Sentient World: Human-Based Procedural Cartography

An Experiment in Interactive Sketching and Iterative Refining

Antonios Liapis[1], Georgios N. Yannakakis[1,2], and Julian Togelius[1]

[1] Center for Computer Games Research
IT University of Copenhagen
Rued Langaards Vej 7
Copenhagen, Denmark
[2] Department of Digital Games
University of Malta
Msida, Malta

Abstract. This paper presents a first step towards a computer-assisted design tool for the creation of game maps. The tool, named *Sentient World*, allows the designer to draw a rough terrain sketch, adding extra levels of detail through stochastic and gradient search. Novelty search generates a number of dissimilar artificial neural networks that are trained to approximate a designer's sketch and provide maps of higher resolution back to the designer. As the procedurally generated maps are presented to the designer (to accept, reject, or edit) the terrain sketches are iteratively refined into complete high resolution maps which may diverge from initial designer concepts. Results obtained on a number of test maps show that novelty search is beneficial for introducing divergent content to the designer without reducing the speed of iterative map refinement.

1 Introduction

In order to address the increasing time and resource requirements of content creation, more and more game companies use algorithmic means to either mass-generate trivial game content such as trees[3] and rocks [1] or to reduce designer effort by automating the mechanizable aspects of content creation, such as feasibility checking. The procedural generation of terrain is applied — to different extents — in many game titles to create the vast landmass of the game's virtual environment. Given the different constraints stemming from the gameworld's theme, mechanics and quests, designers prefer to maintain a level of control over the generated terrain. In most contemporary tools, this equates to manually editing the terrain after it has been (randomly) generated.

To address the requirement for designer control over generated content with minimal investment in human effort, this paper presents a first step towards a computer-assisted design tool for the creation of game maps. This tool, named *Sentient World*, allows a designer to progressively add details to a rough sketch through the process of *iterative refining*. Iterative refining is accomplished by artificial neural networks (ANNs) trained via gradient search to conform to low-resolution sketches submitted by the designer; the infinite resolution potential of ANNs is then used to create higher-resolution

[3] www.speedtree.com

maps which are submitted back to the designer to accept, reject, or edit. The iterative refining process is enhanced by neuroevolutionary novelty search which increases diversity in the pool of networks. Results obtained on a number of test maps show that the coupling of gradient and novelty search introduces divergent content without a significant computational overhead.

This is the first attempt, to the authors' knowledge, to combine novelty with gradient search via backpropagation in order to increase the representational power of ANNs without the uncontrollability often attributed to stochastic search algorithms. Additionally, the paper introduces the concept of iterative refining, where a human and a computer collaboratively add details to a rough concept sketch. While Sentient World is not tested with human designers in this paper, iterative refining is shown to secure the authorial control of human users as it largely conforms to initial designer sketches.

2 Related Work

Most techniques used for generating game terrain, both in the game industry and within academia, use fractals and noise to generate heightmaps [2]; however, there have been several attempts at controlling the generated artifacts. A popular technique allows a designer to specify real-world examples of desired terrain: Ong et al. [3] evaluate heightmaps generated via evolved terrain transformations based on their conformity to example terrains in a database, Li et al. [4] generate landscapes by combining different regions (hills, plains, etc.) selected from a database by a support vector machine trained to differentiate between terrain types and Zhou et al. [5] allow the user to sketch terrain features which are algorithmically refined based on real-world digital elevation models. Such example-based approaches, however, are limited by their corpus of examples, and fail to generate landscapes that do not conform to earth-like geologies and physics. Moving away from the real world, Ashlock et al. [6] rely on designer-specified desirable elevation models, and evaluate heightmaps generated via evolved L-systems based on the RMS error between generated and desired heights. Sentient World is similar to this method in that it relies designer-specified desirable elevation models and uses an error function for evaluating map fidelity; however, the ANN representation used in this paper lends itself better to infinite resolutions and does not suffer from L-systems' poor locality. Other projects try to increase designer control by limiting — but not eliminating — the randomness of the tool. SketchaWorld [7] coins the term "interactive procedural sketching" and allows designers to paint ecotopes on a tile-based grid; the tiles are transformed into detailed 3D representations of mountains and hills using fractal noise and grid interpolation. SketchaWorld provides some authorial control, with randomness being limited to tile-size chunks with specific properties. For even more control, Gain et al. [8] allow users to sketch a freeform terrain feature via its silhouette and boundary, generating a 3D terrain feature via interpolation, deformation and noise; the generated artifacts are faithful to the designer sketch, but require very precise specifications from the designer. Sentient World instead allows for very coarse sketches which are iteratively refined. Finally, control can be asserted via the behavior of terraforming agents, which interact with each other and the world to generate virtual

landscapes [9]; the sheer number of parameters controlling agent behavior makes such a tool cumbersome, as it requires trial and error to discern the impact of each parameter.

Evolutionary art has often focused on the automatic generation of artifacts, but designer intention is usually accommodated via interactive evolution [10]. Interactive evolution does not inherently have a mechanism for designers to specify aesthetic criteria which must be satisfied and is thus likely to create unwanted content; additionally, interactive evolution is likely to evolve artifacts very dissimilar to those chosen by the designer. To provide some direction to evolution and develop the ability to appreciate art, some researchers have used ANNs to evaluate generated content. Pre-training the network to simulate user ratings in a collection of generated content [11], to differentiate between different artists [12] or between human-authored and generated images [13], researchers attempt to create artificial art critics [13] capable of automatically evaluating generated content. The Sentient World tool has dissimilar aims in that it does not attempt to appreciate the designer's work but to conform to it. For that purpose, it uses ANNs to enforce the designer's constraints to its generated artifacts, ensuring that authorial control is maintained. As a computer-assisted design tool, Sentient World aims to accommodate its human designer more than it intends to completely automate the design process.

3 Methodology

The Sentient World tool is geared towards the *iterative refining* of maps, illustrated in Fig. 1. A user manually draws a low-resolution map; the height data from this map are used to train a number of neural networks previously optimized towards novelty via neuroevolution. Once the networks' training (via gradient-search) is completed on the user-provided data, each ANN generates a map of higher detail which is presented to the user. The user can accept or manually edit the detailed maps, and resubmit them for further refining; the process terminates once the designer is content with their final map. The number of maps presented to the user is limited to eight in this study — despite the fact that evolution runs on a larger population — in order to reduce both the training time of ANNs and the cognitive load on the designer when inspecting the detailed maps.

3.1 Representation

The maps generated by the Sentient World tool consist of a number of tiles, with each tile designating a specific *height zone*. The number of tiles (also termed *resolution*) and the number of height zones are interconnected and determine the *level of detail* (LOD) of the sketch (Fig. 2). A map sketch of any LOD can be encoded by a multi-layer ANN using a sigmoid activation function for all its nodes. The map is represented by an ANN in the following fashion: the normalized x, y coordinates of each tile's midpoint (red points in Fig. 3b) are used as input of the ANN, with the output being the tile's height value h. The output h belongs to a height zone i if $h \in [l_i, u_i)$, where l_i the zone's *lower bound* and u_i its *upper bound* (see Fig. 2b).

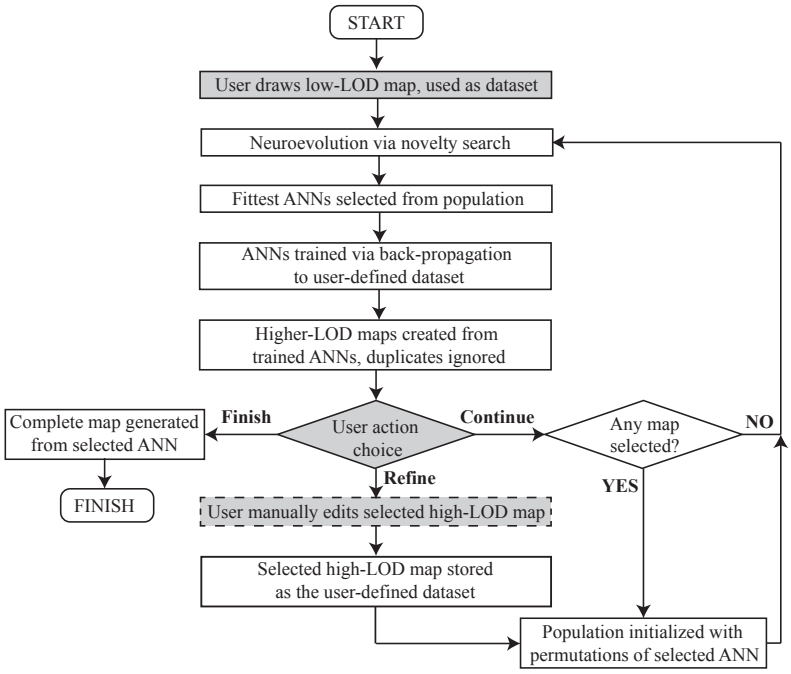
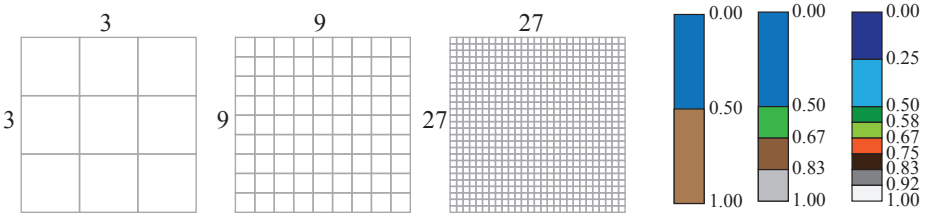


Fig. 1: An outline of the design process via the Sentient World tool, showing the different options for interaction (Finish, Continue and Refine) available to the user. Gray boxes represent user actions.

3.2 Iterative Refining

The key contribution of the Sentient World tool to existing paradigms of computer-assisted design is the process of *iterative refining* which allows the user, through interaction with the tool, to add an increasing number of details to a rough sketch. The process of iterative refining is currently accomplished through the training of multiple ANNs to conform with the rough sketch provided by the user. In order to increase diversity in the refined sketches and increase the ANNs' predicting abilities, a short evolutionary run optimizes these networks towards novelty and larger topologies.

ANN training: Iterative refining is accomplished through the training of multiple ANNs to approximate the patterns of the user-provided low-resolution sketch. In order to train these ANNs, the user sketch is converted to a dataset of input-output pairs. Inputs are the normalized x, y coordinates of the tile's midpoint (red points in Fig. 3b) and the desired output d is the tile's height zone average ($d = \frac{l_i+u_i}{2}$; where i is the tile's height zone and l_i, u_i are the zone's lower and upper bounds, respectively). The error e of the network, for actual output a and desired output d, is calculated as $e = \frac{1}{2}(d-a)^2$. Each network is trained via backpropagation [14] to minimize errors of the entire dataset, and training terminates either once all output values are within the



(a) Resolution LODs (b) Height zone LODs

Fig. 2: The Levels of Detail (LOD) used for the map sketches (from left to right: LOD of 1, 2 and 3), in terms of grid resolution and height zones. Numbers in Fig. 2a refer to the number of tiles available in each row and column for that LOD, while numbers in Fig. 2b refer to the lower and upper bounds (l_i, u_i) of each corresponding height zone.

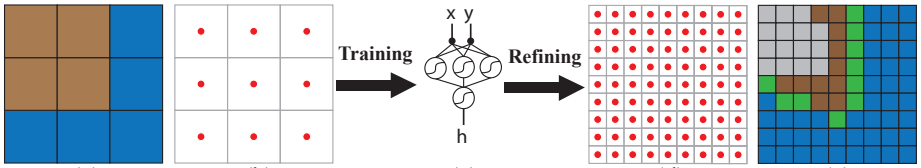


Fig. 3: An example of the iterative refining process. The initial sketch at LOD=1 drawn by the user (a) is used to create a dataset (b) using the height zone average (0.25 for water, 0.75 for land) at each tile's midpoint (red points). The dataset is used to train an ANN (c). Once training is complete, the ANN outputs the height values from the coordinates of a more detailed map (d) which are encoded into height zones of LOD=2, generating the refined map (e).

desirable height zones or after 10^5 epochs. Back-propagation is carried out with non-batch weight updates and a learning rate of 0.1. Once training is complete, the ANN is used to generate a more detailed map, increasing both the resolution LOD and the height zone LOD by one step; thus the network has a larger number of coordinates for inputs, while its h outputs correspond to more precise height zones (see Fig. 3e).

Neuroevolution: As the maps' resolution increases, the dataset of input-output pairs becomes more complex and requires a larger network to approximate. Additionally, as the user is presented with various detailed map suggestions during the iterative refining step, varying the topology and initial weights of the networks prior to training is likely to create more variation in the final results. For these two reasons, a short evolutionary run optimizes the ANNs towards novelty [15]. Evolution is carried out via neuroevolution of augmenting topologies (NEAT), which has a chance of increasing the number of layers, the number of nodes, and the number of links of the neural networks in the population [16]. Following the novelty search paradigm [15], evolution optimizes networks towards maximizing the objective function ρ, which corresponds to the average distance of the k most similar networks in the population and in an archive of novel individuals.

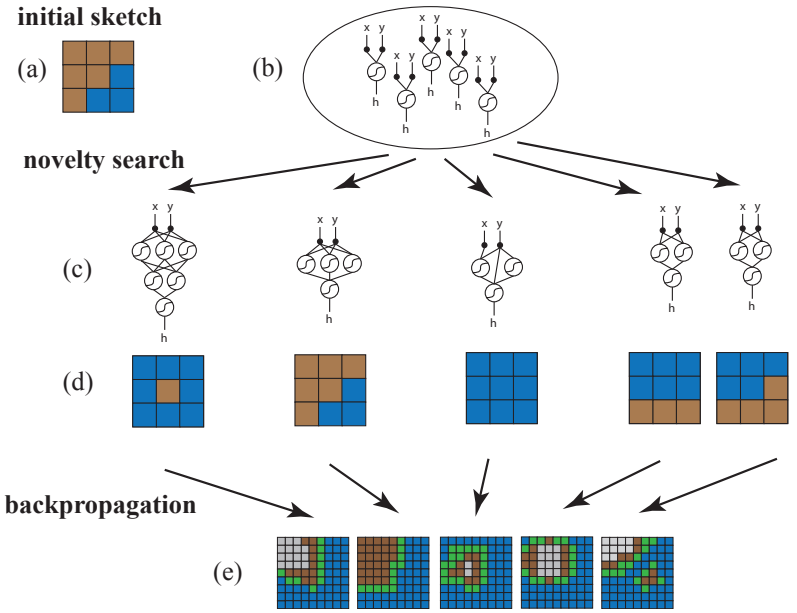


Fig. 4: A visualization of the impact of novelty search: an initial population (b) of similar, simple networks are evolved towards larger topologies (c) and dissimilar phenotypes (d), expanding the expressivity of generated artifacts. Once evolution sufficiently explores the search space, gradient search constricts expressivity towards artifacts that conform to the user-defined dataset of the initial sketch (a). The final, trained ANNs generate more diverse refined maps (e).

The archive stores the highest scoring individuals in the population (in terms of ρ) and is reset at the start of every run of the evolutionary algorithm. The fitness score $\rho(i)$ for individual i is calculated as:

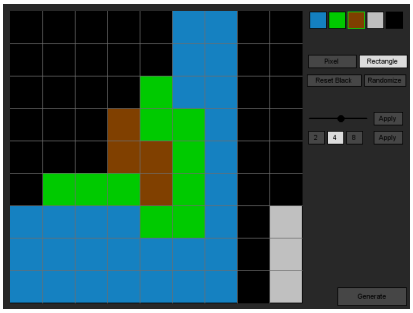
$$\rho(i) = \frac{1}{k} \sum_{j=1}^{k} dist(i, \mu_j) \tag{1}$$

where μ_j is the j-th-nearest neighbor of i (within the population and in the archive of novel individuals). Distance $dist(i, j)$ between networks i and j, is calculated as:

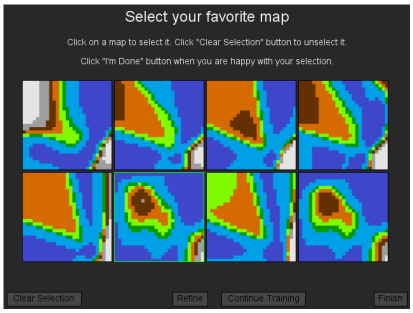
$$dist(i, j) = \frac{1}{T} \sum_{t=1}^{T} |h_i(t) - h_j(t)| \tag{2}$$

where T is the number of tiles of the encoded map on the same resolution, i.e. (c) in Fig. 4 and $h_i(t)$ is the h value at tile t's midpoint of the map encoded by network i.

Evolution is carried out for 20 generations on a population of 20 individuals, with the 5 fittest networks per generation stored in an archive of novel individuals and the closest 5 individuals considered when evaluating ρ. If no prior refining has occurred

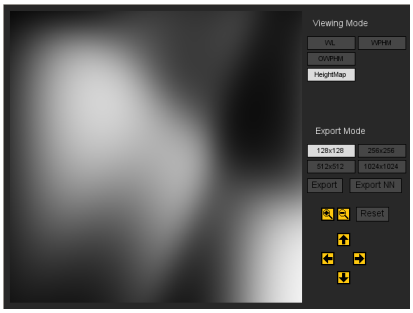


(a) Sketching interface while the user draws a new map sketch (with LOD of 2).



(b) Selection interface while the user selects a higher resolution map sketch (with LOD of 3) to refine the sketch from Fig. 5a.

(c) Sketching interface while the user manually refines the selected map from Fig. 5b.



(d) The full resolution of selected map from Fig. 5b, displayed as a heightmap.

Fig. 5: The User Interface for the Sentient World tool.

during the current session, the initial population in the evolutionary run consists of fully connected networks with randomly initialized weights and one hidden layer with four nodes. If a map and its encoding network has already been selected during previous refining steps, the initial population in the evolutionary run consists of mutations of the selected network, thus preserving its more elaborate topology. In order to bypass the problem of recombining networks of different topologies, evolution takes place only via mutation by adding a new node (10% chance) or a new link (15% chance) to the network, or otherwise modifying the weight of one randomly selected link. The selection of individuals for mutation is made via a fitness-proportionate roulette-wheel scheme. Once evolution is terminated, the eight fittest networks are selected and trained using backpropagation, as described above (see Fig. 4).

3.3 User Interface

The Sentient World generative tool aims to assist the user both in the generation and in the refinement of terrain models; the former is accomplished through a simple map

editor and the latter through the presentation of maps of higher detail. The map editor screen (Fig. 5a) allows the user to paint the map's tiles using brushes for different height zones. In addition to the height zones in Fig. 2b, the user can designate black tiles in the map, which act as wildcards and can be of any height. Black tiles are not included in the dataset for training the ANNs in the iterative refining process.

The map selection screen (Fig. 5b) allows the user to inspect the refined maps generated according to Section 3.2. The interface allows up to eight maps to be shown, although identical maps are omitted. The user may select a single map among presented ones, in which case the following actions become available:

Continue which re-runs the generative algorithms on the current level of detail, with an initial population seeded by the ANN of the selected map.

Refine which runs the generative algorithms on the next level of detail, using the height data of the selected map as the training dataset.

Edit & Refine which allows the user to load the selected map in the map editor and make manual adjustments; the modified map is used as the training dataset to generate the maps of the next level of detail.

Finish which uses the ANN of the selected map to generate the full resolution heightmap (Fig. 5d), allowing for further calculations and for exporting to a file.

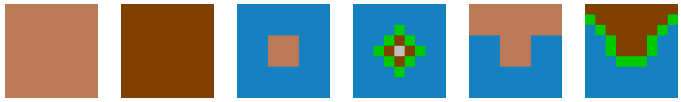
If the user selects no map among those presented, they have the option to re-attempt the map generation process (on the current level of detail) with a new initial population. The available user actions are also shown in Fig. 1; manual editing is an optional component to the process of refining, and appears in a dotted outline.

4 Experiments

To evaluate the potential of the iterative refinement approach and the efficacy of the proposed method, a number of sample maps are refined through the algorithm described in Section 3.2. These sample maps (shown in Fig. 6) have three distinct patterns — i.e. Land (L), Island (I) and Shore (S) — on two LODs. The maps were selected for their diversity — e.g. the patterns in map L1 are much simpler and easier to learn than those of map I2. In a simulated run of Sentient World, the above maps are refined by eight ANNs, since that is the number of presented maps in the Sentient World interface. The impact of gradient and novelty search is tested via two experiments: in the first, backpropagation (BP) is used to train eight randomly initialized fully-connected ANNs with a hidden layer of four nodes. In the second, backpropagation is used to train the eight fittest ANNs evolved via novelty search from a population of 20 ANNs for 20 generations; the initial population's ANNs has the same topology as the randomly initialized ANNs of the first experiment.

The performance measures considered in this study include the *runtime*, derived from an Intel i7 at 2.10GHz with 8 GB of RAM, and the *average distance* between the refined maps; significance is tested through standard t-tests (significance is 5% in this paper). Average distance $\bar{d}$ is calculated as:

$$\bar{d} = \frac{1}{P(P-1)} \sum_{i=1}^{P} \sum_{\substack{j=1 \\ j \neq i}}^{P} dist(i,j) \tag{3}$$



(a) L1 (b) L2 (c) I1 (d) I2 (e) S1 (f) S2

Fig. 6: The six initial maps which will be refined by the Sentient World tool. Maps L1, I1, S1 have LOD=1 while maps L2, I2, S2 have a similar form to their respective coarse counterparts, but with LOD=2.

Template Map	BP		BP with novelty search		
	ANN time (s)	Avg. Distance	Evolution time (s)	ANN time (s)	Avg. Distance
L1	0.14	0.103 (0.023)	0.15	0.16	0.122 (0.031)
L2	0.23	0.013 (0.002)	0.33	0.58	0.028 (0.005)
I1	7.89	0.036 (0.008)	0.08	4.18	0.036 (0.009)
I2	1183.15	0.046 (0.004)	0.27	684.08	0.046 (0.005)
S1	6.81	0.012 (0.006)	0.15	2.67	0.029 (0.019)
S2	569.25	0.016 (0.004)	0.20	680.86	0.031 (0.004)

Table 1: Comparison of the refinement processes for different template maps, using backpropagation (BP) with and without novelty search — i.e. BP trained on random ANNs vs. BP trained on initial ANNs guided by novelty search. Running times of the ANN training (ANN time) and evolution (Evolution time) and the average distance between maps are the performance measures considered.

where P is the number of presented maps (P=8 in this paper) and $dist(i, j)$ is the distance metric of (2) but calculated on the refined maps, i.e. see Fig. 4e.

The results of the different maps' refinement for the two approaches are shown in Table 1 containing the mean values collected across 20 individual runs, with standard deviation among runs shown in parentheses. Figure 7 displays the refined maps encoded by the trained ANNs of the most successful run in terms of average distance. We observe that for L and S patterns novelty search succeeds in significantly increasing the diversity of generated maps. Inspecting the most successful artifacts in Fig. 7, backpropagation combined with novelty search creates far more visually interesting and complex maps for S1 and L2, compared to the repetitive patterns when applied on simpler networks. For the Island patterns (I1, I2), backpropagation with and without novelty search generates maps which conform equally well to the user-defined dataset; however, novelty search does not enhance the diversity of these particular map patterns. With respect to computational time, the larger topology of networks evolved via NEAT increases the training time for backpropagation in the L and S patterns. Small networks appear able to easily encode the simple patterns in the L and S datasets; backpropagation on random small networks can therefore quickly learn such patterns, but creates visually uninteresting results. On the other hand, there is a surprising decrease in the computational time required to train the larger evolved networks for the Island patterns, compared to the training time for random small networks. While not necessarily creating more diverse results, novelty search enhances backpropagation for such maps since

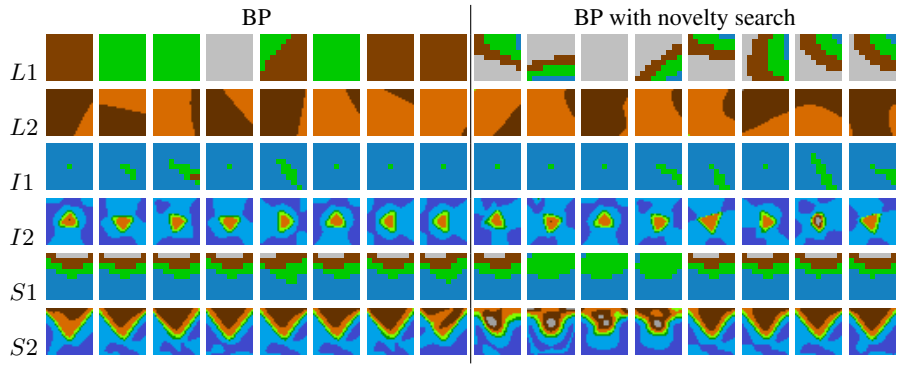


Fig. 7: Comparison of the eight refined maps trained via backpropagation (BP) with and without novelty search, for each template map. The maps are collected from the run with the highest average distance.

the larger networks are able to learn these complex patterns faster than the small random networks. The presented sample maps showcase that novelty search can contribute — with minimal computational overhead compared to that of backpropagation — to faster training and more diverse results, although one often at the expense of the other.

5 Discussion

The combination of stochastic and gradient search for the generation of novel maps conforming to designer intentions has shown promising results on the sample maps used to evaluate the proposed *iterative refinement* algorithm. The most important future research step is to test the tool with human designers, by collecting usability metrics, questionnaires, or verbal feedback and by measuring the impact of back-propagation and novelty search. A concern for the presented approach is the required runtime between iterations of sketching and refining. In larger resolutions and complex patterns, the size of the dataset makes training slow (e.g. 20 minutes for sample map I2). It is not realistic for a computer-assisted design tool promoting human-machine dialog to require such long periods of inaction from the human user. Future work will address the issue by reducing the number of maximum epochs before training terminates, by training each network in its own thread running on parallel and by showing the refined maps while training is under way, allowing users to terminate training prematurely if they find the maps interesting. An alternative solution for reducing training time in high-detail sketches is to take advangage of problem decomposition: an ANN can be pretrained to conform to a low-detail version of the user-defined map before learning the high-resolution map patterns.

Although gradient search helps preserve designer intentions during the generative process, it requires “close” supervision which may not be appropriate in cases where the designer does not wish to specify all the map details. While designers often have specific ideas on the heightmap of their terrain, other properties such as temperature or humidity

are much more difficult to manually specify and would increase the cognitive load on the side of the designer. Future work will explore the unsupervised search of patterns via neuroevolutionary algorithms such as NEAT [16], in circumstances where the designer provides high-level specifications such as vegetation on areas of the map and the algorithm optimizes the underlying conditions (temperature, humidity, and soil consistency maps) for the satisfaction of those specifications. Combining neuroevolution with constrained optimization has been quite successful for the generation of content which satisfies strict design requirements [17, 18]. Additionally, the elevation patterns stored in the trained ANNs can be used as scaffolds [19] for generating complementary maps (such as vegetation or temperature maps) through the use of CPPNs [20].

The visual appearance of the final maps is limited to both the representation employed and the training process followed. The sigmoid functions used in the ANNs often generate very "smooth" landscapes, with rounded shorelines and smooth elevations. More interesting features could be added via noise, but the randomness would remove the controllable aspect of this tool. The use of other activation functions might create more interesting shapes, but such networks can only be trained through evolution, such as CPPN-NEAT [20]. Otherwise, fast and deterministic erosion algorithms [21] could be applied to the heightmaps for a more realistic appearance.

6 Conclusion

The framework presented in this paper is a first step towards a tool supporting and enhancing human creativity, which provides more designer control than most designer-assistive tools available in the literature. Preliminary results show that gradient search (via backpropagation) is able to satisfy most designer-imposed constraints, while evolution via novelty search can increase the networks' representational power and the diversity of the generated results. Future steps should address the computational demand in large datasets, and aim to increase the amount of world features generated (vegetation, cities) and reduce the requirements for designer control to more abstract goal specifications. While the Sentient World tool currently generates heightmaps for use as gamewords, minimal changes — such as increasing the number of network outputs to three for RGB formats — could allow it to become a tool for visual artists, where the human artist begins by creating a rough sketch with basic colors and through the iterative refining process generates a final image which can have endless resolution.

Acknowledgements

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

References

1. Dart, I.M., De Rossi, G., Togelius, J.: Speedrock: procedural rocks through grammars and evolution. In: Proceedings of the 2nd International Workshop on Procedural Content Generation in Games, ACM (2011)

2. Fournier, A., Fussell, D., Carpenter, L.: Computer rendering of stochastic models. Communications of the ACM **25**(6) (1982) 371–384
3. Ong, T.J., Saunders, R., Keyser, J., Leggett, J.J.: Terrain generation using genetic algorithms. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, ACM (2005) 1463–1470
4. Li, Q., Wang, G., Zhou, F., Tang, X., Yang, K.: Example-based realistic terrain generation. In: Advances in Artificial Reality and Tele-Existence. Volume 4282 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2006) 811–818
5. Zhou, H., Sun, J., Turk, G., Rehg, J.M.: Terrain synthesis from digital elevation models. IEEE Transactions on Visualization and Computer Graphics **13**(4) (2007) 834–848
6. Ashlock, D., Gent, S., Bryden, K.: Embryogenesis of artificial landscapes. In: Design by Evolution. Natural Computing Series. Springer Berlin Heidelberg (2008) 203–221
7. Smelik, R.M., Tutenel, T., de Kraker, K.J., Bidarra, R.: A declarative approach to procedural modeling of virtual worlds. Computers & Graphics **35**(2) (2011) 352–363
8. Gain, J., Marais, P., Straßer, W.: Terrain sketching. In: Proceedings of the 2009 symposium on Interactive 3D graphics and games, ACM (2009) 31–38
9. Doran, J., Parberry, I.: Controlled procedural terrain generation using software agents. IEEE Transactions on Computational Intelligence and AI in Games **2**(2) (2010) 111–119
10. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. Proceedings of the IEEE **89**(9) (2001) 1275–1296 Invited Paper.
11. Baluja, S., Pomerleau, D., Jochem, T.: Towards automated artificial evolution for computer-generated images. Musical networks (1999) 341–370
12. Machado, P., Romero, J., Santos, M.L., Cardoso, A., Manaris, B.Z.: Adaptive critics for evolutionary artists. In: Proceedings of EvoWorkshops: Applications of Evolutionary Computing. Volume 3005 of Lecture Notes in Computer Science., Springer (2004) 437–446
13. Machado, P., Romero, J., Manaris, B., Santos, A., Cardoso, A.: Power to the critics — A framework for the development of artificial art critics. In: Proceedings of the IJCAI Workshop on Creative Systems. (2003)
14. Rumelhart, D.: Backpropagation: theory, architectures, and applications. Lawrence Erlbaum (1995)
15. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search for novelty alone. Evolutionary Computation **19**(2) (2011) 189–223
16. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10**(2) (2002) 99–127
17. Liapis, A., Yannakakis, G.N., Togelius, J.: Neuroevolutionary constrained optimization for content creation. In: Proceedings of IEEE Conference on Computational Intelligence and Games. (2011) 71–78
18. Liapis, A., Yannakakis, G.N., Togelius, J.: Adapting models of visual aesthetics for personalized content creation. IEEE Transactions on Computational Intelligence and AI in Games **4**(3) (2012) 213–228
19. Hoover, A.K., Szerlip, P.A., Stanley, K.O.: Interactively evolving harmonies through functional scaffolding. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation. GECCO '11, ACM (2011) 387–394
20. Stanley, K.O.: Exploiting regularity without development. In: Proceedings of the AAAI Fall Symposium on Developmental Systems, AAAI Press (2006)
21. Olsen, J.: Realtime procedural terrain generation: Realtime synthesis of eroded fractal terrain for use in computer games. Technical report (2004)