Chapter 11
Mixed-initiative (DRAFT)

Antonios Liapis, Gillian Smith and Noor Shaker

11.1 What is Mixed Initiative?

Mixed-initiative procedural content generation covers a very broad gamut of generators, algorithms and tools which share one common trait: they require human input in order to be of any use. While most generators arguably require a human to press "generate" or to provide some rudimentary arguments (such as the map size or map type in Civilization maps), mixed-initiative PCG automates only part of the process, requiring significantly more human input than other forms of PCG.

As its title suggests, the generation of content is performed by a human creator and by a computational creator. However, there is a sliding scale on the type and impact of each of these creators' initative. 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); on the other extreme, the map generator in Civilization V is a mixed-initiative process, since the user provides a number of desired properties of 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, however, 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 intutiive way to realize this idea. Closer to a word editor or to Photoshop, such content generators facilitate the human in his creative task, often providing an elaborate user interface; the computer's initiative is realized as it evaluates the human design, testing if it breaks any design constraints and presents alternatives to the human designer. Generators where the creativity stems from the human's initiative, as seen in Figure 11.1, and will be discussed in Section 11.2.

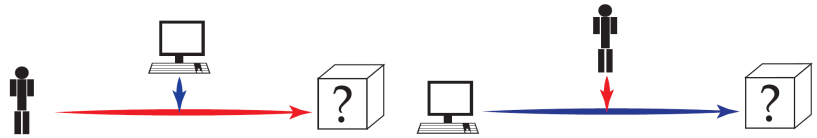

Fig. 11.2: Interactive Evolution: the computer creates content, humans guide it to create content they prefer.

- In other cases, the computer is able to autonomously generate content but lacks the ability to judge whether what it has created is sufficiently good. In cases where evaluating generated artifacts is subjective, unknown in advance, or too daunting to formulate mathematically, such generators request human users to act as the judge and guide their generative processes towards content that they deem better. The most common method for accomplishing this task is interactive evolution, as seen in Figure 11.1, and will be discussed in Section 11.2. 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 favorites does not guarantee in which way the computer will interpret and accomodate their choice.

11.2 Computer Aided Design tools

To understand how mixed-initiative PCG systems are designed today, as well as to attain future inspiration for such systems, it is important to also understand several older systems from which current work draws inspiration. There are three main threads of work that we'll look at in this section: mixed-initiative interaction, computer-assisted design, and creativity support tools. Each of these areas of work have inspired different aspects of the systems we'll talk about in the chapter.

11.2.1 Mixed-Initiative Interaction

In 1960, J.C.R. Licklider [15] 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 as: "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 [1]. SCHOLAR is a text-based instructional system that largely consisted 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 involved ensuring that the computer would only be able to ask questions that were contextually relevant to the discussion thus far, so that large sways in conversation did not occur. Relevancy involves only answering questions with relevant information, rather than all of the information known about the topic.

Mixed-initiative interaction describes a form of interaction between computer and human in which initiative is shared. It can be helpful to think about this style of interaction as 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 [17] 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 [18] (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-Assisted Design and Creativity Support

Doug Engelbart, an early pioneer of computing, posed that computers stand to 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." [3]. Engelbart argued that all technology can serve this purpose. His de-augmentation 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 (in this case, the way we write) has on the ways that we write and communicate.

A peer of Engelbart, Ivan Sutherland's created the Sketchpad system in 1963. 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 design amplifiers. Negroponte was particularly interested in how to support non-expert designers, as he was concerned that experts often push their own agendas without regard for the need of the occupant. However, home-owners 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. 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 crayons are 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 professional designers might look considerably different than 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 in 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 11.

- *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 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

Although not comprehensive, the following Computer-Aided Design tools do not only afford interaction and feedback for the human designer, but also introduce some initiative (in varying degrees and in different forms) to the computational designer.

11.2.4.1 Tanagra

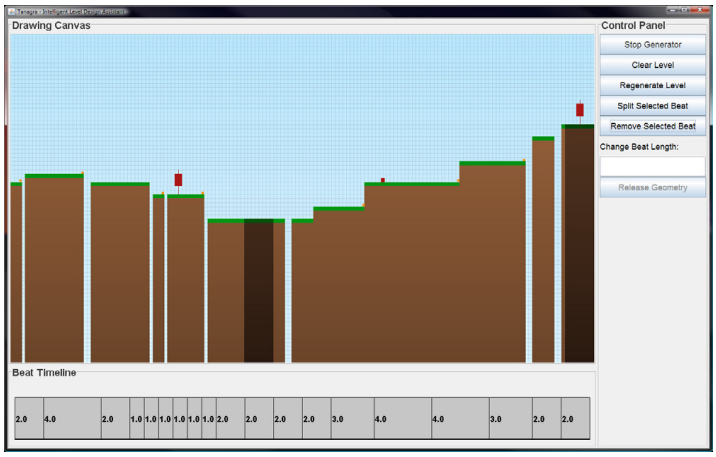

Fig. 11.3: The Tanagra intelligent level design tool. The large area in the upper left is where the level is created. Below that is a beat timeline, allowing manipulation of the pacing of the level. On the right are buttons for editing the level.

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 [4]. 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.

The mixed-initiative approach to design, where content is created through iterative cycles between the human designer and a procedural content generator, capitalizes on the strengths of both human and computer designers. Tanagra's underlying level generator is capable of producing 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. Also, a game-play centric approach to level representation and content generation opens up possibilities for novel editing operations: in addition to controlling physical properties of the world such as platform placement, the designer is also able to control properties related to potential player behavior by influencing the pacing of the level. In Tanagra, the human designer and procedural generator work together in a collaborative way, each taking turns to build on the work of the other.

A combination of reactive planning and constraint programming allows Tanagra to respond to designer changes in real-time. A Behavior Language (ABL) [16] is used for reactive planning, and Choco [27] for numerical constraint solving. Reactive planning allows for the expression of generator behaviors, such as placing patterns of geometry or altering the pacing of the level, which can be interleaved with a human designer's actions. These behaviors monitor multiple aspects of the generator in parallel, and their hierarchical nature allows for complex geometry patterns to be built up from simpler components. The geometric relationship between level components is expressed as a set of numerical constraints that must be satisfied, thus ensuring that the design tool will never allow for the creation of an unplayable level. This architecture sits atop a rhythm-based representation for levels, where each beat in the rhythm corresponds to a single action taken by the player.

There were several iterations on the Tanagra tool, with regard to both its UI and the underlying architecture, in an aim to best support designers within the context of simple 2D platforming levels. The figure below shows a sequence of editing operations being performed using a middle-phase iteration on the Tanagra system. This version of Tanagra incorporated a) the concept of a user "pinning" geometry in place by adding numerical positioning constraints, b) the system attempting to minimize the number of required positioning changes (including never being allowed to move pinned geometry), and c) 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.

More information about the Tanagra tool, including a full description of its system architecture and an evaluation of its expressivity, has been published [4].

11.2.4.2 Sentient Sketchbook

The Sentient Sketchbook is a computer-aided design tool which assists a human designer in creating game levels, such as maps for strategy games [13] or dungeons for roguelike games [14]. The Sketchbook uses the notion of map sketch as a minimal abstraction of a full game; this abstraction limits user fatigue while creating new levels and reduces the computational effort of automatically evaluating such sketches. Like popular CAD tools, the 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 the 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 suggestions are either evolved to maximize one of the predefined objective functions inspired by popular game de-

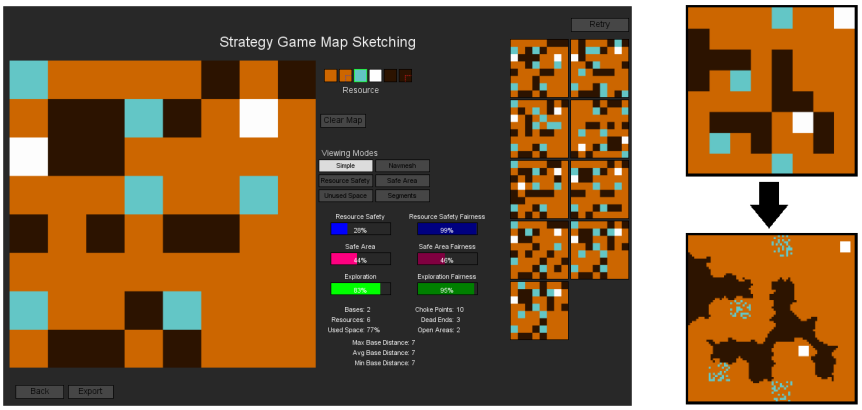

Fig. 11.5: The map sketch can be transformed into a high-resolution map.

sign patterns such as balance and exploration [14], or towards divergence from the user's current sketch through feasible-infeasible novelty search [12].

11.2.4.3 Ropposum

Ropossum is an authoring tool for the generation and testing of levels of the physics-based puzzle game, *Cut the Rope* [22]. 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) optimization of a given design based on playability.

The system constsis of two main modules: an evolutionary framework for procedural content generation [20] and a physics-based playability module to solve given designs [21]. The second module is used both for evolving playable content and for play testing levels designed by humans. The parameters of the evolutionary system and the AI agent were 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) employed by GE. The DG specifies the structure of the levels by defining the positions and properties of the different components of the game and it permits an easy to read and manipulate format by game designers [20].

First-order logic is used to encode the game state as facts specifying the components of the level and their properties (such as position, speed, and moving direction) [21]. The relationships between the components are represented as rules used to infer the possible next actions that can be performed.

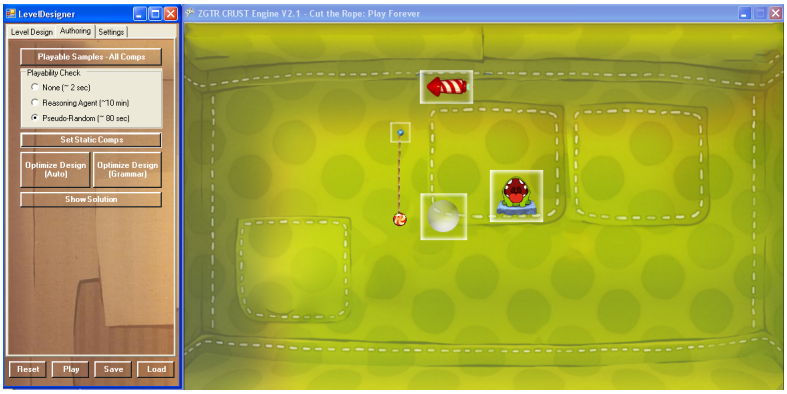

Fig. 11.6: A screenshot from one of the interfaces in Ropossum. The components highlighted are the ones constraint by the designer and therefore will not be changed during evolving complete playable levels.

The two methods for evolving game design and assessing whether the design is playable are combined together in a framework to evolve playable content. An initial level design, according to the design grammar, is generated 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 that 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 was drastically reduced through the use of different priorities for the actions and the employment of domain knowledge encoded by the rules followed by the reasoning agent when inferring the best action to perform.

Figure 11.6 presents a snapshot from the user interface of the tool with a partial level design. The components highlighted are the ones specified by a game designer to be used as constraints by the system when evolving a complete playable level.

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 [24]. 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.

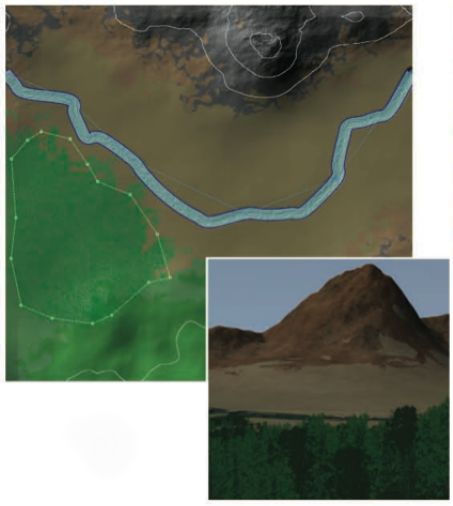

Fig. 11.7: A screenshot from sketchaworld showing the user input consisting of a sketch of a mountain, river and forest and the corresponding 3D terrain generated.

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.

While users sketch on the 2D grid, the effects of their modification is directly visualised in the 3D virtual world. This requires blending the features added with its 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.

Figure 11.7 present a snapshot from sketchaworld interface that illustrate the user sketch of mountains, a river and a forest and the corresponding 3D terrain generated.

11.3 Interactive Evolution

As its name suggests, Interactive Evolution is a variant of artificial evolution 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. 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 optimization task itself. Since it allows for a subjective evaluation of beauty, it is commonly used in evolutionary visual arts [23, 19] and in evolutionary music [6].

11.3.1 User Fatigue and Methods of Combatting 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 number 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.

User fatigue is often induced when the requirement of a large number of selections becomes time-consuming and fatiguing; to combat this factor to user fatigue, content can be evolved *offline* for a number of generations before the next set of individuals is shown to the user, or content can be evolved by more than one user. The first solution requires some form of predicted fitness of unseen individuals, in order to perform a number of generations without requiring any input from the user. 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 which 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 [7]. Such a technique essentially clusters all individuals in the population around the few presented individuals, allowing both a larger population than the number of shown individuals and an offline evolutionary sprint with no human input. Depending on the number of seen individuals and the expressivity of the algorithm's representation, however, a number of strong assumptions are made — the most important of which is for the measure of distance used. In order to avoid extraneous bias of the search from these assumptions, most evolutionary sprints are only for a few generations before new human feedback is required. The second solution requires some form of online database, where users can start evolv-

ing content previously evolved by another user. A good example of this method is PicBreeder, which evolves images created by CPPNs [19]. Since CPPNs become larger and larger with evolution due to the Neuroevolution of Augmenting Topologies algorithm [26], 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[1] allows users to start evolution "from skratch", with a small neural network able to create simple patterns such as circles or gradients, or they can 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 genotypical space and is more likely to have meaningful variations rather than if they were starting from skratch 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 actively interfere directly on the search process by designating an estimated global optimum on a visualization of the search space.

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 side of 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 artificial 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. Alternatively, presented content can be chosen to show a sample of the range of content in the population; this can be accomplished by clustering the population according to a distance function (such as the one used to predict fitness of unseen individuals) and showing a representative example of each cluster, or by showing the fittest and least fit content, with remaining presented content distributed as evenly as possible according to user-defined fitness.

To reduce the cognitive load of evaluating individuals, the most common solution is to limit the number of rating levels. In the simplest of cases, the rating levels can be two, i.e. selected and unselected, limiting the user's effort; they either like

[1] www.picbreeder.org

the content or they don't. Since taste is rarely a boolean value, however, usually more rating levels are included — often using existing scales such as that of "5 'stars" which is popular in restaurant and cinema reviews. Another solution which is expected to limit the cognitive load and subjectivity of ratings is to use rankings [9], i.e. stating preference of A over B without explicitly specifying that A is 3 of 5 stars and B is 1 of 5 stars.

11.3.2 Examples of Interactive Evolution for Games

Recent trends have shown an increase in the use of IEC for evolving game content. As highly interactive experiences themselves, games are ideal for interactive evolution as the user's preference can be inferred based on gameplay traces. In this fashion, the selection of favored content is masqueraded into 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 favored. 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

Fig. 11.8: Galactic Arms Race with multiple players using different weapons.

Galactic Arms Race [5] is one of the more successful examples of games using interactive evolution. The procedurally generated weapon projectiles, which are the main focus of this space shooter game, are evolved interactively based on gameplay data. In Galactic Arms Race, the number of times a weapon is fired is considered an indication of favor; Hastings et al. assume 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 topolgoies [26]; the velocity and color of each particle is defined as the output of a CPPN [25], with the input being the current position and distance from the firing spaceship. 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 the case of multiplayer, the algorithm uses the firing rates of all players when determining which weapons to evolve. The weapons evolved in Galactic Arms Race, and details of its framework have been presented in Chapter 1.

11.3.2.2 TORCS track generation

A more "traditional" form of interactive evolution — with a user directly stating preference of game content — was applied to generate tracks for a car racing game [2]. The system uses the Open Racing Car Simulator[2] (TORCS) and consists of a user interface used to allow interaction with users through a web browser where users can view populations of races and evaluate them (see Figure 11.9). The other component of the system is an evolutionary backend which handles all evolutionary-based operations.

Racing 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 Bezier curves were employed to connect these points and ensure smoothness.

Different variations of the interactive evolution method are implemented to evaluate the generated tracks. In the *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 used as a fitness in the evolution process. In the *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 by all users. The feedback provided by users showed improvements in the quality of the tracks and an increase in their interestingness.

Fig. 11.9: The user interface used to visualise and collect the ranking of tracks.

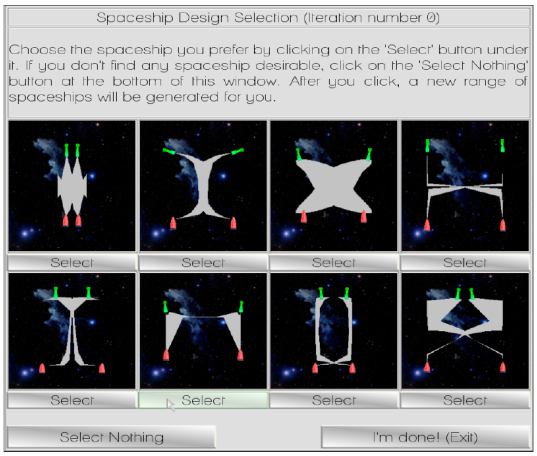

Fig. 11.10: The user interface of the spaceship generator, where the user can select their favorite among a subset of the population of feasible spaceships.

11.3.2.3 Spaceship Generation

The work of [11] is an example of fitness prediction applied for the purpose of speeding up and enhancing the convergence of interactive evolution. This work generates 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 for playability and game balance [10]. The 2D shapes representing the spaceship hulls are encoded as pattern-producing networks (CPPNs) and evolved in two populations using the feasible-infeasible 2-population approach (FI-2pop) [8]. One population contains spaceships which fail ad-hoc constraints pertaining to rendering, physics simulation and game balance, and individuals in this population are optimized towards minimizing their distance to feasibility. The second population contains feasible spaceships, which are optimized 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. This adaptive aesthetic model aims to enhance the visual patterns behind the user's selection and minimize visual patterns of unselected content, thus generating a completely new set of spaceships which more accurately match the user's tastes. A small number of user selections allows the system to recognize their preference, minimizing user fatigue.

The proposed 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, should demonstrate the potential of a flexible tool both for personalizing game content to an end-user's visual taste but also for inspiring a designer's creative task with content guaranteed to be playable, novel and yet conforming to the intended visual style.

11.4 Class 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?

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

(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.

References

1. Carbonell, J.R.: Mixed-initiative man-computer instructional dialogues. Ph.D. thesis, Massachusetts Institute of Technology (1970)
2. 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 13th annual conference on Genetic and evolutionary computation, pp. 395–402. ACM (2011)
3. Engelbart, D.C.: Augmenting human intellect: A conceptual framework. Air Force Office of Scientific Research, AFOSR-3233 (1962)
4. Gillian Smith Jim Whitehead, M.M.: Tanagra: Reactive planning and constraint solving for mixed-initiative level design. IEEE Transactions on Computational Intelligence and AI in Games (TCIAIG), Special Issue on Procedural Content Generation **3** (2011)
5. 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)
6. Hoover, A.K., Szerlip, P.A., Stanley, K.O.: Interactively evolving harmonies through functional scaffolding. In: Proceedings of Genetic and Evolutionary Computation Conference (2011)
7. 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)
8. 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)
9. 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 (CIG) (2013)
10. 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)
11. 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)
12. Liapis, A., Yannakakis, G.N., Togelius, J.: Enhancements to constrained novelty search: Two-population novelty search for generating game content. In: Proceedings of Genetic and Evolutionary Computation Conference (2013)
13. Liapis, A., Yannakakis, G.N., Togelius, J.: Sentient sketchbook: Computer-aided game level authoring. In: Proceedings of ACM Conference on Foundations of Digital Games (2013)
14. Liapis, A., Yannakakis, G.N., Togelius, J.: Towards a generic method of evaluating game levels. In: Proceedings of the AAAI Artificial Intelligence for Interactive Digital Entertainment Conference (2013)
15. Licklider, J.C.R.: Man-computer symbiosis. IRE Transactions on Human Factors in Electronics **HFE-1**, 4–11 (1960)

16. Mateas, M., Stern, A.: A behavior language for story-based believable agents. IEEE Intelligent Systems **17**, 39–47 (2002)
17. Novick, D., Sutton, S.: What is mixed-initiative interaction? In: Proceedings of the AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction (1997)
18. Schon, D.A.: Designing as reflective conversation with the materials of a design situation. Research in Engineering Design **3**, 131–147 (1992)
19. 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 twenty-sixth annual SIGCHI conference on Human factors in computing systems, pp. 1759–1768. ACM, New York, NY, USA (2008)
20. Shaker, M., Sarhan, M.H., Naameh, O.A., Shaker, N., Togelius, J.: Automatic generation and analysis of physics-based puzzle games. In: Computational Intelligence in Games (CIG), 2013 IEEE Conference on, pp. 1–8. IEEE (2013)
21. Shaker, M., Shaker, N., Togelius, J.: Evolving playable content for cut the rope through a simulation-based approach. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (2013)
22. Shaker, M., Shaker, N., Togelius, J.: Ropossum: An authoring tool for designing, optimizing and solving cut the rope levels. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE). AAAI Press (2013)
23. Sims, K.: Artificial evolution for computer graphics. In: Proceedings of the 18th annual conference on Computer graphics and interactive techniques, SIGGRAPH '91, pp. 319–328. ACM, New York, NY, USA (1991)
24. Smelik, R.M., Tutenel, T., de Kraker, K.J., Bidarra, R.: Interactive creation of virtual worlds using procedural sketching. In: Proceedings of eurographics (2010)
25. Stanley, K.O.: Exploiting regularity without development. In: Proceedings of the AAAI Fall Symposium on Developmental Systems. AAAI Press (2006)
26. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10**(2), 99–127 (2002)
27. Team, C.: Choco: an open source java constraint programming library. In: 14th International Conference on Principles and Practice of Constraint Programming, CPAI08 Competition, (2008)