# Real-time Motion Generation for Imaginary Creatures Using Hierarchical Reinforcement Learning

Keisuke Ogaki
DWANGO Co., Ltd.
keisuke_ogaki@dwango.co.jp

Masayoshi Nakamura
DWANGO Co., Ltd.
masayoshi_nakamura@dwango.co.jp

## ABSTRACT

Describing the motions of imaginary original creatures is an essential part of animations and computer games. One approach to generate such motions involves finding an optimal motion for approaching a goal by using the creatures' body and motor skills. Currently, researchers are employing deep reinforcement learning (DeepRL) to find such optimal motions. Some end-to-end DeepRL approaches learn the policy function, which outputs target pose for each joint according to the environment. In our study, we employed a hierarchical approach with a separate DeepRL decision maker and simple exploration-based sequence maker , and an action token, through which these two layers can communicate. By optimizing these two functions independently, we can achieve a light, fast-learning system available on mobile devices. In addition, we propose another technique to learn the policy at a faster pace with the help of a heuristic rule. By treating the heuristic rule as an additional action token, we can naturally incorporate it via Q-learning. The experimental results show that creatures can achieve better performance with the use of both heuristics and DeepRL than by using them independently.

## CCS CONCEPTS

• **Computing methodologies → Evolutionary robotics**; **Animation**; *Machine learning*; *Physical simulation*;

## KEYWORDS

Reinforcement Learning, Q-Learning, Neural Network

**ACM Reference Format:**
Keisuke Ogaki and Masayoshi Nakamura. 2018. Real-time Motion Generation for Imaginary Creatures Using Hierarchical Reinforcement Learning. In *Proceedings of SIGGRAPH '18 Studio.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3214822.3214826

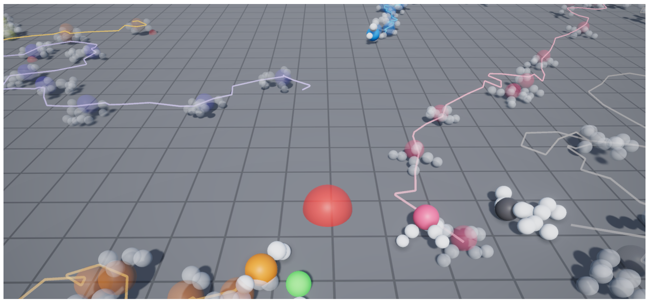

**Figure 1: Creatures with diverse bodies learning to move. They have locomotion skills and can plan their movement direction. Users can control them by feeding them.**

## 1 INTRODUCTION

Different animals have different types of body structures and motor skills. Their bodies and movements are suitable for their respective living environments. Humans can imagine, draw, and animate creatures, such as dragons, Pegasi, or mermaids, which do not exist in reality. In this study, our goal is to generate creatures with bodies and motions that are logically suited for different simulated environments. There exist some challenges in realizing realistic creatures through simulations. These include learning of available body structures for animals, developing motor skills with the generated bodies, and determining which creature survives in specific environments, through simulations [Sims 1994]. In this study, we focused on developing motor skills for imaginary creatures by using machine learning techniques.

Deep reinforcement learning (DeepRL) can control a creature's motion by predicting future rewards; however, it takes a long time to converge, and the result is slightly unstable, especially in a continuous action space [Henderson et al. 2018]. As our goal is to observe how creatures learn to move while inheriting and mutating their bodies through generations, it is essential that each creature starts moving at an early stage so that selection can be achieved. Moreover, we preferred our system to work on a mobile computer. We employed a hierarchical approach using simple bandit algorithm in addition to DeepRL. By using instant rewards, simple bandit algorithm can be used to generate motion; although this cannot maximize future rewards by itself, it can achieve fast convergence.

## 2 METHODOLOGY

Our main contribution is that our system enables tens of creatures to learn to move using DeepRL, in a single mobile device. The key technique is to incorporate the simple bandit algorithm with DeepRL. Figure 2 shows an overview of our learning framework for one creature. Our architecture is mainly composed of two parts, as follows: decision maker $\pi^d(s,t)$ and sequence maker $\pi^s(t,a)$. The decision maker chooses discrete action token $t$ maximizing future rewards $\sum r^d_t$, while the sequence maker generates continuous motion sequences $a$, thereby maximizing immediate rewards. This can avoid the complexity of reinforcement learning.

*SIGGRAPH '18 Studio, August 12-16, 2018, Vancouver, BC, Canada*
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5819-4/18/08.
https://doi.org/10.1145/3214822.3214826

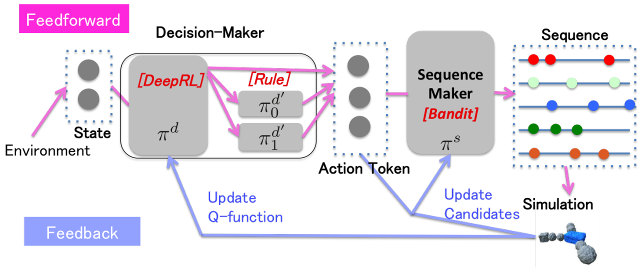

**Figure 2: Overview of our framework. First, the decision maker selects the action token according to the inputs. Then, the sequence maker generates a motion sequence that is appropriate for the creature's body. Finally, the creature enacts the sequence and returns the simulation result as feedback to both the sequence maker and decision maker. Note that the decision maker is divided into deepRL-based and rule-based components.**

The decision maker works based on predictive decision making according to high-level rewards $r^d = r_0^d \ldots r_N^d$. It outputs the high-level discrete action token $t$. Action token $t$ indicates the primitive motions that the sequence maker can perform (e.g., go straight, turn right, go right, and stay there). Each action token has an accompanying reward function $r^s$. By this simplification, we can directly employ the deep Q-learning algorithm [Mnih et al. 2015].

The sequence maker generates a continuous motion sequence $a$ according to action token $t$ decided by the decision maker, and it learns actions according to immediate rewards $r^s$. Action $a$ is the target pose in the next timestep for each joint.

Our system incorporates a heuristic rule with RL. Decision maker $\pi^d$ treats $\pi^{d'}$ as its discrete action $t'$. When $\pi^{d'}$ is chosen, $t$ is decided with the associated rule.

In our system, action $a$ is decided at a frequency lower than the simulation frequency to save the computational cost. Instead of generating a target pose for all the time series, our creatures generate a target pose and their planned time in batches (Figure 2).

For simulating a virtual environment, we implemented this simulator using Unity[1]. Our source code for the simulator, creatures, and learning are available for reproducing our experiments [2].

## 3 EXPERIMENTAL RESULT

We evaluated the cooperation of RL and a rule using a simple move-to-eat situation. The experiment constituted only one creature and a fixed amount of food items in the environment. Note that 30% of the food items were fed at a higher point than the creatures can reach. Once the creature reached the food item, another food item spawned at a random point. Every food item disappeared after 300 s. We evaluated how much food the creatures could obtain in 30,000 s.

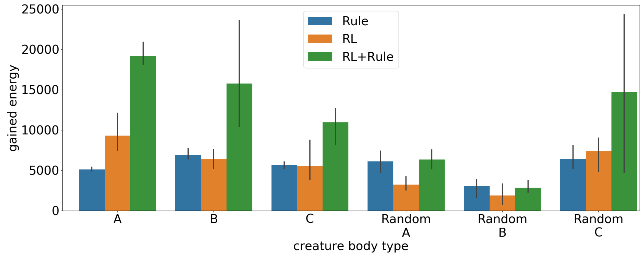

**Figure 3: Total energy acquired after 30,000 s. Each bar shows the mean and standard deviation of three trials. The energy was evaluated for three types of decision makers, as follows: a) RL only, b) Rule only, and c) the proposed method representing a+b. We conducted this evaluation by using various bodies.**

Figure 3 shows the amount of food items the creatures obtained based on the three types of decision makers, as follows: a) rule $\pi^{d'}$ only, b) RL $\pi^d$ only, and c) both $\pi^d$ and $\pi^{d'}$. As shown, for all types of bodies, the proposed method combining $\pi^d$ and $\pi^{d'}$ overcame the baseline $\pi^d$ and rule-based $\pi^{d'}$. The rule-based decision maker was too simple, such that the creatures always moved toward the nearest food item, even if they could not reach it. However, the rule helped creatures to learn the policy at an early stage, so the proposed method worked better than using the reinforcement decision maker alone.

## 4 DEMONSTORATION AND USER EXPERIENCE

For the demonstration, we also present the system for designing creatures. Following Sims's [1994] approach, we provide a technique for generating a new creature by combining the body structures of two creatures. By repeating this process, users can obtain the intended creatures.

We also provide the server where every creature generated by the users live. Users can generate and train creatures on client machines (smart-phones, PCs) and then send their creatures to the server to see whether they survive against other ones. Our system allows more than 30 creatures to live on a single machine.

## REFERENCES

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep Reinforcement Learning that Matters *(AAAI)*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles BeaÂŁie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533.

Karl Sims. 1994. Evolving Virtual Creatures. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. 15–22.

[1] https://unity3d.com

[2] https://github.com/dwango/rlcreature