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

anonymous

ABSTRACT

AI-based games are good. We made CONTRABOT and WHAT DID YOU DO?.

Categories and Subject Descriptors

Applied Computing [**Computers in other domains**]: Personal computers and PC applications—*Computer games*

General Terms

Design

Keywords

Game AI, design patterns

1. INTRODUCTION

AI-based games, woo!

Something to the effect that we need more examples of AI-based games to explore the space.

2. CONTRABOT

[[**copy-pasta from Dagstuhl report**]]

Contrabot (Figure **??**) is a game prototype for an AI-based game, developed at the 2015 Dagstuhl seminar on Computation and Artificial Intelligence in Games. AI-based games foreground interaction with an AI system in the player's experience. Our goal with Contrabot was to build a game based on agents that learn, crafting gameplay around understanding, playing against and ultimately deceiving a machine learning system.

You play as a smuggler trying to secretly label boxes to communicate with a contact on the other side of a customs checkpoint. The smuggler is trying to learn the code you use to indicate a box is contraband—but an inspector is randomly checking boxes too. Can you design and redesign your secret codes to stop the inspector learning your patterns? Will you still manage to sneak your code through to your contact and smuggle your goods out of the country?

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—using a form of least general generalization—in their memory 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 using reverse-engineering how learning works to take advantage of the smuggler forgetting old patterns more quickly than the inspector. The generalization process is simple, when comparing all codes seen the agents memorize exact matches for black or white tiles and generalizing to gray tiles if a position has been occupied by both colors. Despite this simplicity the design accommodates many levels of difficulty and risk-reward considerations for players based on the size of the codes used and memory capacities of each agent.

The game prototype is available to play and fork the design at `https://github.com/gamesbyangelina/contrabot`.

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.

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 mech-

anism, but provided players with the ability to both directly and indirectly manipulate the learned information.

3. WHAT DID YOU DO?

[[copy-pasta from FDG paper]]

Game based on training and editing a companion AI agent to complete in-game tasks without the companion failing. Players control an avatar attempting to raise a computer-controlled child agent in a game environment. The environment contains a variety of items to manipulate (e.g., stones or foodstuffs) that may be helpful or harmful (e.g., moving stones to stand upon or consuming foodstuffs that may be helpful or poisonous). Players aim to guide their child through the environment by acting in the world to overcome various obstacles and complete in-world puzzles. At the same time the child agent applies machine learning techniques to learn to mimic the parent behaviors. Child learning offers benefits and challenges as the child may help the parent or inadvertently harm itself. During the game players have access to the memory of the child and may (at a cost) edit what the child has learned.

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.

https://github.com/gamesbyangelina/whatareyoudoing

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 childrenâĂŹs 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 rules, the game also exhibits the AI as Visualizer 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 âĂŸknowledgeâĂŹ 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.

[1]

4. REFERENCES

[1] G. Smith, A. Othenin-Girard, J. Whitehead, and N. Wardrip-Fruin. PCG-based game design: creating Endless Web. In *7th International Conference on the Foundations of Digital Games*, pages 188–195. ACM, 2012.