An Autonomous Agent Framework for The Game "Don't Starve"

Rodrigo T. Aki[1], Marcos R. O. A. Máximo[1], Mariá C. V. Nascimento[1]

[1]Aeronautics Institute of Technology – São José dos Campos, Brasil

rodrigo.tanaka.aki@gmail.com, mmaximo@ita.br, mariah@ita.br

Abstract. *Applying AI algorithms to find optimal strategies in games has been a heavily researched subject that had a significant impact on video game development and served as the basis for several applications in the real world. In this paper, we investigate developing an AI agent for a game called "Don't Starve", a single-player survival game. The main contribution of this work is a novel agent framework for this game that can survive under the same conditions as a human player. In this regard, the agent uses the game screen as the only source of information and executes actions with mouse and keyboard. After testing the AI capabilities, the agent could identify game objects correctly, plan its next steps, collect important resources, and survive for a few days.*
Keywords *Computer games, Artificial intelligence, Autonomous agent, Heuristics*

1. Introduction

Applying AI algorithms to determine optimal game strategies has been a heavily researched subject. Two popular examples were OpenAI's Dota 2 AI [Berner et al. 2019] and DeepMind's Starcraft II AI [Vinyals et al. 2019]. They created agents that achieved great performance on very complex games, outperforming even some of the best players in the world.

"Don't Starve" [Klei Entertainment 2024] is a single-player survival game where the objective is to survive as long as possible. The game is played on a computer, with the player controlling their character using a mouse and keyboard. "Don't Starve" is classified as a "2.5D Game" because it uses 2D sprites arranged to create a 3D illusion. Players explore to discover and collect essential resources, fend off dangers, and improve access to these resources to aid in the character's survival. In [Almeida e Rui 2018], they created an agent for the game "Don't Starve Together", a multiplayer version of the game "Don't Starve". The authors focused on creating an agent framework, however it did not have planning capabilities or inventory management. As the previously mentioned AI to play games, this employs varying levels of expertise, different approaches, and different objectives.

However, maximizing performance is not the only possible objective when developing an AI. Therefore, in this paper, we introduce a methodology for an agent to play the game. While the objective of this work includes creating an AI that can play the game, its contribution is to serve as a testbed for many different AI and autonomous agent algorithms.

2. Background

YOLO: You Only Look Once (YOLO) is an algorithm that uses a Convolutional Neural Network (CNN) for object detection. In the YOLO v2 algorithm, the 416x416 input image is processed by the CNN exactly once, with no additional reprocessing of the same pixel. It is divided into 32x32 pixel cells and, for each one, it computes the probability of a certain object being in it, its location relative to the cell, and its dimensions relative to the whole image. This creates a tensor with both height and width 32 times smaller. Its final layer outputs probabilities and bounding box coordinates for the annotated objects. We used YOLO v8 in this work to detect objects in the captured screen image, however there is no published paper for that version. Its documentation can be accessed on Ultralytics's site [Ultralytics 2023].

Inverse Perspective Mapping: Inverse Perspective Mapping (IPM) is concerned with determining the 3D position of an object from its position in an image [da Silva et al. 2024]. The well-known pinhole camera model can be used in this problem and it dictates that the perspective projection of a point $[x\ y\ z]^T$ represented in the world coordinate system generates an image point $[u\ v]^T$ [Szeliski 2010] given by

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{1}$$

where K and $[R\ t]$ are the intrinsic and extrinsic matrices, respectively. IPM deals with the inverse problem of projecting a world point to the image. As there are infinite world points mapped to the same image point during perspective mapping, we need to have an additional constraint to determine the world point uniquely. In our case, since the objects never leave the ground, we can simply set $y = 0$. This yields three equations for three variables. Since u and v are known, we can solve for x, z and s. Therefore, we know that the point $[u\ v]^T$ in screen coordinates corresponds to $[x\ 0\ z]^T$ in world coordinates and we were able to map each pixel on the screen to its 3D position. We used IPM in the proposed strategy to calculate an object position in the game's coordinate system from the object position in the captured image (detected using the YOLO algorithm).

3. The Game "Don't Starve"

There are many possible strategies and playing styles for "Don't Starve", but all of them involve managing the character[1] three main attributes: health, hunger, and sanity. The game uses 2D sprites with an isometric view, as seen in Figure 1. In this figure, the character controlled by the player is marked in black and the player attributes are highlighted in red (health), yellow (hunger), and orange (sanity). Each circle is filled proportionally to what it represents.

Health is the representation of the character's vitality, hunger represents how satiated the character is and sanity represents the character's mental state. The player loses when their health reaches zero. The game has a day-night cycle with three phases:

[1]We will use "character" to refer to the player's in-game representation and "player" to indicate the actual person playing the game.

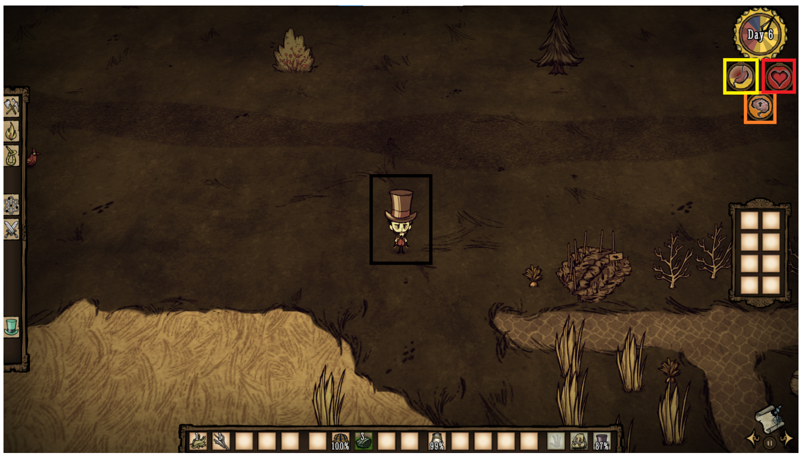

Figure 1. An in-game screenshot.

day, dusk, and night. Being in complete darkness for a few seconds causes the character to receive an unavoidable attack, thus losing health, which means that light is imperative for survival. This means that the most important resources for the first few days are food and light sources. Important skills for survival include time management, forward planning, adaptability, and fighting technique. The game consists of exploring in order to discover where important resources are and collecting resources to aid the character's survival while fighting off dangers and improving access to those resources. For more details on the game, we recommend [Almeida e Rui 2018].

4. Agent Implementation

The AI is divided into five layers as presented in Figure 2: Perception, Modeling, Decision Making, Control, and Action. We shared the code as open source on GitHub[2].

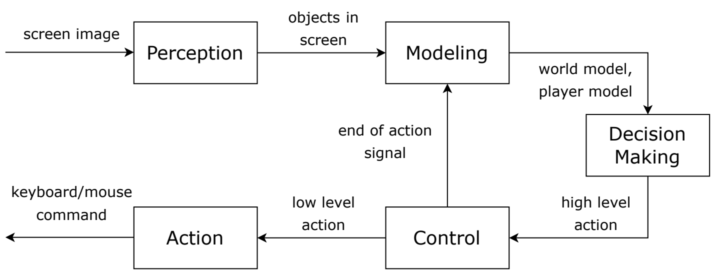

Figure 2. Block diagram showing the AI layers.

Perception: The AI uses a YOLO v8 CNN trained on a dataset with classes considered important. The neural network identifies the objects and their positions from the captured image. In addition to manually annotated images, part of the training dataset was procedurally generated with game assets, split into "important" objects that should be annotated and "not important" objects that should not be marked. This is crucial to expose the CNN to objects present in the game and to help mitigate false positives.

[2]https://github.com/krodrigo71008/DS-AI

Modeling: The Modeling is split into the World Model and the Player Model. The latter simulates the natural decrease of hunger, and sanity, and updates the player's position according to the direction it moves. The World Model is responsible for converting the image coordinates of objects to their positions in the world, keeping track of those positions, and removing nearby objects if they were not observed for some time, which would indicate Perception had a false positive.

Decision Making: There are two main layers in the Decision Making. The high-level layer chooses which action the character should do. The low-level layer breaks it down into simpler actions. Also, an emergency system might override the decision in some cases, such as when starving or when a hostile mob is close. The high-level layer decides a "primary action", according to a behavior tree that decides as follows: since the game's main threats in the first few days are darkness and food, first we gather enough grass and twigs to make two torches, which are the cheapest light source, then craft one torch, gather more grass and twigs to compensate for the resources we spent crafting the torch, then lastly gather as much food as possible. The low-level layer simply decides a small step for the high-level objective. If we want to gather resources (either grass and twigs for a torch or food), we search for sources of those resources (berry bushes or carrots for food, grass tufts for grass, and saplings for twigs) in the World Model and go to the nearest one that is not harvested. If we do not know any, we should explore.

Control: For the most part, this layer simply converts actions to keyboard or mouse commands, for example: eating is right-clicking the item you want your character to eat, equipping is right-clicking the item you wish to equip, going in a direction is pushing and holding some buttons ('w' for up, 'a' for left, 's' for down and 'd' for right), picking something up involves walking closer to it and then clicking the object.

Action: This is a very simple layer that handles interfacing with the keyboard and mouse libraries. It supports the following actions: pressing keys and holding them, pressing keys and releasing them immediately, using the mouse to left-click, and right-click.

5. Results and Discussion

In order to test the YOLO performance, we calculated the mean average precision (mAP) using the test images using a modified version of the code available on this repository[3], developed for [Cartucho et al. 2018]. The average precision for all classes was very high, as we were able to achieve an mAP of 96.32%, which was satisfactory for the project's needs.

To test the inverse perspective mapping, we used some commands in a test world to spawn two objects in known positions. Then, we recorded a video in which we walked around those objects and extracted five images from that video. We estimate the object's world positions using the screen positions and the player's world positions. Then, by comparing those to the real positions, we got an average error of 0.246 in-game units. Considering that the player's in-game speed is 6 in-game units per second, this is a pretty small error. We also recorded the agent playing on two different worlds to ensure that it was able to operate correctly. The videos can be accessed here[4] and here[5].

[3]https://github.com/Cartucho/mAP
[4]https://youtu.be/mrPYFZpQbrs
[5]https://youtu.be/eV34XNymTxQ

6. Conclusions and Future Work

In this paper, we proposed a framework to develop an AI to play the game "Don't Starve". There are several benefits in doing so, the most interesting for us being the development of a platform to try out artificial intelligence techniques and compare them.

Even with all its simplifications, the framework achieved good performance on a very complex problem. The problems that each layer solves are all significant on their own, but making it all work at once is extremely challenging. Even so, we were able to make a functional system with isolated layers, achieving our objective.

However, there are still many possible improvements, since the YOLO network could be trained to detect more classes and its accuracy could be improved, a segmentation network could be useful to detect different types of terrain, player position estimation could be improved with an algorithm such as Simultaneous Localization and Mapping (SLAM), and the decision making could be less simplistic.

7. Acknowledgments

Research carried out using the computational resources of the Center for Mathematical Sciences Applied to Industry (CeMEAI) funded by FAPESP (grant 2013/07375-0). Marcos Máximo is partially funded by CNPq – National Research Council of Brazil through the grant 307525/2022-8. Mariá C. V. Nascimento is partially funded by CNPq – National Research Council of Brazil through the grant 309385/2021-0.

References

Almeida, F. e Rui, P. (2018). Creating an agent-based framework for don't starve together. Master's thesis, IST, University of Lisbon, Portugal.

Berner, C. et al. (2019). Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680.

Cartucho, J., Ventura, R., e Veloso, M. (2018). Robust object recognition through symbiotic deep learning in mobile robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2336–2341.

da Silva, F. B. D. R., de Albuquerque Máximo, M. R. O., Yoneyama, T., Barroso, D. H. V., e Aki, R. T. (2024). Calibration of inverse perspective mapping for a humanoid robot. In Buche, C., Rossi, A., Simões, M., e Visser, U., editors, *RoboCup 2023: Robot World Cup XXVI*, pages 117–128, Cham. Springer Nature Switzerland.

Klei Entertainment (2024). Don't starve. `https://klei.com/games/dont-starve`. Accessed: 2024/05/10.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 1st edition.

Ultralytics (2023). YOLOv8. `https://docs.ultralytics.com`. Accessed: 2024/05/10.

Vinyals, O. et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5.