AI-based Games: Contrabot and What Did You Do?

Michael Cook
Comp. Creativity Group
Goldsmiths, Uni. London
London, UK
mike@gamesbyangelina.org

Mirjam P. Eladhari
Otter Play
Sliema, Malta
info@otter-play.com

Adam M. Smith
Department of Computational
Media
University of California, Santa
Cruz
Santa Cruz, CA
adam@adamsmith.as

Gillian Smith
Northeastern University
Playable Innovative
Technologies Lab
Boston, MA, 02115
gillian@ccs.neu.edu

Tommy Thompson
Department of Computing and
Mathematics
University of Derby
Derby, UK
tommy@t2thompson.com

Julian Togelius
Department of Computer
Science and Engineering
New York University
New York, NY 11201
julian@togelius.com

ABSTRACT

AI-based games foreground interaction with an artificial intelligence system as the core of gameplay. We present AI-based games—Contrabot and What Did You Do?—that use machine learning for novel play experiences.

1. INTRODUCTION

Game designs typically use artificial intelligence (AI) as a tool to create a desired experience. AI-based games have recently emerged as an alternative paradigm that foregrounds interaction with an AI system or agent as the core component of a game [1, 2]. Example AI-based games include *Black & White* (about teaching an AI avatar), *Spy Party* (about mimicking AI routines), and *The Sims* (about guiding simple AI 'pets'). This abstract discusses two prototype AI-based games—Contrabot and What Did You Do?—that explore interacting with machine learning algorithms as a core gameplay loop.

2. CONTRABOT

Contrabot[1] (Figure 1) is a game based on agents that learn abstract codes with gameplay based on understanding, playing against, and deceiving a machine learning system. Players act as a smuggler trying to label boxes to communicate with a contact on the other side of a customs checkpoint. The smuggler is trying to learn the code players use to indicate a box is contraband—but an inspector is randomly checking the same boxes.

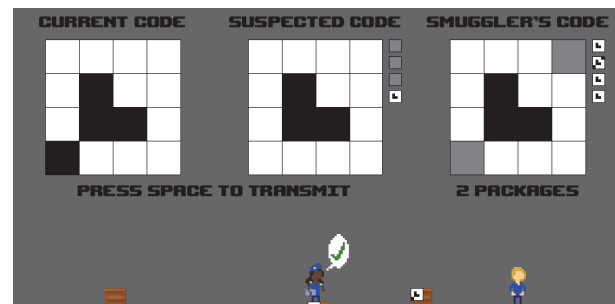[1] https://github.com/gamesbyangelina/contrabot

Figure 1: Contrabot interface: player input code on the left; inspector learned code in the middle; smuggler learned code on the right. Agent memories are shown as the set of smaller codes along the right of the larger central codes.

The game mechanics revolve around how the smuggler and inspector agents learn to check codes based on codes they have seen. These agents have two main processes: learning codes and matching new codes against their learned code. Both agents generalize patterns from example codes in their memory—using a form of least general generalization—to then try to match new codes to these learned patterns. The inspector has a larger memory than the smuggler and gameplay is based on reverse-engineering how learning works to take advantage of the smuggler forgetting old patterns more quickly than the inspector. The generalization process is simple: agents compare all codes seen, memorize exact matches for black or white tiles, and generalize to gray tiles if a position has been occupied by both colors. Despite this simplicity, the design allows many difficulty levels and risk-reward considerations for players based on the size of the codes used and agent memory capacities.

We designed Contrabot gameplay around risk-reward considerations for the player. Both agents start with an empty learned code and empty memory. The smuggler learns from the first crate inspected and only retains a memory of the four most recent codes inspected. The inspector initially only has a random chance to inspect creates; after the inspector chooses to check a crate it begins learning and can either randomly check crates or check crates due
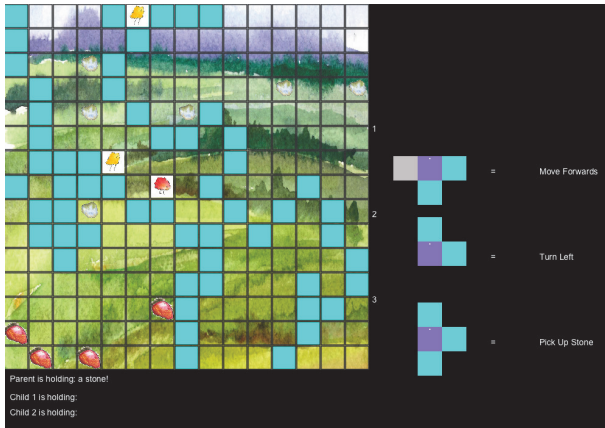
Figure 2: WHAT DID YOU DO? interface: game world on the grid; rules learned by the child on right pane.

to matching. The inspector has an infinite memory, meaning the inspector will eventually learn a code that matches any new code (this ends the game). As both agents learn codes, gameplay centers on creating new codes that the inspector will not check (modulo random checks), but that the smuggler will check. The number of crates the player passes to the smuggler serves as a score as the game will eventually end once the inspector learns to match all crates. While simple, this design highlights how understanding the ways a machine learning algorithm works can be the core part of playing and succeeding at a game: in this case mastering the code learning algorithm to both avoid failure (inspector checks) and achieve success (smuggler checks).

3. WHAT DID YOU DO?

WHAT DID YOU DO?[2] (Figure 2) is a game based on indirectly training and directly editing a child AI agent to complete in-game tasks without the child dying along the way. Players control a parent shrub attempting to raise an AI-controlled child shrub in a simple environment. The challenge derives from the child not being directly controlled by the player: the child learns behavioral rules to mimic the parent (player). The game is turn-based and takes place in a grid world populated with helpful items (e.g., food) and obstacles (e.g., impassable rivers and stones).

Players aim to navigate obstacles in the environment to reach a destination with their child still alive. Navigation includes fording impassable rivers by picking up stones and dropping them into the river. Parents may pick up objects or eat food to restore their energy level—depleting either the parent or child energy fully ends the game. The environment is also more hazardous to the child than the parent: children may be crushed by dropping a stone on themselves after attempting to carry it too long or may drown in rivers.

The child observes the parent and learns rules of the form "when surrounded with certain objects, perform a given action." For example, the child 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 it 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

---

[2] https://github.com/gamesbyangelina/whatareyoudoing

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 could be replaced with the a priori rule learning algorithm.

We designed the rule learning system to play off food scarcity in the environment and the available hazards to the child. On any turn the player may remove rules from the child's mind by spending energy from the parent. Replacing energy requires food, meaning any adjustments made to keep the child from inadvertently harming itself come at the cost of limited resources to stay alive. Thus, promising courses of action can introduce trade-offs because the child is watching and may 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 child sees regularities. However, this takes more time and allows the child to potentially learn other bad behavior in the meantime. These trade-offs induce players to explicitly reason about how the child learns and how to manage in-game resources around that learning.

4. DISCUSSION

These games raise interesting problems with presenting machine learned knowledge to the player. In CONTRABOT learned codes are color-coded and the history of examples can be explicitly shown to the player for the smuggler (but only partially shown for the inspector). Players can readily learn the visual language of the game, but not understand the gameplay implications of code choices. At the same time the code language is very simple and can quickly become boring. WHAT DID YOU DO?'s learned partial rules 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 means that editing knowledge can be exhausting, as there may be multiple cases that are all undesirable (seeming to stem from the same original case to the player), which need to be manually removed. The presentation of machine learned knowledge both in CONTRABOT and WHAT ARE YOU DOING? illustrates possibilities for using these techniques in future game projects, and indicates the richness of new ideas for AI-based games.

5. ADDITIONAL AUTHORS

Alex Zook (School of Interactive Computing, Georgia Institute of Technology), a.zook@gatech.edu.

6. REFERENCES

[1] M. P. Eladhari, A. Sullivan, G. Smith, and J. McCoy. AI-based game design: Enabling new playable experiences. Technical report, University of California, Santa Cruz, 2011.
[2] M. Treanor, A. Zook, M. P. Eladhari, J. Togelius, G. Smith, M. Cook, T. Thompson, B. Magerko, J. Levine, and A. Smith. AI-based game design patterns. In *10th International Conference on the Foundations of Digital Games*, 2015.