Automatic Content Generation in the Galactic Arms Race Video Game

Erin J. Hastings, Ratan K. Guha, and Kenneth O. Stanley
School of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL 32816
{hastings, guha, kstanley}@cs.ucf.edu

Abstract*—Simulation and game content includes the levels, models, textures, items, and other objects encountered and possessed by players during the game. In most modern video games and in simulation software, the set of content shipped with the product is static and unchanging, or at best, randomized within a narrow set of parameters. However, ideally, if game content could be constantly and automatically renewed, players would remain engaged longer. This paper introduces two novel technologies that take steps toward achieving this ambition: (1) A new algorithm called *content-generating NeuroEvolution of Augmenting Topologies* (cgNEAT) is introduced that automatically generates graphical and game content while the game is played, based on the past preferences of the players, and (2) *Galactic Arms Race* (GAR), a multiplayer video game, is constructed to demonstrate automatic content generation in a real online gaming platform. In GAR, which is available to the public and playable online, players pilot space ships and fight enemies to acquire unique particle system weapons that are automatically evolved by the cgNEAT algorithm. A study of the behavior and results from over 1,000 registered online players shows that cgNEAT indeed enables players to discover a wide variety of appealing content that is not only novel, but also based on and extended from previous content that they preferred in the past. Thus GAR is the first demonstration of evolutionary content generation in an online multiplayer game. The implication is that with cgNEAT it is now possible to create applications that generate their own content to satisfy users, potentially reducing the cost of content creation and increasing entertainment value from single player to massively multiplayer online games (MMOGs) with a constant stream of evolving content.

***Index Terms*—content-generating NeuroEvolution of Augmenting Topologies, cgNEAT, Galactic Arms Race, GAR, particle systems, collaborative content generation, CCE, NEAT, interactive evolutionary computation, IEC**

I. Introduction

CONTENT creation requires the dedicated effort of artists and engineers to produce the models, levels, textures, and other content that populate and bring life to the virtual worlds of video games and simulations. Especially in multiplayer settings such as *massive multiplayer online games* (MMOGs), the constant demand for novel content to keep players interested requires significant time and expense to meet [1], [2]. A recent trend motivated by this problem is to provide the players themselves tools to generate their own content and share it with others [3], [4], [5] or to randomize content (e.g. through random map generators). However, such tools force the player to confront the architecture of the game itself, potentially pulling them out of its fictional context. They also require the players to attain a level of expertise and specialized design knowledge. Furthermore, *content randomization* requires tight constraint to avoid undesirable results and does not intrinsically take into account the preferences of the players. To address the demand for perpetual novel content, this paper argues that evolutionary computation (EC) is a potential solution to the content-generation problem. The aim is to demonstrate that it is indeed possible to generate novel content as the game is played, based on the behavior and interests of the players.

EC methods have proven effective at evolving diverse media such as two-dimensional art images [6], [7], [8], [9], [10], [11], three-dimensional forms [12], [13], [14], and even music [15], [16], [17], [18], [19], [20]. However, the feasibility of such methods for creating video game content is little explored. Currently, machine learning techniques are beginning to be applied in mainstream games (e.g. popular genres such as RTS, RPG, racing, shooters, etc.), but only for non-player character (NPC) controllers [21].

The ability of evolutionary methods to evolve media and NPC controllers provides the basis for the idea of *collaborative content evolution* (CCE) for games. The idea of CCE is that the system automatically generates graphical and game content that is extended from and elaborates upon content players preferred in the past. Content that players like (i.e. use often) is evolved to produce new content, whereas content that players dislike is not evolved. Thus, a continuous stream of novel content is produced that potentially (1) keeps players engaged longer and (2) reduces the content creation burden on developers. The first implementation of CCE in a game is described in this paper.

To make such content generation possible, two novel technologies are introduced: (1) the *content-generating NeuroEvolution of Augmenting Topologies* (cgNEAT) algorithm, and (2) Galactic Arms Race (GAR), a multiplayer video game in which unique particle system weapons are automatically evolved based on content players preferred in the past.

The cgNEAT algorithm aims to automatically generate complex graphic and game content in real-time through an evolutionary algorithm based on the content players liked in the past. To show that automatic content generation is genuinely possible in mainstream games, cgNEAT is implemented in this paper in the novel video game called *Galactic Arms Race*. In GAR, *compositional pattern producing networks* (CPPNs), which are a variant of artificial neural networks (ANNs), genetically encode and control particle system weapons. The CPPNs evolve and increase in complexity though cgNEAT, which tracks which weapons the players fire the most. That way, during the game, weapon behavior becomes increasingly sophisticated while consistently evolving to suit player tastes. Thus, it is the *player* rather than the designer who ultimately implicitly determines what kind of content will populate the game.

A. Main Contributions

The major contribution of this work is to show how collaborative content evolution based on player preferences can run seamlessly across a multiplayer client-server architecture, thereby suggesting the potential for eventual implementation of CCE technology across the spectrum of games ranging from single-player to MMOGs. In GAR, novel weapons are evolved on the server and distributed to clients, which collect usage statistics and send them back to the server for further evolution. In the public experiment described in this paper, literally hundreds of thousands of weapons were evolved and over *one million* virtual aliens killed by players across the world participating online. In principle, such a process can evolve any class of parameterized content. Thus the generality of the approach means it can impact future commercial game production and increase the longevity of games that might otherwise become repetitive. In this way, the results in this paper open up a promising new direction in video game design and research.

The paper proceeds as follows. First, Section II covers background material related to the technologies developed in this paper. Next, the cgNEAT algorithm is described in detail in Section III. Then, Section IV demonstrates cgNEAT in practice through evolving weapons in the Galactic Arms Race video game, and the outcome of both single player and multiplayer weapon evolution experiments are described in Section V. Finally, overall implications are discussed and the paper is concluded in Sections VI and VII.

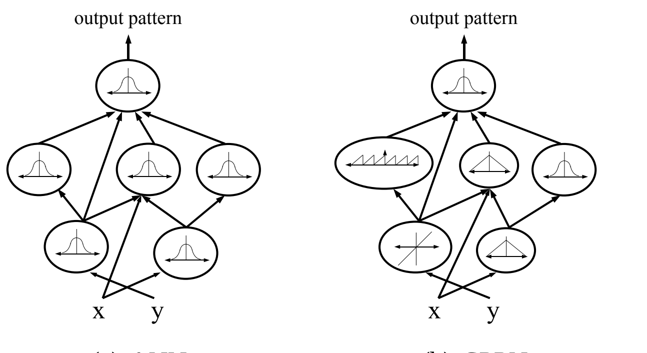
II. Background

This section reviews background material that inspired and enabled the main contributions of this paper (cgNEAT and GAR).

A. Interactive Evolutionary Computation (IEC)

IEC is an approach to evolutionary computation (EC) in which human evaluation replaces the fitness function [22]. A typical IEC application presents to the user the current generation of content. The user then interactively determines which members of the population will reproduce and the IEC application automatically generates the next generation of content based on the user's input. Through repeated rounds of content generation and fitness assignment, IEC enables unique content to evolve that suits the user's preferences. In some cases such content would be otherwise difficult to discover.

IEC aids especially in evolving content for which fitness functions would be difficult or impossible to formalize (e.g. for aesthetic appeal). Thus, graphical content generation is a common application of IEC [13], [14], [23], [24], [25].



(a) ANN (b) CPPN

Fig. 1. **ANNs and CPPNs**. Unlike traditional ANNs (a), which typically only have sigmoid or Gaussian activation functions, CPPNs (b) may have sigmoids, Gaussians, and many other activation functions in the same network. The ability of CPPNs to encode patterns makes them useful for representing graphical content.

IEC was first introduced by Dawkins in Biomorphs [26], which aims to illustrate theories about natural evolution with interactively evolved abstract figures that resemble animals or plants. Since then, representations in *genetic art* (i.e. IEC applied to art [6], [7], [8], [9], [10], [11]) have varied widely, including L-systems, linear or non-linear functions, fractals, and automata, to produce a broad variety of appealing two-dimensional and three-dimensional images and animations.

IEC is demonstrably effective for evolving graphical content. An important prerequisite to evolving such content is that it must be represented by evolvable structures. The evolvable structures utilized in this paper are CPPNs, a specialized type of ANN.

B. Compositional Pattern-Producing Networks (CPPNs)

CPPNs are a variation of artificial neural networks (ANNs) that differ in their set of activation functions and how they are applied [27], [28] (figure 1). While CPPNs are similar to ANNs, the different terminology originated because CPPNs were introduced as pattern-generators rather than as controllers. This section explains the difference in implementation and application between CPPNs and ANNs.

While ANNs often contain only sigmoid or Gaussian activation functions, CPPNs can include both such functions and many others. The choice of CPPN functions can be biased toward specific patterns or regularities. For example, periodic functions such as sine produce segmented patterns with repetitions, while symmetric functions such as Gaussian produce symmetric patterns. Linear functions can be employed to produce patterns with straight lines. In this way, CPPN-based systems can be biased toward desired types of patterns by carefully selecting the set of available activation functions.

Additionally, unlike typical ANNs, CPPNs are usually applied across a broad space of possible inputs so that they can represent a complete image or pattern. Because they are compositions of functions, CPPNs in effect encode patterns at infinite resolution and can be sampled at whatever resolution is desired.

Successful CPPN-based applications such as Picbreeder [24], in which users from around the Internet collaborate to evolve pictures, and NEAT Drummer [20], which evolves drum track patterns to accompany songs, demonstrate that CPPNs can evolve diverse content. The approach in this paper evolves particle systems encoded by CPPNs with the NEAT algorithm.

C. NeuroEvolution of Augmenting Topologies (NEAT)

The NEAT method was originally developed to solve control and sequential decision tasks. The ANNs evolved with NEAT control agents that select actions based on their sensory inputs. While previous methods that evolved ANNs (i.e. neuroevolution methods) evolved either fixed topology networks [29], [30], or arbitrary random-topology networks [31], [32], [33], NEAT begins evolution with a population of small, simple networks and *complexifies* the network topology into diverse species over generations, leading to increasingly sophisticated behavior. A similar process of gradually adding new genes has been confirmed in natural evolution [34], [35] and shown to improve adaptation in a few prior evolutionary [36] and neuroevolutionary [37] approaches. This section briefly reviews the NEAT method; Stanley and Miikkulainen [21], [38] provide complete introductions.

To keep track of which gene is which while new genes are added, a *historical marking* is uniquely assigned to each new structural component. During crossover, genes with the same historical markings are aligned, producing meaningful offspring efficiently. Traditionally, speciation in NEAT protects new structural innovations by reducing competition between differing structures and network complexities. However, in the work in this paper, because a human performs selection rather than an automated process, the usual speciation procedure in NEAT is unnecessary.

Most importantly, complexification, which resembles how genes are added over the course of natural evolution [34], allows NEAT to establish high-level features early in evolution and then later elaborate on them. For evolving content, complexification means that content can become more elaborate and intricate over generations.

In this paper, particle system weapons are controlled by CPPNs evolved by NEAT. NEAT is chosen because (1) it is proven effective for evolving ANNs and CPPNs in a diversity of domains [38], [39], [40], and (2) it is fast enough to run in real time (in the NERO video game [21]), which is required for an interactive system. Because NEAT is a strong method for evolving controllers for dynamic physical systems, it can naturally be extended to evolve the motion of particle effects as well, such as those featured in GAR.

D. Machine Learning in Existing Games

The impact of machine learning so far on the video game industry has been limited, although some games are beginning to incorporate learning techniques. However, content generation continues to be absent from applications of machine learning in commercial games. The most common application of machine learning is to optimize the policy that controls non-player characters (NPCs) (figure 2). For example, the ANN race car controllers in Colin McRae Rally 2.0[1] and Forza Motorsport 2[2], and the creature brains in Creatures 3[3] and Black and White 2[4] are learned. Generally, the NPC behavior in such games is trained by developers before release. Recently, although it is not a commercial game, NeuroEvolving Robotic Operatives (NERO [21]; http://nerogame.org/) enabled players to evolve the tactics for a squad of virtual soldiers in real-time, while the game is played, demonstrating the potential viability of evolution in commercial gaming.

The success of these games suggests the potential to apply machine learning to create content beyond NPC behavior (e.g. maps, items, weapons, etc.). In fact, automatically generating content could further open the video game industry to the possibilities created by machine learning.

E. Procedural Game Content in Existing Games

Since the early days of interactive digital entertainment, games such as Rogue[5] [41] and NetHack[6] have featured randomized map and content placement. These games inspired an entire sub-genre of RPGs known as *Roguelikes* [42], which later directly influenced randomized map and item generation in more modern games, such as the popular Diablo[7] series and Dwarf Fortress[8], which features complex interaction among hundreds of different objects placed in a procedurally-generated game world.

While the path of content generation in these games is determined before the game starts, recent research has investigated enabling players to directly influence the path of content generation during the game. Examples of released such games include Charbitat [43], which traces player behavior and translates it into seed values for future game space generation, and Facade[9] [44], which generates interactive stories based on player choices throughout the game.

These contributions demonstrate successful content generation, but they do not attempt to discern whether the player enjoys the content generated, nor do they obviously extend to a shared multiplayer environment. However, the are successful online collaborative IEC art services [24] that incorporate both features, suggesting the potential of an evolutionary approach to multiplayer procedural game content generation.

F. Evolving Game Content

Evolving procedural game content is an emerging research area with great potential to contribute to the mainstream gaming industry. Some of the few current examples of evolved game content include race tracks [45] and even the rules of the game itself [46], [47].

[1]Copyright 2001 Codemasters, http://www.codemasters.com/
[2]Copyright 2007 Microsoft Game Studios, http://forzamotorsport.net/
[3]Copyright 2004 Creature Labs, http://www.gamewaredevelopment.co.uk/
[4]Copyright 2005 Lionhead Studios, http://www.lionhead.com/
[5]Copyright 1983 Artificial Intelligence Design
[6]Copyright 1983 Stichting Mathematisch Centrum
[7]Copyright 1996, 2000 Blizzard Entertainment, http://blizzard.com/
[8]Copyright 2002 Bay 12 Games, http://bay12games.com/
[9]Copyright 2005 Procedural Arts, http://proceduralarts.com/

Fig. 2. **Evolving NPC Behavior in Existing Games**. Learned policies enable race car controllers to navigate tracks with complex physics in (a) Colin McRae Rally 2.0 and (b) Forza Motorsport 2. Learned policies also control decision making for a variety of characters in (c) Creatures 3 and (d) Black and White 2. In (e) NERO [21], players evolve a squad of virtual soldiers to fight other players. NERO introduced rtNEAT, which demonstrated the viability of NPC evolution in real time. Building on the success of these games, cgNEAT aims to evolve other forms of game content, outside of NPC behavior.

To evolve the race track [45], ANN drivers are evolved by comparing their performance to human controllers on the same track. Then new tracks are generated and the ANN controllers are evaluated on those tracks. The selected tracks are those upon which the ANN controllers perform similarly to how they perform on other tracks, under the assumption that they will provide an appropriate level of challenge to human players.

In a different example [46], the rules of the game itself, rather than NPC controllers that play a specific game well, are evolved. Evolution begins with a grid-like environment containing random sets of walls, an ANN-controlled NPC representing the player, and various game objects that can alter state when the ANN-controlled NPC collides with them. State changes include death, teleportation, bonuses, goals, and other typical two-dimensional game effects. The fitness of the evolved games is based upon the amount of learning the controller must undergo to beat the game. This idea was shown to successfully evolve *Pac-Man*-like games.

Recent works begin to bridge the gap between evolutionary art and games. In Avera [47], [48], the system evolves interactive art pieces for simple puzzle games. In another example [49], players interact with complex swarm systems through an IEC interface, enabling search for well-performing swarm configurations.

These investigations thus represent the cutting edge of an exciting new research direction. Inspired by these works, the aim of cgNEAT is to evolve game content in real time, based on tracked player preferences in a multiplayer setting through a process called *collaborative content evolution* (CCE), as illustrated in figure 3.

CCE is a new concept that has not, until this paper, been implemented in a game. In short, players begin the game with an initial set of content. If players use some of the content often, it is inferred that they enjoy that content, and the game produces new content that extends from or elaborates on that preferred content. However, if players are unhappy with certain content, they will not use it (or may discard it); thus the game will not produce more content of that type. The aim of this process is to continually evolve content based on the preferences of the players. The main type of content evolved in this paper is particle systems.

G. *Particle Systems*

The first computer-generated particle system in commercial computer graphics, called the *Genesis Effect*, appeared in Star Trek II: The Wrath of Khan [50]. Soon after, particle system effects became widespread on television as well. Nearly all modern video games include a particle system engine [51], [52]; special effects in games such as magical spells and futuristic weapons are usually implemented with particle systems.

In addition to diffuse phenomena such as fire, smoke, and explosions, particle systems can also model concrete objects such as dense trees in a forest [53], folded cloth and fabric [54], [55], and simulated fluid motion [56], [57]. Realistic particle movement is often achieved by simulating real-world physics [58]. At a more abstract level, particle systems can simulate animal and insect flocking behavior [59]. The prevalence and diversity of particle system applications demonstrates their importance to computer graphics in modern media and games.

The evolving particle system weapons in this paper (detailed later in Section IV) are based on previous experiments first reported in Hastings, Guha, and Stanley [60], namely, NEAT Particles, an IEC tool that enables users to evolve complex particle effects, and NEAT Projectiles, an IEC tool that evolves effects intended specifically for video game weapons. In both applications, CPPNs represent and control particles and an IEC interface enables the user to guide evolution.

The next section introduces the algorithm for evolving content in this paper.

III. Content Generating NeuroEvolution Of Augmenting Topologies (cgNEAT)

The aim of the cgNEAT algorithm is to automatically generate computer graphics and video game content based on user behavior as the game is played. While there are technologies for evolving content like pictures [6], [7], [8], [24] or three-dimensional models [12], [13], [14], these technologies are not designed to work in real time during a game; rather they require users to explicitly designate which items are the best, which is something that a user playing a game does not want to do. That is, constantly answering questions about what they like and what should be produced in the future would disrupt players' experience. In contrast, the cgNEAT method makes these decisions automatically based on implicit information within usage statistics.

A. *Algorithm Overview*

The main principles of cgNEAT follow:

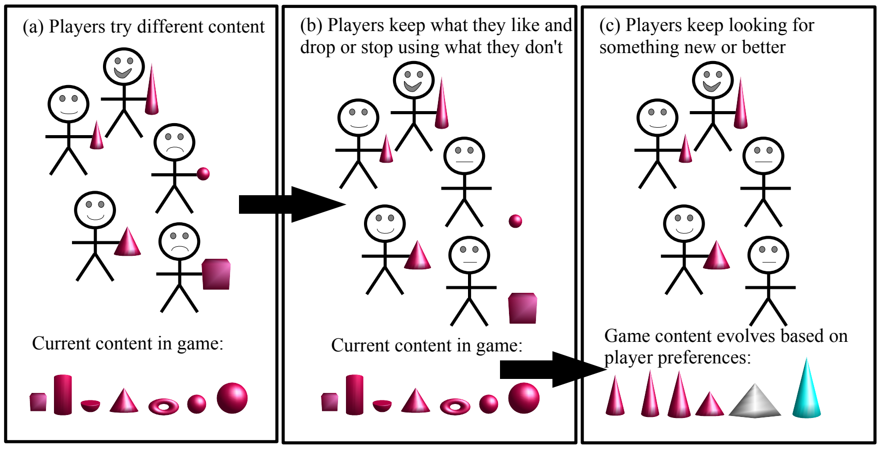

(a) Players try different content
Current content in game:
(b) Players keep what they like and drop or stop using what they don't
Current content in game:
(c) Players keep looking for something new or better
Game content evolves based on player preferences:

Fig. 3. **Illustrating Collaborative Content Evolution (CCE) in Games**. The main idea in CCE is that content evolution in games begins with a diverse population of randomized content (left). As players explore the world and discover new content, they likely keep content with which they are satisfied and discard that with which they are not (center). As evolution continues, content that is widely disliked filters out of the game (right) and content that players enjoy becomes the parents of new generations of content. In this way, players continually explore a succession of changing content. Note that, at the time of writing, no published game has implemented CCE evolution except Galactic Arms Race, which is detailed in this paper.

1) Each content item is represented by a CPPN. Different types of content can be represented by different CPPN input/output configurations (the specific representation for particle weapons is described later). In principle, a different representation than CPPNs can also be evolved.
2) During the game, each content item is assigned a fitness that is computed based on how often players actually *use* the content. That way, the system knows the relative popularity of each content item currently in the game.
3) Players begin the game with either (1) random content or (2) content from the *starter pool*, which is a special pool of content appropriate to beginners in the game. Starter pool content *does not* contribute to evolution and cannot be selected for reproduction.
4) Content is spawned in the game world, which means that it is placed in parts of the world where users can obtain it. However, unlike in most evolutionary systems, spawned content is not eligible for reproduction until players pick it up.
5) Content is reproduced in cgNEAT as follows: The algorithm selects content items from among content that players in the world *already possess* as parents that reproduce to form new content, which is spawned as described in step 4. The content items that are chosen as parents are selected probabilistically based on a roulette wheel scheme in which the chance of being chosen as a parent is proportional to the popularity (i.e. fitness) of the item. Reproduction, including mutation and crossover, is performed in accordance with the NEAT algorithm. Thus, there is a chance that CPPNs may become more complex than their parents.
6) For any new content that is spawned, there is a probability (selected by the designer) that it will be chosen from a *spawning pool*, which is a collection of pre-evolved content, instead of being reproduced from parents. This pool ensures that diversity is not lost and that good types of content from the past (i.e. those that users liked) might reappear. Additionally, it ensures an initial seed of good content when the game first starts and players' preferences are unknown. The game designers initially select content, which may be pre-evolved before the game is released, to include in the spawning pool.
7) Content that obtains a very high fitness (i.e. is popular with players) may optionally be saved to the *content archive*. There are several ways game designers can use archive material including (1) data analysis, (2) cycling it back into the game by adding it to the spawning pool, or (3) giving it to NPCs for use against players.

Finally, note that for cgNEAT in a multiplayer environment, although all players' preferences directly contribute to the course of evolution, the content generated is not an "average" of all player preferences. Rather, unique content is reproduced on a individual basis from individual items that are popular. Thus several diverse trends can flourish simultaneously.

B. Unique Features

The cgNEAT algorithm incorporates some mechanics of NEAT and standard evolutionary computation (EC), yet exhibits several major differences. Unlike in normal EC, the population size (i.e. those items that are eligible at any given time to reproduce) is variable and depends entirely on the number of users in the system. Furthermore, when an offspring is produced, unlike in normal evolutionary computation, it is not immediately placed into the population eligible to reproduce. Instead, it is in a special temporary state (placed somewhere in game world) in which it may join the population only if a user chooses to acquire it. Also unlike normal evolutionary computation, instead of fitness determining which items are eliminated from the population, users entirely determine which items leave the population simply by discarding them.

Unlike standard interactive evolutionary computation (IEC [22]), users never explicitly communicate to the system which content they like. Instead, the preferred content is induced by the system implicitly from natural human behavior. That is, users do not need to know that they are interacting with an evolutionary algorithm yet evolution still works anyway.

Unlike regular NEAT, speciation is not necessary because users determine what is popular and the diversity of the population reflects the diversity of user preferences. Every step of the cgNEAT algorithm is asynchronous. At any time players may cause content to join the population or be eliminated.

C. Starter Pool, Spawning Pool, and Archive Pool

At the beginning of the game or simulation, players must often be provided content with which to start. One approach is to give players randomized content when they start; however, if high-quality content is not a sufficient proportion of the search space, new players might then begin with ineffective content, thereby creating a poor first impression of the game.

The cgNEAT approach is to define a *starter pool* of known effective content items with which players begin the game. Because all players begin the game with it, starter pool content does not contribute to evolution and cannot be selected for reproduction. However, starter pool content does gain fitness, and if during evolution a starter pool item is selected for reproduction, a spawning pool content item is randomly selected instead to spawn into the game world.

The *spawning pool* is a pre-evolved set of content that is known to be effective and can be distributed with the game or application. The primary reason for including the spawning pool is that, at the beginning of the game, there is no past data on player preferences. Thus the spawning pool content effectively jump starts evolution on a promising course. Like the starter pool, the spawning pool helps guarantee a good initial impression of the game to new players, which could be difficult to accomplish with purely random starting content. Additionally, if NPCs in the game are equipped with evolved content, arming them with spawning pool content can ensure an appropriate level of NPC difficulty.

Finally, the *archive pool* consists of all content in the game that achieves a high level of fitness. When such content reaches a certain level of fitness (which by necessity varies among applications), it is automatically saved to the archive pool.

D. Applying cgNEAT

The general approach of cgNEAT described in this paper can be applied to evolving many types of content (e.g. images, models, shader effects, etc.) through other forms of representation (i.e. evolvable structures other than CPPNs). Thus the algorithm is not exclusive to the type of content presented in this paper. Note that the starter pool, spawning pool, archive pool, and roulette parameters can give game designers considerable control of over the course of evolution, should they so desire it. The next section details the first application of cgNEAT in practice, which is to evolve weapons in the Galactic Arms Race video game.

IV. Galactic Arms Race (GAR)

This section introduces the Galactic Arms Race multiplayer video game (available at http://gar.eecs.ucf.edu), which applies cgNEAT to evolve an endless array of unique particle system weapons, with which players can fight space enemies and up to 31 other simultaneous players online. GAR is the first game to enable CCE of novel game content.

A. Development

GAR is intentionally designed to look and feel like a near-commercial quality video game so that it can convincingly demonstrate the promise of automatic content generation for mainstream games. To reach that level of quality, it took over a year to build by a nine-member mostly student team. GAR was first released on June 2, 2009 and is available online at http://gar.eecs.ucf.edu. The *GAR Client 1.1* contains 36,820 source lines of code (SLOC) and over 90MB of three-dimensional models, two-dimensional texture art, music, and sound effects. Additionally, for multiplayer modes, the *GAR Server 1.1* contains 6,796 SLOC and the *GAR Master Server 1.1* contains 873 SLOC.

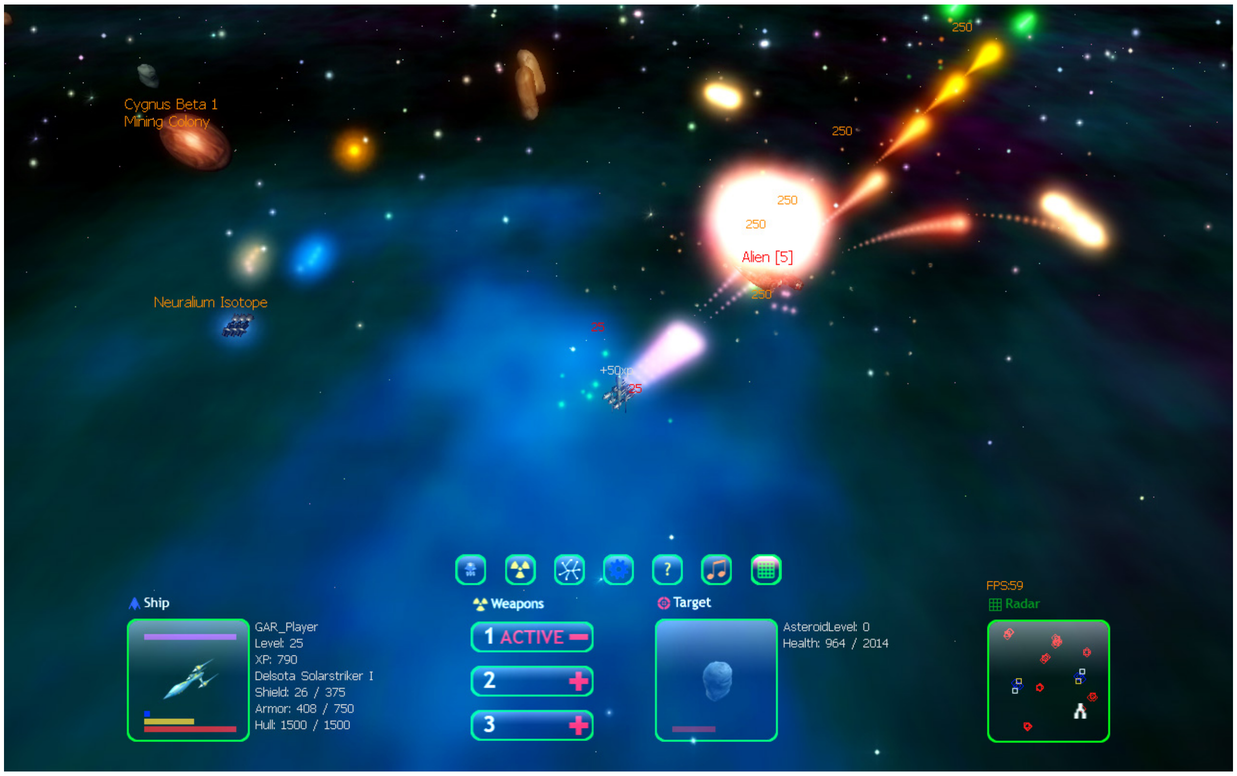
B. Game Mechanics Overview

In GAR (figure 4), the goal is to pilot a space ship to defeat enemies, gain experience, earn money, and most importantly, to find advantageous new weapons that are automatically generated by cgNEAT.

GAR contains both a single player game and a full multi-player game (figure 5), in which weapons evolve based on the aggregate usage of all players online. In GAR's single-player mode, evolution is directed by the actions of a single player battling NPC aliens in the game, which are controlled by scripted steering behaviors [58] and the BOIDS algorithm [59]. The GAR multi-player game enables up to 32 players to fight cooperative online battles against NPCs, or competitive battles against each other. GAR multiplayer evolution is substantially more diverse because the evolutionary population consists of the weapons currently possessed by all players in the game (i.e. it is CCE).

Every weapon found in GAR that is the offspring of other weapons is unique, that is, no more than one copy of any offspring is spawned. Players can continually find novel weapons with characteristics evolved from those weapons players favored in the past. It is important to note that weapons evolved in GAR all fire particle bursts with the same strength and number. Thus it is not sheer power that is evolving, but rather the pattern in which particles spray from the gun, which has complex tactical implications. Therefore, the space of weapons is not a total order from worst to best, but rather a complex multi-objective coevolutionary landscape.

Players are limited to three *weapon slots*, each of which holds a single weapon. Destroyed enemies and enemy bases may drop a *weapon pickup* that contains a novel weapon evolved by cgNEAT. Players choose in which weapon slot to place the new weapon, but doing so drops the existing weapon in that slot. Thus players must be selective about



GAR_Player
Level: 25
XP: 790
Delsota Solarstriker I
Shield: 26 / 375
Armor: 408 / 750
Hull: 1500 / 1500

AsteroidLevel: 0
Health: 964 / 2014

Fig. 4. **GAR Client**. Players in GAR pilot their space ship (screen center) from a third-person perspective. This picture demonstrates a player destroying enemies with an evolved weapon. Left of the player ship is a weapon pickup dropped from a destroyed enemy base. A particle system preview emits from the weapon pickup (i.e. "neuralium isotope," left of player) to visually indicate how the weapon will function before the player picks it up. GAR is designed to look and feel like a near-commercial quality video game to effectively demonstrate the potential of automatic content generation in mainstream games. The GAR Client software is available online at http://gar.eecs.ucf.edu and runs on any Windows PC.

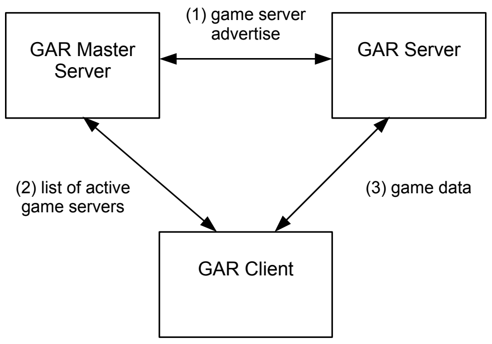


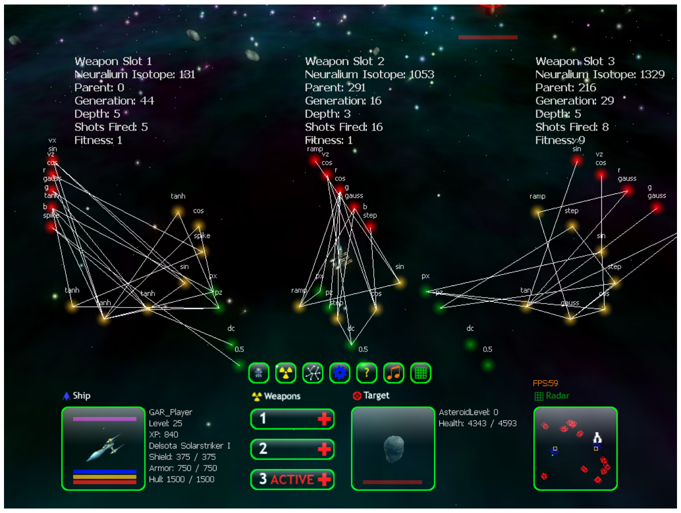
Fig. 5. **GAR Multiplayer Architecture**. This diagram illustrates the networked communication between the GAR Client, Server, and Master Server. (1) When GAR Servers are online, they periodically advertise to the Master Server that indexes all currently active game servers. (2) When a player wishes to join an Internet game, a list of all active game servers is retrieved from the Master Server. (3) The player then chooses from the list of active Internet games and connects to the desired Game Server.

which weapons to keep. In this context, an important goal for any game that generates unpredictable content is to indicate what that content will be like before it is taken. To give players an idea of how a weapon functions before picking it up, weapon pickups emit a miniature particle system preview that behaves exactly as the actual weapon does. In the game this preview is called a *neuralium isotope* (figure 4, left side).

As with most video games, GAR relies on graphical user interfaces (GUIs) for many of its functions. While there are many GUIs in GAR, the major interfaces through which players interact with weapon evolution are the in-game *HUD* (head-up display) and the *Weapons Screen*.

The GAR **in-game HUD** (bottom of figure 4) is always displayed while the player is in space. The HUD displays (1) current player ship armor, shield, and hull strengths, (2) teleport cooldown status, (3) current target and its status, (4) the weapons bank (which enables selecting an active CPPN weapon), and (5) a radar that displays objects in the local star system. The HUD also offers buttons to access to the following screens: (1) the ship screen for detailed ship statistics, (2) the weapons screen with the player's currently equipped CPPN weapons, (3) the galactic map screen, (4) the logoff screen, which exits the current game, (5) the help screen, (6) a music button that toggles the background music on or off, and finally, (7) the mission menu, which displays the player's current mission.

The **in-game weapons screen** (figure 6) displays statistics and CPPNs for the three weapons currently held by the player. In the game's lore, CPPNs are called "neuralium isotopes," each representing a unique isotope of the strange neuralium element that powers weapons and ships. The CPPNs are displayed in three dimensions and slowly rotate. Taking advantage of the three-dimensional rendering view, CPPN inner nodes are aligned in a circle, making viewing them easy and aesthetically pleasing. In effect, players can actually visualize the real CPPNs that realize their weapons.


Weapon Slot 1
Neuralium Isotope: 131
Parent: 0
Generation: 44
Depth: 5
Shots Fired: 5
Fitness: 1

Weapon Slot 2
Neuralium Isotope: 1053
Parent: 291
Generation: 16
Depth: 3
Shots Fired: 16
Fitness: 1

Weapon Slot 3
Neuralium Isotope: 1329
Parent: 216
Generation: 29
Depth: 5
Shots Fired: 8
Fitness: 9

GAR_Player
Level: 25
XP: 840
Delsota Solarstriker I
Shield: 375 / 375
Armor: 750 / 750
Hull: 1500 / 1500

AsteroidLevel: 0
Health: 4343 / 4593

Fig. 6. **GAR Weapons Screen**. The GAR weapons screen displays information on the player's three current weapons, including fitness and number of shots fired. The actual CPPNs for the weapons are displayed as three-dimensional graphs (known in the game as "neuralium isotopes").

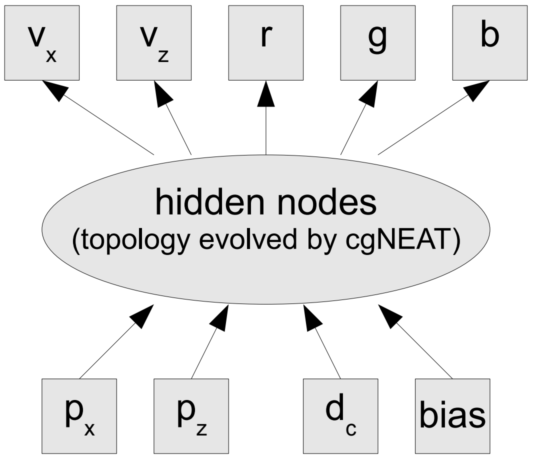


Fig. 7. **How CPPNs Represent Particle Weapons**. For each frame of animation, every particle separately inputs the position (p_x, p_z) and distance (d_c) from where it was *initially* fired into the CPPN (p_y is ignored because the game is situated entirely on the $y = 0$ plane). (b) The CPPN is activated and particle velocity (v_x, v_z) and color (r, g, b) are obtained from CPPN outputs. This method provides GAR with smooth particle animations and a wide variety of possible evolved weapons.

C. Particle System Weapon CPPNs

Particle system CPPNs in GAR are based on the techniques developed in NEAT Particles and NEAT Projectiles [61]. Each player weapon contains a single evolved CPPN (figure 7). Every frame of animation, each particle issued from the weapon inputs its current position relative to the ship (p_x, p_z) and distance from the ship (d_c) into the CPPN. There are two, rather than three, spatial inputs because the game is entirely situated on the $y = 0$ plane. The CPPN is activated and outputs the particle's velocity (v_x, v_z) and color (r, g, b) for that animation frame. Representing particle velocity and color in this manner produces a wide of variety of vivid patterns [61]. It is important to note that because CPPNs are a superset of ANNs, which can approximate any function [62], particle weapons in GAR can theoretically evolve any conceivable pattern.

When GAR was first released, it imposed a forward force on all projectiles (as in NEAT Projectiles) to ensure that all weapons shoot only forwards. However, after further experimentation, it was determined that weapons with particles that move backwards can create compelling patterns (e.g. hurricanes), so the constraint was later lifted.

D. Calculating Weapon Fitness

Two potential challenges to calculating fitness based on usage are that (1) certain weapons, by their nature, require more shots to be effective (e.g. wall guns for blocking incoming projectiles) and (2) players that participate in the game more often might disproportionately influence evolution (i.e. by firing their weapons more often). To address these two issues fitness is calculated in GAR in the following manner.

When a player fires a weapon, that weapon (which is a unique member of the population) gains fitness at a constant rate and the other weapons in that player's arsenal lose fitness at the same rate. This *fitness decay* mechanism for unused weapons emphasizes emerging new weapon trends and ensures that weapons that require more firing do not come to dominate. Furthermore, the *minimum fitness* is 1 and the *maximum fitness* is 1,000, which means that older players do not create a disproportionate effect.

E. Evolving New Weapons

When players destroy an enemy station or space blob (a kind of boss enemy), a new weapon is spawned in one of the following ways: (1) reproduction within the current weapon population selected by roulette [63], (2) from the spawning pool, or (3) random generation. The probabilities for each to occur in GAR are:

1) 80%. A roulette based on weapon fitness is spun to determine which specific weapon will reproduce. If a starter weapon is selected by the roulette, then a spawning pool weapon is reproduced instead. Higher fitness weapons have a higher chance of producing offspring, thereby enabling players to directly affect the course of evolution. Note that there is no recombination (e.g. combining guns), although in principle it could be implemented in a future version.
2) 10%. The spawning pool is a set of pre-evolved weapons chosen by the game designers.
3) 10%. Random weapons have between one and four hidden nodes and random weights.

Novel weapons created by cgNEAT are evolved from the current *weapon population*. In single-player GAR, the weapon population is only the three weapons the player currently possesses. In multiplayer GAR, the weapon population includes the weapons currently held by all players. Thus single-player evolution is to some extent greedy; however, it is not equivalent to a normal evolutionary algorithm with a population of three because the player encounters a significant number of weapon previews *in addition* to the weapons in the ship's current arsenal. Therefore, the player is in effect judging such previews by taking them or not. Furthermore, the spawning pool ensures a diverse set of jumping-off points are injected at regular intervals.

As results in this paper show, the net effect is that a single player can genuinely discover a diverse array of highly specialized and effective weapons. Because in multiplayer mode the population includes every weapon held by every player in the game, multiplayer mode in GAR is genuine CCE. Figure 8 illustrates weapon evolution in GAR with two genealogies of related weapons. To give a sense of the changes produced through mutations from one generation to the next, figure 9 presents a chronological lineage with its corresponding CPPNs.

F. Starter Weapons, Spawning Pool, and Archive Pool

The starter weapons (Section III-C) in GAR shoot only in a straight line and are not eligible to reproduce during evolution (figure 10). Thus new players are guaranteed to begin with viable weapons. Because starter weapons cannot reproduce and players begin the game with only starter weapons, a method is needed to start evolution. For this purpose, the *spawning pool* is a diverse collection of good weapons evolved by the game developers. If cgNEAT selects a starter weapon to reproduce because it is fired often, a random spawning pool weapon is spawned instead. Finally, the *archive pool* is where popular weapons are saved. In GAR weapons are saved to the archive if they have been fired over 800 times. In multiplayer mode they are saved to the host server, creating a history of popular weapons in the game.

V. Automatic Content Generation Results in GAR

The results presented in this section demonstrate the variety of weapons that are evolved by players in GAR. All weapons displayed are created by the cgNEAT algorithm itself, i.e. not by the game developers. The results of both single player and multiplayer weapon evolution are presented.

Results reported in this paper would not have been possible without public interest in such a project, which attracted players to the game. Since before GAR was released, the http://gar.eecs.ucf.edu website (where GAR can now be downloaded for free) has attracted over 30,000 visitors from over 100 different countries. On July 8th, 2009, GAR appeared on the popular Internet news site Slashdot[10], highlighting the public's general interest in automated content generation and attracting many players to GAR from around the world.

Two versions of the GAR client were released: GAR 1.0 on June 2, 2009 and GAR 1.1, which addressed many feature requests by players, on July 6th 2009. GAR (mostly version 1.1) has been downloaded over 8,500 times. The results presented in this section are based on evolution data generated by actual players who downloaded and played the game.

[10]http://games.slashdot.org/story/09/07/08/1419242/Experimental-Video-Game-Evolves-Its-Own-Content

A. Automatic Content Generation Results in GAR Single Player Mode

To investigate the content generation abilities of cgNEAT in collaboration with human players, before the public release of the game a group of over twenty test players piloted space ships in single-player mode for at least one hour each. The results in this section (figure 11) are from these test sessions. Of course, single-player evolution is inherently more limited than multiplayer. However, the main result is that players indeed discovered a variety of genuinely unique weapons with differing tactical implications and aesthetics, suggesting that even the behavior of a single player is sufficient to produce content evolution.

As the weapons showcased in this section will show, the gameplay implications of evolved content sometimes seem intentional, as if designed purposely to create a specific capability. In many cases guns were invented that were unlike anything the developers had seen or imagined before. Yet of course these guns are not the result of random luck either; just as in other evolutionary algorithms, they result from selection pressure, which is wrought by the preferences of the player in GAR. In this way, GAR is a credible demonstration of the potential of this approach.

In GAR it is possible for player projectiles to intercept enemy projectiles. Therefore, several key tactical trade-offs are explored by evolution. Slow projectiles make it easier to block incoming fire whereas fast projectiles are easier to aim at distant enemies. Weapons with a wide spread are more effective at blocking incoming projectiles; however, concentrated patterns more effectively destroy distant targets quickly. Hybrid weapons with variable spread pattern and speed over time evolve as well. Yet these tactical principles are only the beginning. In fact, figure 11 presents samples of the wide range of generated single-player weapons and describes some of their tactical implications. To highlight the creativity of cgNEAT, we have assigned descriptive names to each such gun to help to more easily appreciate their concept. Two especially interesting evolved weapon types are *wallmakers* (figure 11c), which literally create a wall of particles in front of the player, and *tunnelmakers* (figure 11e), which create a line of particles on either side of the player. Both weapon types are defense-oriented, enabling players to switch between them and more offense-oriented weapons, as the tactical situation dictates. These examples demonstrate that cgNEAT evolves unique and tactically diverse weapons as the game is played.

It is important to point out that it does not take long for players to begin to find effective weapons. As figure 11 shows, compelling weapons often arise within the first ten generations (e.g. the *tunnelmaker* in figure 11e is from generation two). Furthermore, weapons continue to evolve into novel forms over dozens of generations, such as the *blue ladder* (figure 11f) from generation 42.

Finally note that, although the population eligible for reproduction is only three in the single-player game, cgNEAT is still capable of generating a large variety of novel content for two reasons. First, players are exposed to many weapons (aside from those they currently possess) by previews of weapons

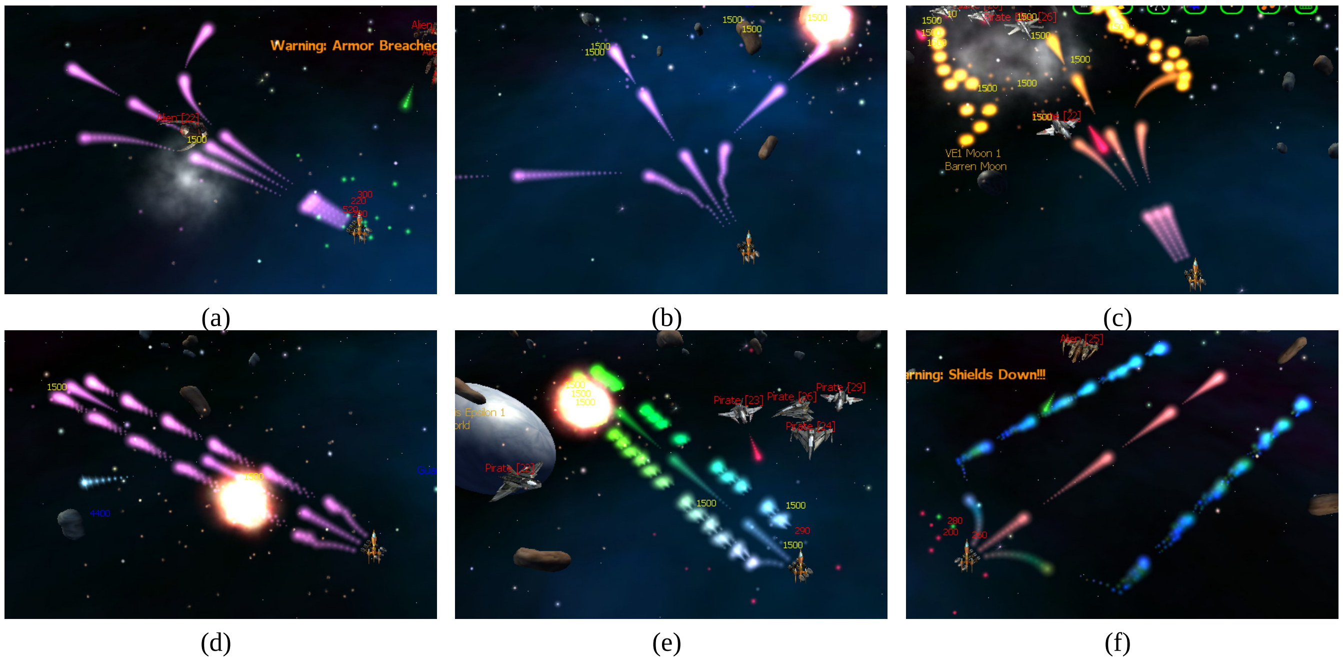
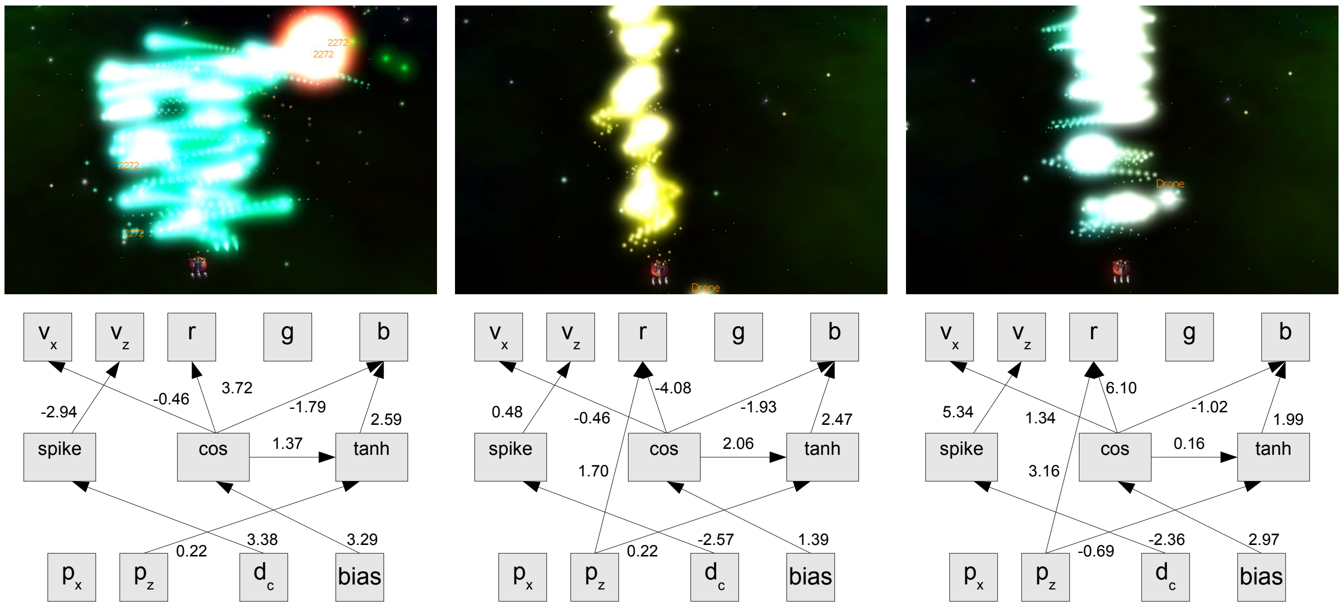


Fig. 8. **Weapon Evolution Examples**. As weapons evolve over the course of the game, players are likely to find weapons with qualities similar to those they favored in the past. In this example from actual single-player gameplay, the player often fired a spread weapon (a). Later in the game, new spread gun variations (b,c) evolved. Another interesting spread gun (d) fires two slower-firing outer projectiles and a fast inner projectile. Later descendants of this weapon (e,f) exaggerated the speed difference between the inner and outer projectiles, diversified the color pattern, and modified the spread width. These examples illustrate how cgNEAT evolves novel content based on past user preferences.



(a) 4 generation genotype (b) 5 generation genotype (c) 6 generation genotype

Fig. 9. **Weapon Genotype Examples**. This figure displays three weapons from the same lineage and their CPPN genotypes. The blue wavy pattern exhibited by Weapon (a) is the result of four generations of evolution. Weapon (b) is the direct offspring of Weapon (a), and has a tighter pattern and yellow color, which results from gaining a connection and altered weights from mutation. Weapon (c) is the direct offspring of Weapon (b). Mutation only changed weights, and it strongly resembles Weapon (a), but with a tighter pattern. These examples illustrate how mutation creates offspring weapons with similar characteristics to their parents.

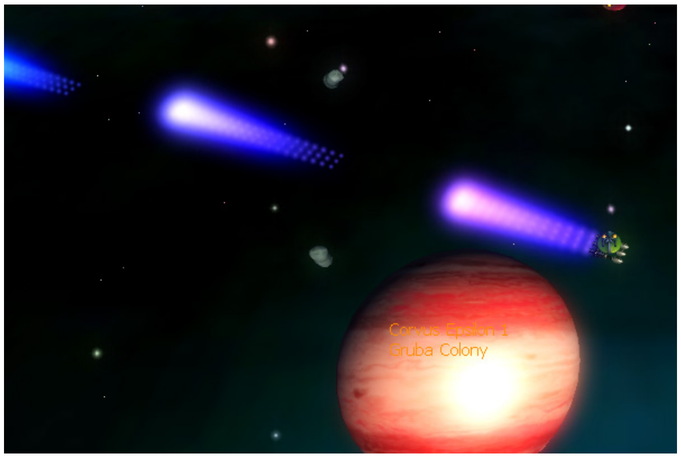


Fig. 10. **GAR Starter Weapons**. When the game begins, players are equipped with straight-shooting starter weapons that do not contribute to evolution. Starter weapons ensure that players begin the game with effective weapons (which would not be guaranteed by randomization) and additionally act as a control to which the effectiveness of evolved weapons is compared. Starter weapons do earn fitness by being fired; however, if a starter weapon is selected by a roulette roll during evolutionary selection, a spawning pool weapon is created instead.

dropped in the game world. Second, for every new weapon generated, there is always a chance of either encountering a random weapon or a weapon from the spawning pool. Thus, a variety of content is possible even with a limited population.

B. *Automatic Content Generation Results in GAR Multiplayer Mode*

In total, the GAR client was downloaded over 8,500 times. The experimental data in this section is from the "GAR Official" 32-player public game server hosted by the Evolutionary Complexity Research Group (Eplex) on the University of Central Florida (UCF) campus. The server began collecting multiplayer data from players across the world on June 2nd, 2009. The data presented in tables 1, 2, and 3 is a snapshot in time on July 30th 2009.

In total, 1,007 unique player accounts (table 1) were created on the server in approximately two months. Excluded from this data are an additional 236 invalid accounts that registered but were lost because of a database error caused by a large simultaneous influx of players from the GAR article on Slashdot.org (the accounts were registered but never fired a shot, nor picked up a weapon in game). At the time of the sample, over 73% of valid player accounts progressed past level 20, indicating substantial time (i.e. at least two or three hours) invested. A substantial proportion of players progressed to much higher levels, which takes several days in-game.

The primary method of obtaining new weapons in GAR is to defeat enemies. The aggregated player kill counts for the snapshot are displayed in table 2. The 1,007 players on the server scored 9,038 PVP kills, 721,456 Alien kills, 714,274 Pirate kills, and 22,670 Space Blob kills. Such a large kill total (over 1.46 *million*) hints at the intensity of game play.

Snapshot data for weapon evolution on the GAR official server is presented in table 3. In total, 379,081 weapons were evolved (note that about 10% of these are from the spawning pool) by players destroying enemies, their bases, and other players. This number is remarkably high for an IEC system. On average, each player encountered over 375 weapon drops. Additionally, players fired over *23.6 million* shots with the evolved weapons they discovered. These results indicate that cgNEAT is capable of exposing players to a large variety of content quickly.

Of the 379,081 weapons dropped, 132,722 were picked up by players, which is roughly 35%. The reasons that players do not always pick up weapons are either (a) the weapon is deemed inferior to those in their arsenal, or (b) the weapon is similar to a weapon they already possess. In this context, that 35% of all weapons are picked suggests that players indeed decide whether or not to pick up content based on the previews. That is, players do not need to pick up every weapon they see. At the same time, it suggests that a considerable proportion of weapons evolved (about one third) are attractive enough to pick up, even with only three weapon slots available.

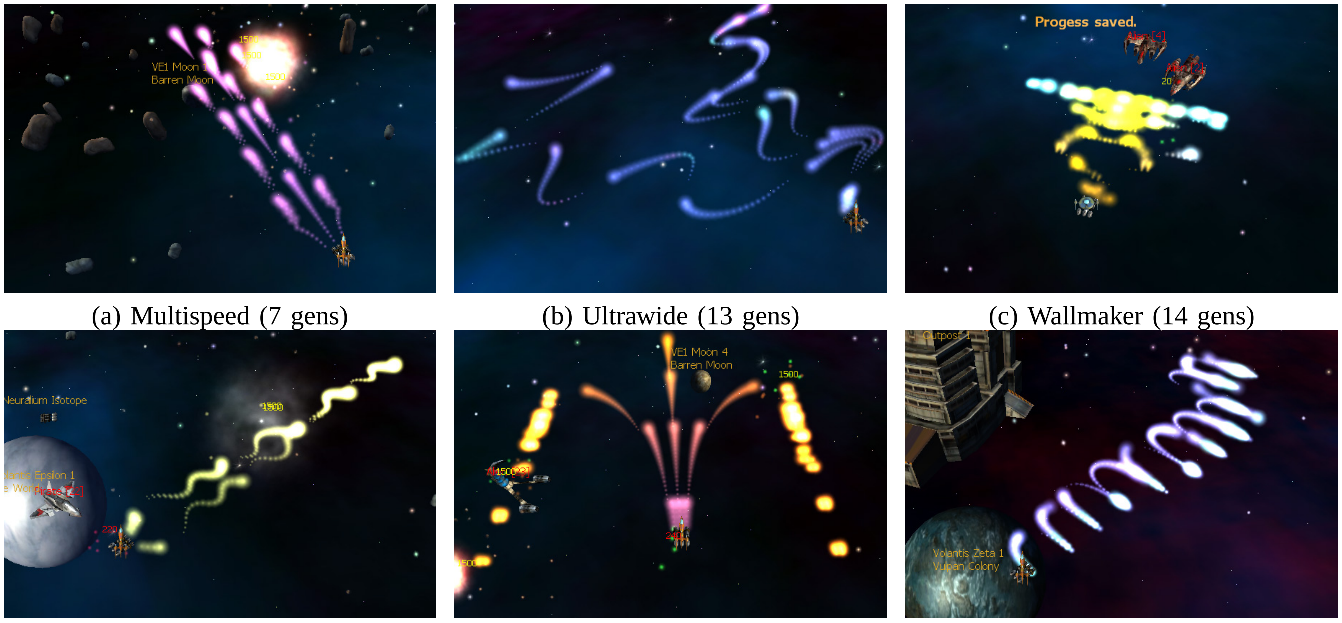
The starter weapons in GAR (which shoot in a straight line) act as an experimental control to compare with the weapons evolved by cgNEAT. During the snapshot, the number of starter weapons possessed by players above level 50 was 70, which is only 4% of the total weapons held by those players.

Player Accounts Data	
Total Player Accounts	1,007
Level 1-20	274
Level 21-40	186
Level 41-60	175
Level 61-80	81
Level 81-100	87
Level 101-120	36
Level 121-140	44
Level 141-160	11
Level 161-180	12
Level 181-200	89

Table 1. **Official Server Player Accounts**. A summary of a snapshot of player accounts on the GAR 1.1 Official 32-player server taken on July 30th, 2009 is shown. Over 73% of players progressed past level 20, indicating substantial time invested. A large number of players progressed to higher levels, which could take several days in-game.

Kill Data	
Total Kills	1,467,438
Total PVP Kills	9,038
Total Alien Kills	721,456
Total Pirate Kills	714,274
Total Blob Kills	22,670
Total Player Deaths	38,409
Max PVP Kills by a Player	1,147
Max Alien Kills by a Player	10,325
Max Pirate Kills by a Player	10,478
Max Blob Kills by a Player	402

Table 2. **Official Server Kill Counts**. Aggregate kill counts and the maximum kill counts for a single player are shown for the 1,007 player accounts on the GAR 1.1 Official 32-player server snapshot taken on July 30th, 2009. Such totals demonstrate the playability of the weapons evolved by cgNEAT in GAR.



(a) Multispeed (7 gens) (b) Ultrawide (13 gens) (c) Wallmaker (14 gens)

(d) Corkscrew (3 gens) (e) Tunnelmaker (2 gens) (f) Blue Ladder (42 gens)

Fig. 11. **Weapons Evolved During Single Player Gameplay**. GAR players discovered many useful and aesthetically pleasing weapons. The number of generations of reproduction taken to evolve each weapon is shown next to its name. The *multispeed* (a) fires two slow outer projectiles, which are useful for blocking incoming enemy fire, and a fast center projectile for quickly striking distant targets. The *ultrawide* (b) emits a wide particle pattern that are effective for fighting many enemies at once. The *wallmaker* (c) literally creates a defensive wall of particles in front of the player. The *corkscrew* (d) emits a pattern that is initially wide, for blocking, but later converges for concentrated damage at a distance. The *tunnelmaker* (e) creates a defensive line of particles as well, but on both sides of the player, yielding a defensive sheath. The *ladder gun* (f) fires a wide wave-like pattern that can swivel around obstacles like asteroids.

Note that in GAR players are able to obtain starter weapons at any time during the game by selling unwanted isotopes in exchange for starter weapons. In fact, because such a sale also yields several credits of game currency for the player, in effect there is an incentive to sell evolved weapons in exchange for starter weapons. Nevertheless, of the over 1.46 million kills of players and NPCs on the GAR official server, only 22,935 were by starter weapons (roughly 1.5%). From these results it can be inferred through player behavior that they preferred evolved weapons to starter weapons.

The total number of combined *generations* of all weapon lineages in the snapshot is 50,646, and the highest generation weapon is 98, suggesting that weapons continue to be used even into later generations. Additionally, the average number of generations per weapon in the population sample is 16, indicating that it does not take many generations for players to find weapons that they want to keep.

Overall, players in the GAR multiplayer experiments discovered a wide variety of both aesthetically and tactically diverse weaponry evolved through their implicit preferences.

Additional server data that was stored includes the *weapon archive*, where all weapons that were fired at least 800 times (i.e. highly fit weapons) are saved, and the *PVP archive*, where all weapons that score PVP kills are stored. By the time of the snapshot, these archives contained 5,209 and 1,662 weapons, respectively. The weapons presented in this section from those archives were evolved by players on the official server from around the world.

Figures 12 through 20 present the visual results of weapon evolution in the form of *weapon trends* on the server. That is, they present general styles of weapons that proved popular with many players on the server. Note that all weapons in the same trend are not necessarily related to each other, but could be members of separate lineages that followed a similar evolutionary path. All weapons displayed are from the GAR server archive (i.e. fired by their owners at least 800 times) or the PVP archive (i.e. scored kills in PVP). Therefore, it is likely that players found all of these weapons either effective or worth keeping for their novelty. The following is a sampling of trends that were discovered:

1) **Fork Guns** (figure 12): The fork trend produces extremely wide triple shots that are good for firing into crowds of enemies or for hitting fast moving enemies.
2) **Goop Guns** (figure 13): The goop trend drops masses of slowly moving particle clouds. Goop guns create effective “space mines” that block incoming bullets and can be dropped while fleeing.
3) **Multi-speed Guns** (figure 14): The multi-speed trend fires fast inner shots and slower outer shots. The fast inner shot can easily hit distant targets, while the slower outer shots act as a shield.
4) **Plasma Guns** (figure 15): The plasma trend fires random-looking bursts of particles that resemble plasma. Plasma guns are good for blocking and their chaotic patterns make them hard to dodge in PVP combat.
5) **Shield Guns** (figure 16): Shield guns create a particle shield that completely encases the player ship.
6) **Spread Guns** (figure 17): The spread gun trend fires tight streams that widen as they travel away from the player. Spread guns deliver concentrated fire at close range but spread later to make distant targets easier to

Weapon Evolution Data	
Total Evolved CPPNs	379,081
Total Looted Weapons	132,722
Total Shots by All Players	23,657,178
Total Evolved Gun Kills	1,444,503
Total Starter Gun Kills	22,935
Total Guns Currently Owned by Players Above Level 50:	1,641
Total Starter Guns Currently Owned by Players Above Level 50:	70
Max Generation	98
Total Generations of All Lineages Combined	50,646
Average Shots Fired Per Gun	178
Average Generations Per Gun	16
Guns Shot Over 800 Times	5,209
Guns That Scored PVP kills	1,662

Table 3. **Official Server Weapon Evolution**. Aggregate weapon evolution data is shown for the GAR 1.1 Official 32-player server snapshot taken on July 30th, 2009. Millions of shots were fired and over one million enemies were killed. The vast majority (98.5%) were killed with evolved weapons.

hit.

7) **Squiggle Guns** (figure 18): Squiggle guns fire curving multi-colored patterns that resemble hand-drawn figures.
8) **Vortex Guns** (figure 19): The vortex trends creates patterns that resemble cyclones. Their wide pattern makes it easy to hit enemies and block projectiles.
9) **Wall Guns** (figure 20): The wall gun trend creates a literal wall of particles in front of the player that can be used as a defensive shield.

To assess viability of evolved weapons against other players, all weapons that scored PVP kills were saved in the PVP archive, examples of which are presented in figure 21. The popular PVP weapons displayed as much variety as the non-PVP weapons; however, one major additional trend that occurred among the PVP weapons is that players significantly more often favored weapons that shoot a particle to a fixed distance at which point it stops and remains stationary, leaving a hazard for other players.

GAR also produces weapons that are not picked up or fired often (e.g. weapons that fire very slowly or in undesirable directions such as due left). Of course, because players do not use such guns, they do not develop into their own evolutionary trends.

The weapon trends demonstrate the variety of tactically diverse weapons automatically generated by cgNEAT. Like trends in the real world, certain weapon trends were popular for a time before their popularity waned and new weapon trends replaced them. Thus an important lesson is that the main contribution of CCE is not necessarily to find the “best” content, but to continually offer new alternatives as once-popular fads lose their luster. Interestingly, while the difference between two weapons from one generation to the next is often small (as in figure 9), occasionally a mutation such as a new connection or node yields a substantial qualitative change, which may become the basis for a new trend.

VI. Discussion and Future Work

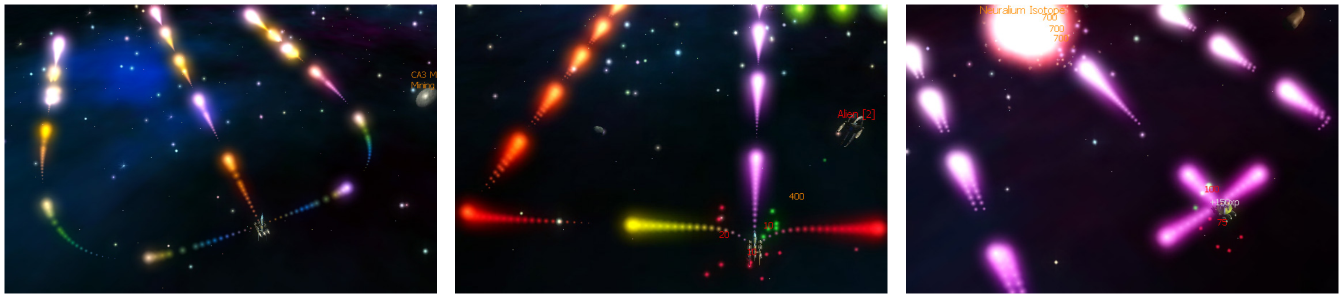
The cgNEAT-evolved weapons in GAR demonstrate that automatic content generation is a viable new technology. The main application is in simulations and games wherein the designers want users to be able to discover and experience a continual stream of new content beyond what the original artists and programmers are able to provide. For players, the main implication is a new kind of experience in which not only is novelty a constant, but the pursuit of novelty itself is an integral part of the game.

For some game designers, this loss of control will be viewed as a risky sacrifice; yet others will see it for its potential, just as any new frontier opens up an unknown world of possibilities. In fact, the interactive evolutionary dynamic automatically creates a kind of implicit game balance because, as soon as a player acquires a weapon that tips the equilibrium, variants of that weapon become available to other players in proportion to its use, thereby continually balancing the game. Just as evolutionary arms races in nature lead to a continual balancing and rebalancing of opposing powers [26], the arms race among *content* to satisfy its users can yield a long-term homeostatic force towards equilibrium.

Note that the benefit of cgNEAT is also potentially a drawback for some game types because content outside of that which the developers envisioned may be discovered. Therefore, for genres such as linear FPS games like Half-Life 2[11], in which it is integral to the story that the player possesses *weapon x* at *time y*, evolutionary content generation would likely not be a good fit.

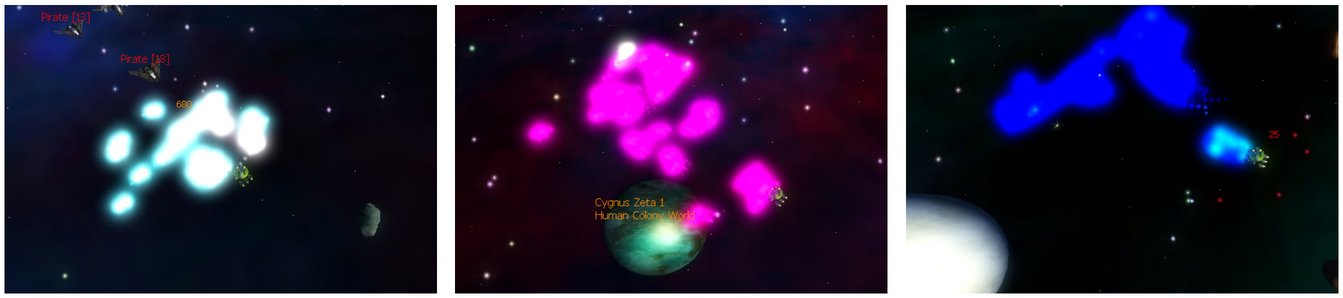
Nevertheless, for those games that can benefit, in addition to weapons, a wide variety of other game content could potentially be generated by cgNEAT including two-dimensional textures, three-dimensional models, many other types of particle effects, and programmable shader effects. Video games that automatically generate their own content (e.g. characters, clothing, weapons, houses, vehicles, music, special effects, etc.) could keep players engaged much longer in such a constantly evolving game world than in a static one. Thus the potential future applications of cgNEAT for automated content generation are broad, especially in virtual

[11]Copyright 2004 Valve Software, http://valvesoftware.com/



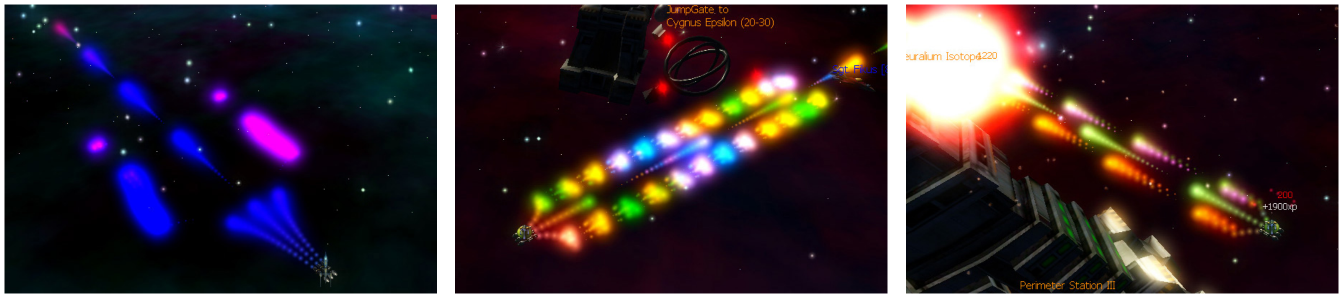
(a) 2 gens (b) 3 gens (c) 5 gens

Fig. 12. **Fork Weapon Trend**. Fork guns fire a wide triple-shot that is effective at firing into crowds of enemies or for hitting distant fast-moving enemies.



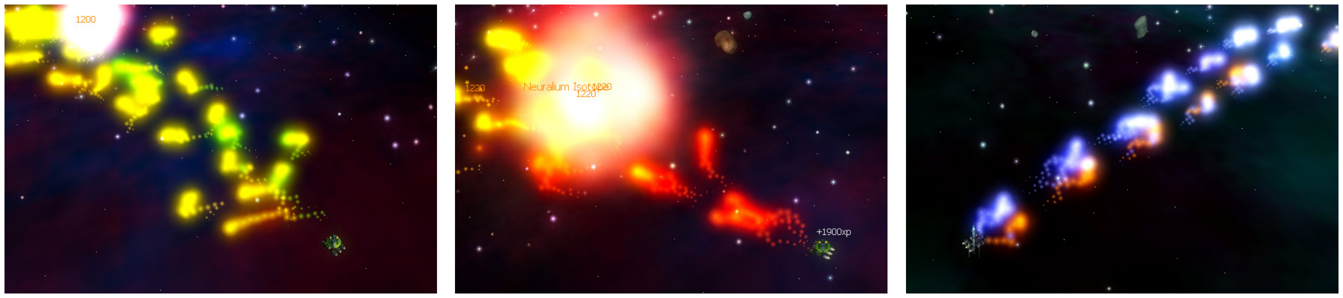
(a) 6 gens (b) 7 gens (c) 13 gens

Fig. 13. **Goop Weapon Trend**. Goop guns drop animated clouds of particles resembling liquids. They create effective "space mines" that can block incoming bullets and that can be dropped as obstacles while fleeing.



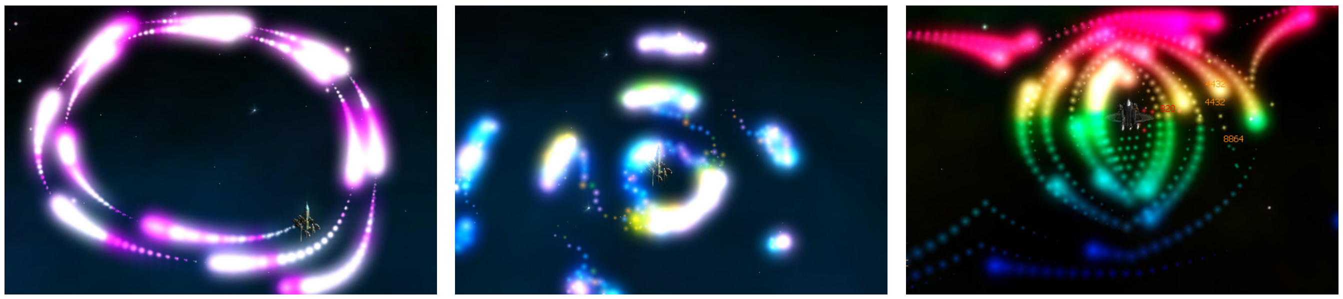
(a) 4 gens (b) 7 gens (c) 12 gens

Fig. 14. **Multispeed Weapon Trend**. Multispeed guns, which proved highly popular on the test server, have a fast center projectile and slower outer projectiles that move in a variety of patterns. Example (a) is also called a "tunnelmaker" because it creates a defensive tunnel of near-stationary particles.



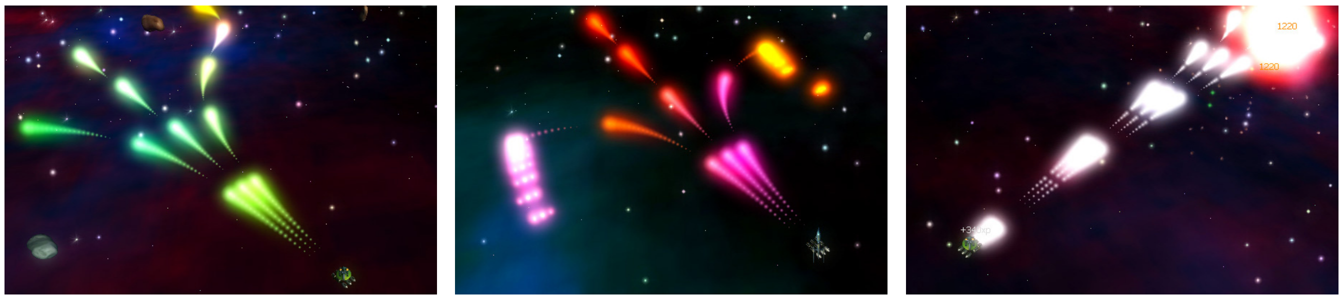
(a) 11 gens (b) 12 gens (c) 18 gens

Fig. 15. **Plasma Weapon Trend**. Plasma guns, which also were popular on the server, fire chaotic streams of colorful particles resembling plasma. Plasma guns fire fast and erratic particles that are excellent for blocking incoming projectiles and are difficult for other players to dodge in PVP mode.



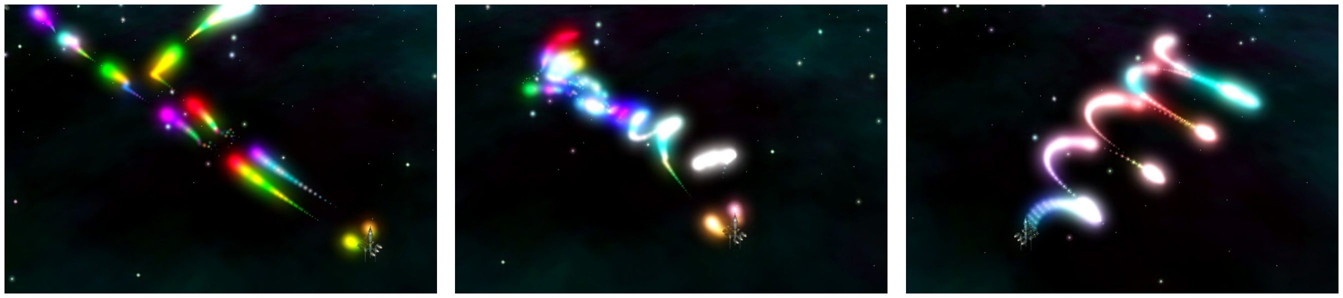
(a) 9 gens (b) 17 gens (c) 38 gens

Fig. 16. **Shield Weapon Trend**. Shield guns are excellent defensive weapons that create a particle shield completely encasing the player ship.



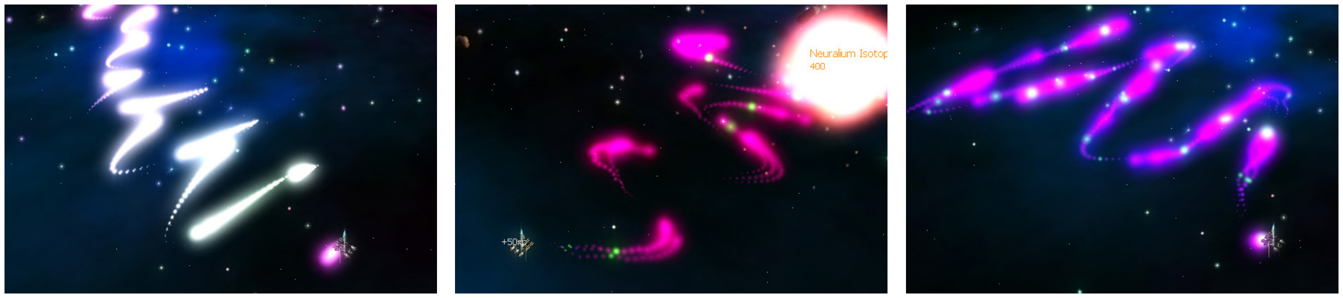
(a) 3 gens (b) 3 gens (c) 4 gens

Fig. 17. **Spread Weapon Trend**. Spread guns fire a tight stream that widens as it travels; thus these weapons deliver concentrated fire at close range but spread later to make distant targets easier to hit. Some spread guns (b) are also called *tunnelmakers* because of the lines of stationary particles created on either side of the player ship.



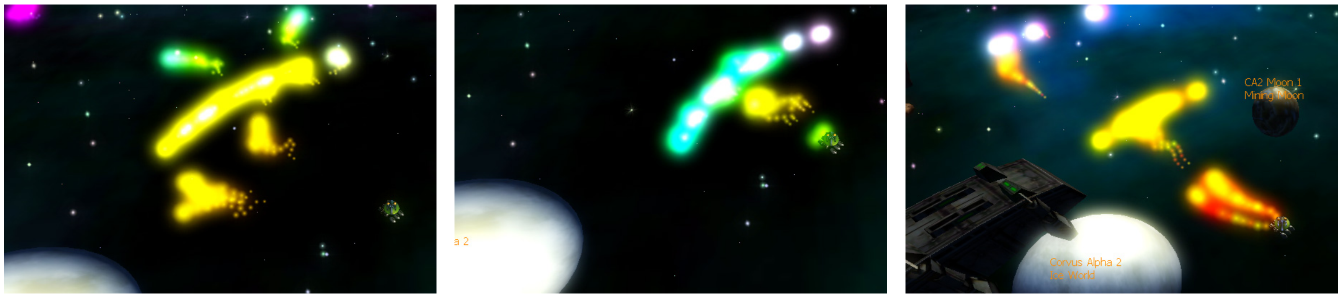
(a) 21 gens (b) 22 gens (c) 36 gens

Fig. 18. **Squiggle Weapon Trend**. Squiggle guns create diverse curved patterns that resemble hand-written script. Squiggle guns display some of the unique color and shape patterns possible through cgNEAT weapon evolution.



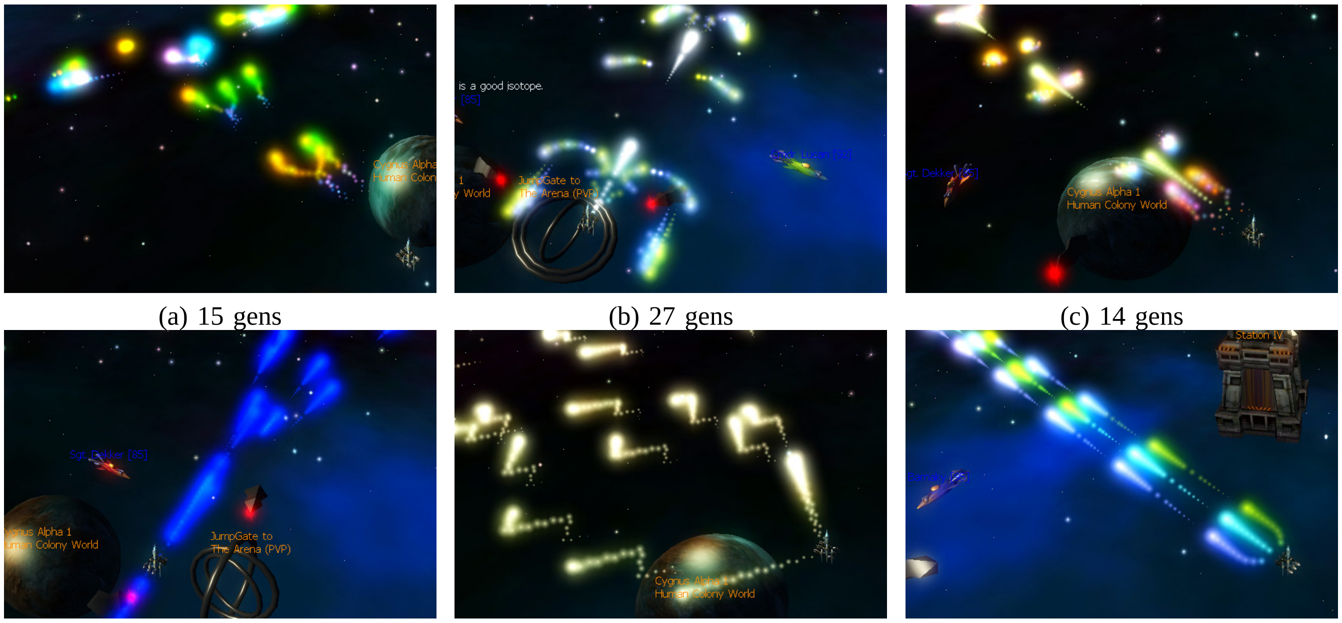
(a) 3 gens (b) 50 gens (c) 51 gens

Fig. 19. **Vortex Weapon Trend**. Vortex weapons produce spinning patterns similar to tornadoes. The seemingly-random wide patterns are effective for blocking incoming projectiles and make it easy to hit targets.



(a) 9 gens (b) 13 gens (c) 27 gens

Fig. 20. **Wall Gun Weapon Trend**. Wall guns create a literal wall of defensive particles in front of the player, useful for blocking or dropping behind while fleeing. Some wall guns such as (c) create multiple lines of particles.



(a) 15 gens (b) 27 gens (c) 14 gens

(d) 21 gens (e) 27 gens (f) 16 gens

Fig. 21. **PVP Archive Weapons**. This example displays PVP archive weapons with which players scored PVP kills on the GAR Official 32-player server. Overall, the popular PVP weapons display as much variety as those used solely against NPCs; thus players employed many evolved weapon tactics to defeat each other.

worlds or MMOGs in which unique content creates value in the virtual economy.

In practice, any class of content that is intended to be evolved must be *parameterized* in some fashion first. In GAR, the CPPN-based particle system representation is the parameterization of the particle weapon class. Ideal parameterizations are *open-ended* in the sense that it should be possible for content to evolve that exceeds the foresight of the game designers. The weapons evolved by GAR satisfy this criterion because many of them represent new concepts (e.g. wallmakers and tunnelmakers) that are novel even to the designers of the game and also that became popular among its players. However, the effort required to parameterize other classes of content will likely vary significantly. For example, parameterizing the space of *cars* or *houses* may require a priori understanding of the most important ways in which such content can vary. At the same time, the potential for open-ended discovery should not be limited by too much constraint. Yet the up-front effort and expense of parameterizing a class of content can pay off in unlimited dividends down the line. While such an effort represents a new kind of investment for developers, this paper shows that it can indeed succeed, bringing the potential for much-needed novelty to future games. That GAR players invested enough time to defeat over one million enemies on a single server suggests the tantalizing potential for much more ambitious such projects. The possibilities for evolving vast classes of content with the full resources and talent of a commercial game company behind the initial parameterization are yet to be imagined.

An interesting future study is to more rigorously assess the perceived value of the procedural content created by the game from the perspective of the players. Potential experiments include (1) two separate versions of a game, one containing only fixed hand-coded content and the other with fully evolved content, or (2) a hidden flag for each player within the game that determines who will be provided with evolved content and who with only a fixed amount of static content. Player behavior in both such scenarios would provide more analytical insight into player evaluation of procedural content. For example, is there any correlation between the types of weapons players find and how long they stay interested in the game?

Another possibility is to seek insight into how much value players place on novelty. This issue could be investigated through an experiment in which players are able to rate their weapons. Then, results would be compared among (1) a subset of players who receive weapons from normal GAR evolution, (2) a second subset of players that receive weapons completely randomly, and (3) a third subset that receives evolved weapons exclusively through novelty search [64].

An additional issue for future research is the role of pure aesthetics in bringing value and entertainment to game content. To what extent do players value aesthetics above tactical value? While many players and game designers may *think* that players ultimately care about the effectiveness of usable content, it is possible that players sometimes *behave* otherwise. That is, players may sometimes choose content that looks nice over content that is more powerful. One appeal of CCE is that it addresses this issue implicitly without requiring designers to ascertain the answer. If players do prefer aesthetic qualities to utility, that will be reflected in their behavior and thereby amplified by cgNEAT. If they do not, the same is true. Thus, by its implicit nature, cgNEAT can potentially transcend tricky philosophical design questions that are difficult to answer explicitly.

The main goal for the near-term future of GAR is to continue developing the game by adding new levels, ship modifications, star systems, and other types of evolvable content. A wider release of future GAR versions could yield a significantly broader explosion of content. Potentially, GAR may be open-sourced, enabling others to utilize GAR for research or game development. In the long term, GAR will be hosted permanently by the Evolutionary Complexity Research Group at UCF, and continue to function as a platform for experimental game development.

VII. CONCLUSIONS

This paper presented two novel works that together establish that automated content generation in mainstream games is possible. First, cgNEAT is an algorithm created explicitly to automatically generate game content based on perceived user preferences in real time, as games are played. The cgNEAT algorithm, unlike standard evolutionary algorithms, selects content for reproduction implicitly through player behavior within the game. That is, content players utilize often is more likely to reproduce. The result is a constant stream of novel content suited to players' tastes. Second, cgNEAT is implemented in Galactic Arms Race, a 32-player persistent online game in which particle system weapons evolve based on the preferences of players.

The GAR client was downloaded by thousands of players and over 1,000 players from around the world participated on the gar.eecs.ucf.edu server hosted at UCF. Experimental results from the official multiplayer server, on which over *one million* enemies were killed and hundreds of thousands of weapons were evolved, suggest that cgNEAT is capable of automatically creating effective content based on player preferences.

The success of the initial GAR release suggests the potential of cgNEAT, and automatic content generation in general, to generate many other types of game content. For players, continual introduction of such novel content may significantly increase game replay value, and enhance the experience of MMOGs. For the game industry, it demonstrates the possibility of building games that automatically generate their own content to satisfy users by leveraging the collective behavior of multiple players, hinting at a promising new paradigm in game design.

ACKNOWLEDGMENT

Special thanks to the Galactic Arms Race (GAR) volunteer team: Nathan Sriboonlue (BOIDs and steering), Jaruwan Mesit (soft-body Space Blobs), Fabian Moncada (music and sound effects), John Martin (testing, balance, and additional server code), Derrick Janssen (additional design, testing, and game balance), Kristen Martin (additional database and file processing), Eric Isles (additional music and sound effects), Gordon Hart (additional music and sound effects), Jonathan "Zarcath" Chan (additional design), FourTwoOmega (additional music), and JRWR (web stats and hosting).

The GAR multiplayer game engine was developed by Erin Hastings with the Microsoft XNA Game Studio SDK available from http://creators.xna.com. GAR multi-player networking code was developed with Lidgren Network Library by Michael Lidgren which is open source and available from http://code.google.com/p/lidgren-network.

Finally, special thanks to our anonymous reviewers for helping us to improve the text substantially.

Galactic Arms Race is available online at http://gar.eecs.ucf.edu and the project's official email address is gar@eecs.ucf.edu.

REFERENCES

[1] R. Edwards, "The economics of game publishing," May 2006. [Online]. Available: http://games.ign.com/articles/708/708972p1.html

[2] M. J. Irwin, "Game developers' trade off," May 2008. [Online]. Available: http://www.forbes.com/2008/05/27/videogame-art-money-tech-personal-cx_mji_0528vgames.html

[3] Valve Software, "Source engine SDK," 2009. [Online]. Available: http://developer.valvesoftware.com

[4] EpicGames, "Unreal engine SDK," 2009. [Online]. Available: http://www.unrealtechnology.com/

[5] IdSoftware, "Quake wars SDK," 2009. [Online]. Available: http://www.idsoftware.com/

[6] K. Sims, "Artificial evolution for computer graphics," *Proceedings of the ACM Special Interest Group on Graphics and Interactive Techniques*, pp. 319–328, 1991. [Online]. Available: http://www.karlsims.com/papers/siggraph91.html

[7] S. Todd and W. Latham, *Evolutionary Art and Computers*. Orlando, FL, USA: Academic Press, Inc., 1992. [Online]. Available: http://www.amazon.com/Evolutionary-Art-Computers-Stephen-Todd/dp/012437185X

[8] L. World, "Aesthetic selection: The evolutionary art of Steven Rooke," *IEEE Computer Graphics and Applications*, vol. 16, no. 1, pp. 4–5, 1996. [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/MCG.1996.481558

[9] P. Machado and A. Cardoso, "All the truth about NEvAr," *Applied Intelligence Special Issue on Creative Systems*, vol. 16, no. 2, 2002. [Online]. Available: http://www.springerlink.com/content/tv79723876wr723x/

[10] J. Romero and P. Machado, Eds., *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Springer, 2007. [Online]. Available: http://art-artificial-evolution.dei.uc.pt/

[11] D. A. Hart, "Toward greater artistic control for interactive evolution of images and animation," *Proceedings of the 2007 Evoworkshops 2007 on Evocomnet, Evofin, Evoiasp,Evointeraction, Evomusart, EvoSTOC and Evotranslog: Applications of Evolutionary Computing*, vol. 4448, pp. 527–536, 2009. [Online]. Available: http://www.dahart.com/research.html

[12] K. Sims, "Evolving virtual creatures," in *Proceedings of the ACM Special Interest Group on Graphics and Interactive Techniques*, 1994, pp. 50–62. [Online]. Available: http://www.karlsims.com/evolved-virtual-creatures.html

[13] P. Husbands, G. Germy, M. McIlhagga, and R. Ives, "Two applications of genetic algorithms to component design," *Evolutionary Computing*, pp. 50–61, 1996. [Online]. Available: http://citeseer.ist.psu.edu/husbands96two.html

[14] H. Nishino, H. Takagi, S. Cho, and K. Utsumiya, "A 3D modeling system for creative design," in *Proceedings of the 15th International Conference on Information Networking*. IEEE Press, 2001, pp. 479–487. [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/ICOIN.2001.905468

[15] G. Nelson, "Sonomorphs: An application of genetic algorithms to growth and development of musical organisms," *Proceedings of the 4th Biennial Art and Technology Symposium*, pp. 155–169, 1993. [Online]. Available: http://timara.con.oberlin.edu/~gnelson/PapersPDF/morph93.pdf

[16] B. Johansen and R. Poli, "GP-music: An interactive genetic programming system for music generation with automated fitness raters," *Proceedings of the Third Annual Conference: Genetic Programming*, pp. 181–186, 1998. [Online]. Available: http://graphics.stanford.edu/~bjohanso/papers/gp98/johanson98gpmusic.pdf

[17] N. T. nd H. Iba, "Music composition with interactive evolutionary computation," *Proceedings of 3rd International Conference on Generative Art*, 2000. [Online]. Available: http://www.iba.t.u-tokyo.ac.jp/papers/2000/tokuiGA2000.pdf

[18] J. McCormack, "Open problems in evolutionary music and art," *Applications on Evolutionary Computing*, vol. 3449, pp. 428–436, 2005. [Online]. Available: http://www.springerlink.com/content/rh40xp73vecbqb20/

[19] P. Husbands, P. Copley, A. Eldridge, and J. Mandelis, "An introduction to evolutionary computing for musicians," *Evolutionary Computer Music*, pp. 1–27, 2007. [Online]. Available: http://www.cogs.susx.ac.uk/users/philh/pubs/IntroECforMusiciansv3.pdf

[20] A. Hoover and K. O. Stanley, "Exploiting functional relationships in musical composition," *Connection Science Special Issue on Music, Brain, and Cognition*, 2009. [Online]. Available: http://eplex.cs.ucf.edu/publications/2009/hoover.connectionscience09.html

[21] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the NERO video game," *IEEE Transactions on Evolutionary Computation Special Issue on Evolutionary Computation and Games*, vol. 9, no. 6, pp. 653–668, 2005. [Online]. Available: http://nn.cs.utexas.edu/?stanley:ieeetec05

[22] H. Takagi, "Interactive evolutionary computation: Fusion of the capacities of EC optimization and human evaluation," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296, 2001. [Online]. Available: http://www.design.kyushu-u.ac.jp/~takagi/TAKAGI/IECsurvey.html

[23] M. Fagerlund, "DelphiNEAT-based genetic art homepage," 2005. [Online]. Available: http://www.cambrianlabs.com/mattias/GeneticArt/

[24] J. Secretan, N. Beato, D. B. D'Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley, "Picbreeder: Evolving pictures collaboratively online," in *Proceedings of the Computer Human Interaction Conference*, 2008. [Online]. Available: http://eplex.cs.ucf.edu/publications/2008/secretan.chi08.html

[25] T. Unemi, "Genetic algorithms and computer graphic arts," *Journal of Japan Society for Artificial Intelligence*, vol. 9, no. 4, pp. 518–523, 1994. [Online]. Available: http://www.intlab.soka.ac.jp/~unemi/sbart/Document.html

[26] R. Dawkins, *The Blind Watchmaker.* Essex, U.K.: Longman, 1986. [Online]. Available: http://www.amazon.com/Blind-Watchmaker-Evidence-Evolution-Universe/dp/0393315703

[27] K. O. Stanley, "Exploiting regularity without development," in *Proceedings of the AAAI Fall Symposium on Developmental Systems*. AAAI Press, 2006. [Online]. Available: http://eplex.cs.ucf.edu/papers/stanley_aaaifs06.pdf

[28] ——, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, vol. 8, no. 2, pp. 131–162, 2007. [Online]. Available: http://www.springerlink.com/content/804411v3703ph210/

[29] F. Gomez and R. Miikkulainen, "Solving non-Markovian control tasks with neuroevolution," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999, pp. 1356–1361. [Online]. Available: http://nn.cs.utexas.edu/downloads/papers/gomez.ijcai99.pdf

[30] N. Saravanan and D. B. Fogel, "Evolving neural control systems," *IEEE Expert: Intelligent Systems and Their Applications*, vol. 10, pp. 23–27, 1995. [Online]. Available: http://portal.acm.org/citation.cfm?id=631355

[31] F. Gruau, D. Whitley, and L. Pyeatt, "A comparison between cellular encoding and direct encoding for genetic neural networks," in *Genetic Programming 1996: Proceedings of the First Annual Conference*. MIT Press, 1996, pp. 81–89. [Online]. Available: http://portal.acm.org/citation.cfm?id=1595547

[32] B. Zhang and H. Muhlenbein, "Evolving optimal neural networks using genetic algorithms with Occam's razor," *Complex Systems*, vol. 7, no. 3, pp. 199–220, 1993. [Online]. Available: http://eprints.kfupm.edu.sa/38471/1/38471.pdf

[33] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999. [Online]. Available: http://www.cs.bham.ac.uk/~xin/papers/published_iproc_sep99.pdf

[34] A. Martin, "Increasing genomic complexity by gene duplication and the origin of vertebrates," *The American Naturalist*, vol. 154, no. 2, pp. 111–128, 1999. [Online]. Available: http://www.jstor.org/pss/2463906

[35] J. Watson, N. Hopkins, J. Roberts, J. Steitz, and A. Weiner, *Molecular Biology of the Gene Fourth Edition*. The Benjamin Cummings Publishing Company, Inc., 1987. [Online]. Available: http://www.amazon.com/Molecular-Biology-Gene-James-Watson/dp/0805396144

[36] L. Altenberg, "Evolving better representations through selective genome growth," in *Proceedings of the IEEE World Congress on Computational Intelligence*. IEEE Press, 1994, pp. 182–187. [Online]. Available: http://dynamics.org/Altenberg/PAPERS/EBR/

[37] I. Harvey, "The artificial evolution of adaptive behavior," Ph.D. dissertation, School of Cognitive and Computing Sciences, University of Sussex, Sussex, 1993. [Online]. Available: http://www.cogs.susx.ac.uk/users/inmanh/inman_thesis.html

[38] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, pp. 99–127, 2002. [Online]. Available: http://nn.cs.utexas.edu/?stanley:ec02

[39] M. E. Taylor, S. Whiteson, and P. Stone, "Comparing evolutionary and temporal difference methods in a reinforcement learning domain," in *GECCO 2006: Proceedings of the Genetic and Evolutionary Computation Conference*, July 2006, pp. 1321–1328. [Online]. Available: http://www.cs.utexas.edu/~pstone/Papers/bib2html-links/GECCO06-matt.pdf

[40] T. Aaltonen *et al.*, "Measurement of the top quark mass with dilepton events selected using neuroevolution at CDF," *Physical Review Letters*, vol. 102, no. 15, 2009. [Online]. Available: http://www-cdf.fnal.gov/physics/preprints/cdf9235_dil_mtop_nn.pdf

[41] G. Wichman, "A brief history of rogue," 1997. [Online]. Available: http://www.wichman.org/roguehistory.html

[42] M. Barton and B. Loguidice, "The history of rogue: Have you, you deadly zs," 1997. [Online]. Available: http://www.gamasutra.com/view/feature/4013/the_history_of_rogue_have__you_.php

[43] M. Nitsche, C. Ashmore, W. Hankinson, R. Fitzpatrick, J. Kelly, and K. Margenau, "Designing procedural game spaces: A case study," *Proceedings of the Future Play Conference*, 2006. [Online]. Available: http://www.lcc.gatech.edu/~nitsche/download/Nitsche_DesigningProcedural_06.pdf

[44] M. Mateas and A. Stern, "Procedural authorship: A case-study of the interactive drama faade," *Proceedings of the Digital Arts and Culture: Digital Experience: Design, Aesthetics, Practice*, 2005. [Online]. Available: http://users.soe.ucsc.edu/~michaelm/publications/mateas1-dac2005.pdf

[45] J. Togelius, R. D. Nardi, and S. M. Lucas, "Towards automatic personalised content creation for racing games," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. IEEE Press, 2007. [Online]. Available: http://julian.togelius.com/Togelius2007Towards.pdf

[46] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. IEEE Press, 2008. [Online]. Available: http://julian.togelius.com/Togelius2008An.pdf

[47] S. Colton and C. Browne, "Evolving simple art-based games," *Applications of Evolutionary Computing*, vol. 5484, 2009. [Online]. Available: http://www.springerlink.com/content/p6p0746n86j7328l/

[48] M.Hull and S. Colton, "Towards a general framework for program generation in creative domains," *Proceedings of the 4th International*

Joint Workshop on Computational Creativity, 2007. [Online]. Available: http://www.doc.ic.ac.uk/~sgc/papers/hull_cc07.pdf

[49] S. von Mammen, "Swarming for games: Immersion in complex systems," *Applications of Evolutionary Computing*, vol. 5484, 2009. [Online]. Available: http://vonmammen.org/science/SwarmGames.pdf

[50] W. Reeves, "Particle systems: A technique for modeling a class of fuzzy objects," *ACM Transactions on Computer Graphics*, vol. 17, no. 3, pp. 91–108, 1983. [Online]. Available: http://design.osu.edu/carlson/history/PDFs/reeves-particles.pdf

[51] J. Lander, "The ocean spray in your face," *Game Developer Magazine*, pp. 13–20, July 1997. [Online]. Available: http://www.double.co.nz/dust/col0798.pdf

[52] J. V. der Berg, "Building an advanced particle system," *Game Developer Magazine*, pp. 44–50, March 2000. [Online]. Available: http://www.mysticgd.com/misc/AdvancedParticleSystems.pdf

[53] W. Reeves, "Approximate and probabilistic algorithms for shading and rendering structured particle systems," *ACM Transactions on Computer Graphics*, vol. 19, no. 3, pp. 313–322, 1985. [Online]. Available: http://portal.acm.org/citation.cfm?id=325250&dl=&coll=

[54] D. Breen, "A particle based model for simulating draping behavior of woven cloth," *Textile Research Journal*, vol. 64, no. 11, pp. 663–685, 1994. [Online]. Available: http://portal.acm.org/citation.cfm?id=193840

[55] B. Eberhardt, A. Weber, and W. Strasser, "A fast, flexible, particle-system model for cloth draping," *IEEE Transactions on Computer Graphics and Applications*, vol. 16, no. 5, 1996. [Online]. Available: http://portal.acm.org/citation.cfm?id=618378

[56] D. Obrien, S. Fisher, and M. Lin, "Automatic simplification of particle system dynamics," in *Proceedings of the 14th Annual Conference on Computer Animation*, 2001, pp. 210–257. [Online]. Available: http://gamma.cs.unc.edu/SLOD/images/slod.pdf

[57] M. Muller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Proceedings of the 2003 ACM Eurographics Symposium on Computer Animation*, 2003, pp. 154–159. [Online]. Available: http://www.matthiasmueller.info/publications/sca03.pdf

[58] C. Reynolds, "Steering behaviors of autonomous characters," in *Proceedings of the Game Developers Conference*, 1999, pp. 763–782. [Online]. Available: http://www.red3d.com/cwr/papers/1999/gdc99steer.html

[59] ——, "Flocks, herds, and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, 1987, pp. 25–34. [Online]. Available: http://www.red3d.com/cwr/papers/1987/boids.html

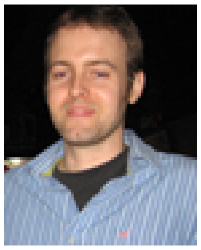
[60] E. Hastings, R. Guha, and K. Stanley, "Interactive evolution of particle systems for computer graphics and animation," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, 2009. [Online]. Available: http://eplex.cs.ucf.edu/publications/2007/hastings.cig07.html

[61] E. Hastings, R. Guha, and K. O. Stanley, "Interactive evolution of particle systems for computer graphics and animation," *IEEE Transactions on Evolutionary Computation*, 2009. [Online]. Available: http://eplex.cs.ucf.edu/publications/2009/hastings.cig09.html

[62] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[63] K. DeJong, *Evolutionary Computation: A Unified Approach*. The MIT Press, 2006. [Online]. Available: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=3749

[64] S. Risi, S. Vanderbleek, C. Hughes, and K. Stanley, "How novelty search escapes the deceptive trap of learning to learn," *Proceedings of the Genetic and Evolutionary Computation Conference*, 2009. [Online]. Available: http://eplex.cs.ucf.edu/publications/2009/risi.gecco09.html



Erin J. Hastings earned a Ph.D. in Computer Science at the University of Central Florida in 2009. He received a B.S. in Computer Science from University of Florida in 2001 and an M.S. in Computer Science from University of Central Florida in 2004. His research interests include automatic graphics and game content generation, interactive evolutionary computation, particle systems, spatial subdivision, and networking. He has recently published papers in the IEEE Symposium on Computational Intelligence and Games and IEEE Transactions on Evolutionary Computation.

Ratan K. Guha received the B.S. degree with honors in Mathematics and the M.S. degree in Applied Mathematics from the University of Calcutta. He received the Ph.D. degree in Computer Science from the University of Texas at Austin in 1970. He is a Professor of Computer Science at the University of Central Florida, Orlando. His research interests include distributed systems, networks, security protocols, modeling, simulation, and graphics. His research has been supported by grants from ARO, NSF, STRICOM, PM-TRADE, NASA, and the State of Florida. Dr. Guha is a member of ACM, IEEE, SCS, and served on the Board of Directors of SCS. He is currently serving on the editorial board for the International Journal of Internet Technology and Secured Transactions and the editorial board for Modelling and Simulation in Engineering.

Kenneth O. Stanley is an assistant professor in the School of Electrical Engineering and Computer Science at the University of Central Florida. He received a B.S.E. from the University of Pennsylvania in 1997 and received a Ph.D. in 2004 from the University of Texas at Austin. He is an inventor of the Neuroevolution of Augmenting Topologies (NEAT) and HyperNEAT algorithms for evolving complex artificial neural networks. His main research contributions are in neuroevolution (i.e. evolving neural networks), generative and developmental systems, coevolution, machine learning for video games, and interactive evolution. He has won best paper awards for his work on NEAT, NERO, NEAT Drummer, HyperNEAT, and novelty search. He is the chair of the IEEE Task Force on Computational Intelligence and Video Games, and chaired the Generative and Developmental Systems track at GECCO for the last three years.