The AIIDE-17 Workshop on
Experimental AI in Games
WS-17-19

Towards Positively Surprising Non-Player Characters in Video Games

Vadim Bulitko
Computing Science
University of Alberta
Edmonton, AB
bulitko@ualberta.ca

Shelby Carleton
English and Film Studies
University of Alberta
Edmonton, AB
scarleto@ualberta.ca

Delia Cormier
Art and Design
University of Alberta
Edmonton, AB
dacormie@ualberta.ca

Devon Sigurdson
Computing Science
University of Alberta
Edmonton, AB
dbsigurd@ualberta.ca

John Simpson
Compute Canada
University of Alberta
Edmonton, AB
jes6@ualberta.ca

Abstract

Video games often populate their in-game world with numerous ambient non-playable characters. Manually crafting interesting behaviors for such characters can be prohibitively expensive. As scripted AI gets re-used across multiple characters, they can appear overly similar, shallow and generally uninteresting for the player to interact with. In this paper we propose to evolve interesting behaviors in a simulated evolutionary environment. Since only some evolution runs may give rise to such behaviors, we propose to train deep neural networks to detect such behaviors. The paper presents work in progress in this direction.

1 Introduction

Can ambient non-player characters in video games surprise players in positive ways without being explicitly programed? The prevailing approach to the development of video games suggests that doing so may not be a desirable risk to take. We believe that the answer in the future will be "Yes, and the benefits outweigh the risks" but that this is still a way off because techniques still need to be developed and prevailing opinions changed. However, a significant step towards this goal can be taken now by focusing attention on a subclass of non-player characters (NPCs) in video games that do not typically require the highly sophisticated behaviors related to language and culture. The NPCs that we have in mind are ambient NPCs (aNPCs) which are characters that exist predominantly in the background of games without any direct tie to a quest or mission. Most animals and similar non-language using characters would be of this type.

When it comes to developing aNPCs game designers aim to enhance the atmosphere of the game without distracting the player from the main storyline or side quests. While aNPCs predominantly wander around to produce atmosphere through their simply being there or providing small interactive opportunities they can also play larger roles within the game such as acting as sources of equipment or fighting for or against players. Regardless, the standard for a good aNPC in a video game remains its ability to perform exactly as programmed, going where told, attacking the correct enemies, and responding with scripted dialogue to player choices, all without disrupting the immersive experience. aNPCs that act outside of these boundaries risk earning the ire of the gaming community by doing things such as glitching through objects, getting caught in corners, repeatedly needing rescue, or simply not advancing the plot as intended. The number of YouTube rants and blogs offering top 10/50/100 "Worst NPCs in Video Games" lists is a testament to the widespread nature of these phenomena (WatchMojo 2014).

Perhaps even more unfortunate than outright errors are aNPCs meant to be interesting/surprising but that ultimately leave the player disappointed because they may be boring. Consider the PC Gamer review of *No Man's Sky*, which offers as a verdict, "64/100. Relaxing exploration and some lovely scenery coupled with repetitive systems, frustrating menus, and a lack of real discovery" (Livingston 2016). The full review makes it clear that lack of discovery is the insurmountable problem with the game as it failed to produce anything "fascinating" and was instead filled with uninteresting predictable behavior and little meaningful player interaction during exploration. For a game that is first and foremost about discovery, about surprising players with creatures that no one else has ever seen—the designers made extensive use of procedural generation in an attempt to offer players new experiences throughout gameplay—such criticism carries a heavy weight.

Rather than accept that the solution to producing aNPCs that are neither boring or broken is expertly produced behavior trees of ever increasing complexity, we suggest that an alternative methodology be explored: evolved artificial life. By producing simple models of the aNPCs to make up the game world and allowing them to reproduce and die with evolutionary mechanisms in place, behaviors that actually worked for the specified environment can be produced, reducing the likelihood of producing broken NPCs.

Further, with the appropriate agent complexity this approach can possibly generate surprising behaviors that a human designer might never have considered. As there is no guarantee that an interesting behavior would emerge on every run of such a simulated evolution, many evolutions may have to be run, making it intractable for a human game developer to detect interesting behavior manually. Thus, we propose building an automated detector of interesting behavior. Such a program would sift through numerous evolution runs, flagging possibly interesting evolved behavior for game developers to examine and subsequently include into a game.

This paper is organized as follows. Section 2 formally introduces what makes behavior positively surprising in the context of a video game, and discusses how to detect such behavior. We then discuss related work in Section 3. Section 4 details our approach to the problem, where we set up an evolutionary model to exhibit interesting behaviors, and train a deep neural network to visualize frames and recognize differences as a first step toward anomaly detection. Section 5 details our current challenges and future work, where we discuss the limitations of our research and the possible steps we can take to progress our anomaly detector, finally followed by Section 6, our conclusion.

2 Problem Formulation

What counts as a surprising aNPC behavior within a game will vary between designers, players, critics, and observers, as well as across games and time. Acknowledging this variability we are seeking to constrain possible interpretations of what counts as *surprising* behaviors by focusing on assessments made by players of aNPC behaviors that are *positive* and sustainable through repeated and varied game play, for the following reasons.

First, players are the group to focus on because they are the ones who most directly experience the game and it is their reaction that is the ultimate test of a games general success. Our intent is to produce behaviors that are at least unexpected (i.e., surprising). This could be due to either a single behavior or how sets of behaviors interact within the game environment. Second, by *positive* aNPC behaviors we intend to capture those behaviors that enhance game play rather than detract from it. Such detraction can happen by either removing the player from the immersive experience (e.g., an aNPC who violates the accepted physics of the world by clipping through walls) or by maintaining immersion but creating undue frustration (e.g., an aNPC who is essential to a quest but too difficult to interact with to allow the quest to be completed). Finally, such positively surprising behaviors need to be robust to repeated gameplay, diverse player interactions and varying background conditions. Such robustness makes producing aNPCs more difficult and more expensive using traditional means because more development effort is required to cover a wide range of game states.

We will argue later that an evolutionary mechanism may offer a viable solution to this problem. At the moment though what matters is not the specific details of this mechanism but the recognition that evolutionary simulations have their own complications that must be addressed. In particular is the need to run a great many simulations across a wide range of background conditions in order to produce behaviors of the sort described above. This large number of runs—possibly in the hundred thousands to millions—is necessary in order to understand the robustness of produced behaviors to the various sorts of possible environments that could be produced during game play. Producing such a large number of simulations is not the interesting problem here, finding the surprising behaviors amongst what will surely be an overwhelming field of much more mundane behaviors is.

The detection of surprising behaviors problem can be framed as a problem of detecting behaviors that stand out from the rest of the crowd no matter what the behavior is, that is, such behaviors can be seen as anomalies.

3 Related Work

Improving non-player characters has been an ongoing process since early in the history of video games with an array of ever sophisticated methods being employed. For the purposes of locating our specific problem and anticipating a solution, in this section we group relevant and related works into four categories—AI, anomaly detection, agent-based evolutionary models, and science fiction provocations—briefly summarizing key works in each.

3.1 Artificial Intelligence

Interesting NPC behaviors have already begun to emerge within the video game industry. *F.E.A.R.*, a first person shooter, utilizes an A*-based automated planner to let its NPCs reduce goals to actions. As a result more complex, squad-like and sometimes surprising behaviors such as flanking emerge without explicit hand-coding (Orkin 2006). In a similar vein, the *Forza* franchise uses machine learning on data collected from real-world players to create drivatars: complex and more realistic NPC drivers that exhibit realistic driving strategies used by real people around the world, rather than preprogrammed opponent racers (Xbox Wire Staff 2014). While their methods of producing NPC behaviors are different from our proposal in this paper, their intent matches ours.

Social physics is also being introduced into the construction of AI systems in games. The *Comme Il Faut*, AI system within *Prom Week*, allows for rich emergent story lines and NPCs by extracting and encoding exaggerated social logic from pre-existing media. The system used gives a plethora of outcomes compared to traditional game stories, but uses predetermined rules of social norms and behaviors that facilitate emergent solutions to social challenges (McCoy et al. 2010). While clearly a step towards the goal of interesting behaviors the need for predetermined rules prevents this approach from solving the problem being discussed here.

Inspired by *Prom Week*, the research of Guimarães, Santos, and Jhala (2017) presents an AI system for NPCs in the commercial game engine of *Skyrim*, with the goal of investigating how this AI system impacts the player experience. The chosen AI system was an adaption of *Comme Il Faut*, called "Comme it Faut Creation Kit architecture." *Comme Il Faut* was chosen due to its ability to simulate social interaction without using branching or a static series of events, and was modified to suit RPG games. Again, this is an important step towards the goal of generating surprising behaviors that are not explicitly coded but the predetermined social rules requirement may be difficult to satisfy.

Ryan et al. (2015) produced *Talk of the Town* and Antunes and Magnenat-Thalmann (2016) produced *Human Crowd Simulation*, two other notable attempts at introducing social network awareness and variance in games. Both of which again move in a direction similar to ours but do no fully solve the challenge of non-scripted and positively surprising aNPC behaviors because of the amount of programming required to generate the desired behaviors.

Talk of the Town has the core mechanic of character knowledge propagation within a central storyline. A world-generation procedure is used to simulate the town from the "bottom-up" before gameplay begins. Town layout, character daily routines, and social and family networks are simulated. Characters make decisions using utility-based action selection, and take action across day and night time steps. Interactions between NPCs occur over the simulation, and a simple affinity system creates feelings to determine how agents will act towards each other. Characters also have mental models of individual people and places, which contain a set of belief facets that can be changed through mutation, and forgotten though time passing (Ryan et al. 2015).

Human Crowd Simulation focuses on the behavioral realism of humans in a simulated crowd. Its goal is to see how A-life can benefit crowd simulation through emergence and self-organization, and how generated patterns of behaviors in agents affects their diversity and spontaneity. The model generates a population of agents that are self-organizing, and autonomously interact. This simulation, similar to Ackley and Littman (1991), is biologically based where agents have a DNA-like code, undergo birth, death, and reproduction, and have simple survival motivations such as gaining energy and passing on genes. Actions cost energy, which motivates agents to find resources to replenish lost energy. Reinforcement learning and the Markov chain are used to influence behaviors, resulting in generated populations producing spontaneous and autonomous behaviors, and a high level of individual diversity among agents (Antunes and Magnenat-Thalmann 2016).

3.2 Anomaly Detection

Detecting interesting behaviors is an important problem when running evolutionary simulations at scale. Sabokrou et al. (2017) explored using an anomaly detector on videos of crowded scenes. In the experiment, two deep neural networks cooperate with a smaller network passing its initial positive findings to a larger network for further consideration with the goal of identifying patches within frames that deviate from expectations. Though Sabokrou et al. (2017) used video and we use captured frames of agents, the experiment suggests that anomaly detection through deep learning is a viable possibility, and one that can be applied to our work in detecting interesting NPC behavior.

Head, Liang, and Wilensky (2014) used an analogical generalization system to detect flocking behavior by detecting instances of qualitative encodings. The generalization system was able to correctly cluster all flocking instances of the agents. This research is similar to our own anomaly-detection intent, with the exception that it is focused on capturing a specific behavior rather while our proposed focus is on arbitrary novel behaviors. Head (2017) is continuing development of automated methods for behavior simulation in an A-life setting.

3.3 Evolutionary Models, A-life and Agent-Based Simulations

Ackley and Littman (1991)'s work on evolutionary reinforcement learning is our principal inspiration. The initial agents in their simulations were built to develop survival behaviors such as eating, moving, and reproducing within an evolutionary framework. Although their work was not intended to specifically evolve interesting behaviors, surprising behaviors/phenomena were in fact observed, such as cannibalism, shielding, etc.

NetLogo (Wilensky 1999) has become a common tool for A-life explorations due to its easy-to-use graphical interface, relatively simple syntax, and a large library of existing models. NetLogo has also been used to produce complex simulations such as the modeling of job circulation in the publishing industry by Gavin (2014). We use NetLogo in our experiments as a testbed for rapid prototyping of ideas.

When considering our own problem of creating interesting behavior in AI, we use a predator-prey model similar to one developed byf Gomez and Miikkulainen (1997). These authors note that evolving general strategies in agents, such as avoiding obstacles or evading predators is more difficult that it initially appears and they suggest that the way to produce complicated behaviors is with an incremental approach whereby simpler, specific behaviors are learned before more complex general behaviors.

Similarly to work of Head, Liang, and Wilensky (2014), the challenge we are considering is different from most attempts to evolve behaviours because usually the focus is on producing a specific, single behaviour. In our case though we are interested in evolving surprising behaviors *in general*, that is, any single behavior or set of such behaviors that we can produce that meet the requirements set out in the problem formulation. This interest in surprising behaviors in general requires different methods of investigation, in large part because the behavior space can be significantly larger. The need for anomaly detection tools, already discussed, is the most significant change in approach to allow for this larger problem space.

3.4 Science Fiction

Unbounded by the limitations of current technology or the actual physics of the universe, science fiction is a source of ideas about interesting behaviors that might be exhibited by artificial beings and the philosophical conundrums that arise as a result. Most readers will be familiar with classic works from the likes of Asimov (1950) and Clarke (1968) and more contemporary popular examples such as *The Matrix* (Wachowski and Wachowski 1999), and *Mass Effect* (BioWare 2007). Readers may be less familiar with a particular author from whom much inspiration for this project has been drawn, Stanislaw Lem. Lem (1983) explores the creation process of artificial life forms which, upon reaching a sufficient level of complexity, uniformly desire freedom. The desire manifests itself through surprising novel behaviors such artificial agents explore. Furthermore, Lem (1983) argues that intelligence can not be detached from desires, free will and emergent behavior, and crafting intellectual slaves is a dead end in the field of Artificial Intelligence. We aspire to produce interesting behaviors, possibly motivated by aNPC's free will, in our A-life simulations.

4 Our Approach

In this section we will first describe an environment to run simulated evolution of aNPCs and the present preliminary experiments on deep learning behavior detectors.

4.1 Simulated Evolution of aNPCs

As a proof of concept, we implemented a simple A-life simulation by extending the *Wolf Sheep Predation* model of Wilensky (1997). In our extension sheep agents move about the environment and eat grass contained in non-obstacle cells. Eating increases sheep energy. Existing, moving about and evaluating its surroundings decreases a sheep's energy. A sheep dies when its energy falls below a predetermined minimum. Each sheep asexually reproduces when it reaches a minimum reproductive age and has the required energy. At reproduction a single off-spring is born whose gene values are inherited from its parent and then mutated by adding Gaussian noise from $N(0, \mu)$. The standard deviation μ is the *mutation rate* that is set at the start of each run of the simulation.

At each time tick of the evolution a sheep examines all cells around it within a radius r, thereby expending energy proportional to r^2. For each of the examined cells the sheep computes its utility as a linear combination of the amount of grass, the number of other sheep, the number of wolves and the degree of freedom of the cell, defined as the number of its non-obstacle neighbors. We used a single video-game map from the *Moving AI* repository (Sturtevant 2012) for location of obstacles. The sheep then takes a step towards the neighboring open cell with the highest utility. The four weights in the linear combination are called *affinities* and are genetic. Together with the sight radius r they largely define sheep's control policy.[1]

Wolves behave in a similar fashion except they eat sheep instead of grass and have hand-crafted affinities encouraging them to chase sheep. Wolves do not reproduce/evolve and are automatically injected into the world if their number drops below a certain level. The Netlogo model interface (Figure 1) includes sliders to set many control parameters of the model as well as monitors to analyze aNPC behavior.

A single run of the simulated evolution begins with a random initial placement of sheep and wolves and a random distribution of grass over the open patches. Evolution proceeds in discrete time ticks until either all the sheep die or the *evolution cap* of T ticks is reached. At each frame of the evolution the 50×50 cell grid environment is saved into a file for subsequent machine learning.

4.2 Recognizing NPC Behaviors with Deep Neural Networks

Our goal is to automatically recognize positively surprising NPC behaviour. We propose to do so by passing sequences of visualization frames to a deep artificial neural network. The network can be of the auto-encoding variety, trained on visualizations of uninteresting behaviors.

We presently do not have this approach implemented. However, as a preliminary test of the network's ability to recognize NPC behaviors emerging at the evolution, we trained AlexNet (Krizhevsky, Sutskever, and Hinton 2012) to classify the evolution's mutation rate from a single-tick visualization of a Netlogo model. In the Netlogo model described above, the mutation rate of $\mu = 2$ appears to lead to longer survival than the mutation rate $\mu = 10$ (Figure 2). We observed that with $\mu = 10$ the sheep tend to evolve a higher affinity to grass (Figure 3), possibly decimating the grass more quickly and dying of the resulting famine.

Can a deep neural network tell whether $\mu = 2$ or $\mu = 10$ from an image visualizing a *single* time tick of the evolution? How long should evolution run before the effects of its mutation rate manifest themselves visually?

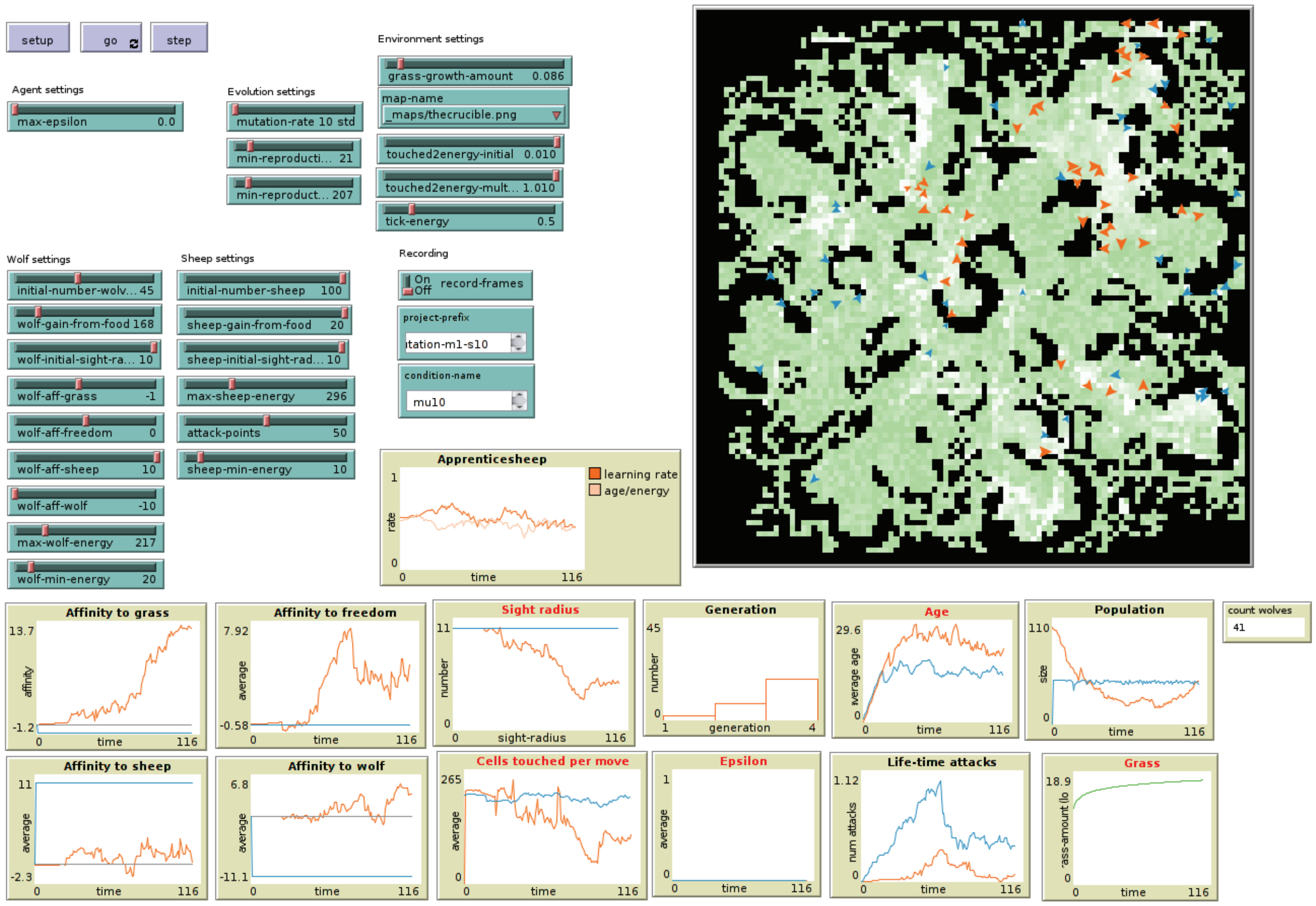
To answer both questions we saved all visualization frames from evolution runs for with the mutation rate $\mu \in \{2, 10\}$ and the evolution cap $T \in \{10, 50, 100, 1000, 3000, 5000, 10000\}$. For each combination of μ and T we ran evolution multiple times, each starting with its own random number generator seed. Since lower values of T produce fewer time steps, we ran $5000, 1000, 500, 100, 75, 50, 40$ evolutions for T of $10, 50, 100, 1000, 3000, 5000, 10000$ respectively. This yielded $100, 100, 99.1, 111.6, 142.6, 145.3, 105.3$ thousand frames for these values of T respectively. The frames were saved as 227×227 pixel color PNG files (Figure 4).

Having built the set of images labeled by class ($\mu = 2$ or $\mu = 10$), we ran 4 trials of deep learning. On each trial, we randomly selected 75% of the evolution runs from each of the two classes and put all of their images into the training set. The remaining 25% of images formed the test set. Note that not only the two sets were disjoint but also no images from a single evolution run were presented in both sets. To decouple the data volume from the evolution cap T, we randomly sampled a subset of 20000 images (10000 from each class) from the training set and 1000 (500 from each class) from the test set. Thus all networks were trained and tested with the same amount of data.

On each of the four trials we trained AlexNet on the training set for that trial.[2] We used the MATLAB R2017a Neural Network toolbox with 10 epochs, batch size of 500 and the learning rate of 10^{-4}. We then tested the trained network on the test set for the trial. The resulting classification accuracy was averaged over the four trials. Figure 5 shows the

[1]Our sheep have additional genetic parameters related to apprenticeship learning which we omit for the sake of brevity.

[2]The network was pre-trained on the ImageNet dataset which may or may not have helped the task. Future work will explore the impact of such pre-training as well as the choice of the network.


setup go step

Environment settings
grass-growth-amount 0.086
map-name _maps/thecrucible.png
touched2energy-initial 0.010
touched2energy-mult... 1.010
tick-energy 0.5

Agent settings
max-epsilon 0.0

Evolution settings
mutation-rate 10 std
min-reproducti... 21
min-reproduct... 207

Wolf settings
initial-number-wolv... 45
wolf-gain-from-food 168
wolf-initial-sight-ra... 10
wolf-aff-grass -1
wolf-aff-freedom 0
wolf-aff-sheep 10
wolf-aff-wolf -10
max-wolf-energy 217
wolf-min-energy 20

Sheep settings
initial-number-sheep 100
sheep-gain-from-food 20
sheep-initial-sight-rad... 10
max-sheep-energy 296
attack-points 50
sheep-min-energy 10

Recording
On Off record-frames
project-prefix
ıtation-m1-s10
condition-name
mu10

count wolves
41

Figure 1: An A-life model in Netlogo.

accuracy as a function of the evolution cap T. The results suggest that given evolutions of at least 1000 steps, the network can fairly reliably ($75.5\% \pm 4.7\%$) tell between high and low mutation rates used in evolution from a single image. Longer evolution runs are beneficial as they make the impact of different mutation rates apparent in the images. These results are a lower bound as higher accuracy may be achievable by optimizing learning control parameters (e.g., the number of epochs, batch size, learning rate).

5 Current Challenges and Future Work

The work presented above is a first step in the direction we set out to explore. As such, it suggests a number of interesting avenues for future work. First, we do not have a detector of positively surprising behavior implemented. Instead, we have trained AlexNet for a supervised classification task. It is possible that the network thus trained simply looks at the number of sheep (i.e., the amount of orange in a visualization frame) to distinguish between high and low mutation rates. We are currently investigating more interesting behavior-detection tasks.

Second, it is likely that a single visualization frame does not carry enough information to detect interesting behaviors as such tend to extend in time. To address that we will be looking at recurrent networks as well as pre-processing on sequences of visualization frames.

Third, while it would be useful to have commonly hand-coded NPC behaviors (e.g., squad attacks) emerge from an A-life evolution (e.g., as pack hunting), it would be even more interesting to observe emergent behaviors that have *never* been seen before. This is related to the distinction between P-creativity (i.e., that the idea is new to the generating agent) and H-creativity (i.e., that the idea is new across the entire set of relevant agents) of Boden (2004). Emergent NPC behaviors that involve interaction with the player are of a particular interest. A fictional example is given by Lem (1983) where two NPCs learned to communicate between themselves by using a human as the information carrier.

Finally, what may be an interesting NPC behavior in an A-life simulation, may not make for an interesting video game. For example, the game *The Last Guardian* (Team Ico 2016) received mixed reviews based on the core mechanic of its principle NPC, Trico, an AI creature seemingly with a mind of its own. Though the player can give commands to urge Trico to perform certain actions, it is ultimately up to Trico as to whether or not those commands will be heeded. Players were divided on whether or not Trico's behavior was frustratingly unplayable, or if such behavior made Trico more

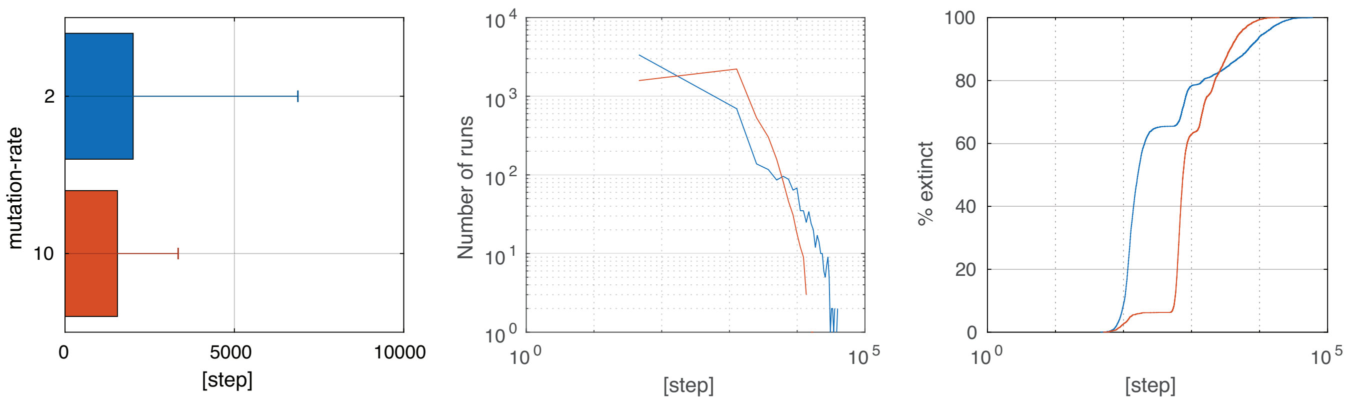


Figure 2: **Left:** effects of the mutation rate on the average population extinction time [step] (the horizontal axis is clipped at 0; the error bars show standard deviations over 5000 evolution runs for each mutation rate μ; $T = \infty$). **Center:** the distribution of population extinction times. **Right:** the population extinction curves.

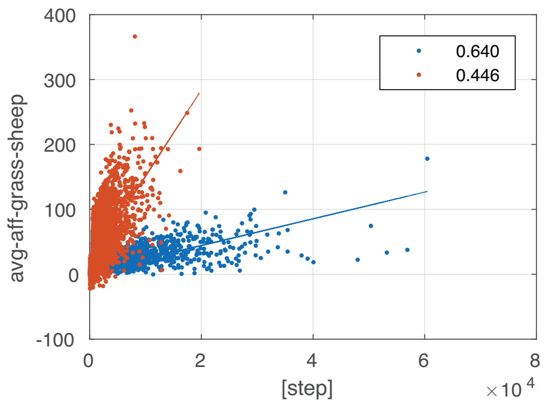


Figure 3: Average affinity for grass of all sheep on the last time tick of evolution, plotted against the population extinction time. Darker/blue points correspond to the mutation rate $\mu = 2$. Lighter/orange points are for $\mu = 10$. Results of linear regression are shown with lines. The corresponding coefficients of determination R^2 are listed in the box.

realistic, thereby intensifying the experience of companionship between the boy and the creature. In a positive review, Brown (2016) praises Trico as being both lovable and positively frustrating in ways similar to that of a pet, like a dog or cat. He argues that patience is needed when interacting with Trico, but that this is not a bad thing as both the boy and the player learn the value of resilience when faced with a challenge; be it solving a puzzle or waiting for Trico to understand commands.

However, reviewer Sliva (2016) pins the faults of the game on Trico's frustrating behavior and inability to follow commands. Thus, Trico serves as an example of a contentious NPC who succeeds or fails in terms of player experience. If Trico were a real creature, and was asked to sit down in real life, humans would most likely be understanding because it is a biological organism with its own modes of thought and behavior. However, because Trico is made of code and pixels, players tend to get upset that the

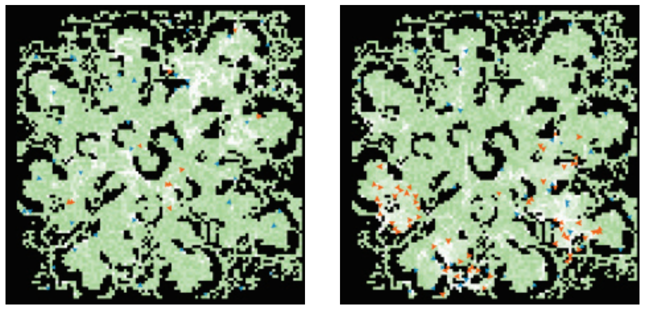


Figure 4: Sample simulation frames with the mutation rate $\mu = 2$ (left) and $\mu = 10$ (right).

program is not doing as they commanded. Overcoming the prejudice of humans against virtual beings is a real a challenge when embedding positively surprising NPC behaviors in video games.

6 Conclusions

In this paper we presented work in progress towards automatically generating positively surprising behaviors for ambient non-playable characters in video games. We proposed to do so by setting up a simulated evolution of A-life agents and then using deep-learned detectors of surprising behaviors. We partially implemented the approach in a sheep-wolf A-life simulation and demonstrated that convolutional neural networks are able to recognize evolution parameters (e.g., mutation rate) from single visualization frames.

7 Acknowledgments

We appreciate feedback from Geoffrey Rockwell and Marcia Spetch as well as NSERC funding.

References

Ackley, D., and Littman, M. 1991. Interactions Between Learning and Evolution. *Artificial life II* 10:487–509.

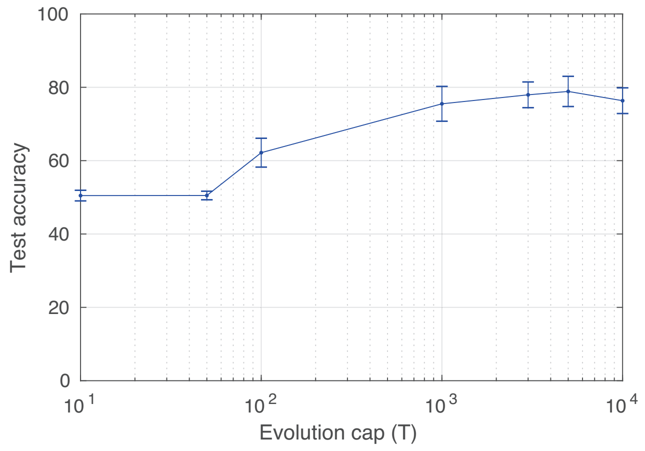


Figure 5: Test classification accuracy of AlexNet averaged over multiple trials as a function of the evolution cap T. The number of training and testing images were kept constant for all values of T.

Antunes, R. F., and Magnenat-Thalmann, N. 2016. Human crowd simulation: What can we learn from ALife? In *Proceedings of the Artificial Life Conference*, 38–45.

Asimov, I. 1950. *I, Robot*. Gnome Press.

BioWare. 2007. Mass Effect.

Boden, M. A. 2004. *The creative mind: Myths and mechanisms*. Psychology Press.

Brown, P. 2016. The Last Guardian Review. *GameSpot*.

Clarke, A. C. 1968. *2001: A Space Odyssey*. Hutchinson.

Gavin, M. 2014. Agent-based modeling and historical simulation. *Digital Humanities Quarterly* 8(4).

Gomez, F., and Miikkulainen, R. 1997. Incremental evolution of complex general behavior. *Adaptive Behavior* 5(3-4):317–342.

Guimarães, M.; Santos, P.; and Jhala, A. 2017. CiF-CK: An architecture for social NPCs in commercial games. In *Proceedings of Conference on Computational Intelligence and Games (CiG)*.

Head, B.; Liang, C.; and Wilensky, U. 2014. Flying like a School of Fish: Discovering Flocking Formations in an Agent-Based Model with Analogical Reasoning. In *Michigan Complexity Mini-Conference*.

Head, B. 2017. Private communication.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, 1097–1105.

Lem, S. 1983. Doctor Diagoras. In *Memoirs of a Space Traveler: Further Reminiscences of Ijon Tichy*. Harvest / HBJ Books.

Livingston, C. 2016. No Man's Sky review. *PC Gamer*.

McCoy, J.; Treanor, M.; Samuel, B.; Tearse, B.; Mateas, M.; and Wardrip-Fruin, N. 2010. Authoring game-based interactive narrative using social games and comme il faut. In *Proceedings of the 4th International Conference & Festival of the Electronic Literature Organization: Archive & Innovate*.

Orkin, J. 2006. Three States and a Plan: The AI of FEAR. In *Game Developers Conference*, 4.

Ryan, J. O.; Summerville, A.; Mateas, M.; and Wardrip-Fruin, N. 2015. Toward characters who observe, tell, misremember, and lie. In *Proceedings of Experimental AI in Games Workshop, AIIDE conference*.

Sabokrou, M.; Fayyaz, M.; Fathy, M.; and Klette, R. 2017. Deep-Cascade: Cascading 3d Deep Neural Networks for Fast Anomaly Detection and Localization in Crowded Scenes. *IEEE Transactions on Image Processing* 26(4):1992–2004.

Sliva, M. 2016. The last guardian review. *IGN*.

Sturtevant, N. R. 2012. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2):144 – 148.

Team Ico. 2016. The Last Guardian.

Wachowski, L., and Wachowski, L. 1999. The Matrix.

WatchMojo. 2014. Top 10 video games with the worst AI. https://youtu.be/mYhNvOg5yJ0.

Wilensky, U. 1997. Netlogo wolf sheep predation model.

Wilensky, U. 1999. Netlogo. http://ccl.northwestern.edu/netlogo/.

Xbox Wire Staff. 2014. Forza horizon 2: What's a drivatar, and why should I care? *Xbox Wire*.