AI-Based Game Design Patterns

Mike Treanor
American University
treanor@american.edu

Alexander Zook
Georgia Institute of Technology
a.zook@gatech.edu

Mirjam P Eladhari
Otter Play
mirjame@gmail.com

Julian Togelius
NYU Polytechnic School of Engineering
julian.togelius@nyu.edu

Gillian Smith
Northeastern University
gi.smith@neu.edu

Michael Cook
Goldsmiths College
mike@gamesbyangelina.com

Tommy Thompson
University of Derby
t2.thompson@gmail.com

Brian Magerko
Georgia Institute of Technology
magerko@gmail.com

John Levine
University of Strathclyde
john.levine@strath.ac.uk

Adam Smith
University of Washington
adam@adamsmith.as

ABSTRACT
This paper proposes a model for designing games around Artificial Intelligence (AI). AI-based games put AI in the foreground of the player experience rather than in a supporting role as is often the case in many commercial games. We analyze the use of AI in a number of existing games and identify design patterns for AI in games. We propose a generative ideation technique to combine a design pattern with an AI technique or capacity to make new AI-based games. Finally, we demonstrate this technique through two examples of AI-based game prototypes created using these patterns.

Categories and Subject Descriptors
I.2.1 [**Artificial Intelligence**] Applications and Expert Systems – Games. K.8.0 [**Personal Computing**] General – Games.

General Terms
Design.

Keywords
Game design, Artificial Intelligence, Machine Learning

1. INTRODUCTION
Almost every game features some kind of Artificial intelligence (AI). The most common role for AI in a game is controlling the non-player characters (NPCs), usually adversaries to the player character. Yet this opposing AI is often rudimentary because the design of the game does not need more complex AI. The perception of AI as controlling adversaries in turn results in games designed to not need richer and more varied AI. However, there are more roles AI can play. Using AI for controlling an adversarial NPC is one of many design patterns for how AI can be used in games. This paper proposes a model and ideation technique for designing games around AI, or AI-based games. A primary goal of this work is to aid discovering new types and potential genres of games.

This paper focuses on AI that is foregrounded in the game, as opposed to AI that operates in the background. We define *foreground AI* as agents the player notices and can reason about. For example, AI that controls a NPC the player either interacts with or observes for sufficient time to learn its behavior is considered foregrounded. Meanwhile, AI that supports gameplay such that the specifics of its behavior is not relevant to the player is considered *background AI*. An example of background AI is the NPC car behavior in *Grand Theft Auto V* that enables the player to quickly speed down the road without crashing too often. Other examples include the "fairer" random number generator in *Civilization IV* that skews probabilities in the player's favor, or the NPC pathfinding systems in first person shooters. While such background AI systems are important to gameplay and smooth the player experience, their operation is not intended to be evident to the player. This paper strives to advance the idea that putting AI in the foreground can enable new types of gameplay experiences.

Accepting a broad definition of AI, games based on simulations of physics can be considered AI-based games. For example, in *Super Mario Bros*, the player must reason about how the system is going to place the character in 2D space based on their input. While the player may not know all of the specifics of how the game simulates 2D physics, they build an approximate model and are able to apply this model to predict how their input will affect the game state while pursuing intentional acts. The physics simulation is central to the gameplay experience.

Super Mario Bros makes use of what have been called "graphical logics" because the player's understanding of its physics simulation is achieved via visually represented entities moving and interacting on a screen [14]. This can be considered foreground AI, as the player's understanding of the system is central to how they make choices. This idea of putting a visualization of the physics simulation central to a game suggests that visualizations of other AI systems might make for interesting games. For example, what types of games could be made when a social simulation, or a learning algorithm are visualized and made central to gameplay? We present several such design patterns for how foreground AI can be used to make new types of games.

As part of this effort, we analyze how AI is used in several existing games and identify design patterns for AI in games. We propose a generative ideation technique to combine a design pattern with an AI technique or capacity to make new AI-based games. Finally, we demonstrate this technique through two examples of AI-based game prototypes created using these patterns.

2. RELATED WORK

Our argument for the value of foregrounding AI in game and our design pattern-based taxonomy builds upon prior research in design patterns, theoretical frameworks for understanding games, and analysis of existing AI-based games.

A design pattern approach to describing games and game content allows us to build a common vocabulary for discussing games, identify common elements between games at the mechanical and player levels, and reason about the structure of games [2, 9, 17, 19]. Design patterns are typically descriptive and informal, drawn from a close analysis of multiple source games. The patterns themselves typically have a short name, a description of how the pattern is abstracted across games, and several motivating examples to show the capacity of the pattern to describe a variety of scenarios across multiple games. Our pattern taxonomy follows the same model: the games we analyzed to extract our patterns come from diverse developers, including large industry studios, academic research, and independent development.

Though the primary purpose of design patterns is typically to provide an analytical lens, they also have the potential to be used generatively. Hullett and Whitehead's [9] FPS level patterns were evaluated via the deliberate design of levels that incorporate those patterns. Dahlskog and Togelius's [4] pattern-based platformer level generator takes this one step further, formalizing the patterns to the extent that a computer can perform the pattern-based design. We pose that the patterns we have identified can be used generatively during the ideation phase of design, to allow us to consider new kinds of playable experiences.

Our taxonomy also builds on previous work in understanding the role of AI in games and the potential it holds for the future of games. Mateas [13] calls for the creation of "expressive AI": playable experiences with complex underlying AI systems where all interaction is framed by the player needing to read meaning into the AI's actions. Eladhari et al. [5] describe a process for designing games where the AI system is an integral part of the game's design. They distill a common process followed during the design of four games: the *Pataphysic Institute* [6], *Prom Week* [15], *Mismanor* [18], and *Endless Web* [16]. With *Endless Web*, Smith et al. pose that an AI-based game is one where the mechanics, dynamics, and aesthetics of the game [10] are deeply linked to the AI system. More recent games designed around their AI system include Horswill's *MKULTRA* [8] and Cook's *A Rogue Dream* [3].

3. DESIGN PATTERNS

Below we discuss several design patterns for AI-based games. These patterns illustrate ways to develop a game mechanic starting from an AI technique (e.g., AI is Visualized) or starting from an intended experience that requires AI (e.g., AI as Role-model). The design patterns and example games are meant to be a tool for thinking about creating AI-based games, rather than serve as a comprehensive taxonomy of methods. Note also that multiple techniques may apply to a single game: Table 1 provides an overview of these patterns and game examples.

3.1 AI is Visualized

Pattern: Provide a visual representation of the underlying AI state, making gameplay revolve around explicit manipulation of the AI state.

Explanation: Many AI techniques revolve around an estimation of the value of actions or game states. Typically these values are hidden from players to promote the sense that an opposing AI agent possesses an intelligence motivating its actions. Visualizing the state of a system or agent enables gameplay as the system is now exposed as a potential obstacle to player progress.

Example: *Third Eye Crime* [11] is a stealth game that illustrates this pattern by visualizing the guard AI position tracking and estimation system. Gameplay involves avoiding guards or throwing distractions to manipulate the guards' predictions of player location. The direct visualization of AI state allows a designer to build a game around manipulating, understanding, and mentally modeling how the AI state changes.

3.2 AI as Role-model

Pattern: Provide one or more AI agents for the player to behave similarly to.

Explanation: AI techniques to date often demonstrate strongly patterned behavior that players come to predict: e.g., finite state machines (FSMs) follow fixed routines that can often be easily noticed. Rather than attempt to make agent behavior more unpredictable, this pattern leverages the behavioral rigidity of a technique to set a stage for the player to act on. Gameplay in this pattern involves acting to mimic the behaviors of AI agents, leading to an "imitation game" judged by an in-game system or opposing players.

Example: *Spy Party* is a game where one player is a spy at a party populated by FSM agents and the opposing player is a sniper watching the party with a single shot to kill the spy. Gameplay for the spy centers on the player attempting to act similarly to the party agents while discreetly performing tasks in the environment like planting a bug or reading a code from a book. Gameplay for the sniper focuses on discerning the human player from AI agents by looking for behavioral cues that differentiate the two. An imitation game thus forces players to explicitly reason about the processes followed by an AI technique.

3.3 AI as Trainee

Pattern: Have player actions train an AI agent to perform tasks central to gameplay.

Explanation: Machine learning techniques revolve around learning new behaviors using examples. By using player actions as a source of examples an AI agent can learn to perform tasks,

Table 1. An overview of AI-based game design patterns and game examples.

Pattern	What player(s) do	Role of AI (in relation to player)	Example(s)
AI is Visualized	Observe AI state	Gives (strategic) information, showing states	Third Eye Crime
AI as Role-model	Imitate AI	Show agent actionas and behaviors, agents as puzzles	Spy Party
AI as Trainee	Teach AI	Child/student	Black & White
AI is Editable	Edit AI	Artifact/agent that player can author/manipulate	Galactic Arms Race
AI is Guided	Guide/manage the AI	Partly independent inhabitants, with players as their Gods	The Sims
AI as Co-creator	Make artifacts assisted by AI	Co-creator, making artifacts	ViewPoints AI
AI as Adversary	Play game against the opponent	Opponent (symmetric)	Chess, Go
AI as Villain	Combat the Villain(s)	Villain in game; mob, boss mob, NPC (asymmetric)	Alien Isolation
AI as Spectacle	Observe	Spectacle, enacting simulated society	Nowhere

with this indirect control (or automation) becoming central to player activity in a game. Note that many paradigms for training exist. Supervised learning requires players to explicitly provide feedback by labeling examples as indicative of a behavior. Unsupervised learning abstracts from examples without explicit guidance. Reinforcement learning uses feedback about the value of actions, rather than labels describing what an action was. Each of these paradigms provides opportunities for different kinds of player action in a game to indirectly control game outcomes.

Example: *Black & White* [1] is a god game where the player trains a creature to act as an autonomous assistant in spatial regions where the player cannot take direct action. The creature learns sets of behaviors through a reward signal based on a needs model; the creature also takes direct feedback through player action (e.g., slapping or petting the creature after it takes actions). Players cannot directly control the actions of the creature, but instead rely on feedback to train the creature to perform actions that align with the player's desired strategy. By training AI agents players are required to consider behind how an agent learns, even without direct representation of that process.

3.4 AI is Editable

Pattern: Have the player directly change elements of an AI agent that is central to gameplay.

Explanation: Most AI techniques have parameters (e.g., weights in a neural network) or other data structures (e.g., nodes in a behavior tree) that can be directly changed to manipulate agent behavior. Typically these are created and tuned at design time and remain fixed and hidden during the running time of a game. Providing players with direct access to manipulate these parameters can be the foundation for a game about player indirect action via an AI agent. Note that direct player editing can supplement indirect player training (e.g., as an alternate gameplay mode or resource-gated mechanic).

Example: *Galactic Arms Race* [7] is a space shooter where how the player uses different weapons evolves an underlying neural network representation to change weapon firing behavior. Base gameplay revolves around finding a set of firing behaviors that together enable a player to succeed at destroying opposition (another example of the AI as Trainee pattern). One gameplay mode allows the player to explicitly manipulate the network weights on weapons, allowing more precise control over the firing patterns of the evolved weapons. This control enables players to more finely explore the space of parameterizations, leading to an indirect way to understand the processes of the AI system.

3.5 AI is Guided

Pattern: The player assists a simple or brittle AI agent that is threatened with self-destruction.

Explanation: Many AI algorithms are brittle and likely to break unless constrained to highly limited environment. Rather than avoid exposing the AI to situations where its behavior would be detrimental, build gameplay around the player acting to avoid those situations. Gameplay then emphasizes players acting around the AI to protect it or directly acting to continually maintain the AI in the face of gradual degradation.

Example: *The Sims* addressed the problem of "human-like" agents in a social world by making gameplay revolve around the player addressing the needs of simple agents. AI agents have a set of needs and desires they attempt to pursue while players intervene to provide for the needs of the agents through food, shelter, work, socialization, and eventually more grand life aspirations. By having players care for the AI, players come to (at least indirectly) model some of the processes used by the AI.

3.6 AI as Co-creator

Pattern: Involve the player in a creative task where an AI agent directly contributes to the task as an equal partner.

Explanation: In games based around performance or creation of an artifact an AI system can directly contribute to that creation. Rather than leave all content creation in the hands of players, an AI system can participate in content creation, creating gameplay around the shared construction. Enabling AI participation in the process affords gameplay around the ongoing negotiation of meaning and goals for an artifact or performance, rather than traditional game structures based on overcoming obstacles.

Example: *Viewpoints AI* [12] has a human and AI performer create a shared movement experience through improvised, turn-based interactions. Players are projected into a 2D plane shared with an AI agent where the pair act and react to one another's movements. Having players share authoring an AI agent guides players to consider the processes being used by a system.

3.7 AI as Adversary

Pattern: Require players to overcome an (embodied or not) AI opponent in a contest.

Explanation: This pattern is arguably one of the oldest uses of AI in games: providing players with opponents when none (or few) may be found. Many games depend on multiplayer competition where it is impossible to play without a computer opponent. Gameplay against AI adversaries revolves around understanding the strategies and tactics the AI executes to succeed at the competition.

Example: Computer *Chess* is a classic example of this pattern, where AI agents enable people to play the game at any time and against an opponent with adjustable capabilities ("difficulty"). Within global chess tournaments AI agents have had sweeping influence: players study the strategies of AI systems and use these to achieve high-level play. This influence has reached the point that high-level chess success requires players to innovate in ways that take advantage of the reasoning processes used by these AI systems, evidencing the deep understanding of AI processes the AI as Adversary pattern can induce.

3.8 AI as Villain

Pattern: Require players to complete a task or overcome an AI opponent where the AI is aiming to create an experience (e.g., tension or excitement) rather than defeat the player.

Explanation: In games developed around players overcoming opposition the AI agents can be "pulling punches" to intentionally create a desired experience for the player. Rather than the AI being a *character* in the game world, it is an *actor* attempting to create an experience for the player while maintaining a facade of being a character. For the player, gameplay still revolves around defeating the opponent, yet for the opponent, gameplay revolves around shaping player behavior in a desired way.

Example: *Alien: Isolation* is a first-person survival horror game where the opposing alien was designed to harass the player without using an optimal strategy that would always kill the player directly. The enemy alien spends the game hunting the player, displaying behaviors of seeking the player's location (a weak version of AI is Visualized), and gradually learning from tactics the player uses repeatedly (an oppositional application of AI as Trainee). By having players continually reason on what the alien has learned and where it will go the player is forced to consider the state of the AI and (after repeated play) the processes involved in the AI learning.

3.9 AI as Spectacle

Pattern: Have an AI or group of AI agents implement a complex system, such as a social hierarchy, that the player may observe or interfere with.

Explanation: AI agent architectures are often capable of acting and interacting fully autonomously. This pattern leverages AI autonomy to create a game experience around watching the unfolding of an AI society and potentially intervening in controlled ways to observe the outcomes. For the player, gameplay revolves around watching events unfold and formulating ideas about how the agent society functions.

Example: *Nowhere* is a 'psychedelic RPG' that aims to implement a complex society of AI agents that the player can influence at any point in its history, and at any level in its hierarchy. The game is designed to be confusing and alien to the player – communication with the AI agents is through an 'alien vocabulary' of 27 words, so the player is forced into unfamiliar experiences. The scale and complexity of the society the player is faced with is part of the game's design aesthetic.

4. USING PATTERNS

4.1 Generative Ideation Technique

AI is used to achieve ends: automated classification of data, object recognition, planning, and language generation are just a few AI capacities. We have found AI-based games can be described as a combination of one or more patterns and one or more AI capacities. For example, *Spy Party* combines the 'AI as Role-model' pattern with AI's capacity to manage character behavior with finite state machines. The social simulation game *Prom Week* combines the AI is Visualized and the AI is Editable patterns with the AI capacities to select actions and generate language.

We believe the space of AI-based games is vast and underexplored: the patterns above can be used generatively to explore possibilities for other AI-based games. Combining each pattern with an AI capacity yields many potential games. For instance, starting with the AI as Trainee pattern and the AI capacity for facial expression recognition, we can imagine a game where the player sits in front of a camera with make up, and tries the trick the game into thinking they are someone they are not (e.g., "Try to be an old angry man!"). This is just one of many potential games that could come from this combination.

Below are two concrete examples of game prototypes we developed with this approach in mind.

4.2 Examples

The design patterns we presented are largely descriptive of how AI-based games have been created. To understand how these patterns might be used generatively we developed two AI-based games in short hackathon sessions. These games were used to explore different ways of instantiating and combining some of the patterns from the table and explore the challenges of making game mechanics, theme, and content choices when developing AI-based games.

4.2.1 Contrabot

In *Contrabot*[1] players attempt to send crates of contraband material past an inspector to a partner (see Figure 1). The player's goal is to get as many crates collected by their partner as possible, while minimizing the crates intercepted by the inspector. To

[1] code available at: https://github.com/gamesbyangelina/contrabot

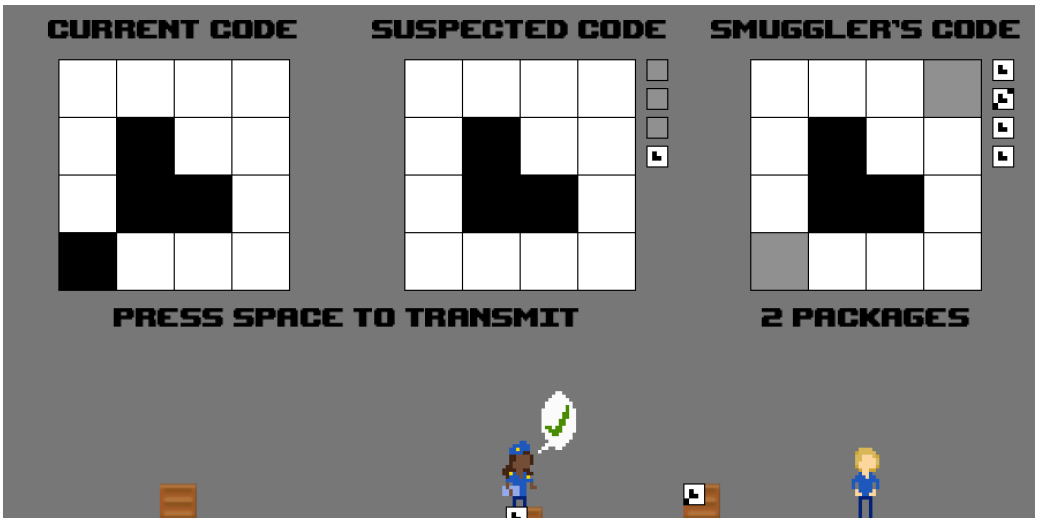

Figure 1. The *Contrabot* game interface. The leftmost tiles indicate the code the player will send, the middle tiles are the learned code for the inspector, and the rightmost tiles are the learned code for the partner. The smaller boxes adjacent to the larger tiles indicate the 4 most recent codes in the memory of the inspector and partner, respectively.

achieve this goal the player stamps crates with a code indicating to their partner that the crate should be opened. The crates, and their associated codes, are viewed first by the inspector and then, if the inspector does not confiscate it, the player's partner. The core AI in the game involves how agents learn the codes sent by the player, creating gameplay around managing what the inspector learns to check as opposed to what the partner learns to check.

The game mechanics revolve around how agents learn to check codes based on codes they have seen. Agents have two main processes: learning codes and matching new codes against their learned code. To do so agents posses a memory of previously viewed codes and a learned code that generalizes over those remembered codes.

Learning proceeds as follows. When the inspector or partner agent looks at a crate they add the code to their memory. The agent then uses a simplified form of least general generalization to produce a code that can match any of the crates. Codes take the form of a grid of tiles all set to a binary value of white or black (0 or 1, respectively). Learning considers each tile location: if all codes from memory have the tile white (or black) the learned code considers that position white (black). If the tile has been both white and black in the agent's memory the tile is assigned grey, indicating a wildcard. When new codes pass the agent they attempt to match the new code against their learned code. Matching checks whether every tile matches the color of the learned code as white or black, with grey allowing either white or black (i.e., generalization to match any option). If a code matches, the agent adds the code to their memory.

We designed *Contrabot* gameplay around risk-reward considerations for the player. To do so we started both agents with an empty learned code and empty memory. The partner learns from the first crate inspected and only retains a memory of the 4 most recent codes inspected. The inspector instead has a random chance to inspect creates – until the inspector randomly chooses to check a crate, the inspector will not match any crates. However, the inspector has an infinite memory, meaning the inspector will eventually learn a code that matches any new code (this ends the game). After both agents have learned some code gameplay revolves around creating codes that the inspector will not check (modulo random checks), but that the partner will check. The number of crates the player passes to the partner serves as a score as the game will eventually end once the inspector learns to match all crates.

Contrabot implements three AI-based game patterns. AI is Visualized is implemented through showing the codes learned by both the inspector and partner. This supports player reasoning about what codes the agents have learned to consider new codes to create. AI as Adversary is implemented through the inspector agent checking crates and learning to match their codes. Players must reason about how the inspector (and partner) learns codes, how the inspector (and partner) match codes, and recognize the random inspection chance for the inspector. AI is Guided is implemented through the partner, where the player must manage the limited partner memory and simple learning mechanism. By providing a limited partner memory buffer we force players to recognize the simplicity of the learning technique, particularly under conditions with limited codes to learn from. This game is highly extensible and customizable: with a variety of gameplay difficulties and options that can be achieved by implementing target numbers of received packages, limiting the number of packages sent by the player and manipulating the size of both the inspector and smugglers memories.

We chose these patterns as they highlight how a simple learning technique, when visualized, can readily become the core of gameplay. Once this is established there are many other patterns to use to create different gameplay loops – in this case we took a metaphor of code transmission and interception and used it to choose patterns for both adversary (inspector) and teammate (partner). Many other combinations are possible – below we discuss an alternative approach where we explored visualization of a different learning mechanism, but provided players with the ability to both directly and indirectly manipulate the learned information.

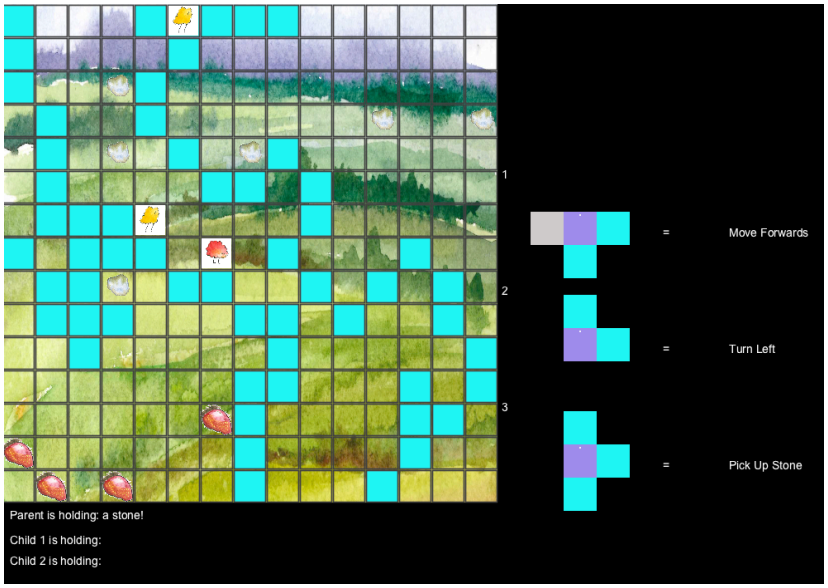

Figure 2. The *What are you doing?* game interface.

4.2.2 What are you doing?

In *What are you doing?*[2] the player plays as a parent shrub in charge of protecting and guiding one or several child shrubs. The challenge the player faces is that the children will not do as they are told, but instead learn from and mimic the player. This can be positively lethal for them, however the player possesses a limited power to edit their minds to unlearn some of the bad habits they've learned.

The game is turn-based and plays out on a grid as shown in Figure 2. There are five types of entities in the world: the player character (parent shrub), childshrubs, stones, strawberries, and ponds. The parent shrub can move in any of the four cardinal directions, eat or pick up whatever is in the direction it is facing, or drop something it is carrying. The parent has an energy level which can be recharged by eating. Child shrubs have mostly the same actions available, but they have somewhat different effects: in particular, if the child picks up stone it might be crushed under its weight, and it will drown if moving into a pool. The energy levels of the child shrubs are also constantly decreasing. Both stones and ponds block the way, but stones can be picked up and moved elsewhere. If a stone is placed into a pond both the pond and stone disappears, so the tile becomes empty (passable). Strawberries can be eaten by parents or kids and will increase energy. As children will rarely seek out strawberries, the parent can pick up strawberries and drop them on child shrubs so they eat.

In a pane to the right of the main game area the shared mind of the children is visualized. Their mind is initially empty, but as the children watch the parent act, they learn rules. Rules are of the form "when faced with certain surroundings, perform a certain action". For example, they could learn a rule that when standing with a stone to the left and a pond behind, move forward. Or if in front of a strawberry, pick up. Rules are re-learned every ten turns, and there is no limit to how many rules can be learned – this depends only on how many regularities the rule learning algorithm can find within the recent history of what the parent has done. The rule learning algorithm is a simple brute force search for all rules that can be constructed from recent activity; if using longer action histories or more actions, this would have to be replaced with an *a priori* implementation.

The player has the option at any turn to remove one or more rules from the mind of the children. This, however, costs energy for the parent, so this possibility must be used sparingly. In order to replace energy, strawberries can be eaten – but then the same strawberries cannot be used to feed your children. The reason for removing rules from the children's mind is that they would otherwise hurt themselves, for example drowning in a pond or picking up a stone and crushing themselves.

Tension is created in the game by scarce resources (energy and strawberries), but also by the nature of the learning algorithm. There could be many good courses of action which should be avoided because your children are watching and could learn the wrong things. For example, one might want to approach a stone from the opposite direction than what the shortest path would suggest, to avoid that the kids see regularities. However, this takes more time and allows the children to learn more bad behavior in the meantime. It is also taxing the player's memory to try to remember what situations the parent has been in recently. However it is done, there is a clear advantage to reasoning about the AI, which is reinforced by the visualization of the kids' mind and the possibility of editing it.

The child shrubs in *What Are You Doing?* exhibit both the AI as Trainee pattern and the AI is Editable pattern. In the former case, player actions are observed by the shrubs, and so the player knows that by solving problems in the game world they are also giving new knowledge to the AI agents. In the latter case, pruning knowledge from the shrubs is a restricted form of editing, in that knowledge can be explicitly removed from the agents (although new knowledge cannot be explicitly added, only implicitly through behaving in a certain way). Because the learned knowledge is displayed to the player through a visualization of the

[2] code available at:
https://github.com/gamesbyangelina/whatareyoudoing

rules, the game also exhibits the AI is Visualized pattern, through which the player can understand what knowledge has been picked up by the shrubs, and then edit it appropriately to remove dangerous behavior.

The game raises interesting problems with presenting machine learned knowledge to the player. Our method of machine learning learned partial rules that may not be thought of as 'knowledge' to a human player unfamiliar with machine learning (such as what to do when a rock is to your left, but not what to do in the general case of being adjacent to rocks). This also means that editing knowledge can be exhausting, since there may be multiple cases that are all undesirable, which need to be manually removed when to the player they may seem to all stem from the same original case. The presentation of machine learned knowledge both in *Contrabot* and *What Are You Doing?* presents interesting possibilities of how to use these technique in future game projects, and indicates the richness of new ideas to be found in applying these patterns to new aspects of gameplay.

5. CONCLUSION

A potential critique of most of the design patterns we present is that human players can replace the AI to produce a comparable or improved experience. Why not have people act as adversaries or be the targets for imitation? We are not claiming that AI agents can pass a game-based version of the Turing test and thereby provide new or improved play experiences. Rather, we believe that a serious consideration of the strengths and limitations of various AI techniques can be the foundation for new kinds of games. By using AI as the core of gameplay experiences we can leverage how people reason about other agents (e.g., adversaries, "people" to imitate, or creatures to raise) and create gameplay based around thinking about how agents work. Just as much as human puppeteers could play the roles of AI agents (consider Jason Rohrer's *Sleep Is Death* that replaces a disembodied AI as Villain with a human), humans could also manually fill in the physics used in many 2D platformer games. By creating automated processes, gameplay can be based on the reliability of these underlying systems derived from their algorithmic structure.

Designing games that use AI techniques in a new way as a core of their gameplay diversifies and enriches the role of artificial intelligence in games. This not only improves the breadth of the medium with new games and genres, but also opens up new research questions, as players begin to interact with software in novel ways. Developing AI-based games also pushes us to tackle existing research problems from a new, practical perspective. Building AI agents capable of taking over from absent players in online games or developing agents capable of assisting and enhancing the creativity of other players are research problems that are very relevant to the modern games industry. By building games in which these problems are approached as a question of game design, we can evaluate solutions directly, in contexts where the problem is the very focus of the player activity, rather than being one element in a much larger game. This might prove to be a new and effective lens through which to examine other problems in game AI research.

6. ACKNOWLEDGMENTS

The ideas in this paper were formed in a working group at the Dagstuhl seminar 15051: Artificial and Computational Intelligence In Games – Integration.

7. REFERENCES

[1] Barnes, J. 2002. Testing Undefined Behavior as a Result of Learning. *AI Game Programming Wisdom*.

[2] Bjork, S. and Holopainen, J. 2004. *Patterns in Game Design (Game Development Series)*. Charles River Media.

[3] Cook, M. and Colton, S. 2014. A Rogue Dream: Automatically Generating Meaningful Content For Games. (2014).

[4] Dahlskog, S. and Togelius, J. 2012. Patterns and Procedural Content Generation. *Proceedings of the Workshop on Design Patterns in Games ({{DPG}} 2012), co-located with the Foundations of Digital Games 2012 conference* (Raleigh, NC, May 2012).

[5] Eladhari, M.P. et al. 2011. *{{AI}-Based} Game Design: Enabling New Playable Experiences*.

[6] Eladhari, M.P. 2014. The Mind Module: Using an Affect and Personality Computational Model as a {Game-Play} Element. *Affective Computing, IEEE Transactions on*. 5, 1 (Jan. 2014), 3–16.

[7] Hastings, E.J. and Stanley, K.O. 2010. Galactic Arms Race: An Experiment in Evolving Video Game Content. *SIGEVOlution*. 4, 4 (Mar. 2010), 2–10.

[8] Horswill, I. 2014. Game Design for Classical {AI}. *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference* (2014).

[9] Hullett, K. and Whitehead, J. 2010. Design Patterns in {FPS} Levels. *Proceedings of the 2010 International Conference on the Foundations of Digital Games ({{FDG}} 2010)* (Monterey, CA, Jun. 2010).

[10] Hunicke, R. et al. 2004. MDA : A Formal Approach to Game Design and Game Research. *Challenges in Game AI Workshop Nineteenth National Conference on Artificial Intelligence* (2004).

[11] Isla, D. 2013. Third Eye Crime: Building a Stealth Game Around Occupancy Maps. *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference* (2013).

[12] Magerko, B, Permar, J, Jacob, M, Comerford, M, Smith, J. 2014. An Overview of Computational Co-creative Pretend Play with a Human. *Proceedings of the Playful Characters workshop at the Fourteenth Annual Conference on Intelligent Virtual Agents* (2014).

[13] Mateas, M. 2007. Expressive Intelligence: Artificial Intelligence, Games and New Media. *{AI*IA} 2007: Artificial Intelligence and {Human-Oriented} Computing*. Springer Berlin Heidelberg. 2.

[14] Mateas, M. and Wardrip-Fruin, N. 2009. Defining Operational Logics. *Digital Games Research Association (DiGRA)* (2009).

[15] McCoy, J. et al. 2011. Prom Week: Social Physics as Gameplay. *The Foundations of Digital Games Conference (Poster)* (2011).

[16] Smith, G. et al. 2012. PCG-Based Game Design: Creating Endless Web. *Foundations of Digital Games Conference (FDG 2012)* (2012).

[17] Smith, G. et al. 2011. Situating Quests: Design Patterns for Quest and Level Design in {Role-Playing} Games. *Proceedings of the 2011 International Conference on Interactive Digital Storytelling* (Vancouver, BC, Canada, Dec. 2011).

[18] Sullivan, A. et al. 2012. The Design of Mismanor: Creating a Playable Quest-based Story Game. *Proceedings of the International Conference on the Foundations of Digital Games* (New York, NY, USA, 2012), 180–187.

[19] Zagal, J.P. and Bruckman, A. 2008. The game ontology project: Supporting learning while contributing authentically to game studies. *Proceedings of the 8th international conference on International conference for the learning sciences-Volume 2* (2008), 499–506.