EvoIsland: Interactive Evolution via an Island-Inspired Spatial User Interface Framework

Alexander Ivanov
University of Calgary
aaivanov@ucalgary.ca

Wesley Willett
University of Calgary
wesley.willett@ucalgary.ca

Christian Jacob
University of Calgary
cjacob@ucalgary.ca

ABSTRACT

We present EvoIsland, a scalable interactive evolutionary user interface framework inspired by the spatially isolated land masses seen on Earth. Our generalizable interaction system encourages creators to spatially explore a wide range of design possibilities through the combination, separation, and rearrangement of hexagonal tiles on a grid. As these tiles are grouped into island-like clusters, localized populations of designs form through an underlying evolutionary system. The interactions that take place within EvoIsland provide content creators with new ways to shape, display and assess populations in evolutionary systems that produce a wide range of solutions with visual phenotype outputs.

CCS CONCEPTS

• **Computing methodologies** → *Genetic algorithms; Neural networks;* • **Human-centered computing** → *Visualization.*

KEYWORDS

Interactive Evolutionary Systems, User Interfaces, Visualization.

ACM Reference format:

Alexander Ivanov, Wesley Willett, and Christian Jacob. In *Genetic and Evolutionary Computation Conference (GECCO '22), July 9-13, 2022, Boston, MA, USA*. ACM, New York, NY, USA, 8 pages.
https://doi.org/10.1145/3512290.3528722

1 INTRODUCTION

Over the centuries, as life flourished across the crust of the Earth, species that were unfit for their environment were eliminated through the process of evolution [11]. Meanwhile, shifting tectonic plates and fluctuating sea levels have caused the geographic boarders of land masses to change, sparking new interactions between distinct groups of species. With these geological and evolutionary processes in mind, we have created an interactive system for guiding evolutionary design. Our EvoIsland system, inspired by Earth's tectonic events, supports the rapid exploration and generation of design paths across a large search space of multiple branching solutions. Unlike common existing evolutionary user interface approaches [1, 38], EvoIsland enables viewers to continuously regroup and evolve multiple subpopulations of spatially distinct solutions through platform agnostic spatial interactions. To demonstrate the unique characteristics of our framework, we present a prototype multitouch application that employs the EvoIsland framework design to evolve the input vectors of a BigGAN generative adversarial network (GAN) [2]. We then compare our framework to existing evolutionary systems [17, 21, 35] and reflect on how EvoIsland supports the rapid interactive generation of multiple subpopulations of solutions at different scales.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '22, July 9–13, 2022, Boston, MA, USA
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9237-2/22/07. . . $15.00
https://doi.org/10.1145/3512290.3528722

2 INTERACTIVE EVOLUTIONARY SYSTEMS

Through selective breeding processes and the application of fitness optimization functions, evolutionary algorithms build upon Darwinist principles to iteratively generate solutions [15]. Evolutionary algorithms have demonstrated a capacity to generate a diverse set of solutions to design problems [25]. However, some tasks – such as generating works of art – have much more loosely defined goals which can be challenging to describe, leading to an underlying alignment problem between generative systems and humans [7]. Unlike purely computer generated approaches, interactive evolutionary systems prompt a human advisor to guide the generative process [13]. Through an implemented user interface, designers iteratively rank how fit each result is amongst a population in different evolutionary rounds, while an underlying evolutionary algorithm evolves the ranked results to generate new solutions over time. Some evolutionary interfaces have been shown to cause significant user fatigue due to the use of repetitive evaluation tasks [33]. Reducing the number of interactions required by interactive systems can reduce this user fatigue, but often limits the number of generated solutions presented at a given time.

2.1 User Interfaces for Evolutionary Systems

A common evolutionary user interface is the grid selection design, which is present across a variety of interactive systems, such as GAN input evolution [1, 38] and generative race track paths for video games [5] . The grid approach displays a 2D matrix of results on screen and prompts viewers to iteratively select one or more preferred solutions in distinct evolutionary rounds using a variety of scoring systems [17, 22, 28]. While grid-style systems can display many results simultaneously, user fatigue can increase with the number of ranking tasks required in each evolutionary round [33]. Due to this constraint, grid-style systems typically display fewer than 20 solutions at a time. In the fitness zone framework [21], viewers drag and position generated solutions into different spatial regions on screen in consecutive evolutionary rounds to rank their fitness. These regions may have discrete fitness values or include a continuous range of values that change when phenotypes are positioned along an axis. Viewers using tournament evaluation interfaces select preferred evolved solutions in a perfect binary tree of branching rounds [35]. As deeper tree depths are generated, filtered solutions compete in "quarter final" rounds, similar to how athletes play a sports match.

As an alternative approach for shaping the evolutionary paths of evolving agents, Mammen & Jacob invited designers to grow swarm data like a gardener by spatially adding nutrients to continuously sprouting designs within an evolving 3D scene [24]. The gardening analogy masks the complex operations taking place within the system's underlying code and breaks free from the ridged rounds used in traditional evaluation interfaces. In a unique tangible approach, the novel PETRI system uses a number of small cylindrical smart devices with phenotypes displayed on their tops [26]. By shaking the tangible units, colliding multiple devices, and performing other gestures, viewers can physically control solutions generated by the evolutionary system. While only nine PETRI prototype nodes were developed, the authors speculate that the approach may be particularly helpful for evolving separate subpopulations of generated solutions. Hyperinteractive evolutionary computation systems can invoke a more positive user experience, while producing higher quality designs by putting individual operations of an evolutionary system in the control of the user [4]. These systems have a reduced reliance on automated underlying functions and provide more creative control over the generation of results when compared to pure evaluation task-based systems.

2.2 Visualizing Evolutionary Data

With clearly traceable histories of evolving solutions and consistent genome encodings, evolutionary systems are particularly well suited for exploring new visualization techniques. In an early investigation of evolutionary data visualization best practices, Pohlheim compared a variety of simple visualization techniques for a range of datasets [29], allowing viewers to review the path of an evolutionary algorithm over time. New approaches for visualizing tree structures that occur in evolutionary algorithms have also been developed by Daida et al [10], condensing the evolution of large populations into a *small multiples* [36] style grid. Taking design cues from desktop media library interfaces, *EvoShelf* [12] encourages viewers to organize and scrub through thousands of individual phenotype images of evolved swarm data by moving a computer cursor from left to right. Additionally, EvoShelf displays the full evolutionary histories of the data through the use of star plots and a *ThemeRiver* visualization technique [16]. This overview+detail [8] approach to EvoShelf's user interface provides viewers with the ability to simultaneously examine datasets at multiple levels of granularity. More recently *ELICIT* [9], a general-purpose tool for exploring evolutionary algorithms, introduces a variety of techniques for visualizing the lineage of evolving solutions. Entering immersive environments, *EvoVersion* [20] uses stacks of 3D rings to visually encode evolutionary sessions, generations, and individual phenotype representations of the data itself. While retroactively visualizing generations of evolutionary populations can help viewers build an understanding of how a solution is achieved, interactive evolutionary systems often require viewers to continuously assess the current state of the evolutionary populations presented on screen. In developing our EvoIsland framework, we set out to incorporate visualization techniques into the interactive evolutionary process itself to allow viewers to quickly revisit and reflect on their previous selection choices as they evolve generated designs.

3 EVOISLAND FRAMEWORK DESIGN

The EvoIsland Framework features a generalizable interactive system of operators that support the rapid exploration of a wide range of outputs through spatial subpopulation management at different scales. In this next section, we provide an overview of the framework design and outline the *Island Operators* that serve as the system's foundation.

3.1 Overview

EvoIsland is a generalizable interactive design framework that enables viewers to rapidly create, divide, grow, and combine subpopulations of evolutionary data. The system supports both fine-tuned manipulation of individual genomes and higher-scale directed evolution across evolutionary populations using a hyperinteractive approach [4]. As viewers interact with evolutionary data, EvoIsland provides a visual overview of individual genome states and groupings of similar solution clusters to communicate underlying evolutionary processes.

The EvoIsland framework user interface is comprised of two primary elements: an evolutionary grid that snaps elements into a hexagonal pattern and evolutionary tiles that are placed onto the grid itself (Fig. 1). Each tile in the system contains a genome generated by an evolutionary algorithm and displays a visual phenotype representation of the genome. As viewers position tiles onto the grid, they can interactively and iteratively shape generated populations. Upon placement, a new genome is generated for the tile, taking into account the proxemic position and distances of neighboring tile genomes using a von Neumann neighborhood search [3]. Unlike four-sided rectangular grid systems, hexagonal grids support a search across six neighboring sides, providing viewers with more flexibility for generating related solutions. After adding multiple evolutionary tiles to the grid, interconnected island groupings begin to form that visually communicate distinct evolving subpopulations of solutions. The tile positions within the islands are then tracked and updated by the EvoIsland system as interactions take place, enabling unique operators that provide a means for directing results generated by an underlying evolutionary algorithm.

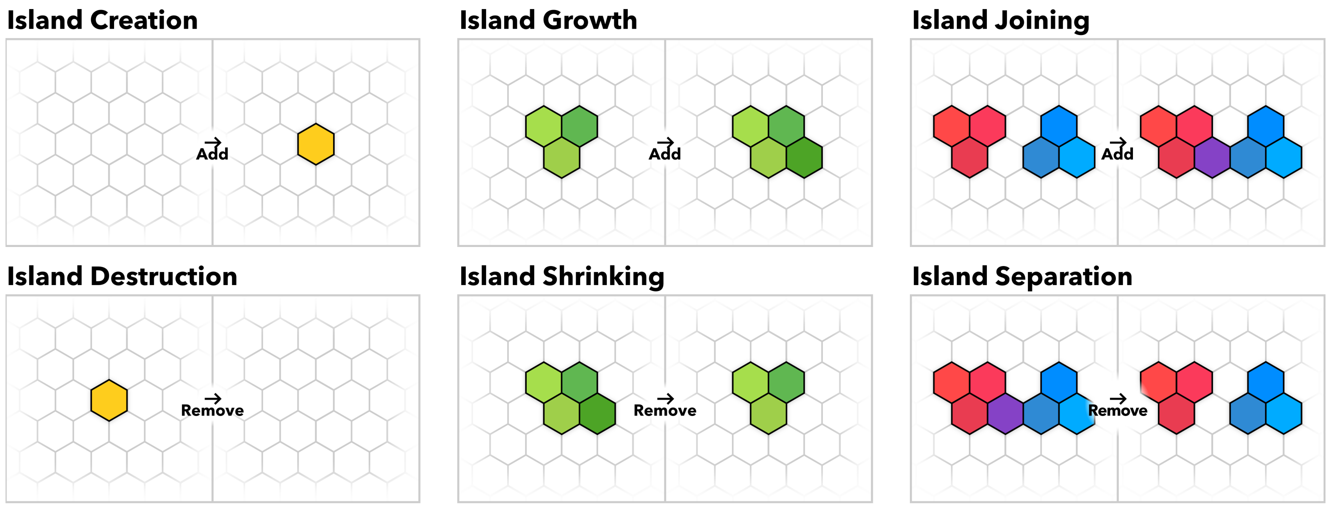


Figure 1: Underlying *Island Operators* in the EvoIsland Framework alter relationships between evolved subpopulations through the addition and removal of tiles on a grid.

3.2 Island Operators

Principal island operations performed on the EvoIsland grid alter relationships between distinct subpopulation groupings of tiles. In the EvoIsland Framework, we refer to these subpopulation groupings of tiles and the embedded genomes they encode as *Islands*. To communicate these operations, we refer to an example evolutionary system that evolves basic colors (Fig. 1) (for more details on implemented EvoIsland algorithms see Appx. A.1-3). The operations supported by the EvoIsland system include:

1. **Island Creation.** An island is created when the viewer adds an evolutionary tile to the grid and the new tile does not have a neighboring tile. The system generates a new genome for the tile that serves as a starting point for the subpopulation.
2. **Island Growth.** The growth of an island takes place when a tile is placed next to a single existing island formation. When growth occurs, the system's crossover+mutation function uses the relative distances of existing tiles on the island to generate a spatially aware genome for the new tile.
3. **Island Joining.** When the viewer adds a tile to the grid and two or more previously disconnected islands are connected, the populations of both islands combine to form a new, larger island. The crossover+mutation function uses the relative distances of existing tiles on both islands to generate a unifying genome for the new tile.
4. **Island Separation.** An island is separated when the viewer removes a tile that solely connects two parts of a larger island, splitting an island into multiple subpopulations. After a separation occurs, the two islands follow different evolutionary paths that stem from the isolated populations.
5. **Island Shrinking.** The shrinking of an island occurs when a tile is removed from an island and the remaining tiles in the formation continue to be interconnected after the removal.
6. **Island Destruction.** An island is destroyed when the last remaining tile in an island formation is removed.

4 EVOISLAND TOUCH

Employing the EvoIsland framework, we developed EvoIsland Touch (Fig. 2). The interactive evolutionary iPad app supports the above Island Operators through multi-touch gestures and builds upon the framework with additional support for rapid subpopulation editing and selection features.

4.1 System Architecture

While EvoIsland Touch can be customized to support different genetic algorithms that produce visual outputs, to demonstrate its capabilities we developed an evolutionary system that evolves the input vectors of a BigGAN network [2]. BigGAN builds upon image synthesis techniques [27] to generate a diverse range of images across 1000 ImageNet classes [14]. Its input is comprised of a *one-hot* vector to specify the class of image to generate, a noise vector for activating different image class compositions, and a truncation value which modifies the result variation. While these networks can produce millions of output images, it can be challenging to predict how changing the inputs of the model affects generated images. Our EvoIsland framework seeks to improve interactions with such complex systems by using a human-in-the-loop approach that supports viewers as they search through large solution spaces.

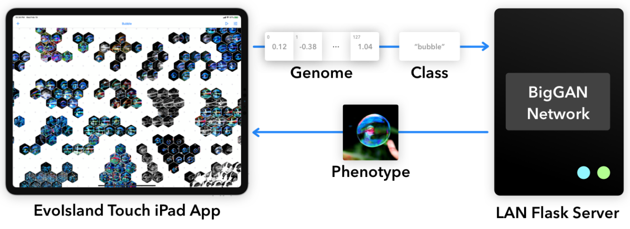


Figure 2: EvoIsland Touch System Architecture.

BigGAN Genome. To generate a wide range of images, the BigGAN neural network utilizes an input noise vector comprised of 128 unbounded float values. In EvoIsland Touch, this noise vector is embedded within the genome present in each evolutionary tile and is evolved through the interactive interface.

BigGAN Phenotype. In our EvoIsland Touch app, the phenotype of generated images corresponds to the class vector of the BigGAN model and can be changed to update the tile views.

BigGAN Crossover + Mutation. This BigGAN example for EvoIsland Touch uses *CrossoverMutate* as a crossover+mutation function that receives three inputs (Appx. A.1). The first input is a 3D *parent* array that contains all genomes in the existing island formation and their relative distances to the newly added genome. The second input is the *mutation radius* value between 0 and 1. Finally, the third input is a *gradient* value between 0 and 1. When the viewer creates a new island in the user interface, the function receives an empty parent array, then generates a new genome with random float values between 0 and 1 as a starting point for the evolving population. While the BigGAN neural network accepts input values outside of this range, values from the [0,1] interval produce images with relatively minimal artifacts. When the viewer adds to an existing island formation, the function selects genome values for the tile from the set of existing values on the island. Depending on the gradient value (Appx. A.2), genomes located within directly neighboring tiles have a higher probability of their values being selected for the crossover operation than tiles at further distances (Appx. A.2). We set this gradient value to 0.75, causing the genome values present in direct neighbor tiles to have a 75% probability of selection for the crossover function, while the tiles at further tile distances have increasingly less likelihoods of selection. Each genome value is then offset by a random amount within the mutation radius.

BigGAN Selection. Our BigGAN evolutionary system ranks the fitness of each generated design by its similarity to other genomes in the island subpopulation such that more unique genomes are favored (Appx. A.3). The *FindLeastFit* algorithm calculates an average noise vector across the genomes of all tiles on the island. It then determines the fitness of each tile on the island by taking the sum difference between the tile's genome noise vector values and the average noise vector values. Finally, the function returns the least fit genomes, based on this calculated fitness, for removal from the system. With this approach, as islands evolve over time in the app's auto-evolve mode (Fig. 6), the generated tiles on each island diverge towards an appearance that is less homogenous.

4.2 User Interface

The EvoIsland Touch (Fig. 2) user interface includes two primary components, an interactive *evolutionary scene* for directly manipulating evolutionary tiles and an upper toolbar for performing various genetic actions on the scene. Standard pinch to zoom and two finger pan gestures are present throughout the evolutionary scene for navigation and popover menus include additional operators for subpopulation interactions.

Evolutionary Scene. Here viewers have access to all tile operators outlined in the EvoIsland Framework and additional capabilities for more precise control of evolutionary subpopulations. Viewers can add, duplicate, select, edit, and remove evolutionary tiles to create and destroy islands. Tiles

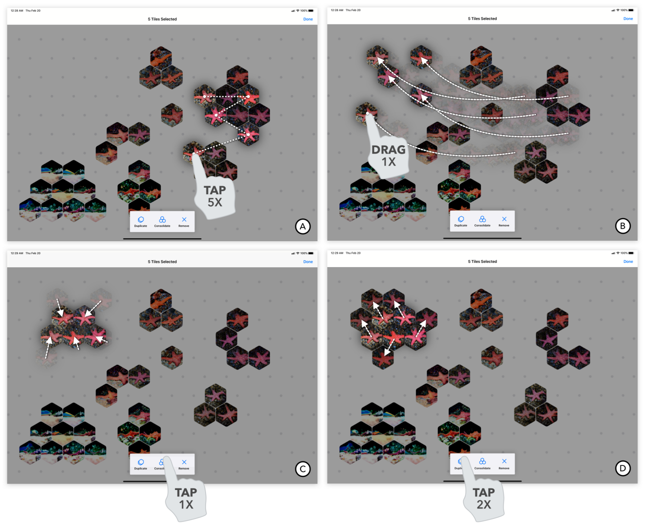


Figure 3: EvoIsland Touch Selection Mode. The viewer taps to select 5 tiles (A) and drags them to a new position (B). The viewer then taps *Consolidate* to merge the tiles into a unified island formation (C) and taps *Duplicate* twice to place copies of the selection onto the grid (D).

placed throughout the scene snap into formation on the evolutionary grid, visually indicated by light grey dots. Tapping on an empty space adds a new evolutionary tile to the scene and tapping on an existing tile removes it. Tiles may also be relocated to other grid positions by dragging. When a tile is added, an evolutionary algorithm generates a genome for the tile with a crossover+mutation algorithm, and then displays a visual phenotype image as its texture. Meanwhile, a long press on a tile presents a popover menu with the following options:

1. ***Select.*** Tapping this option activates *Selection Mode* (Fig. 3), where viewers can tap to select multiple tiles and drag to relocate them. When tiles are selected, they hover above the main grid for positioning and snap to the closest available location on the grid upon deselection. A lower selection toolbar provides the ability to *Consolidate* the selected tiles into a single clustered island formation, *Duplicate* the selected tiles onto the main grid, or *Remove* the selected tiles from the scene entirely.
2. ***Duplicate.*** Choosing this option also enters Selection Mode, but instantly duplicates the selected tile onto a nearby vacant space in the main evolutionary grid.
3. ***Edit.*** Picking this option presents the *Genome Editor* window for adjusting the noise vector values of a selected genome directly (Fig. 4). The top half of the window displays the current visual phenotype representation of the tile, while selecting and editing the genome values is reserved for the bottom half of the window. Viewers can drag the large horizontal slider bar in the center of the window to update the value of the currently selected genome trait index or scrub their finger along miniature vertical bars on the bottom portion of the screen to select a different parameter to adjust. Updates to the genome are then reflected in the upper phenotype view.

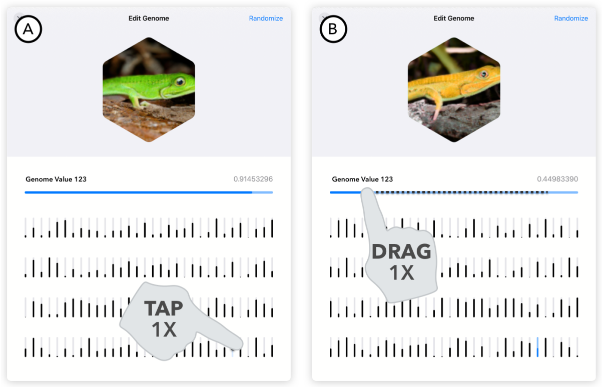


Figure 4: EvoIsland Touch Genome Editor. The viewer opens, scrubs, then taps on genome trait #123 (A). As they drag their finger along the middle horizontal bar, the phenotype view above reflects the changes (B).

Toolbar. The upper toolbar provides access to a variety of additional tile manipulation actions through popover menus:

1. **Add Tile Menu.** This menu contains two special tiles: a *grower tile* that automatically adds a set number of tiles to a connected island, and a *destroyer tile* that removes a specified number of least fit tiles from an island. Unlike many of the other features available in the toolbar, these actions operate on the population of a particular island, rather than affecting all islands on the grid.
2. **Phenotype Picker.** Viewers can select a new phenotype from a list of 1000 ImageNet classes. After picking a new phenotype value, each tile texture on the grid updates with a new image class, while the underlying genomes of each tile remain the same. This allows viewers to quickly generate entirely new sets of images that retain the underlying population and layout of the generated images (Fig. 5).
3. **Playback Toggle.** Tapping the playback toggle starts and stops *auto-evolve mode* (Fig. 6). When enabled, auto-evolve mode destroys the least fit tiles, then crossbreeds the most fit tiles found on each island over a specified number of evolutionary rounds.
4. **Options Menu.** The options menu includes parameters for adjusting auto-evolve mode. The *Growth Count* parameter affects the number of tiles to add to each island in each evolutionary round, while the *Death Count* parameter determines the number of unfit tiles to remove from each island in every auto evolving round. When auto-merge is enabled, auto-evolve mode will also permit the automatic placement of tiles that perform an *Island Joining* operation. Finally, the menu also includes controls for adjusting the mutation radius, prototype debug options, and a *Reset Map* button that removes all tiles from the grid.

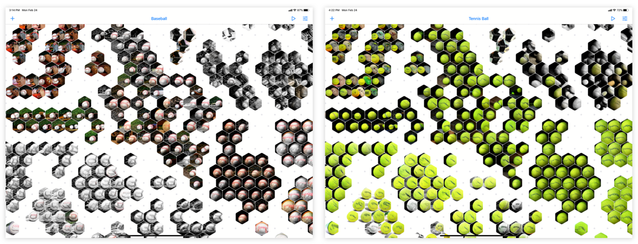
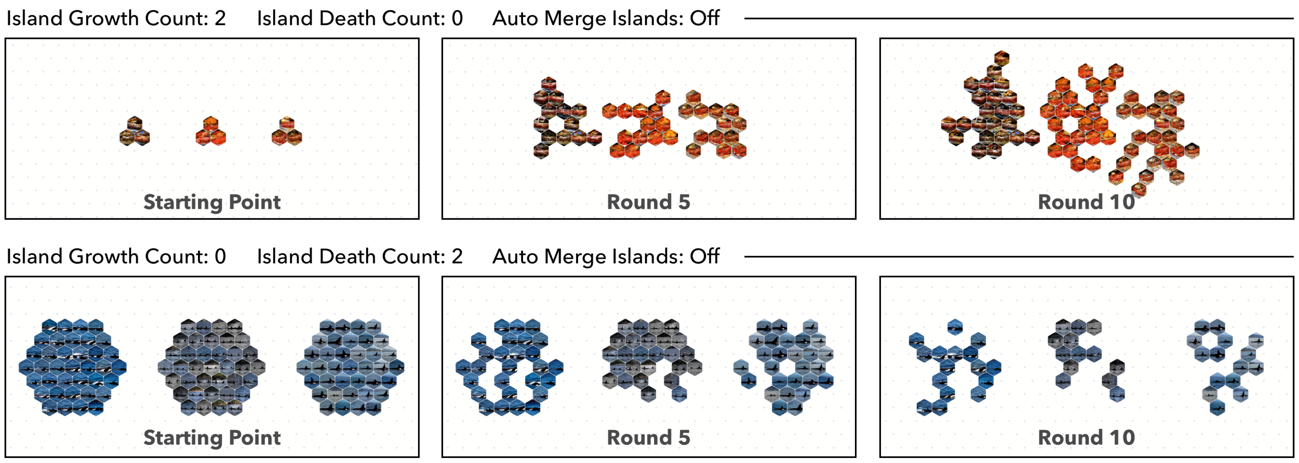


Figure 5: Phenotype Picker. After evolving images of baseballs (left) the viewer uses the Phenotype Picker to change the image class of the results to tennis balls.

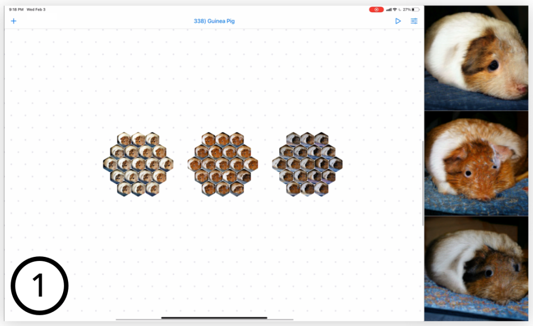
4.3 Demonstration

In Fig. 7 and the included reference video, we demonstrate the gamut of capabilities our EvoIsland framework provides through an HCI toolkit demonstration [23] of EvoIsland Touch. The example highlights the unique spatial subpopulation management workflows present in the EvoIsland framework and reveals how the system supports the interactive simultaneous search of multiple areas of the BigGAN latent design space. In addition to system screenshots, three additional phenotype images are displayed to the right of each frame to communicate changes more clearly in the evolving population.

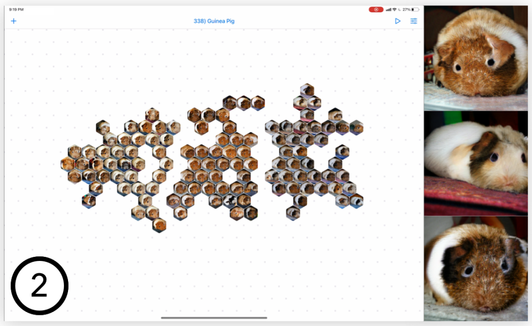

Island Growth Count: 2 Island Death Count: 0 Auto Merge Islands: Off

Island Growth Count: 0 Island Death Count: 2 Auto Merge Islands: Off

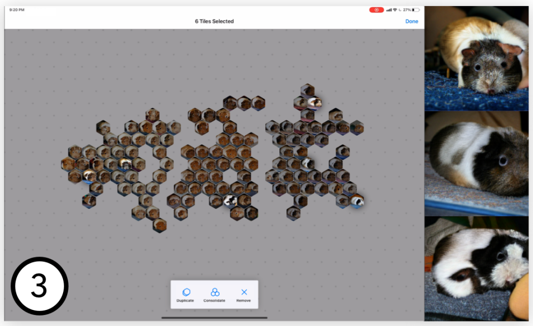
Figure 6: Auto-Evolve Mode in EvoIsland Touch. A *Growth Count* of 2 quickly increases the number of tiles in each island (above). *Death Count* set to 2 shrinks the islands rapidly by removing unfit results from each island (below).



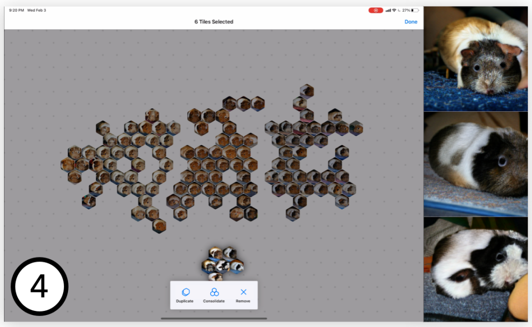
Pete is creating a website for his new pet shop and would like to find a unique guinea pig image to feature. He starts by selecting the guinea pig phenotype in EvoIsland Touch and taps repeatedly in 3 spirals to generate 3 separate islands.



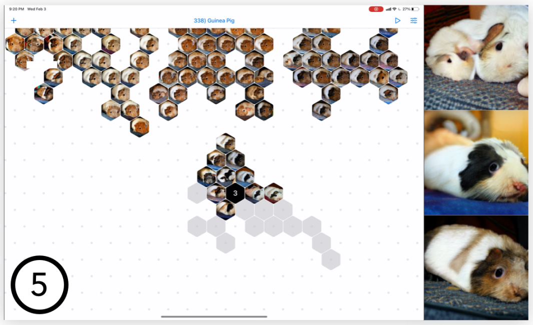
Seeking a highly unique design, he then sets and runs *Auto-Evolve Mode* with parameters to rapidly increase the number of tiles. The *Growth Count* is set to 10, *Death Count* set to 1, *Auto-Merge* is disabled, and *Mutation Radius* is set to 50%.



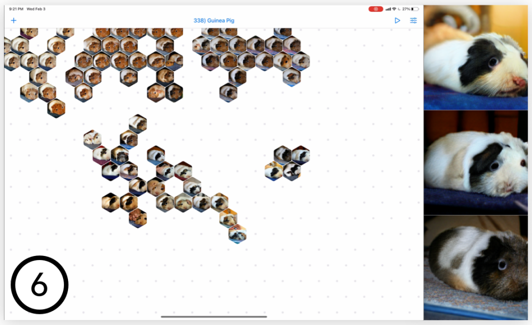
After the evolution, he notices some creatures have a fur pattern with a white stripe down the middle. He enters *Selection Mode* then taps to select his favorite designs.



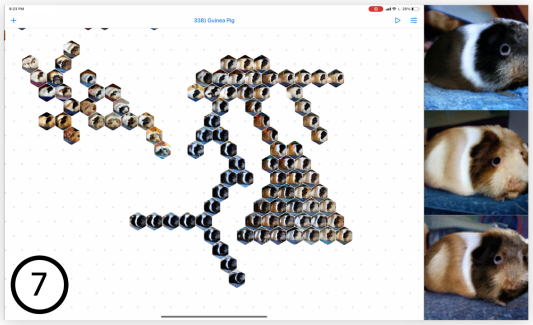
He then taps the *Consolidate* button in *Selection Mode* to combine the tiles into a new island formation.



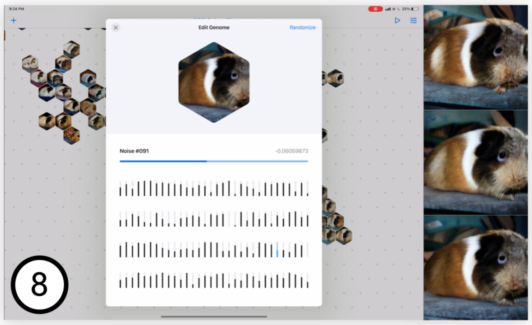
Next, Pete attaches a *Grower Tile* to the island to rapidly increase its size by 20 tiles.



He then drags four of his favorite designs into another new island formation, separating them from the other islands with higher design variation.



After reducing the *Mutation Radius* to 20% and iteratively tapping to create branching patterns that stem from the island, he discovers a creature with a dark brown head, white stripe in the middle, and light brown back.



Finally, Pete enters the *Genome Editor* to manually edit the genome traits of the guinea pig and removes the white spot on its back.

Figure 7: A demonstration of a pet shop website owner, Pete, evolving a guinea pig image in our BigGAN EvoIsland Touch application.

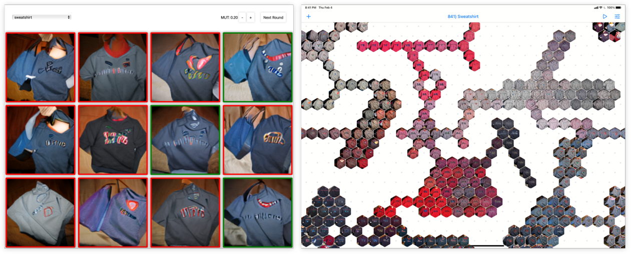


Figure 8: Evolving BigGAN sweatshirt images with our grid interface (left) and EvoIsland Touch (right).

5 COMPARISON WITH EXISTING SYSTEMS

We compare our EvoIsland Framework, as implemented through EvoIsland Touch, to common interactive evolutionary interfaces present in published systems. For a more direct comparison, we also developed a complementary twelve tile grid-style interface that resembles these common approaches [19] for the underlying BigGAN evolutionary system present in EvoIsland Touch (Fig. 8). In the grid interface, viewers may adjust the phenotype ImageNet class of the genome, change the mutation radius of the evolutionary algorithm, and click to highlight preferred designs before clicking the *Next Round* button to generate new results. Unlike our EvoIsland system, however, all parent values provided to the crossover+mutation algorithm have an equal influence on generated genomes, regardless of their spatial position on the grid. After exploring variations of phenotypes in the systems and guiding the networks towards target designs over hundreds of sessions, we contrast how interactions with evolving populations in EvoIsland differ from traditional grid-based interfaces.

Evolving Subpopulations. Evolutionary interfaces commonly support the ability to evolve a single global population towards a goal. After viewers select their preferred candidates in grid-style interfaces [5], or position phenotypes in the fitness zone [21] framework, a new generation replaces the entire population in the scene (for example, by fitness-proportionate selection). While the PETRI system supports multiple simultaneously evolving populations [26], the initial exploration only included nine physical tangible nodes, limiting subpopulation explorations at larger scales. EvoIsland supports the evolution of multiple larger subpopulations using spatial grouping techniques as viewers add and remove phenotype tiles from the grid. In our system, subpopulations can be quickly combined, split into multiple groups, or kept isolated throughout. In our experience with evolving BigGAN populations, we found having multiple subpopulations in EvoIsland enables the exploration of many distinct search spaces simultaneously. This provides additional precision over the combination of phenotypes from differing search areas. EvoIsland Touch further encourages subpopulation management with a variety of selection mode options for the positioning, duplication, and consolidation of disjoint tiles into new formations. Meanwhile, evolving BigGAN images via the traditional grid-style interface often led to each evolutionary round presenting results from a similar search space within a local minimum. While algorithmic adjustments and additional mutation controls can lead to wider search space explorations in each round, standard grid-style selection does not afford the same level of fine-tuned subpopulation control present in EvoIsland.

Proxemic Evolution. Although fitness zone systems [21] prompt viewers to interactively position phenotypes to rank their fitness, EvoIsland's spatially-aware evolution of tiles provides deeper hyperinteractive [4] control over crossover combinations. When a tile is added to an island, genome values in directly neighboring tiles have a higher probability of recombination within the crossover+mutation function. With the proxemic evolutionary approach in EvoIsland, viewers can spatially select genomes for breeding. It is also possible to perform side-by-side comparisons of identical evolving subpopulations through the duplication functions in EvoIsland Touch. This high level of control in EvoIsland contrasts starkly with our grid system, as the grid lacks an interface for directly controlling the precise crossover combination of selected parents in each evolutionary round. While other grid-style systems have been supplemented with user interface elements for ranking the fitness of results, such as star rating controls [5] or sliders [34], they traditionally do not have a means for directly selecting crossovers within their presented populations. When evolving BigGAN images in the grid-style system, it is challenging to predict and review parent-child relationships across evolutionary rounds.

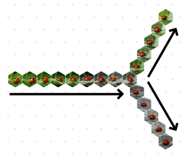
History of Evolutionary Paths Taken. Some systems enable viewers to reflect on and revisit histories of evolutionary actions. *MusicCube*, for example, provides historical access to previous selection choices by displaying the system's entire search space and highlighting viewer song rankings on a scatterplot [31]. Meanwhile, the *PicBreeder* system contains a separate view that visualizes evolved nodes in a graph [32]. The visualization of evolutionary histories using EvoVersion's stacked ring system also includes a method to review previous selections performed in grid-style or fitness zone interfaces [20]. EvoIsland systems provide an implicit history to past evolutionary operations by retaining selected results on a grid (although removed tiles cannot be reviewed). Rather than visualizing previous evolutionary operations in a separate view, EvoIsland displays selection histories within the evolutionary interface itself. The approach supports interactions with earlier stages of evolution and promotes the combination of results from multiple search spaces as viewers reposition tiles. When reviewing the screenshot of our traditional BigGAN grid-style interface in Fig. 8, it is challenging to approximate how far along the viewer is in the evolutionary process. The static layout and number of presented results in the grid remains constant throughout. In EvoIsland, the depth of explored search spaces is implicitly communicated as the number of tiles often increases over time.

6 DISCUSSION

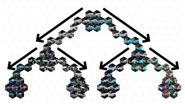
The affordances present in the EvoIsland platform permit a variety of unique interaction strategies for directing evolutionary processes. In our discussion, we highlight interaction strategies that support population explorations in EvoIsland and discuss potential opportunities for future development.

6.1 Evolution Patterns

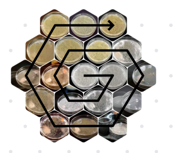
Using EvoIsland Touch, we found several unique interaction patterns that can support search space explorations including:



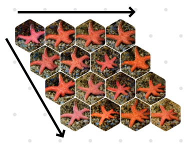
Branching. Iteratively positioning mutating tiles outwards in two or more directions diverges solutions from a starting position. Adding branches provides a rapid approach for exploring multiple possible paths of evolution simultaneously.



Trees. Recursively branching islands reveals a tree pattern with increasing variation at deeper levels. Trees provide an organized approach for exploring repeated child parent relationships.



Spirals. Spiralling outward from a center tile results in a cluster of related tiles with increasing variation from originals. Creating spirals provides a fast and space-efficient mechanism for exploring possible variations based on a small number of parents.



Gradients. Gradually adding repeating lines of tiles increases variation from the starting corner of the shape towards its adjacent corner. Gradient formations provide a compact mechanism for exploring compounding variations along two axes.

6.2 Subpopulation Management

Continuous Evolution. Unlike interactive evolutionary systems that present a set number of solutions in rigid ranked rounds of phenotype selection tasks, interacting with populations in EvoIsland is a continuous process that consists of adding and removing tiles on a hexagonal grid at will. Viewers can display an unbounded number of phenotypes and quickly revisit earlier generated solutions. While many solutions are displayed simultaneously, the coherent mental model of a grid map encourages viewers to choose how to spatially situate results and trace their progress. Through an adjustable viewport, viewers can also pan and zoom to focus on smaller segments of the grid.

Locking Controls. Some evolutionary systems include locking controls to prevent parts of a genome from evolving any further. In CG-GAN [38], locking controls prevent certain facial features from drastically changing as the generative adversarial network inputs evolve. In our EvoIsland Touch application, viewers can spatially place designs aside to save their state and duplicate preferred tile designs to increase their influence on the grid. In future versions of the system, additional locking mechanisms may be helpful for providing fine-tuned evolution or for preventing the removal of some tiles from the grid entirely.

Scaling to Larger Populations. Scaling the grid in EvoIsland Touch reveals different quantities and levels of phenotype details. From an overview perspective viewers can observe larger trends in populations, then zoom in for a closer examination of phenotypes. This grid scaling could be pushed even further to support dot-map style visualizations [37] of much larger numbers of phenotypes. While EvoIsland Touch includes multi-selection capabilities, multi-touch data visualization interaction techniques [30] could support larger populations.

6.3 Generalizable Design

The EvoIsland Framework can extend into computing mediums beyond the 2D touch interface in EvoIsland Touch. We also created an augmented reality (AR) iOS app where viewers can tap to add and remove evolutionary tiles on real-world surfaces to evolve parametric 3D models (Fig. 9). The prototype demonstrates the framework operating in a 3D situated spatial layout. As a platform-agnostic framework, we hope to see EvoIsland-inspired systems on a wider range of computing modalities in the future.

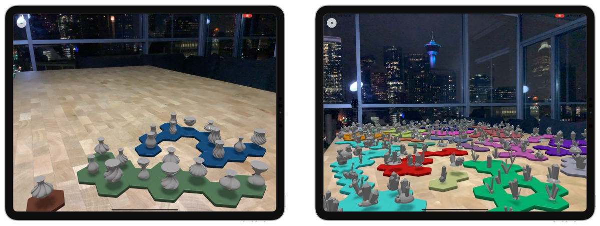


Figure 9: EvoIsland AR for parametric 3D design.

6.4 Evaluating Creativity Systems

Evaluating computationally creative systems is touted as 'Grand Challenge' in the field [6] as the definition of creativity is open to interpretation. Addressing this challenge, SPECS [18] proposes a three-step process where researchers to adopt a definition of creativity for their system, then derive experiments from the definition for evaluation. While we compare EvoIsland to existing approaches, a creative systems evaluation could reveal and better quantify additional characteristics about the EvoIsland system.

7 CONCLUSION

Through the spatial positioning of phenotype tiles, the EvoIsland Framework includes an expressive vocabulary of subpopulation management techniques for exploring evolutionary systems. As viewers repeatedly perform island operations, isolated landmasses on a grid produce disparate evolutionary results. EvoIsland Touch exemplifies the potential for production systems to further build upon the framework, with multi-selection controls and rapid evolution options for BigGAN images. Nonetheless, we suspect that branching, tree, spiral, and gradient patterns are only scratching the surface of how the framework can guide evolving populations. We finally encourage the community to consider adopting EvoIsland over traditional interfaces that can stifle the creative exploration of evolving populations.

ACKNOWLEDGMENTS

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) [RGPIN-2016-04564, RGPIN-2021-02492, RGPIN-2019-06267] and the Canada Research Chairs Program.

REFERENCES

[1] Bontrager, P., Lin, W., Togelius, J. and Risi, S. 2018. Deep Interactive Evolution. *International Conference on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART EvoStar '18)*. Springer, Cham. (Jan. 2018), 267–282.

[2] Brock, A., Donahue, J. and Simonyan, K. 2019. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *7th International Conference on Learning Representations (ICLR '19)*. (May 2019).

[3] Burks, A. 1969. *Von Neumann's Self-Reproducing Automata*. The University of Michigan.

[4] Bush, B.J. and Sayama, H. 2011. Hyperinteractive Evolutionary Computation. *IEEE Transactions on Evolutionary Computation*. 15, 3 (Jun. 2011), 424–433. DOI:https://doi.org/10.1109/TEVC.2010.2096539.

[5] Cardamone, L., Loiacono, D. and Lanzi, P.L. 2011. Interactive Evolution for the Procedural Generation of Tracks in a High-End Racing Game. *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11)*. ACM Press. (2011), 395–402. DOI:https://doi.org/10.1145/2001576.2001631.

[6] Cardoso, A., Veale, T. and Wiggins, G.A. 2009. Converging on the divergent: The history (and future) of the international joint workshops in computational creativity. *AI magazine*. 30, 3 (2009), 15.

[7] Christian, B. 2020. *The Alignment Problem: Machine Learning and Human Values*. W. W. Norton & Company.

[8] Cockburn, A., Karlson, A. and Bederson, B.B. 2009. A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM Computing Surveys (CSUR '09)*. Association for Computing Machinery. 41, 1 (Jan. 2009), 1–31. DOI:https://doi.org/10.1145/1456650.1456652.

[9] Cruz, A., Machado, P., Assunção, F. and Leitão, A. 2015. ELICIT: Evolutionary Computation Visualization. *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO Companion '15)*. ACM Press. (2015), 949–956. DOI:https://doi.org/10.1145/2739482.2768443.

[10] Daida, J.M., Hilss, A.M., Ward, D.J. and Long, S.L. 2005. Visualizing Tree Structures in Genetic Programming. *Genetic Programming and Evolvable Machines*. Springer. (Mar. 2005), 79–110.

[11] Darwin, C. 1964. *On the Origin of Species: A Facsimile of the First Edition*. Harvard University Press.

[12] Davison, T., Von Mammen, S. and Jacob, C. 2010. EvoShelf: A System for Managing and Exploring Evolutionary Data. *International Conference on Parallel Problem Solving from Nature (PPSN '10)*. Springer. 6239 LNCS, (2010), 310–319. DOI:https://doi.org/10.1007/978-3-642-15871-1_32.

[13] Dawkins, R. 2015. *The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design*. W. W. Norton & Company.

[14] Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li and Li Fei-Fei 2009. ImageNet: A Large-Scale Hierarchical Image Database. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*. IEEE. (Jun. 2009), 248–255. DOI:https://doi.org/10.1109/cvpr.2009.5206848.

[15] Fogel, L.J., Owens, A.J. and Walsh, M.J. 1966. Intelligent Decision-Making Through a Simulation of Evolution. *Behavioral Science*. 11, 4 (1966), 253–272. DOI:https://doi.org/https://doi.org/10.1002/bs.3830110403.

[16] Havre, S., Hetzler, B. and Nowell, L. 2000. ThemeRiver: Visualizing Theme Changes Over Time. *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS '00)*. IEEE. (2000), 115–123. DOI:https://doi.org/10.1109/infvis.2000.885098.

[17] Hua, H. 2012. Planning Meets Self-Organization: Integrating Interactive Evolutionary Computation with Cellular Automata for Urban Planning. *Frontiers of Architectural Research*. Elsevier. 1, 4 (Dec. 2012), 400–404. DOI:https://doi.org/10.1016/j.foar.2012.08.002.

[18] Jordanous, A. 2012. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation*. Springer. 4, 3 (2012), 246–279.

[19] Kelly, J. and Jacob, C. 2018. evoExplore: Multiscale Visualization of Evolutionary Histories in Virtual Reality. *International Conference on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART EvoStar '18)*. Springer, Cham. (2018), 112–127.

[20] Kelly, J. and Jacob, C. 2016. EvoVersion: Visualizing Evolutionary Histories. *2016 IEEE Congress on Evolutionary Computation (CEC '16)*. IEEE. (Jul. 2016), 814–821. DOI:https://doi.org/10.1109/CEC.2016.7743875.

[21] Khemka, N., Hushlak, G. and Jacob, C. 2009. Interactive Evolutionary Evaluation Through Spatial Partitioning of Fitness Zones. *Workshops on Applications of Evolutionary Computation (EvoWorkshops '09)*. Springer. 5484 LNCS, (2009), 432–441. DOI:https://doi.org/10.1007/978-3-642-01129-0_49.

[22] Kowaliw, T., Dorin, A. and McCormack, J. 2012. Promoting Creative Design in Interactive Evolutionary Computation. *IEEE Transactions on Evolutionary Computation*. IEEE. 16, 4 (Aug. 2012), 523–536. DOI:https://doi.org/10.1109/TEVC.2011.2166764.

[23] Ledo, D., Houben, S., Vermeulen, J., Marquardt, N., Oehlberg, L. and Greenberg, S. 2018. Evaluation Strategies for HCI Toolkit Research. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery. (Apr. 2018), 1–17. DOI:https://doi.org/10.1145/3173574.3173610.

[24] Von Mammen, S. and Jacob, C. 2007. Genetic Swarm Grammar Programming: Ecological Breeding Like a Gardener. *2007 IEEE Congress on Evolutionary Computation (CEC '07)*. IEEE. (2007), 851–858. DOI:https://doi.org/10.1109/CEC.2007.4424559.

[25] Miller, J., Job, D. and Vassilev, V. 2000. Principles in the Evolutionary Design of Digital Circuits—Part II. *Genetic Programming and Evolvable Machines*. Springer. 1, 3 (2000), 259–288. DOI:https://doi.org/10.1023/A:1010066330916.

[26] Mitchell, T., Bennett, P., Madgwick, S., Davies, E. and Tew, P. 2016. Tangible Interfaces for Interactive Evolutionary Computation. *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery. (May 2016), 2609–2616. DOI:https://doi.org/10.1145/2851581.2892405.

[27] Odena, A., Olah, C. and Shlens, J. 2017. Conditional Image Synthesis with Auxiliary Classifier Gans. *34th International Conference on Machine Learning (ICML '17)*. PMLR. 6, (Oct. 2017), 4043–4055.

[28] Peng, H., Hu, H., Chao, F., Zhou, C. and Li, J. 2016. Autonomous Robotic Choreography Creation via Semi-interactive Evolutionary Computation. *International Journal of Social Robotics*. Springer. 8, 5 (Nov. 2016), 649–661. DOI:https://doi.org/10.1007/s12369-016-0355-x.

[29] Pohlheim, H. 1999. Visualization of Evolutionary Algorithms - Set of Standard Techniques and Multidimensional Visualization. *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation (GECCO '99)*. ACM Press. (1999), 533–540.

[30] Sadana, R. and Stasko, J. 2016. Expanding Selection for Information Visualization Systems on Tablet Devices. *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces: Nature Meets Interactive Surfaces (ISS '16)*. ACM Press. (2016), 149–158. DOI:https://doi.org/10.1145/2992154.2992157.

[31] Saito, Y. and Itoh, T. 2011. MusiCube: A Visual Music Eecommendation System Featuring Interactive Evolutionary Computing. *In the Proceedings of the 2011 Visual Information Communication (VINCI '11)*. ACM Press. (New York, New York, USA, 2011), 1–6.

[32] Secretan, J., Beato, N., D'Ambrosio, D.B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J.T. and Stanley, K.O. 2011. Picbreeder: A Case Study in Collaborative Evolutionary Exploration of Design Space. *Evolutionary Computation*. IEEE. 19, 3 (Sep. 2011), 373–403. DOI:https://doi.org/10.1162/EVCO_a_00030.

[33] Takagi, H. 2001. Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation. *Proceedings of the IEEE*. 89, 9 (2001), 1275–1296. DOI:https://doi.org/10.1109/5.949485.

[34] Takenouchi, H., Tokumaru, M. and Muranaka, N. 2013. Interactive Evolutionary Computation Using a Tabu Search Algorithm. *IEICE Transactions on Information and Systems*. E96-D, 3 (2013), 673–680. DOI:https://doi.org/10.1587/transinf.E96.D.673.

[35] Takenouchi, H., Tokumaru, M. and Muranaka, N. 2012. Performance Evaluation of Interactive Evolutionary Computation with Tournament-Style Evaluation. *2012 IEEE Congress on Evolutionary Computation (CEC '12)*. IEEE. (Jun. 2012), 1–8. DOI:https://doi.org/10.1109/CEC.2012.6256128.

[36] Tufte, E.R. 1990. *Envisioning Information*. Graphics Press.

[37] Turner, E. and Allen, J.P. 2010. Issues in Depicting Population Change with Dot Maps. *Cartography and Geographic Information Science*. Taylor & Francis. 37, 3 (Jan. 2010), 189–197. DOI:https://doi.org/10.1559/152304010792194921.

[38] Zaltron, N., Zurlo, L. and Risi, S. 2020. CG-GAN: An Interactive Evolutionary GAN-Based Approach for Facial Composite Generation. *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*. AAAI Press. 34, 03 (Apr. 2020), 2544–2551. DOI:https://doi.org/10.1609/aaai.v34i03.5637.