Towards Orchestrating Game Creativity Facets: a Case study fusing Audio, Visuals and Gameplay

Omitted for Anonymous Review

Omitted for Anonymous Review

Abstract. Computer games are unique creativity domains in that they elegantly fuse several facets of creative work including visuals, narrative, music, architecture and design. While the exploration of possibilities across facets of creativity offers a more realistic approach to the game design process, most existing autonomous (or semi-autonomous) game content generators focus on the mere generation of single domains (creativity facets) in games. Motivated by the sparse literature on multifaceted game content generation, this paper introduces a multifaceted procedural content generation (PCG) approach which is based on the interactive evolution of multiple artificial neural networks that orchestrate the generation of visuals, audio and gameplay. The approach is evaluated on a spaceship shooter game. The generated artifacts — a fusion of audiovisual and gameplay elements — showcase the capacity of multifaceted PCG and its evident potential for computational game creativity.

1 Introduction

Computer games are creative domains of multiple facets including audio, visuals, gameplay, narrative, level architecture and game design [7]. For the vast majority of game genres the interplay between audio (e.g. character themes, sound effects, foreshadowing) and visuals (e.g. object and level design) is of critical importance for the creative expression of game designers and, ultimately, for the immersion of players via gameplay [7]. As *nonlinear games* surge in popularity, game composers and designers are increasingly challenged to design a meaningful player experience for players who are dynamically constructing their own game experiences. While traditional composers and designers are adapting to this new practice [2] approaches for procedurally generating assets for games are emerging to enhance game personalisation, increase game replayability and alleviate some of the additional demands on designers, musicians, and artists [13].

Procedural content generation (PCG) methods as employed in games have traditionally been used to either fully automate aspects game development such as asset creation or as assistive authoring tools for designers [16]. The dominant focus of PCG until recently has been on generating single game creativity facets (domains) — such as visuals or levels — into the standard game development pipeline [14] with a few notable exceptions [3, 10**?**].

In this paper, instead, we introduce a *multifaceted* PCG approach for the simultaneous generation of audio, visuals and gameplay. The generation of the game domains is multifaceted in that the creation process on one domain informs and is informed by the creation process on the others. Our approach relies on *implicit* interactive evolution [13] where human input (gameplay) determines the fitness of the generated artifacts: in our case the fusion of audio, visuals and gameplay. Both audio and visuals are represented by compositional patterns producing networks (CPPNs) [11] which co-evolve throughout gameplay (i.e. real-time) and are both interlinked with gameplay. The proposed methodology is tested in the AudioInSpace game, a spaceship shooting game, where both weapons trajectories (gameplay, visuals), their color (visuals) and their sounds effects (audio) are orchestrated and interlinked with (and affected by) the gameplay.

2 Background

While there are many approaches to procedurally generating content or facets for video games (e.g. visuals, audio, narrative, game design, level design and gameplay), the focus is often on the single facet rather than how they can combine to create a complete game experience. For instance, world maps in *Civilization V* (Firaxis 2010), dungeons in *Diablo* (Blizzard 1996), racing tracks [12], petal colors and shapes in Petalz [9], and weapons in *Borderlands* (Gearbox 2009) are all procedurally generated to provide the player with increased personalization and replayabity. Simiarly, *Audiosurf* [8] generates visual game play elements depending on the sound file provided by the player. While these generated elements may increase replayability and alleviate the art and design requirements on developers, these approaches serve as means to an end rather than celebrating the creativity of procedural content generation itself.

Other approaches aim to interweave procedurally generated elements to enhance the game experience. For instance, *Galactic Arms Race* [5] encourages players to interactively evolve weapon visuals and bullet trajectories to their aestetic and gameplay preferences, while *Game-o-matic* [15] helps players visually construct the topic of their game and then procedurally generates the rules and visuals. Similarly, *A Rogue Dream* [3] is 2D maze game that gathers and parses internet data to discover relationships between natural language concepts and create game facets from them, challenging players to interpret the designer's intent on topics like religion or politics. While Proteus [**?**] is a completely procedurally generated pixel-art world where users can spacially and sonically explore their environments, the sonic output does not affect the previously generated landscapes. Even in these integrated environments, the output of one procedurally generated facet has little affect on the other.

While integrating procedurally generated game facets is a growing trend, few investigate bidirectional communication between elements. One notable exception is *AudioOverdrive*, side-scrolling space shooter which creates bidirectional

communication between enemies, gameplay, and level visuals [6]. However, but the approach requires heavy human initiave by the developers.

The approach in this paper is to extend the ideas of Audiooverdrive to create an environment where the visuals and audio are closely coupled with minimal interference from the developers. AudioInSpace attempts to orchestrate the generation of audio, visuals and gameplay: in the testbed space shooter game, the appearance of the spaceship's weapons (e.g. color) and their gameplay effects (e.g. whether it hit an enemy) affect the game's soundtrack; the soundtrack in turn affects the firing rate, color and movement patterns of the weapon's particles, thus ensuring a bidirectional communication across creative facets. While similar to AudioOverdrive in terms of goals, theme and creative game facets tackled, AudioInSpace specifically uses information on the spaceship's gameplay behavior and on-screen visuals rather than treating gameplay events such as firing a weapon solely on account of its sound effect.

3 Approach

While many approaches to generating content create singular game facets to augment the developers' creativity, the approach in this paper is to explore mixed initiative co-creativity, where the machine and player combine to create an augment the other's experience. The machine's creativity is enhanced through a *conceptual blending* of the audio and visual modules in figure 1. Here, audio inputs are sent to the visual module, which are looped back to inputs for the audio. Players exercise their own creativity when deciding which weapons and audio to select. Because it is fundamental for the individual audio and visual domains to interact, each PCG module bases decisions on information gathered from both domains. Together, the machine and the player create an experience unique for each player.

These relationships between the domains and generated outputs are represented by a special type of artificial neural network (ANN) called a compositional pattern producing network (CPPN; shown in figure 2) [11]. Like traditional ANNs, each CPPN is an interconnected network of nodes and connection weights that when provided input, calculate an output value. However, unlike traditional ANNs that only compute sigmoid functions at the hidden nodes, CPPNs can compute any function (e.g. Gaussian, sigmoid, sine, ramp etc.), in effect making the CPPN a patten generator biased toward certain regularities. Through this representation, the aim is that each

Through the NeuroEvolution of Augmenting Topologies (NEAT) (NEAT; **?**) algorithm, each CPPN can start minimally and complexify as necessary over evolutionary time. By adding hidden nodes and connections, and changing activation functions and weight values, each new individual can expand the relationship between inputs and hidden nodes.

Each PCG module is represented by a separate CPPN that inputs domain information and outputs instructions for generating either audio or visual patterns that not only affect player immersion but also gameplay. To personalize the

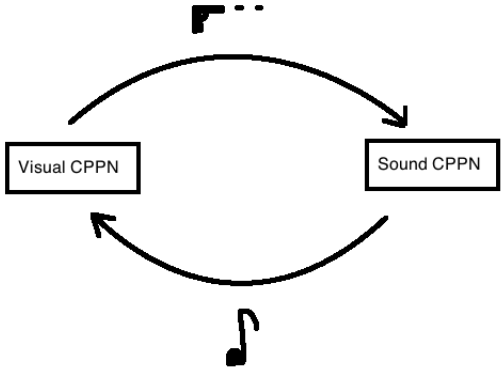


Fig. 1. In AudioInSapce, audio and weapon visuals are represented by two separate modules that each provide relevant domain information to the other (i.e. the visual module is not only informed by current visual elements in the level, but also by the concurrently generated audio and vice versa).

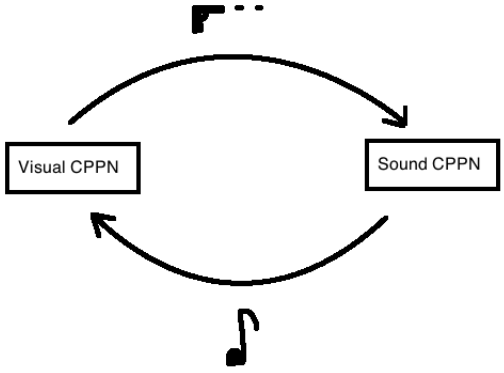


Fig. 2. Compositional pattern producing networks (CPPNs; 11) are an interconnected network of nodes with inputs, hidden nodes, and outputs that can theoretically compute any function[4]. They are a special type of artificial neural network (ANN) wherein hidden nodes can compute activation functions beyond the typical sigmoid restrictions. By allowing the hidden nodes to compute Gaussian, Sine, and other functions, resulting patterns can be biased toward producing particular regularities.

relationship between music, visuals, and gameplay, these CPPNs can be evolved through a process similar to animal breeding called interactive evolutionary computation (IEC), wherein the human use rather than an explicit fitness function rates candidate individuals. CPPNs are evolved through the NeuroEvolution of Augmenting Topologies (NEAT; **?**) algorithm which was originally developed to solve control and decision tasks but generates sound and visuals in this paper.

Because CPPNs produce patterns of the inputs, it is important to choose meaninful inputs Because patterns are important, generating desirable melodic and visual patterns is dependent on the types of inputs sent through the CPPN.

Relationships in TODO music-game are represented as compositional pattern producing networks evolved with neat. Each game asset has a different set of inputs and outputs.

3.1 Game

This mixed initiave design is presented through a space shooter, called AudioInSpace, which extends approaches by Hastings et al. [5] for weapon particle generation by adding musical inputs to the CPPN. Shown in figure 3, the player moves through levels in an outerspace environment attempting to avoid obstacles and shoot enemies while simultaneously evolving weapon trajectories and visuals with the generated accompanying audio. By directing the two CPPNs to create patterns from information gathered from both domains, this approach is one of the first to incorporate mixed-initiative co-creativity while combining two different PCG modules.

Upon launch, the colors and trajectory of the first bullet are calculated when a random note from the C Major pentatonic scale is played and input to the visual CPPN in figure 4a. Together with the pitch information from the initially random MIDI note, this CPPN also inputs the $(\delta x, \delta y)$ position between where the bullet was fired and where it is currently located with respect to where it was fired, and the time t since firing. Note that before the bullet is fired, the initial $(\delta x, \delta y) = (0, 0)$ and $t = 0$. Outputs of the visual CPPN determine the red, green, and blue (RGB) color values for each bullet, and the (x', y') outputs dictate the bullet's new position (i.e. its trajectory). The color and trajectory of each bullet is calculated every time that a new note sounds.

New notes are fired when the bullet hits an enemy or otherwise at the end of the current note's duration. Shown in figure 4b, the music CPPN determines pitch and a notes duration through the respective Pitch and DeltaTime outputs. These values are based on the (x, y) position of where the last bullet was fired, time since firing t, whether the bullet struck an enemey h, and its RGB values.

As the game progresseses, the player moves through different levels encountering obstacles and hostile enemies. large rocks to avoid, and small rocks to avoid and shoot, preference ratings are given to weapons for every bullet fired and the audio CPPNs fitness is increased for every new note. The longer each weapon and audio network are in play, the higher that CPPN is rated. It is assumed that the longer weapons and visuals are in play, the more preferable

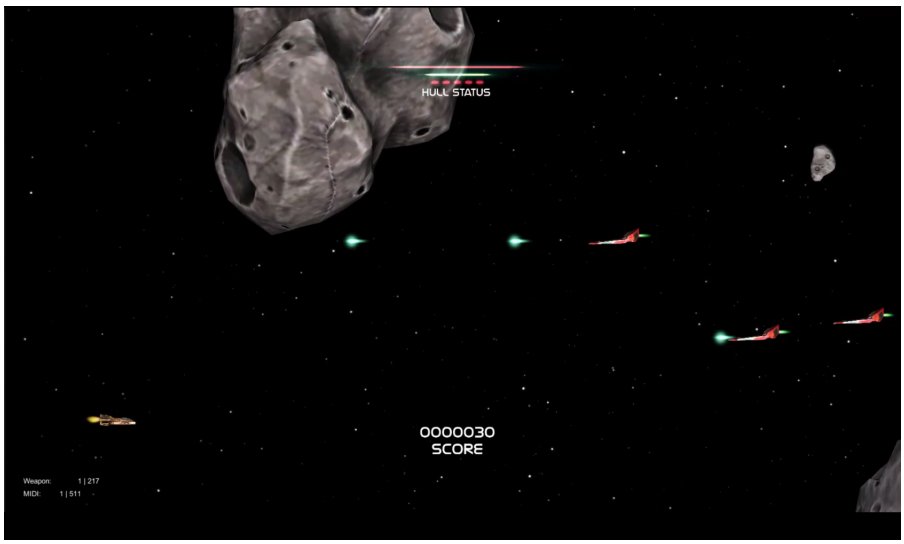
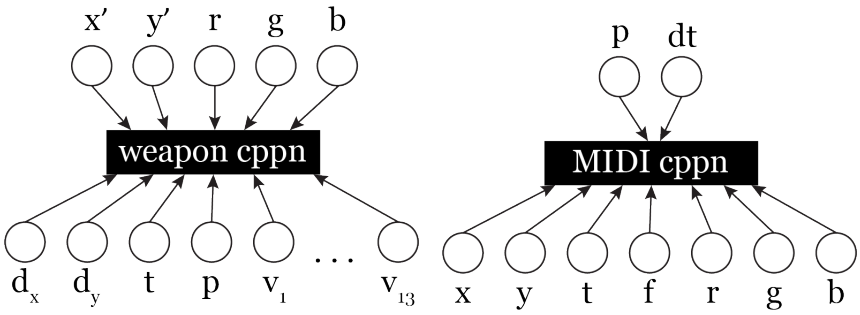


Fig. 3. AudioInSpace. In AudioInSpace, the user controls the space ship on the left side of the screen while moving through each space level. The health of the ship is shown at the top of the screen as "hull strength" while the weapon score is displayed at the bottom. Two mirrored visual beams project from the player's spaceship shooting patterns directed by the current visual CPPN. The rate of fire and the pitches heard result from the audio CPPN's Pitch and DeltaTime outputs.



(a) Weapon CPPN (b) Audio CPPN

Fig. 4.

they are to a particular player. Therefore through IEC players evolve the visual and audio networks toward their aestetic preferences.

4 Experiments

Experiments in this section illustrate the potential power of this proof-of-concept by exploring the interactions and effects of the model. In the first experiment, inputs to each CPPN are simplified to examine their impact on the alternate CPPN. To illustrate the effects of the MIDI on the visual CPPN, rather than generate MIDI notes from the CPPN, they are hard coded and the resulting weapon visuals are observed. Both a constant MIDI value and scaling MIDI notes between MIDI values 20 and 90 are input to the visual CPPN. The simplified visual network obtained by restricting the MIDI input to a single note is then loaded back into the game where the musical CPPNs are evolved without input to the visual CPPN. Then, in this way, the relationship between audio and visuals can be more thouroughly investigated.

The second experiment explores the complete interaction between the two networks when evolved by players. In these experiments, a note in the pentatonic C major scale is sent to the visual network. The visuals are then generated in loop with the MIDI notes.

In each of the experiments, tournament selection occurs amongst a population of fifty individuals, ten of which are rated by the player through IEC. Players can switch the current weapon or audio reprentation by pressing a button to activate a different CPPN chromosome whose effects are immediately heard and seen in AudioInSpace. The evolutionary parameters are set through preliminary testing, and the probability of adding a new node or connection is 30% and 15% respectively. Similarly, the activation functions at any node can be replaced by one of the following with an even chance: Hidden Node Functions: Sigmoid, Hyperbolic Tangent, Sine, Cosine, Bipolar Sigmoid, Gaussian, Ramp, Step, Spike [5]. However, the activation function of any particular node is only changed with a 20% chance. Weights mutated with a 90% chance real values between $[-10, 10]$. Ocassionally (1% chance), connections are disabled.

5 Results

Videos of the results in this section are available at http://todo-link. The first set of results explores the combination of visual and audio CPPNs by first examining the weapons with a predetermined MIDI pattern. MIDI note TODO is input to the visual network every TODO-deltatime. While the bullets shown in figure **??** are TODO because of the other network inputs, the bullet timing is set and TODO.

time results

number of generations

likes for each gun

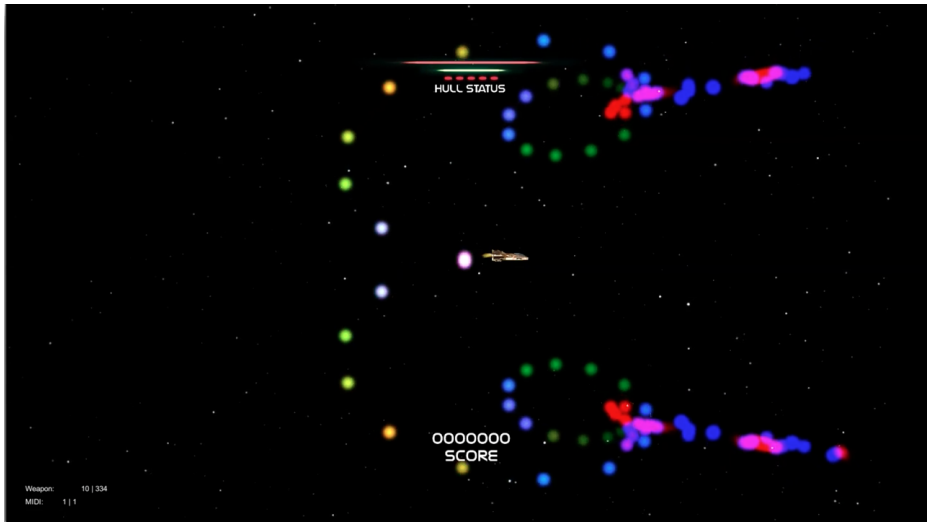


Fig. 5. TODO Title TODO capition

initial feed for music, constant on either side, see which generate what

Because the CPPNs can potentially represent any relationship, sometimes an inverse relationship Note trajectory outputs can go backwards, may be better for different types of levels

pitch helps determine speed

6 Discussion and Future Work

While this paper is a proof-of-concept about fusing game and player creativity, it is important that players enjoy this new mixed-initiave experience. Future work aims to explore players' appreciation of the visuals, audio, and game play, and whether the mixed-initiative interaction enhances the overall experience. The question is whether players prefer the sound of the additional audio and the control that it provides over the weapon visuals. AudioInSpace will be play tested with the work by Cachia et al. [1] as a control, examining average game time, perceived enjoyment, generations per audio and visual CPPNs, and complexity of the relationships between inputs and outputs of each CPPN.

While it is currently assumed that the number of times a gun is fired and the length of time an audio network is heard reflect player preference, another interesting question is why players choose certain weapons and audio. For example, do players focus more on the look of the weapon or the rate of fire determined by the audio? Answers to these questions could help identify player styles and optimize potential development areas.

However, there are also many different avenues for increasing the quality of the visuals and audio, and the cooperation between the two. Like Galactic Arms

Race for instance, the initial generation of weapon and audio CPPNs could be preselected for quality and gameplay.

Monitors on the music to trigger events in game - appearance of enemies, frequencies, boss battles

Narrative?Level design

Game about learning weapon patters, evolving weapon patterns

7 Conclusion

This paper presented a mixed initiative co-creative goal for developers and players. Through a space shooter called AudioInSpace, players evolve their own weapons and audio to suit their aestetic tastes and tactical demands of the current level. Results from this proof-of-concept game illustrate that a significant impact is made on the visuals when combined with simultaneously generated audio patterns.

However,

Extensibility to other games, other disciplines, out of the box

8 Acknowledgements

[Omitted for anonymous review]

References

[1] William Cachia, Luke Aquilina, Hector P. Martinez, and Georgios N. Yannakakis. Procedural generation of music-guided weapons. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, page TODO.

[2] Karen Collins. An introduction to procedural music in video games. *Contemporary Music Review*, 28(1):5–15, 2009.

[3] Michael Cook and Simon Colton. A rogue dream: Automatically generating meaningful content for games. In *Proceedings of the AIIDE Workshop on Experimental AI and Games*, 2014.

[4] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[5] Erin J. Hastings, Ratan K. Guha, and Kenneth O. Stanley. Automatic content generation in the galactic arms race video game. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4):245–263, 2009.

[6] Nils Iver Holtar, Mark J. Nelson, and Julian Togelius. Audioverdrive: Exploring bidirectional communication between music and gameplay. In *Proceedings of the 2013 International Computer Music Conference*, 2013.

[7] Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Computational game creativity. In *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.

[8] Audio Surf LLC. Audiosurf. www.audio-surf.com, 2011.

[9] Sebastian Risi, Joel Lehman, David D'Ambrosio, Ryan Hall, and Kenneth O. Stanley. Combining search-based procedural content generation and social gaming in the petalz video game. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.

[10] Marco Scirea. Mood dependent music generator. In Dennis Reidsma, Haruhiro Katayose, and Anton Nijholt, editors, *Advances in Computer Entertainment*, volume 8253 of *Lecture Notes in Computer Science*, pages 626–629. Springer International Publishing, 2013.

[11] Kenneth O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 8(2):131–162, 2007.

[12] Julian Togelius, R De Nardi, and Simon M. Lucas. Towards automatic personalised content creation for racing games. In *Proceedings of IEEE Symposium on Computational Intelligence and Games*, pages 252–259. IEEE, 2007.

[13] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186, 2011.

[14] Julian Togelius, Noor Shaker, and Mark J. Nelson. Introduction. In Noor Shaker, Julian Togelius, and Mark J. Nelson, editors, *Procedural Content*

Generation in Games: A Textbook and an Overview of Current Research. Springer, 2014.

[15] Mike Treanor, Bryan Blackford, Michael Mateas, and Ian Bogost. Game-o-matic: Generating videogames that represent ideas. In *Proceedings of the FDG Workshop on Procedural Content Generation*, 2012.

[16] Geogios N. Yannakakis. Game ai revisited. In *Proceedings of the 9th conference on Computing Frontiers*, pages 285–292, 2012.