Engineering Document Design for CSCI599 midterm

Zhihan Zhang, ChuanZhe Li, Jia Xu

March 2020

1 Project Goal

As the history passes, there are only two things stay in stale, survive and competition. The life in earth can be simplified to the maximum their ability to live, to become healthy, the simplest goal for nature beings may induce millions of undocumented strategies for every unique species to sustain themselves.
Here we want to mimic this status and find hidden strategies based on simulator, thus we want to create many agents fight for living everyone need to acquire daily necessities to survive and by adopting deep reinforcement we may mimic the process of each species can learn from the terrain and conditions provided by the infinite and may induce serendipitous results.

2 Background and Game Overview

2.1 The genre of the game

RPG(Role-playing games) is a popular genre with thousands of famous Ips such as Final Fantasy and pokemon, they will create many scenarios and players need to make curtain decisions in a given term for further steps.
Here we want to use Massively Multiplayer Role-playing game for this sand-box simulator which allows hundreds of players play on the same map to make their decisions and live as long as possible.

2.2 How the game played

The artificially intelligent agents compete to survive, and learn new skills in a long the way. Every agent has three parameters: food, water and health. At each server tick (time step), agents may move one tile.The AI agents must acquire food and water to stay alive, and they move across the map to gain both. This brings them into conflict with other agents and requires the AI to move carefully in order to maximize the chance of finding resources as it explores. Agents forage for food and must refill their water supplies .[6]

terrain types	properties
Stone	inaccessible
Grass	Traversable
Forest	have food & traversable
Water	have water & inaccessible
Scrub	Traversable

Table 1: Terranin types

We set environment with different terrains to increase the reality of the game. there are six types of terrain types and each of them have different properties.

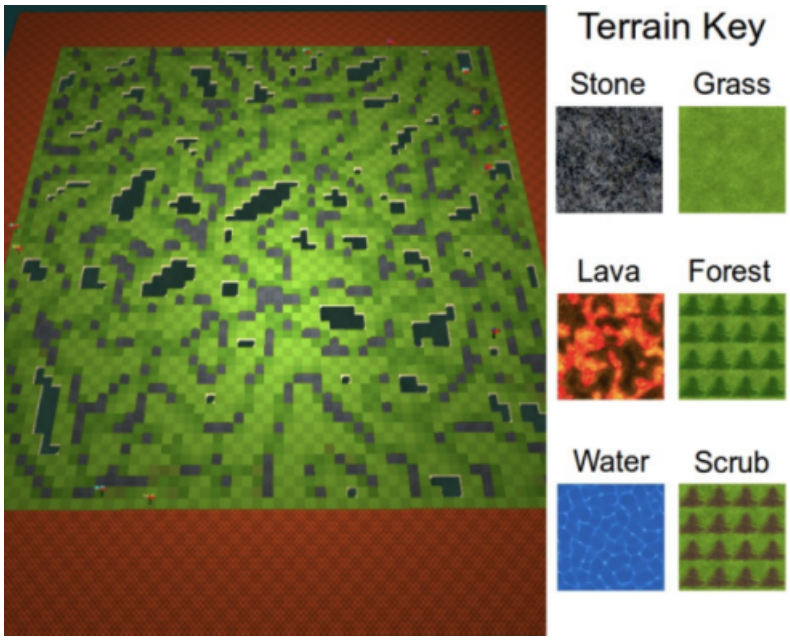

Figure 1: Terrain and Map Example

At each server tick (time step), agents may move one tile or just stay there. Stepping on a forest tile or next to a water tile refills a portion of the agent's food or water supply, respectively.
The training purpose is to make the agents stay alive as long as much. Each agent has an full observation. They make the best decision according to the current status and gain wise decisions on the next step.

3 Prior Researches

3.1 stage 1

At the very begining, we explored classic algorithms such as model-based learning like Monte Carlo Tree Search[4] and their application in tic-tac-toe[3] and harvesting game made by myself https://github.com/ZhiHanZ/monte_carlo_tree_search This strategy is perfect if we have small amount of agents and need less generalization but for our current problem, it cannot fulfill our requirement because the performance is bad in scale.

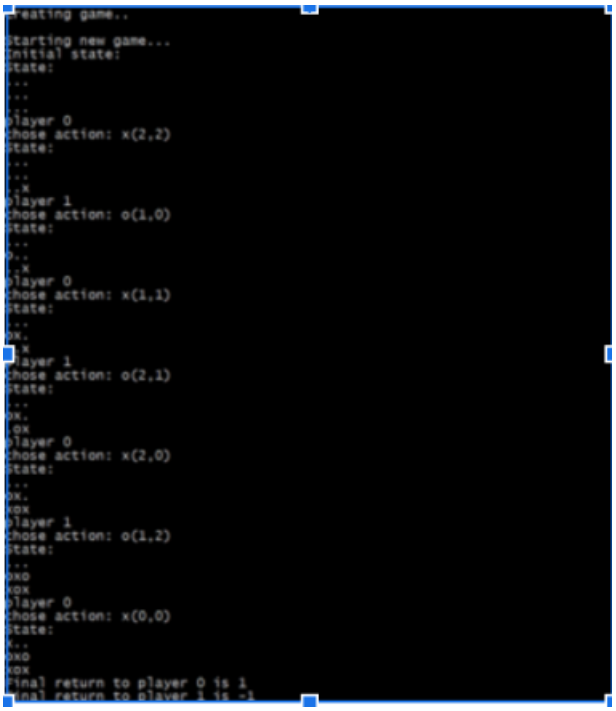

Figure 2: Tic Tac Toe Example

3.2 stage 2

After that we try to work on the turn-based strategy board game which the human player can fight against the AI player. In the mean time we learned the skills and knowledge by adopting Machine Learning-Agent which benefited a lot. We step by step deployed it on the unity environment but the problems are found that the turn-based game needs an on-demand decision making process which is still under development in Unity and we have no example as reference.[2]

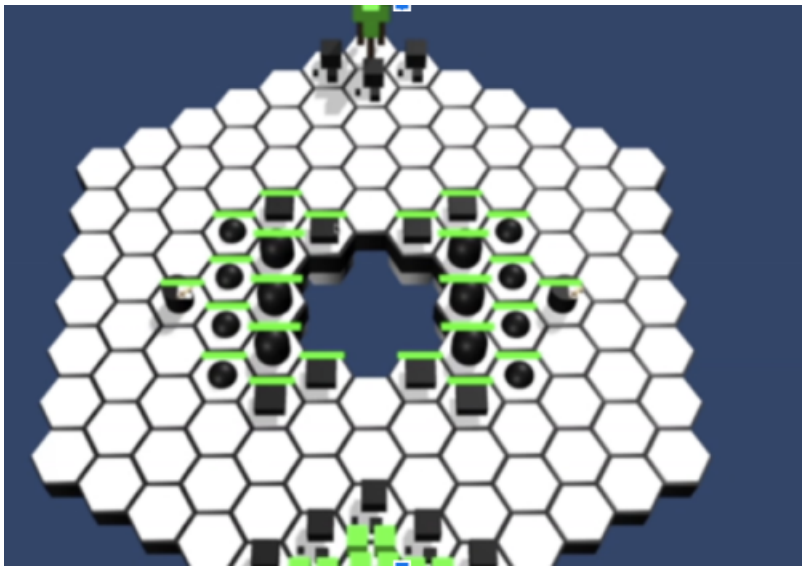

Figure 3: Unity Turn Based Game example

4 Methodology

4.1 why using Deep Reinforcement Learning

After many trials, we finally decided to use Deep Reinforcement Learning as our final methodology for life simulator. Traditional Reinforcement Learning needs to design complex rules for each agents to maximize their rewards during gaming play, That is not make sense because the intelligence made by beings come from learning.
There are something like walking or sitting come in nature and everybody can do it.
But something people can just learn from practice such as driving or dance. We can learn a huge variety of things even very complex one such as Go game or StarCraft II [1]
There are many contemporary works made by Google or DeepMind which train their agents to achieve high degree of proficiency in domain governed by simple known rules such as The Game of Go [5].

4.2 Define given terms used in our model

s_t : the current state of given agents given time tick t, state contains position information, current food, water, health and so on
o_t : the current observation based on one agent.
a_t : the action made by given agent which have five types, North, West, East, South, Pass

$\pi_\theta(a_t|s_t)$: Policy(fully observed)
τ: the trajectory of given agents which is a sequence of observation, action and reward. $\tau = (o_t, a_t, r_t, \ldots, o_T, a_T, r_T)$
$\mathbf{R}(\tau)$: reward given trajectory which is the sum of each survival one in every time titch with discount γ which by default is 0.99 $\mathbf{R}(\tau) = \sum_t^T \gamma^t r_t$

4.3 Training steps

The purpose is to maximize the reward in given trajectory

$$\underbrace{p_\theta\left(\mathbf{s}_1, \mathbf{a}_1, \ldots, s_T, \mathbf{a}_T\right)}_{p_\theta(\tau)} = p\left(\mathbf{s}_1\right) \prod_{t=1}^{T} \pi_\theta\left(\mathbf{a}_t|\mathbf{s}_t\right) p\left(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t\right) \tag{1}$$

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right] \tag{2}$$

Here we want to optimize the learned parameters maximize the expected survival rewards by adopting **policy gradient method**

$$J(\theta) = E_{\tau\sim p_\theta(\tau)}\left[\sum_t r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right] \approx \frac{1}{N}\sum_i \sum_t r\left(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}\right) \tag{3}$$

$$\nabla_\theta J(\theta) = E_{\tau\sim \pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t|\mathbf{s}_t\right)\right)\left(\sum_{t=1}^{T} r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right)\right] \tag{4}$$

We will update based on epoch

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{5}$$

4.4 Training Model

Here the model is just a linear model, we use embedding and flatten a agents' reward to a one-dimensional encoded vector and fit it into a linear layer of with trainable parameters.

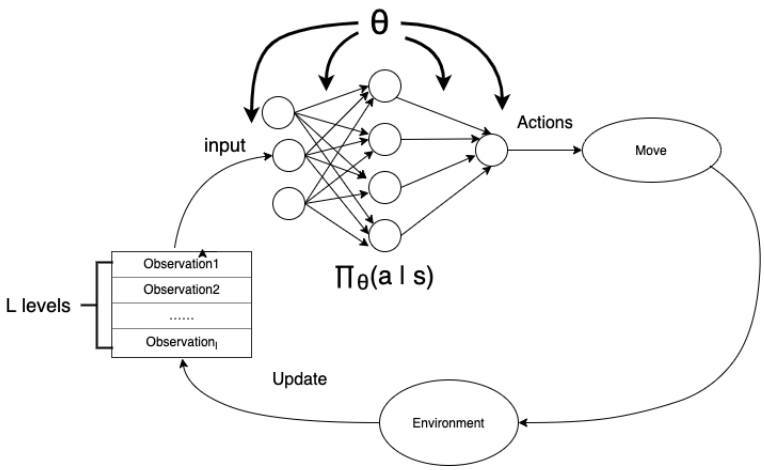

Figure 4: current Net word model

4.5 Network Architecture

Network here is simple and easy to configure, we used Twisted Network Library to communicate packet information between client and server

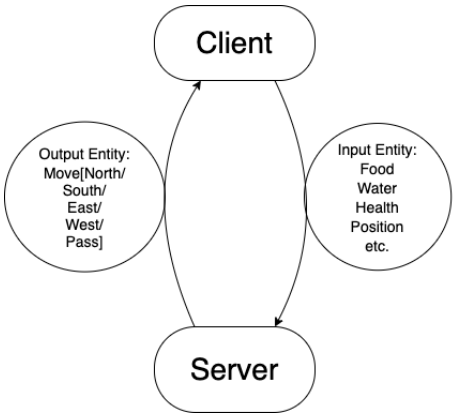

Figure 5: current Net word model

References

[1] Kai Arulkumaran, Antoine Cully, and Julian Togelius. Alphastar: An evolutionary computation perspective. 2019.

[2] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents, 2018.

[3] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Paul Muller, Timo Ewalds, Ryan Faulkner, János Kramár, Bart De Vylder, Brennan Saeta, James Bradbury, David Ding, Sebastian Borgeaud, Matthew Lai, Julian Schrittwieser, Thomas Anthony, Edward Hughes, Ivo Danihelka, and Jonah Ryan-Davis. OpenSpiel: A framework for reinforcement learning in games. *CoRR*, abs/1908.09453, 2019.

[4] Marc Lanctot, Mark H. M. Winands, Tom Pepels, and Nathan R. Sturtevant. Monte carlo tree search with heuristic evaluations using implicit minimax backups, 2014.

[5] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017.

[6] Joseph Suarez, Yilun Du, Phillip Isola, and Igor Mordatch. Neural mmo: A massively multiagent game environment for training and evaluating intelligent agents, 2019.