Playable Experiences at AIIDE 2014

Nathan R. Sturtevant
University of Denver
sturtevant@cs.du.edu

Jeff Orkin
Giant Otter Technologies
jorkin@media.mit.edu

Robert Zubek
SomaSim LLC
robert.zubek@gmail.com

Michael Cook
Goldsmiths, University of London
michael.cook06@imperial.ac.uk

Stephen G. Ware
University of New Orleans
sgware@uno.edu

Christian Stith
Clemson University
cstith@g.clemson.edu

R. Michael Young, Phillip Wright
North Carolina State University
{young,pcwright}@csc.ncsu.edu

Squirrel Eiserloh
SMU Guildhall
squirrel@eiserloh.net

Alejandro Ramirez-Sanabria, Vadim Bulitko
University of Alberta
{ramirezs,bulitko}@ualberta.ca

Kieran Lord
Strange Loop Games
cratesmith@cratesmith.com

Abstract

AIIDE 2014 is the second AIIDE event that has featured a playable experience track. This paper describes the seven entries that were accepted in the 2014 track, as well as the motivation behind the track and the criteria used to evaluate and accept entries.

Introduction

In his 2010 AIIDE Keynote, Chris Hecker concluded with a slide stating: "You are hosed. By 'you', I mean anyone who tries to do work in game AI without doing the game design part. By 'doing the game design part', I mean actually making compelling games."[1]

In 2013 the AIIDE community recognized this challenge and the need to highlight not only research being performed on AI in games, but also the products that are being created as part of this research. As a result, a new track was begun for playable experiences.

The 2014 Playable Experience track, chaired by Jeff Orkin and Nathan Sturtevant, includes seven entries. This paper highlights the work that appears in the 2014 track as well as the evaluation criteria used for accepting entries.

Evaluation Criteria

The definition of a playable experience is intentionally broad; it has been used both for developers in industry to show off the AI they are doing with their games and also for researchers to show their work. This necessarily results in experiences with a range of polish, which we believe is both necessary and acceptable.

Evaluation criteria for the track has focused on (1) articulable innovation in the use of AI, and (2) that the experiences are sufficiently complete and polished enough for naive users to play them. The first criteria is the most important; submissions were rejected primarily because the authors did not sufficiently articulate the AI innovation in their playable experiences. Under this criterion we could accept *To That Sect*, an experience that would never be accepted on its own merits. This experience was accepted because of the novel AI process used to create the playable experience itself.

The experiences accepted this year include:

- *1849*, a commercial game which uses an AI-based production-rule system, by Robert Zubek and Matthew Viglione
- *To That Sect*, a game created by ANGELINA – an automated game-creation program written by Michael Cook
- *The Best Laid Plans*, a research prototype that uses an automated planner to drive gameplay, created by Stephen Ware, R. Michael Young, Christian Stith, and Phillip Wright
- *Everyday Genius: SquareLogic*, a commercial puzzle game with smart content generation, created by Ken Harward and Squirrel Eiserloh
- *PAST: Player-Specific Automated Storytelling*, an adaptive text-based game by Alejandro Ramirez-Sanabria and Vadim Bulitko

[1] http://chrishecker.com/My_AIIDE_2010_Lecture_on_Game_AI

- *Penguin Push*, a puzzle game with automated content generation by Nathan Sturtevant[2]
- *Vessel*, a commercial game featuring many AI technologies by Kieran Lord, John Krajewski, Martin Farren, Mark Filippelli, and Milenko Tjunic.

The following sections are written by the authors of each experience and highlight the unique aspects of what they have created.

1849

1849 is a simulation game set during the California Gold Rush, released in May 2014 for PC/Mac computers, and iOS and Android tablets.

Figure 1: A screen shot of the game *1849*.

The game is a city builder simulation. As the mayor, your task is to build towns, populate them with workers, and keep your citizens housed, fed, and entertained. You'll have to manage and coordinate extensive manufacturing and trade networks to make sure your towns thrive.

At the core of the game is a production rules engine: the economic and logistics simulation is written as sets of rules in this engine that get evaluated periodically and change the game state over time.

We can talk about the rules engine in terms of layers:

- A simple data model for representing knowledge about each unit and about the world
- Rules of the simulation expressed in a succinct domain-specific language
- Rules engine itself to match and execute the rules

Simulation in *1849* is focused on resources: production, conversion, and consumption, by a variety of actors such as the player, buildings (eg. wheat farm produces wheat), natural elements (eg. trees produce lumber), and even the board tiles themselves (eg. gold stored in the ground, waiting to be mined). Rules of the simulation are most often concerned with checking if some conditions have been met, then consuming and/or producing resources, and executing side-effects. Three blog posts contain some examples:

- Intro: http://tiny.cc/1849-rules-1
- Details: http://tiny.cc/1849-rules-2
- Performance: http://tiny.cc/1849-rules-3

To make the system efficient, we adopted two drastic constraints. First, we required that each rule element (precondition and post-condition) must be possible to evaluate in constant or near-constant time. In order to support this, we don't allow for any free variables in the rules, to eliminate the need for potentially expensive unification. Instead, we use deictic variables that get bound ahead of evaluation time. Second, we've chosen a succinct and efficient data model, using mostly flat pre-allocated vectors and avoiding runtime memory allocation. This allows for query and update operations with $O(1)$ time complexity. These two constraints limit the expressiveness of our rule set, but result in an efficient rule system that runs the entire game on a fraction of the available processor time, and doesn't affect frame rate even on underpowered mobile devices.

Performance was clearly a high point of the system, which can run cities with hundreds of active entities without breaking a sweat, even on comparatively underpowered tablet devices. We could probably push production rules even further, if the rendering subsystem had not claimed all available processor cycles already.

Finally, the last piece of the puzzle was expressing these rules in a domain specific language built on top of JSON. Having major parts of gameplay be defined in external files, which get loaded up at runtime and are not subject to compilation, greatly improved our iteration speed and authorial control.

ANGELINA - To That Sect

To That Sect is a game jam entry to Ludum Dare 28, made by ANGELINA - a piece of software being developed as part of research into computational creativity. ANGELINA has been through many iterations in the past, in which it designed arcade games (Cook and Colton 2011), news-inspired platforming games (Cook, Colton, and Pease 2012), as well as generating mechanics and levels for puzzle-platformers (Cook et al. 2013). All versions of ANGELINA have been built using co-operative co-evolution (Potter and De Jong 2000), a kind of evolutionary computation that uses multiple evolving systems working in tandem. Each system evaluates the content it produces in the context of what the other systems are producing, and in doing so a form of co-operation arises between the content generated by each individual system. This means that a system for evolving levels will try to design levels which work well with the rules being evolved by the rest of the system, for example.

The latest version of ANGELINA is being developed to investigate two new objectives: the generation of program code during game design, and the integration of creative software within creative communities. The former objective is still being worked on, however the latter has been a concern since the start of the project. A major objective for us in developing the new version of ANGELINA was to ensure that it could engage with communities of game developers in natural ways. One way in which we've begun to work towards this goal is by having the software enter

[2]As this author was also a chair of the Playable Experience track, it was evaluated independently and subject to a higher bar for acceptance.

game jams, time-limited competitions in which people make games based around a certain theme. ANGELINA can now be given a natural-language phrase and will use this phrase to start a process of media acquisition and linguistic analysis (to assess any emotions commonly associated with the input phrase, for example). This provides a palette of 3D models, sound effects, music and text to use during the evolutionary design process.

ANGELINA came 500th out of 780 entries in the 'Overall' score category, after a peer review process by other entrants to the contest. Alongside *To That Sect* we also entered a second game, Stretch Bouquet Point. This game was entered to the jam without any public mention of ANGELINA or of the game being designed by software. This game ranked lower than *To That Sect* in every category except humour (which we believe was an unintentional win for Stretch Bouquet Point, since the game was met with an incredulous reaction from many reviewers). By considering the relative scores of the two games, and the language used in comments written alongside the scores, we believe there is currently considerable positive bias towards software like ANGELINA in both the player and developer communities. We explore this in more detail in (icc 2014).

The Best Laid Plans

The Best Laid Plans is an interactive narrative point-and-click adventure game that uses narrative planning techniques to generate its story at run time. As a research prototype and proof-of-concept, it demonstrates how fast planning algorithms can create interactive stories by leveraging computational models of narrative like character intentionality and conflict.

The player takes on the role of a hapless goblin minion who has been ordered by the Dark Overlord to go to town and fetch a bottle of hair tonic. The game tells the story of how that quest goes horribly wrong by alternating between two modes. In *Make Your Plan* mode, the player acts out how the goblin will fetch the hair tonic while the NPCs do nothing. Play then changes to *Watch Your Story Unfold* mode, and the player watches that plan happen. NPCs will now act to thwart the goblin's plan. At any time (or if the goblin dies) the player can return to *Make Your Plan* mode to change the goblin's plan. Play continues in this fashion, with the player acting out plans and seeing those plans thwarted, until the goblin successfully reaches the Dark Tower with the hair tonic in hand.

For example, the player can act out a plan to walk to town, purchase the tonic, and walk back to the Dark Tower. When watching this plan unfold, a bandit will intercept the goblin on the road and kill him to steal the tonic. The player must now find some way around the bandit, and so might chooses to return to town, steal a sword, and fight his way past the bandit. When watching that plan unfold, the bandit chases the goblin into town. The town guard attacks the bandit, but then proceeds to also attack the goblin for stealing the sword. Now the player must find some way to avoid the town guard, and so on.

Neither the game's story nor the NPC behaviors are prescripted at design time. The story is constructed by the

Figure 2: The interface for *The Best Laid Plans*. The current game mode of *Make Your Plan* is displayed at the top of the screen. The Dark Overlord (top left) watches telepathically and occasionally offers information and guidance. The player's current plan is displayed under the portrait of the Dark Overlord. The player's inventory, mana, and score are shown at the bottom of the screen. The location pictured here is the town. The characters in the scene are, from left to right, the town guard, the goblin, and the weapons merchant. The player can interact with the other characters and items with a simple point-and-click interface. Yellow arrows allow the goblin to walk to other locations, such as the tavern (west), the alley (north), the junction (south), and the potion shop (east).

Glaive narrative planner (Ware 2014) from a set of 10 atomic action building blocks (e.g. "Character x picks up item y at location z.") at run time.

A planner is an algorithm which finds a sequence of these atomic actions that achieve some goal (Russell and Norvig 2003). The planner's goal is the opposite of the player's goal: that the goblin not have the hair tonic or that the goblin be dead. Glaive extends classical planning by also reasoning about narrative phenomena. It uses Riedl and Young's intentional planning framework (2010) to ensure that every action taken by a character is clearly motivated and goal-oriented for that character. It also uses Ware and Young's model (2014) of narrative conflict to represent plans which fail or are thwarted by other characters.

Unlike the original narrative planning algorithms described by Riedl and Young (2010) and Ware and Young (2011), Glaive is fast enough to be used at run time for this game. Glaive is based on Hoffmann and Nebel's Fast-Forward planner (Hoffmann and Nebel 2011). It uses advances from state-space heuristic search planning as well as additional reasoning about the structure of goal-oriented plans to quickly discover stories where NPCs act believably and create conflict with the protagonist to make the game more challenging. This game serves as a proof-of-concept for how advanced AI techniques can be used to create stories intelligently from atomic building blocks. It can be downloaded from http://liquidnarrative.csc.ncsu.edu/blp.

Everyday Genius: SquareLogic

Everyday Genius: SquareLogic (*SquareLogic* for short) is an

Figure 3: A 7x7 double-board, hidden-cage puzzle from *Everyday Genius: SquareLogic*. This puzzle is one of 200 found in the game's 23rd location (Summit) of 42 locations included in the PC version.

award-winning puzzle game created by Ken Harward and Squirrel Eiserloh at TrueThought LLC, released for Windows/PC in 2009, and subsequently ported (third-party) to iPad. The game received very positive reviews ("sublime", "a phenomenal game", "hauntingly beautiful", "quite ingenious"), was named Best of 2009 by GameZebo, and has been used in numerous classrooms. Players have logged thousands of hours in the game, making it one of the "stickiest" games on Steam (where the game has currently received 92 positive and 0 negative reviews).

SquareLogic's gameplay is a combination and extension of concepts from numerous Latin-squares-based math games, and involves the selection of single-digit numbers in cells in a square grid adhering to all of the rules imposed by a given puzzle. One cell might be required to be less than another cell, while another "cage" of cells are required to add up to 7 or form a sequence (e.g. 3,4,5,6).

The game provides 20,000+ procedurally-generated puzzles in 42 categories. With each category, a new rule (or rule combination) is introduced; for example, double-board puzzles (in which two different boards share a set of parallel answers) or hidden-cage puzzles (in which cage topography must be deduced, Figure 3). Puzzles increase in complexity, initially starting with two rules and a 4x4 grid, and culminating with 9x9 puzzles which use all 12 rules and require 400+ steps. Puzzles are presented in order of increasing difficulty, and each is guaranteed to be deductively solvable (without guessing) regardless of the solution approach chosen.

AI was employed heavily throughout *SquareLogic*'s development in order to assist with puzzle creation, puzzle verification, difficulty balancing, player instruction, and hints. A number of different AI entities/systems were created:

The **PuzzleGenerator** takes a random seed and a set of mechanics and generates a puzzle employing those mechanics. Puzzles generated in this fashion are guaranteed to be filled and Latin-squares compliant, and to make use of all required mechanics (only).

The **PuzzleSolver** plays each generated puzzle: first, to validate its integrity (exactly one solution, obtainable deductively); and second, to estimate its difficulty. 25 solving inferences of varying difficulty were identified and tuned by expert human players; the solver attempts each puzzle using the simplest inferences possible. If solvable, the puzzle is assigned a difficulty rating based on the variety, number, and complexity of inferences used. A "puzzle recipe" – the random seed, rule set, and difficulty – is then saved.

The **PuzzleChef** reads in the puzzle recipes for each of the game's 42 locations, sorts them by difficulty, and divides the list into thirds (Easy, Medium, and Hard). A set of 12 "practice puzzles" are chosen from Easy and Medium. Players may play any or none of these practice puzzles, but must at least solve a "challenge puzzle" before unlocking the next location, revealing puzzles of a new variety. Hundreds of puzzles like the one just solved are also presented, by difficulty, should the player wish to return and experience more.

The **Instructor** monitors player progress throughout the game, noting rules she has encountered and techniques she has exercised. Since player progression through the game is non-deterministic (players may skip or play practice puzzles in any order), the instructor watches for opportunities to teach new concepts as needed. If a new puzzle introduces the Comparison rule for the first time, the Instructor will select a Comparison cage in that puzzle and explain its use within the puzzle's context. If the player has not yet used cage-painting or candidate elimination by the time they are needed, the Instructor will seize the opportunity to instruct the player in their use. The Instructor does not offer tips about techniques already demonstrated by the player.

Finally, a dynamic **Hint System** provides context-specific hints to players when requested, starting vague ("Look at this addition cage") and becoming more specific as requested. The Hint System uses the PuzzleSolver to recommend the next simplest move (which depends on the solution approach the player has taken to this point), or offers other options if the player has committed faulty assumptions (in which case it offers to do nothing, or provide contextual hints about where the player went wrong, or to rewind the player's moves to the moment before the mistake was made).

PAST: Player-Specific Automated Storytelling

Storytelling is an important facet of human life: from entertainment to everyday communication, we are constantly telling and immersing in stories. Understanding how to make these more aesthetically pleasing is of interest to artificial intelligence (AI) research and can be used to improve video games. Specifically, research in *interactive storytelling*—responding to player actions while unfolding a narrative according to a model of experience quality—must tackle several challenges. One key challenge is to strike a balance between *player agency* (i.e., allowing the player to meaningfully influence the story) and *authorial control* (i.e., guiding the player's experience through a desired story trajectory). Increasing the player's agency can allow the player to take ownership of the story but is resource-demanding on the part of the developers. Additionally, it may be difficult to guarantee that the player will have a particular experience when they are given the freedom to change to the story.

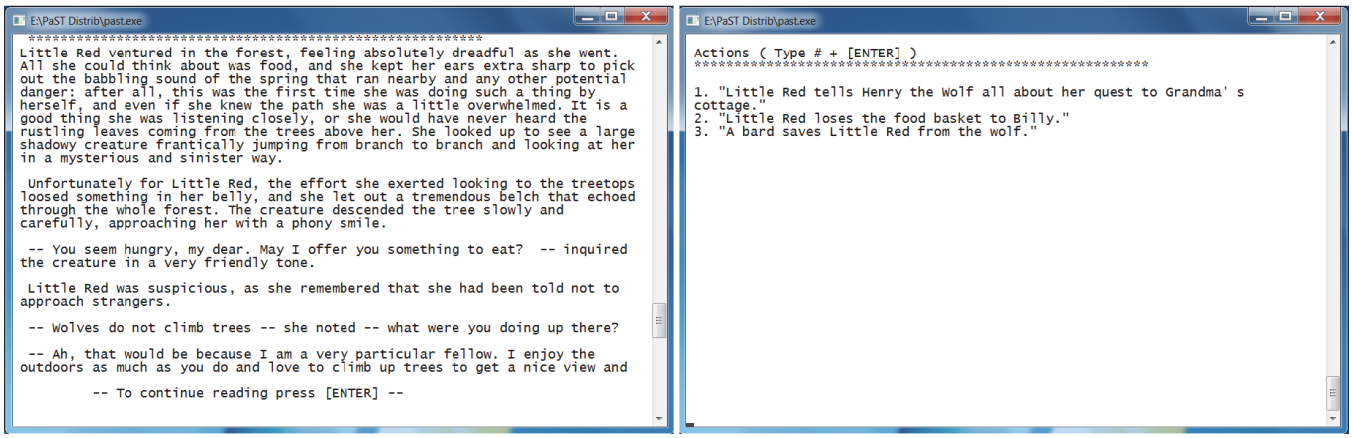

Figure 4: The text-based command line in *PAST*. In the left frame, a story state description; in the right, a series of player actions available.

Player-specific Automated Storytelling (PAST) (Ramirez and Bulitko 2012; Ramirez, Bulitko, and Spetch 2013; Ramirez 2013) is an interactive storytelling system that employs AI to tackle the aforementioned problem. In doing so, PAST relies on *player modelling* and *automated planning*. In line with previous work, *PAST* requires that the story developer describe the story world in a formal, computer-readable format. Subsequently, whenever the player exercises narrative freedom and deviates from the original story (e.g., the player as the Little Red Riding Hood kills the wolf on sight), *PAST* uses *automated planning* to shape the story on the fly so that it still meets authorial goals (e.g., another wolf appears to deceive and eat the Granny). Second, *PAST* shapes the story not only to satisfy authorial goals but also to accommodate the player's play-style inclinations. In the example above, a player who has shown a tendency to play as a fighter will see another mean wolf replacing the former whereas a storytelling inclined player may witness a magic fairy resurrecting the original wolf. These inclinations are learned automatically from the player's previous actions, using *player modeling* techniques.

To evaluate the efficacy of our approach towards improving perceived agency, we created an interactive version the Little Red Riding Hood story. The results of these user studies indicated that *PAST*'s approach to storytelling is more likely to increase the perception of player agency (Ramirez, Bulitko, and Spetch 2013). The playable interactive narrative experience developed for those user studies is hereby presented (Figure 4). A player can advance through the interactive narrative world by making narrative choices via a text-based interface and reading resulting narrative off the screen. The length of the experience depends on the players choices and reading speed, with 10-30 minutes being typical.

Technically, the program is a self-contained Windows executable—a Lisp code base running on Steel Bank Common Lisp—and can be executed on any modern computer running either a 32 or 64-bit version of Windows XP, Vista, 7 or 8.

Penguin Push

Penguin Push was built around the hypothesis that when constraints exist that will help someone solve a puzzle, the puzzles should be selected to maximize the gain acquired by the user from understanding the constraints. We use search tools to build levels for a game which maximizes this gain.

Background

There is wide interest in the general idea of procedural content generation, particularly for generation of maps and natural structures. We are interested in a slightly different problem of procedurally assisted design – tools that will help designers create interesting experiences, but where the designer still has a strong role in the creation process. The design role is just assisted by the strengths of the computer - the ability to efficiently perform significant brute-force computations.

Designer Jonathan Blow has often talked about his process of discovering the puzzles that exist when a set of mechanics are created for a game. In a sense, our goal is to automate parts of this process so that the mechanics can be explored and understood more quickly and thoroughly.

Project Goals

This specific project arose from an analysis of the game *Fling!*[3], a popular mobile game. In our previous analysis (Sturtevant 2013) we looked at this game from the perspective of the number of reachable states in each puzzle (we refer to a single puzzle as a "board"). There are 35 levels of difficulty in the game which are presumably of increasing difficulty. While brute-force analysis reveals that there is a loose correlation between reachable states and the level of a board, there is also over an order of magnitude of variation between boards in the same level.

The game also has constraints – in particular that there is only one solution for every board. This means that there is exactly one move in each board configuration that will lead

[3] http://www.candycaneapps.com/fling/

to a solution. Looking at the reachable states in the reduced state space (eliminating moves that would require the solution to violate the above constraint) we still found the same trends. In particular, that there wasn't a strong correlation between the size of the search space and the difficulty of a level.

We hypothesize that there is a correlation between the size of the search tree and the difficulty of a problem - this is one possible measure of the difficulty of a problem. (Although it isn't difficult to design puzzles for which this rule will not hold, such problems often contain many unnecessary elements which can easily be abstracted away.) Furthermore, we believe that one mark of a good puzzle is that it is relatively easy for experts, but difficult for novices. This suggests further than an expert that deeply understands the constraints in a puzzle should be able to solve it quickly (a small search tree), where a novice that doesn't understand these constraints should have a difficult time (a large search tree). So, the ratio of the brute-force tree size to the constraint-reduced tree size is a second measure that we believe is interesting for level design.

Penguin Push is a re-design of *Fling!* which has levels selected to maximize the ratio of the brute-force search tree to the constraint-reduced tree. An introductory set of levels teaches the mechanics of the game using the small set of levels for which all moves lead to a solution. 50 boards each with 7, 8, or 11 initial pieces respectively were chosen to demonstrate the levels chosen by this metric. An additional 31 boards with 8 pieces were selected that minimize the ratio; these levels provide a contrast in that knowing that only a single solution is possible does not make the level any easier.

Vessel

Vessel is a puzzle platformer by Strange Loop Games that uses fluid physics and AI creatures as the core mechanics in an adventure though an intricately detailed world.

The player takes the lonely role of Arkwright, the inventor of liquid automotons called fluros which provide a near limitless supply of "just add water" labour that has revolutionized the world. When new mysterious types of fluros appear and run amok, Arkwright must investigate using his creations and the mutated fluros he discovers to unearth the greater mystery of where these new creatures came from.

Puzzles in *Vessel* are as much the clockwork interaction of the liquid fluro characters as they are the buttons, pipes and other level elements. Every aspect of the world is physically simulated, and all puzzles and characters besides the player are based in the liquid simulation. Often the player will need to lure a fluro to press a button they can't reach using a light source, or set them moving on a path that will trigger a switch. These creatures and the ability to create them from any source of liquid become the players tools. Progression through the game relies on learning and understanding how they interact with one another and their environment.

Instead of being separate, aesthetic elements even the background machinery, sounds, and music are driven by the state of the game's puzzles, AI and physics simulation. Intricate gears and pumps deliver force and liquids about the levels in the background, steam is a liquid and can lift objects up rather than just be a particle effect. Even subtle hints are given to the player without their knowing, supplied by an adaptive music system which gently creates and transitions between variations of the soundtrack by Jon Hopkins.

By far *Vessel*'s greatest innovation is its optimized liquid simulation and rendering engine; capable of simulating large quantities of flowing water, scalding lava and steam, reactant chemicals, glowing goo, the mysterious protoplasm, and more. Each liquid can have unique properties on its own and and mix with other liquids for dramatic effects. This is directly linked to its skeletal animation system allowing characters made of these liquids to inhabit the environment in large numbers.

Vessel is available on Steam and PS3 and was written by Kieran Lord, John Krajewski, Martin Farren, Mark Filippelli, and Milenko Tjunic.

Conclusions

Playable experiences are an integral part of creating game AI. The 2014 AIIDE Playable Experience track recognizes this need and features seven playable experiences. These experiences use a broad range of AI approaches to create compelling play. We hope that future playable experiences will build on what has been created here and further push the bounds of the use of AI in games.

References

Cook, M., and Colton, S. 2011. Multi-faceted evolution of simple arcade games. In *Proceedings of the IEEE Conference on Computational Intelligence and Games*.

Cook, M.; Colton, S.; Raad, A.; and Gow, J. 2013. Mechanic miner: Reflection-driven game mechanic discovery and level design. In *Proceedings of 16th European Conference on the Applications of Evolutionary Computation*.

Cook, M.; Colton, S.; and Pease, A. 2012. Aesthetic considerations for automated platformer design. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference*.

Hoffmann, J., and Nebel, B. 2011. The ff planning system: Fast plan generation through heuristic search. *Journal Artificial Intelligence Research* 14:253–302.

2014. Ludus ex machina: Building a 3D game designer that competes alongside humans. In *Proceedings of the Third International Conference on Computational Creativity*.

Potter, M. A., and De Jong, K. A. 2000. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evol. Comput.* 8(1).

Ramirez, A., and Bulitko, V. 2012. Telling interactive player-specific stories and planning for it: ASD + PaSSAGE = PAST. In *Proceedings of the Eight AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 173–178.

Ramirez, A.; Bulitko, V.; and Spetch, M. 2013. Evaluating planning-based experience managers for agency and fun in text-based interactive narrative. In *Proceedings of the Ninth*

AAAI Conference on AI and Interactive Digital Entertainment, 65–71.

Ramirez, A. 2013. *Automated Planning and Player Modelling for Interactive Storytelling*. M.Sc. dissertation, University of Alberta.

Riedl, M. O., and Young, R. M. 2010. Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research* 39:217–268.

Russell, S., and Norvig, P. 2003. *Artificial intelligence: a modern approach*. Pearson.

Sturtevant, N. 2013. An argument for large-scale breadth-first search for game design and content generation via a case study of fling! In *AI in the Game Design Process (AI-IDE workshop)*.

Ware, S. G., and Young, R. M. 2011. Cpocl: A narrative planner supporting conflict. In *Proceedings of the 7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 97–102.

Ware, S. G.; Young, R. M.; Harrison, B.; and Roberts, D. L. 2014. A Computational Model of Narrative Conflict at the Fabula Level. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games*.

Ware, S. G. 2014. Glaive: A State-Space Narrative Planner Supporting Intentionality and Conflict. In *Proceedings of the 10th International Conference on Artificial Intelligence and Interactive Digital Entertainment*.