Playable Experiences at AIIDE 2015

Michael Cook
AIR Lab, Falmouth University
mike@gamesbyangelina.org

Squirrel Eiserloh
SMU Guildhall
squirrel@eiserloh.net

Justus Robertson
R. Michael Young
North Carolina State University
{jjrobert,young} @ncsu.edu

Tommy Thompson
James Tatum, Neall Dewsbury
Table Flip Games /
University of Derby
tommy@t2thompson.com

David Churchill
Lunarch Studios /
University of Alberta
dave.churchill@gmail.com

Martin Cerny
Charles University in Prague
cerny.m@gmail.com

Sergio Poo Hernandez
University of Alberta
pooherna@ualberta.ca

Vadim Bulitko
University of Alberta
bulitko@ualberta.ca

Abstract

This paper describes entries to the third Playable Experiences track to be held at the AIIDE conference. We discuss the five entries accepted to the track for 2015, as well as the ongoing development of the track as part of AIIDE.

Introduction

The AIIDE Playable Experiences track began as a means to celebrate and present games which demonstrate compelling uses of AI, and perhaps AI research, within their design. Now in its third year, the track is an opportunity to reflect on how researchers can embed their work in playable games, and the many different ways this can manifest itself. Its intentionally broad reference to '*playable experiences*' is a respectful nod to the breadth and diversity of the modern games industry and its surrounding communities and movements.

In the time since the last Playable Experiences track we have seen the founding of both the Procedural Generation Jam and the AI Jam, events which attracted researchers and practitioners focused on the use of AI techniques within games. Dagstuhl Seminar 15051 also took place earlier this year, bringing together a group of games researchers in an event which had a strong focus on implementing ideas and led to the creation of several game prototypes. The relationship between games research and game development is shifting, and the Playable Experiences track serves as a good way to celebrate and record this.

The 2015 AIIDE Playable Experiences track, chaired by Michael Cook and Squirrel Eiserloh, includes five accepted entries. This paper highlights each entry with a short summary, briefly summarises the criteria used for accepting entries, and poses some thoughts for where the track might progress in the future.

Evaluation Criteria

As the breadth of game AI research and practice has expanded in all directions, so naturally has the variation in scope and focus of Playable Experiences submissions. The selection process has accordingly become more subjective, as direct comparisons between entries are apples-to-oranges. Evaluation of submissions for the track has therefore focused on the articulable and demonstrable innovation in the approach and use of AI as it relates to the playable experience, as well as the level of completeness, playability, and accessibility of the experience itself.

Accepted submissions for 2015 include:

- *Base Case*, a top-down sneaking game by Justus Robertson in which the world is generated from a plan-based experience manager, where the player directly manipulates the planner's underlying state via in-game mechanisms;
- *iGiselle*, an interactive narrative by Sergio Poo Hernandez in which the player influences an AI experience manager and branching narrative choices by assuming dance positions perceived by the Microsoft Kinect;
- *Prismata*, a unique commercial turn-based digital strategy card game in development by Lunarch Studios, submitted by lead AI programmer Dave Churchill, featuring sophisticated AI opponents navigating a large state-space;
- *Sarah & Sally*, a puzzle-platformer by Martin Cerny with a cooperative AI sidekick that telegraphs its search state in-game, set in a problem space designed to highlight and simplify AI search while creating perceived complexity for the human player;
- *Sure Footing*, an infinite runner building on rhythm-based approaches to real-time procedural level design.

The following sections have been provided by the authors of each experience, and provide highlights and insights into the role of AI in that experience and its development.

Base Case

Figure 1: A screenshot from Base Case.

Base Case is a top-down sneaking game where the player must free their captured comrades from a military compound while avoiding detection by the guards that patrol the base. What makes Base Case unique is that the game is generated using a plan-based experience manager's declarative state transition system. This allows the experience manager to control not only the behavior of the game's NPCs but also the configuration of the procedurally generated game world. In order to escape the base for good, the player must find a computer console that allows them to modify the underlying declarative state and open an otherwise hidden exit.

Base Case's gameplay begins with an initial game world state specified in PDDL (McDermott et al. 1998). Using this specification, a plan-based experience manager (Robertson and Young 2014) creates a declarative state transition system and plan to guide the game's NPCs. Based on this state transition system generated by the experience manager, a procedurally generated content pipeline creates an interactive game world and populates it with game objects. The PCG pipeline is implemented with Unity which instantiates and destroys game objects based on the declarative state.

This declarative experience manager and PCG pipeline game engine is called the Unity General Mediation Engine and is used to create Base Case. Base Case is a sneaking game where the player moves through rooms patrolled by enemy NPCs in search of a prisoner. When the prisoner is found, the two must escape the base. When the player escapes the base with the rescued prisoner, the game restarts in a new configuration. This new configuration is one of several pre-authored PDDL problems the system chooses between on initialization. The game continues to repeat as the player explores new configurations of the base.

In order to finish the game, the player must find a special world object and use it to modify the underlying declarative state. In every world state configuration there is a computer that allows the player to directly manipulate the truth value of fully ground atomic formulae describing aspects of the world's state. In order to open the true exit and escape the base for good, the player must find the computer system and use it force the underlying world state into a configuration that will allow the player to access the hidden exit. Once the player escapes from the secret exit, the game stops repeating.

iGiselle

Figure 2: A photograph of iGiselle being played on a Kinect.

iGiselle is an AI-managed interactive narrative inspired by the Romantic ballet *Giselle*. In iGiselle the player takes control of Giselle, a young ballerina, and experiences the interactive narrative through a series of still images, voice overs and music. To further immerse the player in the world of ballet, a traditional game controller was eschewed in favor of having the player indicate his/her narrative choices by assuming dance positions which are perceived by a Microsoft Kinect connected to a PC.

The game utilizes PACE (Player Appraisal Controlling Emotions) as its AI experience manager. PACE determines the next narrative event to show to the player in an attempt to keep him/her on an author-specified target emotional curve. PACE models the player's emotions by determining the player's playstyle from the actions he/she has taken in the game so far. Candidate narrative events that PACE selects from are computed automatically by the Fast Downward AI planner as the narrative domain is encoded in the Planning Domain Description Language (PDDL).

The multimedia content for the game was developed in two phases. First, working with writers, a non-linear narrative graph was developed which allows the player to explore various narratives via choices made during the game. The graph contains 102 narrative events and 4 choice points, resulting in 9 distinct narrative trajectories and 10 possible endings. In phase two, developers worked with ballet dancers and choreographers, voice actors and recording engineers, photographers and graphic artists to create 162 cell-shaded images and 270 lines of studio-recorded voice overs.

The game interface was coded in C# and interfaces with the Kinect framework. A pose recognition module was im-

plemented within the framework to read in the player's poses and interpret them as his/her narrative choices. In total, 44 people were involved in iGiselle production which took approximately a year and a half.

Prismata

Figure 3: A screenshot of the Prismata play area.

Prismata is a fast-paced hybrid strategy game from Lunarch Studios which combines elements from real-time strategy games, collectible card games, and tabletop games. In Prismata players build up an economy, spend resources to buy armies, and unleash attacks on their enemies while simultaneously defending incoming enemy barrages. Think "turn-based StarCraft", but without a map, or Hearthstone with workers and build orders instead of decks. Games in Prismata last just a few minutes, and have infinite replay value due to the randomly selected units up for purchase at the start of a game. Each game there are new units to construct, new build orders to discover, and entirely new strategies to unleash on your opponent.

Several challenges are faced when creating AI systems for modern online strategy games like Prismata:

- Strategy games often have enormous action spaces, with millions of possible action combinations to consider on any given turn.
- Different difficulty settings must be offered so that players of all skill levels can play against and enjoy their experience with the AI.
- As new units are frequently added to the game, the AI system must be robust enough to handle design and balance changes made to the game over time.

To take on these challenges, the Prismata AI system uses Hierarchical Portfolio Search (HPS), a new search technique developed by the presenter, David Churchill, which is described in the accompanying AIIDE 2015 publication: "Hierarchical Portfolio Search: Prismata's robust AI architecture for games with large search space" (Churchill 2015). HPS offers a modular and generic approach to dealing with abstract games with large action spaces, and is especially powerful in strategy games like Prismata or real-time strategy games like Starcraft, where turns have multiple actions which can be broken down into tactical categories. Another benefit of HPS to game designers is that the modular nature of the search system allows for complex strategic behaviours to be constructed from smaller tactical pieces. These modular configurations allow the AI to have multiple difficulty settings and strategic play styles, which are easily created and tuned within seconds. The AI bots in Prismata have many difficulty settings, ranging from the virtual punching bag *Docile Bot* all the way up to experienced player level *Master Bot*, so that anyone can play the game and have fun.

Sarah & Sally

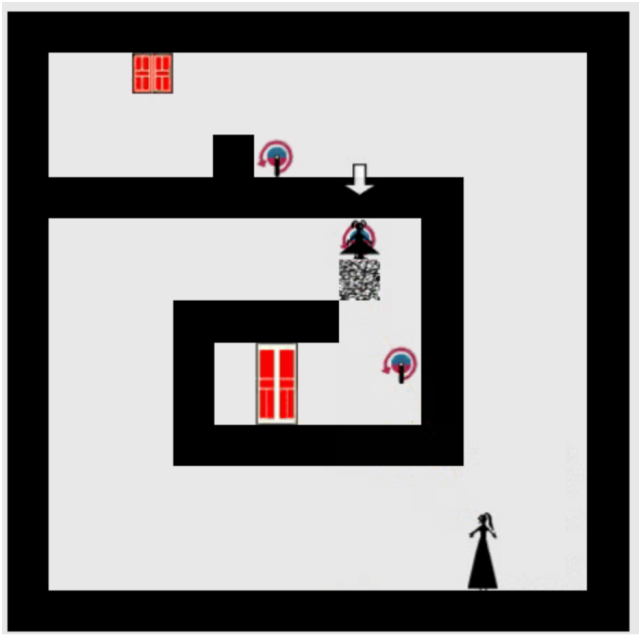

Figure 4: A screenshot of a puzzle in Sarah & Sally.

Creating reasonable AI for sidekicks in games has proven to be a difficult challenge. Sarah & Sally is an experiment in designing around the problems inherent in cooperative AI development: it is a cooperative 2D puzzle-platformer that looks similar to mainstream examples of the genre, but allows for an easy implementation of a quality sidekick AI. The design of the game allows the AI to find optimal solutions using a straightforward search-based approach while the problem remains relatively hard for a human player.

Exploring the experience of cooperating with an AI-controlled character that matches human-level intelligence (in the context of the game) was one of our main interests. We also see the game as an interesting exercise in AI-motivated game design and, to a lesser extent, in communicating the inner state of a search algorithm to the player.

In a cooperative puzzle-platformer, the player switches control between multiple characters and has to complete a task requiring complex collaboration of all the characters – hence the "puzzle" part. The "platformer" ingredient is the fact that the characters are generally affected by gravity and

an important part of solving the puzzles is figuring out how to reach certain platforms in each level.

The most desirable property of those games is that regardless of player skill, the game simply cannot be completed without the characters cooperating. We thus designed a game with two characters where one is controlled by the player, and the other one is an AI-controlled sidekick. We chose the protagonists to be two girls with distinct appearances: Sarah (the player character) is small while Sally (the AI sidekick) is tall. The cooperation of the characters is enforced by their complementary abilities. To increase the feeling that the sidekick is helping, her ability (levitating the player character) is very powerful.

To make the game amenable to a simple search-based AI, the game logic operates on a grid. At the same time we try to hide this fact from the player by making the movement of the characters smooth. Moreover, only one character is active at a time and the other character cannot perform any action until the active character ends her turn.

One of the most interesting challenges in developing the game was exposing some of the AI's internal state to the player to make the collaboration possible. Note that it is not desirable to expose its complete state, because the AI knows the solution for the level and showing it to the player would render the game uninteresting. The most important moment to communicate is when the AI decides to perform no action at all as these situations can easily lead to player frustration. A nice side-effect of the search-based sidekick AI is that a lot of information that is of interest to the player (e.g. the AI character expects the player to help her) can be acquired by analysing the optimal action sequence and/or the top levels of the search tree.

In early playtests we noticed that players often think that a solution is feasible, when in fact it is not. They would then get frustrated because the AI refuses to help them. This state is very difficult to communicate, as the AI would have to understand what the player is trying to do and explain to him why it is not possible. An even trickier situation arises when a solution is found, but the player arrived at a different (longer or incorrect) solution. Luckily, the simplistic design of our game lets us detect both of these situations with reasonable accuracy and develop simple behaviours for the AI that allow the game to progress and usually lead to the player's realization of the flaws in his approach. In a more complex game this would likely be a very difficult problem.

The game is also a foundation for a paper entitled "Sarah and Sally: Creating a Likeable and Competent AI Sidekick for a Videogame" (Cerny 2015) accepted to the EXAG workshop at AIIDE 2015, where the game is discussed in more detail and a basic evaluation of player experience is given.

Sure Footing

Sure Footing is an infinite runner game currently in development by Table Flip Games Ltd and is programmed by Neall Dewsbury, James Tatum and Tommy Thompson. The game is intended as both a framework for procedural content generation research, as well as a fully fledged commercial product that will allow for broader assessment of prototypical research outputs.

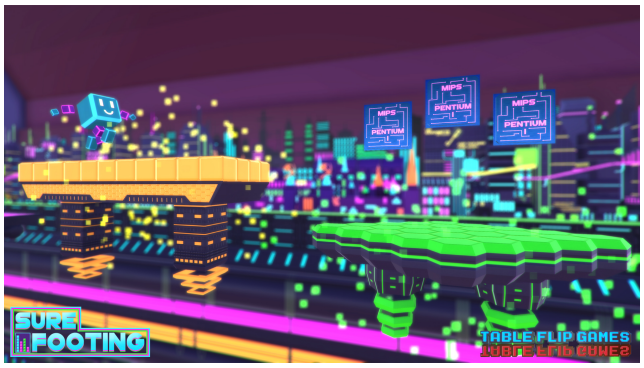

Figure 5: A promotional screenshot from Sure Footing.

Gameplay Overview

Players take the role of one of several characters known as 'pixellites', each with its own unique skills, who are attempting to escape from an evil force chasing them known as 'The Deletion Wave'. Players are tasked with navigating a series of increasing more challenging environments that are procedurally generated at runtime. *Sure Footing* is an 'infinite runner', a subset of the platforming genre of games in which the player has limited to no control over the continued speed and acceleration of his/her character and instead must focus on timing to avoid obstacles hazards. This has genre is largely popularised by titles such as *Canabalt* and *Temple Run*.

In *Sure Footing*, the emphasis is to maintain a steady running pace to avoid the enemy non-player character chasing you. However, this becomes increasingly more difficult as the number of activities players must handle in quick succession continues to increase, with hazards becoming much harder over time.

Current Research

The genre of infinite runner was selected for this project given the natural pairing of procedural content generation research with a type of game that is largely reliant upon it.

The game is broken up into segments known as *sprints*, separated by prefabricated chunks of environment - hereby referred to as *prefabs* - that are designed to look like empty streets. When the player reaches a new street segment, the next batch of platforms is built before it comes within the player's view. The generation process is largely inspired by existing work in the adoption of rhythm for the creation of gameplay segments (Smith et al. 2011). Each sprint is built in two phases: a grammar-driven action generation, followed by the construction of geometry. Each of these phases is reliant upon a *budget*, which allows for designers to constrain the expressiveness of the system at a given point: the former constrains the number of activities the player will expect to complete in a sprint, while the latter dictates the difficulty of said activity given how it is built in the game world.

Action Generator The first phase of level construction is a constraint-driven system that is reliant upon establishing rhythm in gameplay. It adopts a series of specific activities for the player to conduct.

Run: A flat section of terrain which the player must run across.

Jump: A gap between platforms which may carry a small variation in height.

Incline: A series of platforms or a ramp that gradually increases in height.

Decline: A series of platforms or a ramp that gradually decrease in height.

Hopscotch: A series of platforms with one in the middle that is higher than the others, forcing the player to hop atop or over it.

Fall: Two platforms separated by a significant vertical drop. Players are expected to fall or jump down to the lower platform.

Spring: A long platform with a spring attached to the end that will launch the player to a much higher platform.

Constraints are in place that prevent particular activities appearing within close proximity to one another, or with a given frequency. Furthermore, each activity carries a cost, with the action generator responsible for minimising costs of activities to fit within the assigned budget.

Geometry Generator With a complete action sequence prepared for a given sprint, the geometry generator selects from one of multiple prefabs that embody this particular activity. The selected platform types, their length and the potential difficulty they represent, dictate the cost of the prefab. This allows us to decouple the number of activities we wish to place in the sprint from the difficulty of navigating them. Given a particular budget, the geometry generator will aim to utilise it as best it can. In addition, upon placing particular prefabs, the game will add obstacles or collectables onto platforms.

Future Goals

The current system allows for an increasingly more difficult game with relative ease: by increasing the action and/or geometry budget for each sprint, the level segments become gradually more challenging. We are currently experimenting with a number of improvements: including more expressive geometry generation that reflects particular character traits and skills, a more modular generation processes which is not reliant upon prefabricated level segments and options for players to create their own prefabs and levels akin to that previously discussed in the Launchpad project (Smith et al. 2011).

Conclusions

Bringing together academic research and modern game development has always been a challenging aim of the AIIDE conference. The playable experiences track is a growing, exciting new aspect of this process, allowing us to celebrate existing fusions of research and game development, and also to encourage researchers to try developing a broad range of experiences themselves. This year's track shows both familiar ideas and brand new kinds of experience, and we hope to see both sides of the track continue into the future.

References

Cerny, M. 2015. Sarah and sally: Creating a likeable and competent ai sidekick for a videogame. In *Proceedings of the Experimental AI in Games Workshop at AIIDE*.

Churchill, D. 2015. Hierarchical portfolio search: Prismata's robust ai architecture for games with large search space. In *Proceedings of the Artificial Intelligence in Interactive Digital Entertainment Conference*.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. *PDDL - The Planning Domain Definition Language*.

Robertson, J., and Young, R. M. 2014. Gameplay as On-Line Mediation Search. In *Experimental AI in Games*.

Smith, G.; Whitehead, J.; Mateas, M.; and Treanor, M. 2011. Launchpad: A Rhythm-Based Level Generator for 2D Platformers. *IEEE Transactions on Computational Intelligence and AI in Games* 3.