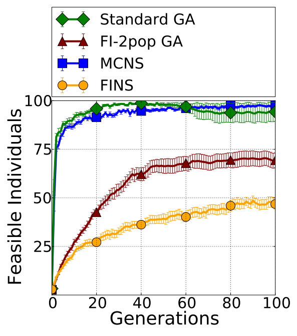
where μ_j is the j-th-nearest neighbor of i (within the feasible population and in the archive of novel individuals). Distance $dist(i, j)$ between maps i and j is calculated as the number of tiles which are not of the same type (passable, impassable, base or resource) for both maps at the same location.

Both feasible and infeasible populations evolve via fitness-proportionate roulette-wheel selection of parents; parents are recombined via two-point crossover. Mutation may occur on parents (5% chance) or offspring (1% chance); during mutation, either the map is rotated by 180^o (10% chance) or a portion of the map's tiles (between 5% and 20%) are altered (swapped with a neighboring tile, transformed from impassable to passable or from passable to any other tile type). This aggressive mutation scheme is appropriate for the short evolutionary runs of map suggestions.

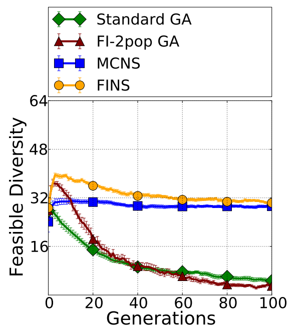
4. EXPERIMENTS

The Sentient Sketchbook tool introduces constrained novelty search via the FI-2pop paradigm, and uses Feasible-Infeasible Novelty Search (FINS) to generate map suggestions while a user edits the map. Before the generated map suggestions can be evaluated by human designers, the optimization behavior of FINS must be evaluated in a number of controlled experiments and compared against minimal criteria novelty search (MCNS) which has been used for the constrained optimization of novelty [7]. The experiments presented in this section use a random initial population of 100 individuals (including feasible and infeasible) and run for 100 generations; while such parameters cannot be afforded by the real-time suggestion generator included in Sentient Sketchbook, the algorithms should exhibit the same — if more robust — optimization behavior. Both novelty search approaches are compared with objective-driven constrained optimization using a standard GA and FI-2pop GA; both MCNS and the standard GA assign a fitness score of 0 to infeasible maps. The argument for allowing infeasible individuals to survive and reproduce in a separate population is twofold: a) while no feasible individuals exist, infeasible parents move closer to the border of feasibility and the discovery of feasible individuals is much more likely than with random search, and b) infeasible parents can eventually generate unforeseen feasible offspring, increasing diversity in the feasible population. The performance metrics are therefore: a) the number of feasible individuals and b) the *diversity* of the feasible population, measured as the average tile difference between all pairs of feasible maps in the population. The experiments test the optimization of a small map where constraints can be easily satisfied, and that of a large map where the discovery of feasible individuals is unlikely.

The first experiment tests the optimization progress of a map sketch of 8 by 8 tiles, where 2 player bases and 4 to 10 resources must be present; due to the small map size and few bases, it is likely that such maps are feasible even when generated randomly. Figure 4 displays the progress of the feasible population's diversity and size for MCNS, FINS, standard GA and FI-2pop GA; the standard GA and FI-

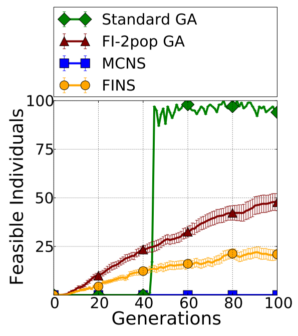


(a) Feasible individuals

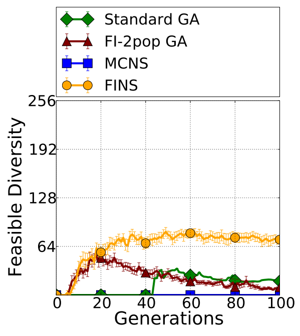


(b) Feasible avg. diversity

Figure 4: The progress of the number of feasible individuals and their average diversity when small, simple maps are optimized via different constrained optimization methods. Displayed values are averaged across 20 independent runs, and error bars denote the standard error.



(a) Feasible individuals



(b) Feasible avg. diversity

Figure 5: The progress of the number of feasible individuals and their average diversity when large, complex maps are optimized via different constrained optimization methods. Displayed values are averaged across 20 independent runs, and error bars denote the standard error. Note that for Standard GA, values refer to the single run on which a feasible individual was discovered.

2pop GA optimize the fitness dimension of resource safety, although their performance (in terms of the metrics being evaluated) is similar for all fitness dimensions. The optimization behavior of FI-2Pop GA in terms of the fitness dimensions used in Sentient Sketchbook is explored in [14].

Observing the number of feasible individuals for all approaches in Figure 4a, it is immediately obvious that single-population approaches (MCNS and standard GA) swiftly generate a large number of feasible individuals (almost 100% of the population). Since the feasible space is relatively large, the occurrence of feasible individuals in the (random) initial population is very likely. Infeasible individuals have a fitness of 0 for MCNS and standard GA and are swiftly killed in favor of fitter feasible ones; on the other hand, FI-2pop GA and FINS retain the most promising infeasible individuals and therefore the number of feasible individuals is relatively small. For FINS and FI-2pop GA the number of feasible individuals grows slowly as infeasible individuals converge to the border of feasibility, increasing migration from the infeasible to the feasible population. Novelty search is more likely to create an infeasible offspring from feasible parents, since it encourages exploration of the search space which may lead back to infeasible solutions; this increases migration from the feasible population to the infeasible and explains the lower number of feasible individuals for FINS.

Observing the diversity of the feasible population for all approaches in Figure 4b, it is expected that the diversity of feasible individuals drops swiftly for objective-driven approaches (FI-2pop GA and standard GA), as the population converges to a few highly fit solutions. Unsurprisingly, novelty search approaches maintain or increase their feasible individuals' diversity, as they optimize towards the same heuristic for distance used in the diversity metric. FINS maintains a higher diversity than MCNS, which can be attributed to the constant migration of novel feasible individuals from the infeasible population. This is most pronounced in early stages of evolution, when both FINS and FI-2pop GA have higher diversity values than their single-population counterparts due to infeasible individuals becoming feasible.

The second experiment tests the optimization progress of a map sketch of 16 by 16 tiles, where 8 player bases and 12 to 30 resources must be present; the numerous bases and resources are much more likely to be unreachable in this large map, and feasible individuals are less likely to be discovered via random search. Figure 5 displays the progress of the feasible population's diversity and size for MCNS, FINS, standard GA and FI-2pop GA — the latter two optimizing the dimension of resource safety. No feasible individuals were present in the initial population on any run and for any approach. For MCNS and standard GA, all members of the initial population have a fitness of 0 and the resulting random search makes discovery of a feasible individual both unlikely and unpredictable. Table 1 displays the number of runs that any feasible individual was discovered, and the average generation on which it occurred. Since both FI-2pop approaches (FINS and FI-2pop GA) perform an objective-driven optimization of the infeasible population towards the border of feasibility, random search is avoided and a feasible individual is consistently discovered in all runs within a few generations. Once a feasible individual is found within single-population approaches, the entire population converges to it and the number of feasible individuals remains high. For FI-2pop approaches, the fact that feasible parents are very likely to generate infeasible offspring results in few feasible individuals, even after extensive optimization. For FINS, migration from the feasible to the infeasible population is more pronounced, and feasible individuals rarely amount to more than 25% of the total population. As with the smaller map in the first experiment, FINS attains a higher diversity

Method	Runs	First Generation
MCNS	0	-
Standard GA	1	44 (0.0)
FI2pop GA	20	10.9 (2.1)
FINS	20	11.15 (1.6)

Table 1: The number of runs (out of 20) in which a feasible individual was discovered when optimizing large, complex maps, and the first generation in which this discovery occurred. The first generation is averaged across runs in which an individual was found, and the first generations' standard error is shown in parentheses.

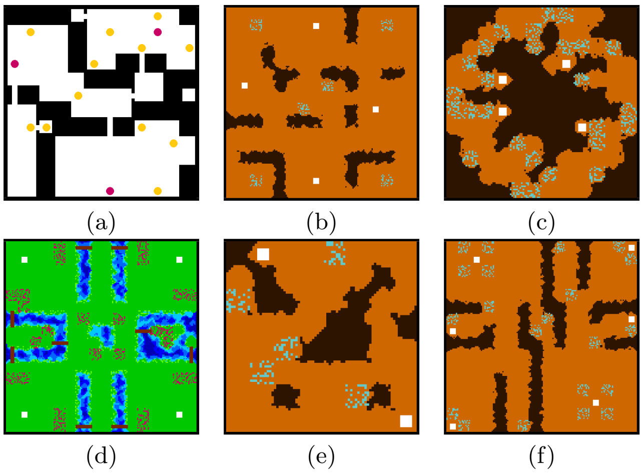


Figure 6: A sample of the final maps created by five participants interacting with Sentient Sketchbook; the range of map sizes and visualizations is shown.

in the feasible population than objective-driven approaches.

While different from the Sentient Sketchbook generator for their larger, randomly initialized populations and more generations, the controlled experiments presented in this section showcase several desirable properties of FINS which make it more appropriate than MCNS for the generation of map suggestions. As demonstrated by the experiment with large maps, FINS and all FI-2pop GAs are better at discovering feasible individuals in cases where the feasible space is small. Since FINS consistently finds feasible individuals within few generations, it complies to the real-time requirements of the map suggestion generator. On the other hand, the experiment with smaller maps demonstrated that FINS can achieve a higher diversity in the population than MCNS, especially in early stages of evolution. Given the short evolutionary sprint for generating map suggestions, FINS is preferable to MCNS for generating diverse maps — although in problems with easily satisfied constraints MCNS creates more feasible maps, which may also be desirable.

5. USER SURVEY

The Sentient Sketchbook tool assists human designers in level creation via real-time map evaluation and playability checks, via a number of map displays and via generated suggestions. In order to evaluate its usability, a small-scale user survey was conducted with 5 expert users consisting of independent game developers, game programmers and level designers. The tool was sent via e-mail to the participants, who were asked to interact with it as they saw fit on their spare time; feedback and interaction data was returned via e-mail. Each participant had several map design sessions; in each session a user creates a complete map starting from a blank canvas. Data is collected from 24 design sessions, with each participant contributing a minimum of 4 sessions.

A sample of the final maps created by users are displayed in Figure 6. Users could choose between small maps (8 by 8 tiles), medium maps (12 by 12 tiles) and large maps (16 by 16 tiles). Participants favored medium-sized maps in 42% of design sessions; remaining 29% of maps were small, and 29% large. Most maps had 2 bases (50%) or 4 bases (25%); all small maps had 2 bases. Unlike bases, resources vary — from 2 to 12 on Small maps, 6 to 27 on Medium maps and 6 to 40 on Large maps. Regarding map appearance, participants predominantly favored symmetrical features, with bases placed at the corners of the map as in Figure 6d or following symmetries on non-Cartesian axes as in Figure 6b.

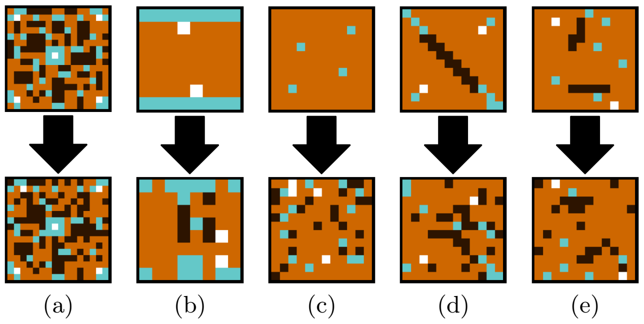


Figure 7: A sample of user maps before (top) and after (bottom) a suggestion was selected, showcasing different instances where suggestions were deemed useful.

The number of user actions taken (placing tiles, clearing the map or applying suggestions) in each design session range from 13 to 168 (58 on average). Observing the changes in fitness scores after each action, it appears that safe area and resource safety balance are consistent with design patterns as user actions lead to increased scores in these fitness dimensions. Using the difference between positive and negative changes over the total number of fitness changes as the test statistic, obtained values (0.41 and 0.24, respectively) are significant, with $p < 10^{-6}$ for both fitness dimensions.

Across all design sessions, map suggestions were selected 22 times to replace the user's sketch; although that averages to 0.92 suggestions selected per session, no map suggestions were selected in 9 sessions. Users offered a number of reasons why suggestions were not always useful. The simplest reason is that for some design sessions the user had a specific map in mind even before starting the tool; in such cases the design process did not include much map inspection or interaction with the suggestions. A related reason concerns cases where the number of bases differed between the user's map and the map suggestions; since adding or removing bases (and therefore players) would need a thorough re-evaluation of the designer's current ideas, such suggestions were not preferred. A final reason was the "organic" appearance of generated maps; while users favor straight lines for impassable regions (see Figure 6f) or completely symmetrical map bases, the generated maps rarely possess such neatly arranged features.

Users often selected one map suggestion per session (for 12 sessions). However, in two sessions map suggestions were selected 4 times; since the two sessions belong to different users, it is worthwhile to investigate the reasons behind this behavior. In one session, the user created an elaborate map (see Figure 7a), and then sequentially applied a map suggestion and saved the new map, thus creating four variations of their original map as a cheap shortcut for generating more content. In the other session, the user created a very simple symmetrical map (see Figure 7b) with all resources on two map edges, and successively used the map suggestions to add impassable regions and randomize tile placement. In other sessions with selected suggestions, a number of different us-

Objective of map suggestion	Times selected
Resource safety	3
Safe area	3
Exploration	5
Resource safety balance	2
Safe area balance	2
Exploration balance	1
Novelty search	6
Total	22

Table 2: The objectives for which map suggestions were optimized, among suggestions selected by the five users.

age patterns were identified. In 3 sessions, map suggestions were selected early in the design process to quickly generate a map "draft" which was then edited by the designer (see Figure 7c); such design sessions were short, since the suggestion was already playable and did not need significant changes. In 10 sessions, map suggestions were applied as one of the last steps before saving the final map; this last step either aimed at breaking the simple authored patterns and create more organic maps (see Figure 7d) or to increase the score in one or more fitness dimensions (see Figure 7e).

The type of map suggestions selected by users are shown in Table 2. Despite the fact that novelty search generated as many suggestions as all the objective-driven GAs together, its suggestions are selected less frequently than those generated to optimize specific objectives. Novelty search was preferred by users requesting large changes in the map, usually in early stages of the design process when user-created maps were still uninteresting; e.g. Figures 7b, 7c and 7d feature suggestions generated via novelty search. In late stages of the design process, when the authored maps were almost final, suggestions were used to fine-tune the map and were selected based on their improvements to the fitness dimensions. Surprisingly, suggestions targeting balance were not chosen often; a likely reason is that user-created maps were often optimal in most balance dimensions because of the strict map symmetries of human designs. Suggestions which increased the Exploration score, on the other hand, were more preferable as they create long, winding paths which would require significant effort to create manually.

Overall, the expert users' reception of Sentient Sketchbook was positive. Participants stated that the different map sketch displays were useful, especially Navmesh, Base Safety and Resource Safety, while the final map visualizations (especially the Dungeon) allowed them to envision level designs outside the realm of strategy games. Participants also stated that most of the parameters they evaluate strategy maps on are somehow represented in the included metrics, fitness dimensions and map displays. While one participant initially commented on the suggestions' apparent lack of visual structure and symmetry, they noted that after a while "the tool starts pushing you to places you didn't really consider" such as "working with some high level abstract ideas for the maps [...] instead of doing something symmetrical and intricate". Several suggestions for improvement were also provided, such as speeding up the generation of suggestions for large maps, giving users the option to "lock" certain features such as the number of bases in the map suggestions and improving the detection of chokepoints.

6. DISCUSSION

The experimental results for the feasible-infeasible novelty search paradigm show that it is faster to discover feasible individuals in cases where feasibility is not easily achieved. Based on the small survey with industry experts, the rapid generation of map suggestions was identified as an important feature; since participants in the survey preferred larger maps with a small feasible space, FI-2pop GA and FINS are ideal for finding feasible individuals quickly. Additionally, the survey demonstrated that maps optimized for novelty were selected when human designers needed inspiration; on the other hand, suggestions optimized for specific objectives were also popular when refining an elaborate human-authored map. Based on these results, it is clear that both objective-driven evolution via FI-2pop GA and novelty search via FINS are valuable for generating map suggestions.

The usability of Sentient Sketchbook as a computer-assisted level design tool was tested with five industry experts; based on their feedback, the option of alternating map displays both on the sketch and on the final map was useful. Additionally, the interface was deemed simple and intuitive, while the metrics and fitness dimensions reduce the effort of calculating path distances visually. Designer feedback from this small survey will help to refine some aspects of the tool, such as to introduce more useful map displays and improve how chokepoints are detected.

Future work for Sentient Sketchbook should aim to improve the quality of generated map suggestions. An interface option allowing the user to "lock" certain map metrics could transform such "locked" values into additional constraints. Moreover, users expressed the need for more visually consistent maps, favoring patterns like symmetry, straight lines and placement of bases on map edges. Such patterns can be detected on the human-authored map by e.g. an artificial neural network (ANN) where the inputs include all map tiles; the output of a trained ANN can replace the objective function for the feasible population. Finally, the map generator could use the data from design sessions to simulate humans' design processes, such as optimizing base placement first before starting to optimize impassable tile placement.

In its current form, the Sentient Sketchbook tool is more appropriate for generating maps for strategy games; the base and resource tiles and the fitness dimensions used assume strategic gameplay and can be used for authoring maps for different game genres only if creatively reinterpreted. For instance, First Person Shooter maps can be generated via the Dungeon visualization, treating player bases as the team's spawn location and resources as weapons (with powerful weapons placed in unsafe areas). Future work should allow maps for different game genres to be created, either by removing the current fitness dimensions or by allowing the designer to create their own types of tiles and even their own objectives. Alternatively, a specific game can be targeted by the map sketches; this affords better context for the designer who can now envision a full playthrough of a game. Additionally, the level can be playtested, and collected gameplay data can inform future design iterations.

In the context of Lubart's classification of roles which enhance idea generation [12], Sentient Sketchbook is at times a "nanny" by assisting the quick authoring of game levels and by overseeing their playability, and as a "colleague" by providing playable and often useful suggestions. A future addition to the tool should explore the role of "penpal", connecting the user to a larger group of designers and encouraging the exchange of ideas. This collaboration can be achieved via an online, persistent database of map designs which can be viewed, edited and resubmitted by any user. The collaborative creation of content has shown potential in projects such as PicBreeder [17], enhancing creativity and limiting user fatigue by crowdsourcing design choices.

7. CONCLUSION

This paper presented the Sentient Sketchbook tool which allows a designer to create game levels via a computer-assisted sketching interface. The tool supports the designer by automating map evaluations, visualizing them on-screen and proposing alternative designs. The creation of map suggestions combines the Feasible-Infeasible two-population paradigm with novelty search, and is shown to be more robust than state-of-the-art constrained novelty search in cases where discovery of feasible individuals is difficult. A user survey with industry experts demonstrated the tool's potential, attested to the different instances where map suggestions can prove useful and provided important feedback which will guide future additions to Sentient Sketchbook.

8. ACKNOWLEDGEMENTS

We would like to thank the participants of the user survey for their valuable feedback.

9. REFERENCES

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

[2] C. Browne and F. Maire. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16, 2010.

[3] Entertainment Software Association. Essential facts about the computer and video game industry, 2012. Retrieved December 12, 2012 from http://www.theesa.com/facts/pdfs/ESA_EF_2012.pdf.

[4] E. J. Hastings, R. K. Guha, and K. O. Stanley. Evolving content in the galactic arms race video game. In *Proceedings of IEEE Conference on Computational Intelligence and Games*, pages 241–248, 2009.

[5] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, 1992.

[6] 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*, 190(2):310–327, 2008.

[7] J. Lehman and K. O. Stanley. Revising the evolutionary computation abstraction: Minimal criteria novelty search. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2010.

[8] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223, 2011.

[9] A. Liapis, G. Yannakakis, and J. Togelius. Limitations of choice-based interactive evolution for game level design. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.

[10] A. Liapis, G. N. Yannakakis, and J. Togelius. Neuroevolutionary constrained optimization for content creation. In *Proceedings of IEEE Conference on Computational Intelligence and Games*, pages 71–78, 2011.

[11] 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*, 4(3):213–228, 2012.

[12] T. Lubart. How can computers be partners in the creative process: classification and commentary on the special issue. *International Journal of Human-Computer Studies*, 63(4-5):365–369, 2005.

[13] Z. Michalewicz. Do not kill unfeasible individuals. In *Proceedings of the Fourth Intelligent Information Systems Workshop*, pages 110–123, 1995.

[14] Omitted for the purposes of the double-blind review.

[15] J. F. Reintjes. *Numerical control: making a new technology.* Oxford University Press, Inc., 1991.

[16] M. Schoenauer and Z. Michalewicz. Evolutionary computation at the edge of feasibility. In *Proceedings of the 4th Parallel Problem Solving from Nature*, pages 245–254, 1996.

[17] J. Secretan, N. Beato, D. B. D'Ambrosio, A. Rodriguez, A. Campbell, J. T. Folsom-Kovarik, and K. O. Stanley. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, 19(3):373–403, Sept. 2011.

[18] N. Sorenson, P. Pasquier, and S. DiPaola. A generic approach to challenge modeling for the procedural creation of video game levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):229–244, 2011.

[19] 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*, pages 252–259. IEEE, 2007.

[20] 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*, (99), 2011.

[21] G. N. Yannakakis. Game AI revisited. In *Proceedings of ACM Computing Frontiers Conference*, 2012.

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