A General Level Design Editor for Co-creative Level Design

Matthew Guzdial, Jonathan Chen, Shao-Yu Chen, and Mark O. Riedl
School of Interactive Computing
Georgia Institute of Technology
mguzdial3@gatech.edu, jonathanchen@gatech.edu, shao-yu.chen@gatech.edu, and riedl@cc.gatech.edu

Abstract

In this paper we describe a level design editor designed as an interface to allow different AI agents to creatively collaborate on level design problems with human designers. We intend to investigate the comparative impacts of different AI techniques on user experience in this context.

Introduction

Co-creation refers to the practice of pairing together human and artificial intelligent (AI) designers to collaborate on the same design task. In this paper we describe a general level design editor for co-creative level design. By general we refer to the tool's ability to slot in different AI agents in the co-creative design activity. For example, the system could go from a AI level design agent driven by convolutional neural nets to one driven by genetic algorithms. Further we note that the tool is general to tile-based 2D games, though we use Super Mario as a test case. With this tool we intend to investigate the comparative effects of different AI techniques on the human designer's experience. During this demo we hope to solicit expert feedback on the tool as we prepare for a human subjects test with both novice and expert level designers.

Related Work

Co-creation as a framework for human-computer interaction exists across many fields and under many names, such as mixed-initiative design (Schaffner and Meyer 2006) or human-robot interaction (Goodrich and Schultz 2007). In the field of games there have been many intelligent, co-creative design tools (Young and Riedl 2003; Smith, Whitehead, and Mateas 2010; Bauer, Cooper, and Popovic 2013; Butler et al. 2013; Yannakakis, Liapis, and Alexopoulos 2014; Machado, Nealen, and Togelius 2017). These tools vary in terms of their focus, such as visualizing stealth in levels (Tremblay et al. 2013), and their intended audience, such as children (Banerjee et al. 2016). However, as far as the authors are aware, these design tools generally focus on a single AI approach to inform the intelligence of the intelligent tools.

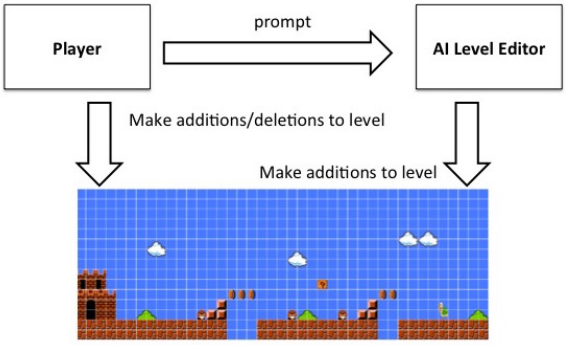

Figure 1: A diagram of the intended flow of control through the system.

Danesh (Cook, Gow, and Colton 2016), a tool to help designers use procedural content generation, represents the most similar tool to our level editor. It allows for multiple different AI approaches. However, there is a difference in focus, Danesh's AI approaches do the majority of design work with a user acting as a curator rather than an equal partner. Further, our editor focuses on comparing different AI approaches' performance on the same task while Danesh uses different AI approaches to generate different content.

Editor Interface Overview

The primary design question for our level editor was how to facilitate the interaction between human and AI designer. At first we considered real-time interaction. In this format the agent would make suggestions for additions each time the human designer changed the level. We recognized that this could prove annoying to human designers making rapid changes. Further, some AI techniques may require more processing time in effect causing lagging or long load times.

We note a similarity between a human designer waiting on an AI designer's edits and players dealing with latency in games. Drawing on literature on design approaches for combatting latency (Claypool and Claypool 2006; Shea et al. 2013), we decided on a turn-based approach. We visualize the format of this interaction in Figure 1. The player/user starts all actions, prompting the AI level editor for suggested additions to the level. Further, while the user can make addi-

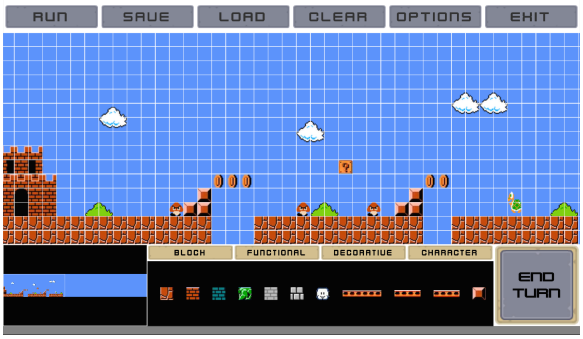

Figure 2: A screenshot of the level editor.

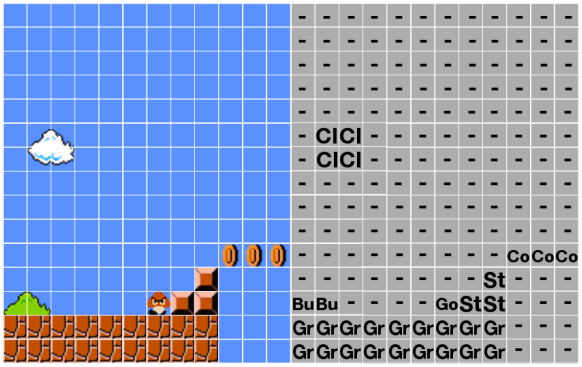

Figure 3: A comparison of the level design editor display and the same section in the grid-based representation.

tions and deletions (including removing the AI's work), the AI designer is only capable of making additions, in order to minimize user frustration.

We visualize the final level editor interface in Figure 2. We note that beyond the previous design decisions we took inspiration from 2015's *Super Mario Maker* (Nintendo 2015). While our level editor allows for level components from any 2D tile-based game, we make use of level components from the Super Mario series at present. Thus it follows to emulate the design experts behind Super Mario in their attempt to create an approachable level design interface.

Figure 2 represents all of the features of the level design interface. At the top of the image one can see the run button that allows designers to play the current level design, along with saving, loading, and clearing functionality. At the bottom of the screen one can see a minimap and a palette of level components that users can paint onto the grid in the middle, that represents the current level. A large end turn button on the right allows the designer to give up agency to the AI designer. After a brief "thinking" pop-up to indicate that the AI designer is processing the current level the level design editor animates the agent's suggested additions appearing on the screen one-by-one, to give the illusion that the AI designer is using the same interface.

AI Level Design Agents

In this section we discuss two of the AI level design agents implemented in our system: Guzdial and Riedl's probabilistic level design model (Guzdial and Riedl 2016) and Summerville and Mateas' LSTM Super Mario Bros. level generator (Summerville and Mateas 2016). We note that we have explored many AI techniques for level design agents as part of the Co-creative level design tool, but lack the space for a full break down of each and every one. Instead we will note both how the two systems listed above integrate into the tool and how the other AI systems do in general.

When the user pressed the 'End Turn' button the system creates a grid-based representation of the current level as seen in Figure 3. This grid is a 2D map with string values, with the string values corresponding to the file name of the tile that was placed there (e.g. "Coin" for coins, "Ground" for ground, etc). Notably in Figure 3 we only include the first two letters of this string value. The grid is sent to a server that transforms the grid to the representation required for the current AI level design assistant. For example, a list of shapes for Guzdial and Riedl, and a 1D sequence for the Summerville and Mateas. This parser is hand-authored for each AI assistant, though notably many approaches can take an untransformed grid.

Once the level design assistant has a representation of the current level is queried for additions. Because all of the approaches we make use of can function as autonomous level designers, this is accomplished by feeding in the current level representation and taking as output an updated version of the same level. For the Guzdial and Riedl assistant, the system adds level components above a hand-defined probability threshold. For Summerville and Mateas we query the system for each currently empty tile of the level. We run an inverted form of the hand-authored transformative parser on the outputted, updated level in order to get a grid-based level back. By comparing the 2D map of string values from the original input level and updated output level in this grid representation we gather a list of changes of the form (x,y,string value). This list of additional level components are passed back to the level editor and displayed as visualized back in Figure 1.

Planned User Study

We created this system with the goal of comparing different AI approaches within a level design framework. For example consider a Convolutional Neural Network (CNN)-based AI agent compared to a Long Short-Term Memory (LSTM) Recurrent Neural Network-based AI agent. Because a CNN makes decisions based on local windows of information and a LSTM can make decisions based on longer-dependencies, a CNN level designer might make suggestions that make more sense to a human co-designer, but an LSTM may make decisions that appear more surprising.

We intend to use a within-users study in which designers interact with multiple AI assistants (2-3) to produce levels, and then ask each designer to rank each AI assistant according to a set of features. We chose rankings over ratings (**?**) due to the advantages of ranking over rating systems. We intend to ask users to rank on a set of features that requires

users to reflect on the level design tool as software (ease of use, confusion, etc.) and as a partner (creativity of suggestions, value of suggestions, etc). We hope to gain feedback from the community and solidify these features during the workshop.

Discussion

In this paper we describe the design for a general level design editor for co-creative level design. During the demo we hope to demonstrate the quality of the level editor along with the experience of interacting with several different AI level design agents. We plan to take any advice or suggestions forward towards an eventual user study of novice and expert users to investigate the comparative effects of different AI agents on human designer experience.

Acknowledgments

The authors would like to thank Adam Summerville for his work and insight on the initial version of this project and all the members of the Entertainment Intelligence Lab.

References

Banerjee, R.; Yip, J.; Lee, K. J.; and Popović, Z. 2016. Empowering children to rapidly author games and animations without writing code. In *Proceedings of the The 15th International Conference on Interaction Design and Children*, 230–237. ACM.

Bauer, A. W.; Cooper, S.; and Popovic, Z. 2013. Automated redesign of local playspace properties. In *FDG*, 190–197.

Butler, E.; Smith, A. M.; Liu, Y.-E.; and Popovic, Z. 2013. A mixed-initiative tool for designing level progressions in games. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, 377–386. ACM.

Claypool, M., and Claypool, K. 2006. Latency and player actions in online games. *Communications of the ACM* 49(11):40–45.

Cook, M.; Gow, J.; and Colton, S. 2016. Danesh: Helping bridge the gap between procedural generators and their output. In *Proc. PCG Workshop*.

Goodrich, M. A., and Schultz, A. C. 2007. Human-robot interaction: A survey. *Foundations and trends in human-computer interaction* 1(3):203–275.

Guzdial, M., and Riedl, M. 2016. Game level generation from gameplay videos. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Machado, T.; Nealen, A.; and Togelius, J. 2017. Cicero: Computationally intelligent collaborative environment for game and level design. In *Proceedings of the 3rd Computational Creativity and Games Workshop*. ACC.

Nintendo. 2015. Super Mario Maker. Nintendo Entertainment System.

Schaffner, J., and Meyer, H. 2006. Mixed initiative use cases for semi-automated service composition: A survey. In *Proceedings of the 2006 international workshop on Service-oriented software engineering*, 6–12. ACM.

Shea, R.; Liu, J.; Ngai, E. C.-H.; and Cui, Y. 2013. Cloud gaming: architecture and performance. *IEEE network* 27(4):16–21.

Smith, G.; Whitehead, J.; and Mateas, M. 2010. Tanagra: A mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, 209–216. ACM.

Summerville, A., and Mateas, M. 2016. Super Mario as a string: Platformer level generation via LSTMs. *DiGRA/FDG*.

Tremblay, J.; Torres, P. A.; Rikovitch, N.; and Verbrugge, C. 2013. An exploration tool for predicting stealthy behaviour. *IDP* 13.

Yannakakis, G. N.; Liapis, A.; and Alexopoulos, C. 2014. Mixed-initiative co-creativity. In *FDG*.

Young, R. M., and Riedl, M. 2003. Towards an architecture for intelligent control of narrative in interactive virtual worlds. In *Proceedings of the 8th international conference on Intelligent user interfaces*, 310–312. ACM.