Chapter 11
Mixed-initiative content creation

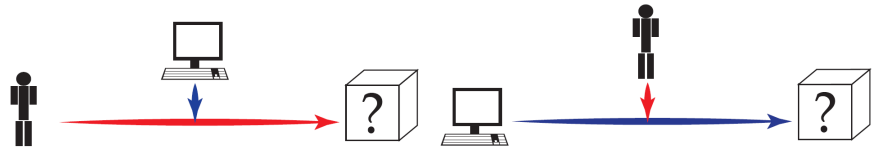
Antonios Liapis, Gillian Smith, and Noor Shaker

Abstract Algorithms can generate game content, but so can humans. And while PCG algorithms can generate some kinds of game content remarkably well and extremely quickly, some other types (and aspects) of game content are still best made by humans. Can we combine the advantages of procedural generation and human creation somehow? This chapter discusses mixed-initiative systems for PCG, where both humans and software have agency and co-create content. A small taxonomy is presented of different ways in which humans and algorithms can collaborate, and then three mixed-initiative PCG systems are discussed in some detail: Tanagra, Sentient Sketchbook, and Ropossum.

11.1 Taking a step back from automation

Many PCG methods discussed so far in this book have focused on fully automated content generation. *Mixed-initiative* procedural content generation covers a broad range of generators, algorithms, and tools which share one common trait: they require human input in order to be of any use. While most generators require some initial setup, whether it's as little as a human pressing "generate", or providing configuration and constraints on the output, mixed-initiative PCG automates only part of the process, requiring significantly more human input during the generation process than other forms of PCG.

As the phrase suggests, both a human creator and a computational creator "take the initiative" in mixed-initiative PCG systems. However, there is a sliding scale on the type and impact of each of these creators' initiative. For instance, one can argue that a human novelist using a text editor on their computer is a mixed-initiative process, with the human user providing most of the initiative but the text editor facilitating their process (spell-checking, word counting or choosing when to end a line). At the other extreme, the map generator in *Civilization V* (Firaxis 2014) is a mixed-initiative process, since the user provides a number of desired properties of



(a) Computer-aided design: Humans have the idea, the computer supports their creative process

(b) Interactive evolution: The computer creates content, humans guide it to create content they prefer

Fig. 11.1: Two types of mixed-initiative design

the map. This chapter will focus on less extreme cases, however, where both human and computer have some significant impact on the sort of content generated.

It is naive to expect that the human creator and the computational creator always have equal say in the creative process:

- In some cases, the human creator has an idea for a design, requiring the computer to allow for an easy and intuitive way to realize this idea. Closer to a word processor or to Photoshop, such content generators facilitate the human in their creative task, often providing an elaborate user interface. The computer's initiative is realized as it evaluates the human design, testing whether it breaks any design constraints and presenting alternatives to the human designer. Generators where the creativity stems from human initiative, as seen in Figure 11.1a, will be discussed in Section 11.2.
- In other cases, the computer can autonomously generate content but lacks the ability to judge the quality of what it creates. When evaluating generated content is subjective, unknown in advance, or too daunting to formulate mathematically, generators can request human users to act as judges and guide the generative processes towards content that these users deem better. The most common method for accomplishing this task is interactive evolution, as seen in Figure 11.1b, and discussed in Section 11.3. In interactive evolution the computer has the creative initiative while the human acts as an advisor, trying to steer the generator towards their own goals. In most cases, human users don't have direct control over the generated artifacts; selecting their favourites does not specify how the computer will interpret and accommodate their choice.

11.2 A very short design-tool history

To understand mixed-initiative PCG systems, as well as to gain inspiration for future systems, it is important to also understand several older systems on which current work builds. There are three main threads of work that we'll look at in this section: mixed-initiative interaction, computer-aided design (CAD), and creativity support tools. Today's research in game-design and mixed-initiative PCG tools has been

influenced by the ways each of these three areas of work frames the idea of joint human–computer creation [1, 18, 28], and the systems we'll talk about in the chapter all take inspiration from at least one of them.

11.2.1 Mixed-initiative interaction

In 1960, J.C.R. Licklider [24] laid out his dream of the future of computing: man–computer symbiosis. Licklider was the first to suggest that the operator of a computer take on any role other than that of the puppetmaster—he envisioned that one day the computer would have a more symbiotic relationship with the human operator. Licklider described a flaw of existing interactive computer systems: "In the man-machine systems of the past, the human operator supplied the initiative, the direction, the integration, and the criterion."

Notice the use of the term "initiative" to refer to how the human interacts with the computer, and the implication that the future of man-computer symbiosis therefore involves the computer being able to share initiative with its human user.

The term "mixed-initiative" was first used by Jaime Carbonell to describe his computer-aided instruction system, called SCHOLAR [3]. SCHOLAR is a text-based instructional system that largely consists of the computer asking quiz-style questions of the student using the system; the mixed-initiative component of the system allows the student to ask questions of the computer as well. Carbonell argued that there were two particularly important and related aspects of a mixed-initiative system: context and relevancy. Maintaining context involves ensuring that the computer can only ask questions that are contextually relevant to the discussion thus far, ensuring that large sways in conversation do not occur. Relevancy involves only answering questions with relevant information, rather than all of the information known about the topic.

It can be helpful to think about the sharing of initiative in mixed-initiative interaction in terms of a conversation. Imagine, for example, two human colleagues having a conversation in the workplace:

Kevin: "Do you have time to chat about the tutorial levels for the game?"
Sarah: "Yes, let's do that now! I think we need to work together to re-design the first level. Do you—."
Kevin: "Yeah, I agree, players aren't understanding how to use the powerups. I was thinking we should make the tutorial text bigger and have it linger on the screen for longer."
Sarah: "Well, information I got from the user study session two days ago implied that players weren't reading the text at all. I'm not sure if making the text bigger will help."
Kevin: "I think it will help."
pause
Kevin: "It's easy to implement, at least."

Sarah: "Okay, how about you try that, and I'll work on a new idea I have for having the companion character show you how to use them."
Kevin: "Great! Let's meet again next week to see how it worked."

There are several ways in which Kevin and Sarah are sharing initiative in this conversation. Novick and Sutton [30] describe several components of initiative:

1. Task initiative: deciding what the topic of the conversation will be, and what problem needs to be solved. In our example, Kevin takes the task initiative, by bringing up the topic of altering the tutorial levels, and by introducing the problem that, specifically, players don't understand how to use the powerups.
2. Speaker initiative: determining when each actor will speak. Mixed initiative is often characterized as a form of turn-taking interaction, where one actor speaks while the other waits, and vice versa. Our example conversation mostly follows a turn-taking model, but deviates in two major areas: a) Kevin interrupts Sarah's comments because he thinks he already knows what she will say, and b) Kevin later speaks twice in a row, in an effort to move the conversation along.
3. Outcome initiative: deciding how the problem introduced should be solved, sometimes involving allocating tasks to participants in the conversation. For this example, Sarah takes the outcome initiative, determining which tasks she and Kevin should perform as a result of the conversation.

The majority of mixed-initiative PCG systems focus entirely on the second kind of initiative: speaker initiative. They involve the computer being able to provide support during the design process, an activity that design researcher Donald Schön has described as a reflective conversation with the medium [32] (more on this in the next section). However, they all explicitly give the human designer sole responsibility for determining what the topic of the design conversation will be and how to solve the problem; all mixed-initiative PCG systems made thus far have prioritized human control over the generated content.

11.2.2 Computer-aided design and creativity support

Doug Engelbart, an early pioneer of computing, posited that computers might augment human intellect. He envisioned a future in which computers were capable of "increasing the capability of a man to approach a complex problem situation, to gain comprehension to suit his particular needs, and to derive solutions to problems" [10]. Engelbart argued that all technology can serve this purpose. His deaugmentation experiment, in which he wrote the same text using a typewriter, a normal pen, and a pen with a brick attached to it, showed the influence that the technology has on the ways that we write and communicate.

A peer of Engelbart's, Ivan Sutherland, created the Sketchpad system in 1963 [42]. This was the first system to offer computational support for designers; it was also the first example of an object-oriented system (though it did not involve programming). Sketchpad allowed designers to specify constraints on the designs they

were drawing; for example, it was possible to draw the general topology of an item such as a bolt. The user could then place constraints on the edges of the bolt to force them to be perpendicular to each other. The system was object-oriented in that individual sketches could be imported into others to produce entire diagrams and drawings; if the original sketch was altered, that change would propagate to all diagrams that imported the sketch. The idea of letting users create through adding and removing constraints has carried forward into mixed-initiative tools such as Tanagra and Sketchaworld, described later in this chapter.

A decade after Engelbart and Sutherland's work, Nicholas Negroponte proposed the creation of what he called design amplifiers. Negroponte was particularly interested in how to support non-expert architectural designers, as he was concerned that professional architects often pushed their own agendas without regard for the needs of the occupants [27]. However, homeowners do not have the domain expertise required to design their own home. His vision was for tools that could help the general population in creating their own homes using the computer to support their designs and ensure validity of the design. The idea that human creators should take the forefront and have the majority of control over a design situation is reflected in all mixed-initiative design tools; in general, the computer is never allowed to override a decision made by the human. However, all tools must push an agenda to some extent, though it may not be intentional: the choices that go into how the content generator operates and what kind of content it is capable of creating vastly influences the work that the human designer can create with the system.

More recently, Chaim Gingold has pushed the idea of "magic crayons": software that supports a novice's creativity while also being intuitive, powerful, and expressive [11]. Gingold argues that using design support tools should be as simple and obvious as using a crayon, and allow for instant creativity. Any child who picks up a crayon can quickly and easily grasp how to use it and go on to create several drawings quite rapidly. The "magic" part of the magic crayon comes in the crayon's computational power and expressive potential: the crayon is imbued with computational support that allows a user to create something better than what they would normally create themselves, while still echoing their original design intent.

11.2.3 Requirements, caveats, and open problems for mixed-initiative systems

When designing a mixed-initiative system, there are several main questions to consider. These points are based on the authors' experiences creating their own prototype mixed-initiative tools:

- *Who is your target audience?*

How to design both the underlying technology and the interface for a mixed-initiative system depends wildly upon who the target audience is. A tool for pro-

fessional designers might look considerably different from a tool intended for game players who have no design experience.

- *What novel and useful editing operations can be incorporated?*

A mixed-initiative environment offers the opportunity for more sophisticated level-editing operations than merely altering content as one could do in a non-AI-supported tool. The way the generation algorithm works might prioritize certain aspects of the design. For example, Tanagra's underlying generator used rhythm as a driver for creating levels; thus, it was relatively straightforward to permit users to interact with that underlying structure to be able to directly manipulate level pacing.

- *How can the method for control over content be balanced?*

Mixed-initiative content generators can involve both direct and indirect manipulation of the content being created. For example, the tool will typically support a user directly drawing in aspects of the content (e.g. level geometry), but also allow the computer to take over and make new suggestions for the generator. How to balance these forms of control can be challenging (especially when the human and computer conflict, see next point). Should the computer be allowed to make new suggestions whenever it wants, or only when specifically requested? How much of the content should be directly manipulable?

- *How to resolve conflicts that arise due to the human stating conflicting desires?*

In situations where both human and computer are editing content simultaneously, editing conflict inevitably arises. The majority of mixed-initiative tools follow the principle that the human has final say over what is produced by the tool. However, when the human user states contradictory desires, the system must decide how to handle the situation. Should it simply provide an error message? Should it randomly choose which desire is more important for the human? Should it generate several plausible answers and then ask the human to choose which solution is most reasonable?

More generally, the issue is: how can the computer infer human design intent via an interface where the human simply interacts with the content itself.

- *How expressive is the system?*

All content that a human can produce using a mixed-initiative PCG system must be possible for the computer to generate on its own. Thus it is vital for the system to be expressive enough to offer a meaningful set of choices to the human user. More information about expressivity evaluation is in Chapter 12.

- *Can the computer explain itself?*

It is difficult for a human and computer to engage in a design collaboration if neither is able to explain itself to the other. In particular, a human designer may become frustrated or confused if the computer consistently acts as though it is not following the model that the human designer has in her head for how the system should work. The computer should appear intelligent (even if the choices it is making do not

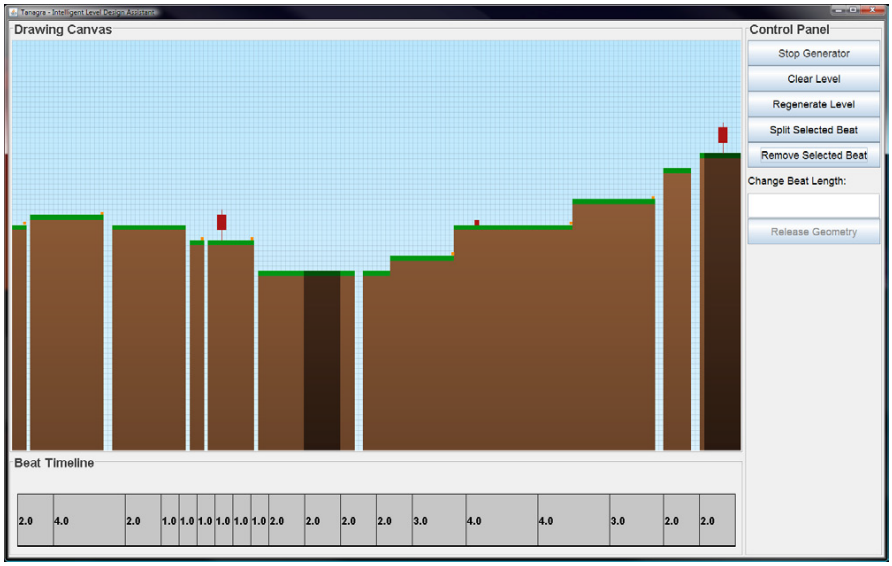


Fig. 11.2: Tanagra, an intelligent level design tool. The level is created in the large area at the upper left. Below is a beat timeline, where the pacing of the level can be manipulated. On the right are buttons for editing the level. Adapted from [40]

involve a sophisticated AI system), and ideally should be able to explain its actions to the human. Being able to communicate at a meta-level about the design tasks and outcomes has not been well explored in mixed-initiative PCG work thus far.

11.2.4 Examples of CAD tools for games

Now we'll go through several examples of computer-aided design tools that both allow interaction and feedback for the human designer and introduce some initiative taken by the computational designer (in varying degrees and in different forms).

11.2.4.1 Tanagra

Tanagra is a mixed-initiative tool for level design, allowing a human and a computer to work together to produce a level for a 2D platformer [40]. An underlying, reactive level generator ensures that all levels created in the environment are playable, and provides the ability for a human designer to rapidly view many different levels that meet their specifications. The human designer can iteratively refine the level by placing and moving level geometry, as well as through directly manipulating the pacing of the level. Tanagra's underlying level generator is capable of produc-

ing many different variations on a level more rapidly than human designers, whose strengths instead lie in creativity and the ability to judge the quality of the generated content. The generator is able to guarantee that all the levels it creates are playable, thus refocusing early playtesting effort from checking that all sections of the level are reachable to exploring how to create fun levels.

A combination of reactive planning and constraint programming allows Tanagra to respond to designer changes in real time. A Behavior Language (ABL) [25] is used for reactive planning, and Choco [45] for numerical constraint solving. Reactive planning allows for the expression of generator behaviours, such as placing patterns of geometry or altering the pacing of the level, which can be interleaved with a human designer's actions.

The version of Tanagra displayed in Figure 11.2 incorporates (1) the concept of a user "pinning" geometry in place by adding numerical positioning constraints, (2) the system attempting to minimise the number of required positioning changes (including never being allowed to move pinned geometry), and (3) direct changes to level pacing by adding, removing, and altering the length of beats. Later versions of Tanagra altered the UI to make it clearer what geometry was "pinned" and what was not. The latest version of Tanagra also added the idea of geometry preference toggles, allowing designers an additional layer of control over the system by letting them state whether or not particular geometry patterns are preferred or disliked on a per-beat basis.

11.2.4.2 Sentient Sketchbook

Sentient Sketchbook is a computer-aided design tool which assists a human designer in creating game levels, such as maps for strategy games [22] (shown in Figure 11.3) or dungeons for roguelike games [23]. Sentient Sketchbook uses the notion of map sketch as a minimal abstraction of a full game level; this abstraction limits user fatigue while creating new levels and reduces the computational effort of automatically evaluating such sketches, but is rendered into a high-resolution map before use. Like popular CAD tools, Sentient Sketchbook supports the human creator by automatically testing maps for playability constraints, by calculating and displaying navigable paths, by evaluating the map on gameplay properties and by converting the coarse map sketch into a playable level.

The innovation of Sentient Sketchbook is the real-time generation and presentation of alternatives to the user's sketch. These alternatives are evolved from an initial population seeded by the user's sketch, and thus a certain degree of map integrity is maintained with the user's designs. The shown suggestions are guaranteed to be playable (i.e. have all vital components such as bases and resources for strategy games connected with passable paths) via the use of constrained evolutionary optimisation with two populations [16]. The suggestions are either evolved to maximise one of the predefined objective functions inspired by popular game design patterns such as balance and exploration [23], or towards divergence from the user's current sketch through feasible-infeasible novelty search [21].

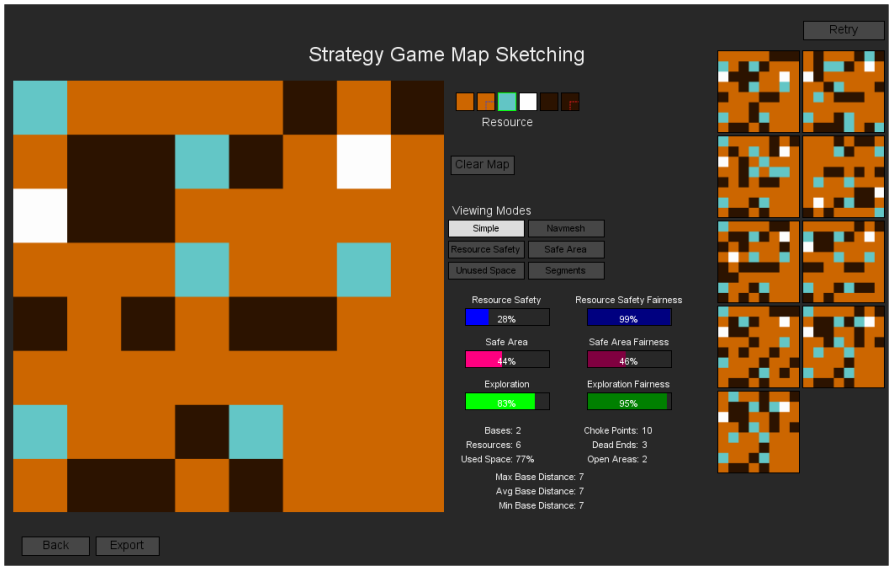


Fig. 11.3: The user interface of Sentient Sketchbook. A human designer edits their sketch (left) and a generator, acting as the artificial designer, creates map suggestions in response (right). Adapted from [22]

11.2.4.3 Ropossum

Ropossum is an authoring tool for the generation and testing of levels of the physics-based puzzle game, *Cut the Rope* [36]. Ropossum integrates many features: (1) automatic design of complete solvable content, (2) incorporation of designer's input through the creation of complete or partial designs, (3) automatic check for playability, and (4) optimisation of a given design based on playability.

Ropossum consists of two main modules: an evolutionary framework for procedural content generation [34] and a physics-based playability module to solve given designs [35]. The second module is used both for evolving playable content and for playtesting levels designed by humans. The parameters of the evolutionary system and the AI agent are optimised so that the system can respond to the user's inputs within a reasonable amount of time. Grammatical evolution (GE) is used to evolve the content. The level structure is defined in a design grammar (DG) which defines the positions and properties of different game components and permits an easy to read and manipulate format by game designers [34]. First-order logic is used to encode the game state as facts specifying the game components and their properties (such as position, speed, and moving direction) [35]. The relationships between the components are represented as rules used to infer the possible next actions.

The two methods for evolving game design and assessing whether the design is playable are combined in a framework to evolve playable content. An initial level design, according to the design grammar, is generated or created by the designer (see

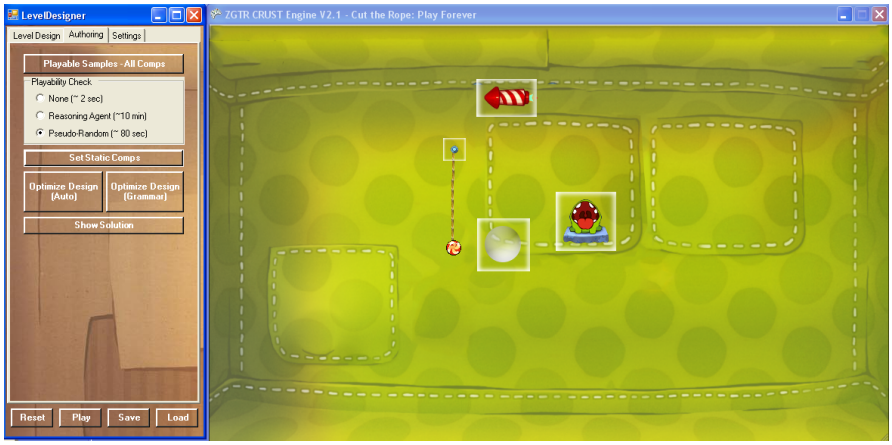


Fig. 11.4: One of the interfaces of Ropossum. The components highlighted are the ones constrained by the designer, and therefore will not be changed when evolving complete playable levels. Adapted from [36]

Figure 11.4) and encoded as facts that can be used by the AI reasoning agent. Given the game state, the agent infers the next best action(s) to perform. The actions are then sent to the physics simulator, which performs the actions according to a given priority and updates the game state accordingly. The new game state is sent to the agent to infer the next action. If the sequence of actions does not lead to winning the level, the system backtracks. A state tree is generated that represents the actions and states explored. For each action performed, a node in the tree is generated and the tree is explored in a depth-first approach. The size of the explored branches in the solution tree is reduced by assigning priorities to the actions and employing domain knowledge encoded in reasoning agent's rules to infer the best action to perform.

11.2.4.4 Sketchaworld

Sketchaworld is an interactive tool created to enable a non-specialist user to easily and efficiently create a complete 3D virtual world by integrating different procedural techniques [39]. Sketchaworld integrates many features: (1) it facilitates easy interaction with designers, who can specify procedural modelling operations and directly visualize their effects, (2) it builds 3D worlds by fitting all features with their surroundings and (3) it supports iterative modelling.

The tool allows user interactions in two main modes: *landscape mode* and *feature mode*. The first mode consists of the user input which is a 2D layout map of the virtual world formatted in a colouring grid that includes information about elevations and soil materials painted with a brush. In the feature mode, users add more specific land features such as cities and rivers (see Figure 11.5).

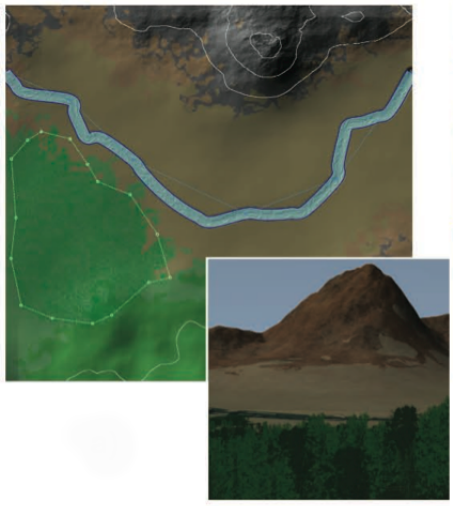


Fig. 11.5: A screenshot from Sketchaworld. The user sketches as input a mountain, river, and forest, and corresponding 3D terrain is generated. Adapted from [39]

While users sketch on the 2D grid, the effects of their modification are directly visualized in the 3D virtual world. This requires blending the features added with their surroundings. To this end, whenever a new feature is created, an automatic local adaptation step is performed to ensure smoothness and correctness. This includes for example removing trees on a generated road's path or adding a road to connect a generated bridge.

11.3 Interactive evolution

As its name suggests, interactive evolutionary computation (IEC) is a variant of evolutionary computation where human input is used to evaluate content. As artificial evolution hinges on the notion of survival of the fittest, in interactive evolution a human user essentially selects which individuals create offspring and which individuals die. According to Takagi [44], interactive evolution allows human users to evaluate individuals based on their subjective preferences (their own *psychological space*) while the computer searches for this human-specified global optimum in the genotype space (*feature parameter space*); as such, the collaboration between human and computer makes IEC a mixed-initiative approach. In interactive evolution a human user can evaluate artifacts by assigning to each a numerical value (proportionate to their preference for this artifact), by ordering artifacts in order of preference, or by simply selecting one or more artifacts that they would like to see more of. With more control to the human user, the artifacts in the next generation may

match users' desires better; the user's cognitive effort may also increase, however, which results in *user fatigue*, which is covered in Section 11.3.1

Interactive evolution is often used in domains where designing a fitness function is a difficult task; for instance, the criteria for selection could be a subjective measure of beauty as in evolutionary art, a deceptive problem where a naive quantifiable measure may be more harmful than helpful, or in cases where mathematically defining a measure of optimality is as challenging as the optimisation task itself. Since it allows for a subjective evaluation of beauty, IEC has often been used to create 2D visual artifacts based on L-systems [26], mathematical expressions [37], neural networks [33] or other methods. Using interactive evolution in art is often motivated by a general interest in artificial life, as is the case with Dawkins' Biomorph [7]. In evolutionary art, human users often evaluate not the phenotypes but outputs specified by the phenotypes, in cases where the phenotypes are image filters or shaders [9]. Apart from 2D visual artifacts, IEC has also been used in generating 3D art [6], animated movies [38], typography [31], and graphic design [8]. Evolutionary music has also used IEC to generate the rhythm of percussion parts [46], jazz melodies [2], and accompaniments to human-authored music scores [14], among others. Outside evolutionary art and music, IEC has been used for industrial design [12], image database retrieval [5], human-like robot motion [29], and many others. The survey by Takagi [44] provides a thorough, if somewhat dated, overview of IEC applications.

11.3.1 User fatigue and methods of combating it

Since interactive evolution is entirely reliant on human input to drive its search processes, its largest weakness is the effect of human fatigue in human-computer interaction. Human fatigue becomes an issue when the users are required to perform a large number of content selections, when feedback from the system is slow, when the users are simultaneously presented with a large amount of content on-screen, or when users are required to provide very specific input. All of these factors contribute to the *cognitive overload* of the user, and several solutions have been proposed to counteract each of these factors.

As human users are often overburdened by the simultaneous presentation of information on-screen, user fatigue can be limited by an interactive evolutionary system that shows only a subset of the entire population. There are a number of techniques for selecting which individuals to show, although they all introduce biases from the tool's designers. An intuitive criterion is to avoid showing individuals which all users would consider unwanted. Deciding which individuals are unwanted is sometimes straightforward; for instance, musical tracks containing only silence or 3D meshes with disconnected triangles. However, such methods often only prune the edges of the search space and are still not guaranteed to show wanted content. Another technique is to show only individuals with the highest fitness; since fitness in interactive evolution is largely derived from user choices, this is likely to

result in individuals which are very similar—if not identical—to individuals shown previously, which is more likely to increase fatigue due to perceived stagnation.

User fatigue is often induced when the requirement of a large number of selections becomes time-consuming and cumbersome. As already mentioned, fewer individuals than the entire population can be shown to the user; in a similar vein, not every generation of individuals needs to be shown to the user, instead showing individuals every 5 or 10 generations. In order to accomplish such a behaviour, the fitness of unseen content must be somehow predicted based on users' choices among seen content. One way to accomplish such a prediction is via distance-based approaches, i.e. by comparing an individual that hasn't been presented to the user with those individuals that were presented to the user: the fitness of this unseen individual can be proportional to the user-specified fitness of the closest seen individual while inversely proportional to their distance [15]. Such a technique essentially clusters all individuals in the population around the few presented individuals; this permits the use of a population larger than the number of shown individuals as well as an offline evolutionary sprint with no human input. Depending on the number of seen individuals and the expressiveness of the algorithm's representation, however, a number of strong assumptions are made—the most important of which pertains to the measure of distance used. In order to avoid biasing the search by these assumptions, most evolutionary sprints are only for a few generations before new human feedback is required.

Another solution to the extraneous choices required of IEC systems' users is to crowdsource the selection process among a large group of individuals. Some form of online database is likely necessary towards that end, allowing users to start evolving content previously evolved by another user. A good example of this method is PicBreeder [33], which evolves images created by compositional pattern-producing networks (CPPNs). Since evolution progressively increases the size of CPPNs due to the Neuroevolution of Augmenting Topologies algorithm [41], the patterns of the images they create become more complex and inspiring with large networks. This, however, requires extensive evolution via manual labor, which is expected to induce significant fatigue on a single user. For that reason, the PicBreeder website allows users to start evolution "from scratch", with a small CPPN able to create simple patterns such as circles or gradients, or instead load images evolved by previous users and evolve them further. Since such images are explicitly saved by past users because they are visually interesting, the user starts from a "good" area of the genotype space and is more likely to have meaningful variations than if they were starting from scratch and had to explore a large area of the search space which contains non-interesting images.

Another factor of user fatigue is the slow feedback of evolutionary systems; since artificial evolution is rarely a fast process, especially with large populations, the user may have to sit through long periods of inaction before the next set of content is presented. In order to alleviate that, interactive evolution addresses it by several shortcuts to speed up convergence of the algorithm. This is often accomplished by limiting the population size to 10 or 20 individuals, or by allowing the user to in-

terfere directly in the search process by showing a visualization of the search space and letting them designate an estimated global optimum [43].

To reduce the cognitive load of evaluations, a common solution is to limit the number of rating levels, either to a common five-star rating scale, or even to only two: the user either likes the content or doesn't. Another option is to use rankings [17], i.e. the user is presented with two options and chooses the one they prefer, without having to explicitly specify that e.g. one is rated three stars while the other is five-star content.

11.3.2 Examples of interactive evolution for games

As highly interactive experiences themselves, games are ideal for interactive evolution, since the user's preferences can be inferred from what they do in the game. Instead of an explicit selection process, selection masquerades behind in-game activities such as shooting, trading, or staying alive. Done properly, interactive evolution in games can bypass to a large extent the issue of user fatigue. However, the mapping between player actions and player preference is often not straightforward; for instance, do humans prefer to survive in a game level for a long time, or do they like to be challenged and be constantly firing their weapons? Depending on the choice of metric (in this example, survival time or shots fired), different content may be favoured. Therefore, gameplay-based evaluations may include more biases on the part of the programmer than traditional interactive evolution, which tries to make no assumptions.

11.3.2.1 Galactic Arms Race

Galactic Arms Race [13], shown in Figure 11.6, is one of the more successful examples of a game using interactive evolution. The procedurally generated weapon projectiles, which are the main focus of this space-shooter game, are evolved interactively using gameplay data. The number of times a weapon is fired is considered a revealed user preference; the assumption is that players who don't like a weapon will not use it as much as others. Weapon projectiles, represented as particles, are evolved via neuroevolution of augmenting topologies (NEAT); the velocity and colour of each particle is defined as the output of a CPPN, with the input being the current position and distance from the firing spaceship.[1] Newly evolved weapons are dropped as rewards for destroying enemy bases; the player can pick them up, and use them or switch among three weapons at any given time. *Galactic Arms Race* can be also played by many players; in multiplayer play, the algorithm uses the firing rates of all players when determining which weapons to evolve.

[1] NEAT and CPPNs are discussed in detail in Chapter 9.

Fig. 11.6: *Galactic Arms Race* with multiple players using different weapons. Adapted from [13]

11.3.2.2 TORCS track generation

A more traditional form of interactive evolution, in which a user directly states preferences, was used to generate tracks for a car racing game [4]. The system uses The Open Racing Car Simulator (TORCS)[2] and allows user interaction through a web browser where users can view populations of race tracks and evaluate them (see Figure 11.7). This web front-end then communicates with an evolutionary backend. Race tracks are represented in the engine as a list of segments which can be either straight or turning. In the evolution process, a set of control points and Bézier curves are used to connect the points and ensure smoothness.

Different variations of interactive evolution are used to evaluate the generated tracks. In *single-user mode*, human subjects were asked to play 10 generations of 20 evolved tracks each and evaluate them using two scoring interfaces: like/dislike and rating from 1 to 5 stars. The feedback provided by users about each track is the fitness used for evolution. In *multi-user mode*, the same population of 20 individuals is played and evaluated by five human subjects. The fitness given to each track in the population is the average score received from all users. The feedback provided by users showed improvements in the quality of the tracks and an increase in their interestingness.

[2] http://torcs.sourceforge.net/

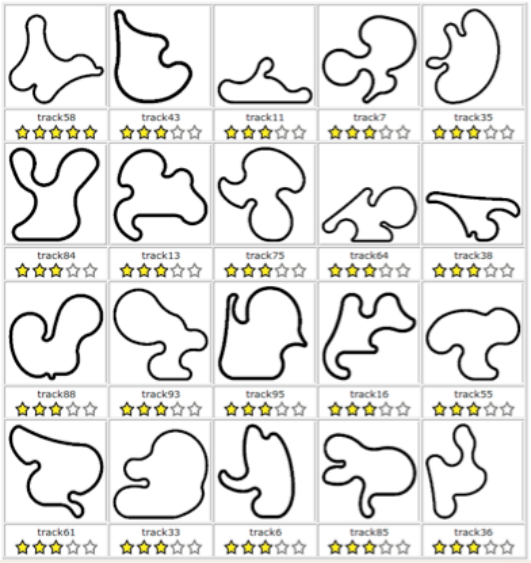


Fig. 11.7: The TORCS track generator visualizes tracks, and asks the player to rank them. Adapted from [4]

11.3.2.3 Spaceship generation

Liapis et al.'s [20] work on spaceship generation is an example of fitness prediction for the purpose of speeding up and enhancing the convergence of interactive evolution. They generate spaceship hulls and their weapon and thruster topologies in order to match a user's visual taste as well as conform to a number of constraints aimed at playability and game balance [19]. The 2D shapes representing the spaceship hulls are encoded as pattern-producing networks (CPPNs) and evolved in two populations using the feasible-infeasible two-population approach (FI-2pop) [16]. One population contains spaceships which fail ad-hoc constraints pertaining to rendering, physics simulation, and game balance, and individuals in this population are optimised towards minimising their distance to feasibility. Removing such spaceships from the population shown to the user reduces the chances of unwanted content and reduces user fatigue.

The second population contains feasible spaceships, which are optimised according to ten fitness dimensions pertaining to common attributes of visual taste such as symmetry, weight distribution, simplicity, and size. These fitness dimensions are aggregated into a weighted sum which is used as the feasible population's fitness function. The weights in this quality approximation are adjusted according to a user's selection among a set of presented spaceships (see Figure 11.8). This adaptive aesthetic model aims to enhance the visual patterns behind the user's selection and minimise visual patterns of unselected content, thus generating a completely new

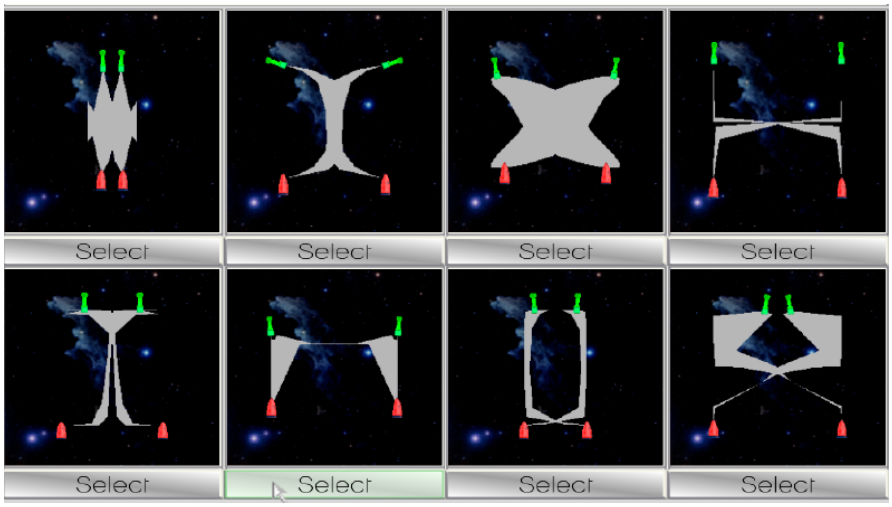


Fig. 11.8: In this evolutionary spaceship generator, the user is presented a set of spaceships from the feasible population, and selects their favourite. Adapted from [20]

set of spaceships which more accurately match the user's tastes. A small number of user selections allows the system to recognize the user's preference, reducing fatigue.

The two-step adaptation system, where (1) the user implicitly adjusts their preference model through content selection and (2) the preference model affects the patterns of generated content, is intended to make for a flexible tool both for personalizing game content to an end-user's visual taste and also for inspiring a designer's creative task with content guaranteed to be playable, novel, and conforming to the intended visual style.

11.4 Exercise

1. Choose one of the tools described in this chapter. Perform a design task similar to that which is supported by the tool without any computational support. Reflect upon this process: What was easy and what was hard? What did you wish the computer could do to help? What do you feel the computer would not be able to assist with? If the tool is available for download, try to perform the same design task using the AI-supported tool. What were some of the key differences in your experience as a designer?
2. Create a requirements analysis document and mock-up architecture diagram for a mixed-initiative design tool that operates in a domain of your choice. Make sure to consider: (a) Who is your audience? (b) What, specifically, is your domain?

(c) What is the PCG system capable of creating? (d) What is the mixed-initiative conversational model the system will follow?
3. Create a paper prototype of the tool you designed in exercise two. Test the prototype with someone else in the class, with you acting as the "AI system" and your partner acting as the designer. Be careful to only act according to how the AI system itself would be able to act.

11.5 Summary

Mixed-initiative systems are systems where both humans and computers can "take the initiative," and both contribute to the creative process. The degree to which each party takes the initiative and contributes varies between different systems. At one end of this scale is computer-aided design (CAD), where the human directs the creative process and the computer performs tasks when asked to and according to the specifications of the user. At the other end is interactive evolution, where the computer proposes new artifacts and the user is purely reactive, providing feedback on the computer's suggestions. Both of these approaches have a rich history in games: computer-aided design in many game design tools that include elements of content generation, and interactive evolution in games such as *Galactic Arms Race*. "True" mixed-initiative interaction, or at least the idea of such systems, has a long history within human-computer interaction and artificial intelligence. Within game content generation, there are so far just a few attempts to realize this vision. Tanagra is a platformer level-generation system that uses constraint satisfaction to complete levels sketched by humans, and regenerates parts of levels to ensure playability as the levels are edited by humans. Sentient Sketchbook assists humans in designing strategy game levels, providing feedback on a number of quality metrics and autonomously suggesting modifications of levels. Ropossum is a level editor for *Cut the Rope*, which can test the playability of levels and automatically regenerate parts of levels to ensure playability as the level is being edited.

References

1. Almeida, M.S.O., da Silva, F.S.C.: A systematic review of game design methods and tools. In: Proceedings of the International Conference on Entertainment Computing, pp. 17–29 (2013)
2. Biles, J.A.: GenJam: A genetic algorithm for generating jazz solos. In: Proceedings of the International Computer Music Conference, pp. 131–137 (1994)
3. Carbonell, J.R.: Mixed-initiative man-computer instructional dialogues. Ph.D. thesis, Massachusetts Institute of Technology (1970)
4. Cardamone, L., Loiacono, D., Lanzi, P.L.: Interactive evolution for the procedural generation of tracks in a high-end racing game. In: Proceedings of the Conference on Genetic and Evolutionary Computation, pp. 395–402 (2011)
5. Cho, S.B., Lee, J.Y.: Emotional image retrieval with interactive evolutionary computation. In: Advances in Soft Computing, pp. 57–66. Springer (1999)

6. Clune, J., Lipson, H.: Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In: Proceedings of the European Conference on Artificial Life, pp. 144–148 (2011)
7. Dawkins, R.: The Blind Watchmaker. W. W. Norton & Company (1986)
8. Dipaola, S., Carlson, K., McCaig, G., Salevati, S., Sorenson, N.: Adaptation of an autonomous creative evolutionary system for real-world design application based on creative cognition. In: Proceedings of the International Conference on Computational Creativity (2013)
9. Ebner, M., Reinhardt, M., Albert, J.: Evolution of vertex and pixel shaders. In: Genetic Programming, *Lecture Notes in Computer Science*, vol. 3447, pp. 261–270. Springer (2005)
10. Engelbart, D.C.: Augmenting human intellect: A conceptual framework. Air Force Office of Scientific Research, AFOSR-3233 (1962)
11. Gingold, C.: Miniature gardens & magic crayons: Games, spaces, & worlds. Master's thesis, Georgia Institute of Technology (2003)
12. Graf, J.: Interactive evolutionary algorithms in design. In: Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms, pp. 227–230 (1995)
13. Hastings, E.J., Guha, R.K., Stanley, K.O.: Automatic content generation in the Galactic Arms Race video game. IEEE Transactions on Computational Intelligence and AI in Games **1**(4), 245–263 (2009)
14. Hoover, A.K., Szerlip, P.A., Stanley, K.O.: Interactively evolving harmonies through functional scaffolding. In: Proceedings of the Conference on Genetic and Evolutionary Computation (2011)
15. Hsu, F.C., Chen, J.S.: A study on multi criteria decision making model: Interactive genetic algorithms approach. In: IEEE International Conference on Systems, Man, and Cybernetics, vol. 3, pp. 634–639 (1999)
16. Kimbrough, S.O., Koehler, G.J., Lu, M., Wood, D.H.: 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)
17. Liapis, A., Martínez, H.P., Togelius, J., Yannakakis, G.N.: Adaptive game level creation through rank-based interactive evolution. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (2013)
18. Liapis, A., Yannakakis, G.N., Alexopoulos, C., Lopes, P.: Can computers foster human users' creativity? Theory and praxis of mixed-initiative co-creativity. Digital Culture & Education **8**(2), 136–153 (2016)
19. Liapis, A., Yannakakis, G.N., Togelius, J.: Neuroevolutionary constrained optimization for content creation. In: Proceedings of the IEEE Conference on Computational Intelligence and Games, pp. 71–78 (2011)
20. Liapis, A., Yannakakis, G.N., Togelius, J.: Adapting models of visual aesthetics for personalized content creation. IEEE Transactions on Computational Intelligence and AI in Games **4**(3), 213–228 (2012)
21. Liapis, A., Yannakakis, G.N., Togelius, J.: Enhancements to constrained novelty search: Two-population novelty search for generating game content. In: Proceedings of the Conference on Genetic and Evolutionary Computation (2013)
22. Liapis, A., Yannakakis, G.N., Togelius, J.: Sentient Sketchbook: Computer-aided game level authoring. In: Proceedings of the 8th International Conference on the Foundations of Digital Games, pp. 213–220 (2013)
23. Liapis, A., Yannakakis, G.N., Togelius, J.: Towards a generic method of evaluating game levels. In: Proceedings of the Artificial Intelligence for Interactive Digital Entertainment Conference, pp. 30–36 (2013)
24. Licklider, J.C.R.: Man-computer symbiosis. IRE Transactions on Human Factors in Electronics **1**(1), 4–11 (1960)
25. Mateas, M., Stern, A.: A behavior language for story-based believable agents. IEEE Intelligent Systems **17**(4), 39–47 (2002)
26. McCormack, J.: Interactive evolution of L-system grammars for computer graphics modelling. In: Complex Systems: From Biology to Computation, pp. 118–130. ISO Press (1993)

27. Negroponte, N.: Soft Architecture Machines. MIT Press (1975)
28. Nelson, M.J., Mateas, M.: A requirements analysis for videogame design support tools. In: Proceedings of the 4th International Conference on the Foundations of Digital Games, pp. 137–144 (2009)
29. Nojima, Y., Kojima, F., Kubota, N.: Trajectory generation for human-friendly behavior of partner robot using fuzzy evaluating interactive genetic algorithm. In: Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp. 114–116 (2003)
30. Novick, D., Sutton, S.: What is mixed-initiative interaction? In: Proceedings of the AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction (1997)
31. Schmitz, M.: genoTyp, an experiment about genetic typography. In: Proceedings of Generative Art (2004)
32. Schön, D.A.: Designing as reflective conversation with the materials of a design situation. Research in Engineering Design **3**(3), 131–147 (1992)
33. Secretan, J., Beato, N., D'Ambrosio, D.B., Rodriguez, A., Campbell, A., Stanley, K.O.: Picbreeder: Evolving pictures collaboratively online. In: CHI '08: Proceeding of the 26th SIGCHI Conference on Human factors in Computing Systems, pp. 1759–1768 (2008)
34. Shaker, M., Sarhan, M.H., Naameh, O.A., Shaker, N., Togelius, J.: Automatic generation and analysis of physics-based puzzle games. In: Proceedings of the IEEE Conference on Computational Intelligence and Games, pp. 1–8 (2013)
35. Shaker, M., Shaker, N., Togelius, J.: Evolving playable content for Cut the Rope through a simulation-based approach. In: Proceedings of the Conference on Artificial Intelligence and Interactive Digital Entertainment, pp. 72–78 (2013)
36. Shaker, M., Shaker, N., Togelius, J.: Ropossum: An authoring tool for designing, optimizing and solving Cut the Rope levels. In: Proceedings of the Conference on Artificial Intelligence and Interactive Digital Entertainment, pp. 215–216 (2013)
37. Sims, K.: Artificial evolution for computer graphics. In: Proceedings of the 18th Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '91, pp. 319–328 (1991)
38. Sims, K.: Interactive evolution of dynamical systems. In: Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, pp. 171–178 (1992)
39. Smelik, R.M., Tutenel, T., de Kraker, K.J., Bidarra, R.: Interactive creation of virtual worlds using procedural sketching. In: Proceedings of Eurographics, pp. 29–32 (2010)
40. Smith, G., Whitehead, J., Mateas, M.: Tanagra: Reactive planning and constraint solving for mixed-initiative level design. IEEE Transactions on Computational Intelligence and AI in Games **3**(3), 201–215 (2011)
41. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10**(2), 99–127 (2002)
42. Sutherland, I.E.: Sketchpad: A man-machine graphical communication system. In: Proceedings of the Spring Joint Computer Conference, AFIPS '63, pp. 329–346 (1963)
43. Takagi, H.: Active user intervention in an EC search. In: Proceedings of the International Conference on Information Sciences, pp. 995–998 (2000)
44. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. Proceedings of the IEEE **89**(9), 1275–1296 (2001)
45. The Choco Team: Choco: An open source Java constraint programming library. In: 14th International Conference on Principles and Practice of Constraint Programming (2008)
46. Tokui, N., Iba, H.: Music composition with interactive evolutionary computation. In: International Conference on Generative Art, pp. 219–226 (2000)