Advanced Games Engineering Report

Cameron Pearson

December 7, 2024

Contents

Chapter 1

Introduction

This report discusses the development of my economy simulation and AI agents. It begins by outlining the exploratory phase, where my project's concept and the inspiration behind it is discussed. This section also covers the research conducted to shape the approach to the project and the formulation of the final plan.

The report then progresses to the base deliverable, detailing the initial development of the simulation, challenges faced, and insights gained during this phase

Finally, the report explores the extension phase, discussing deviations from the initial plan, improvements made to both the simulation and the AI, and the incorporation of features such as multi-agent interaction and customisable agents.

Chapter 2

Exploratory Phase

2.1 Initial Project Idea

The idea for this project stemmed from my interest in economy games, which I occasionally play in my free time. However, I often found these games becoming repetitive, which led me to consider how their repetitive aspects could be offloaded to an AI to handle more efficiently.

This sparked the concept of creating a simplified economy simulation, loosely mimicking real-world dynamics, where individuals work, buy, and consume to drive a steadily growing market. The goal was to develop an AI that could interact with this simulation efficiently, bypassing the repetitive nature that might bore a human player. I felt this idea presented an intriguing premise for the project, combining my interest in games with the challenge of developing AI for dynamic, interactive environments.

2.2 Research

To begin the research phase, I started by determining the programming language and libraries most suited for my project. Python stood out for its simplicity, extensive library support, and active community, which would simplify development and facilitate rapid prototyping.

For the simulation's graphical interface, I chose PyGame [2]. It allowed me to easily implement interaction and visualisation, enabling real-time observation of resource dynamics and market behaviours. PyGame's event-driven system streamlined the integration of user controls and interactive features.

To implement AI agents, I decided to use Stable Baselines3 [4] alongside the Gym [1] framework. Stable Baselines3 provides robust reinforcement learning algorithms, such as PPO [3] and A2C, in a user-friendly package. Its compatibility with Gym environments made it an ideal choice for developing and training agents to interact with the simulation. Gym's flexibility allowed me to structure the environment to include detailed observations and custom reward systems, ensuring agents could learn and adapt effectively within the economic simulation.

By selecting these tools, I established a streamlined development process with powerful libraries tailored to the needs of the simulation and AI training.

2.3 Final Plan

I developed the concept for my economy game with a focus on creating a self-sustaining, simplified economic system. The core idea revolves around a population of people who consume resources daily. To replenish these resources, they must purchase them from the market. To afford these purchases, individuals work in factories, which in turn produce the goods that supply the market. This creates a continuous cycle of consumption, production, and trade, allowing the economy to develop over time.

For the AI component, my final plan was to design an intelligent agent capable of interacting with this simulation efficiently. The AI would observe the state of the economy, such as market prices and resource availability, to make decisions on when to buy, sell, or work. The goal was to train the AI to maximise profit. By using reinforcement learning from stable baselines 3, the AI could learn optimal strategies over time, adapting to the changing dynamics of the simulated economy.

Chapter 3

Base Deliverable

3.1 Introduction

The base deliverable for this project focused on creating a functioning economy simulation, setting up an environment for interaction, and developing a basic AI capable of engaging with the simulation.

3.2 Simulation Development

Developing the simulation presented several challenges. The first step was designing and implementing the necessary classes for the key components, such as people, factories, and the market. Each class had to accurately represent its role within the economy.

Next, I established the parameters for the simulation, including base values for resources, consumption rates, production capacities, and pricing mechanisms. These parameters were crucial to ensuring the simulation behaved realistically. Once these values were set, I developed the main function to define the order in which events occurred each day, such as resource consumption, purchasing, and production cycles.

To gain insights into the simulation's behaviour, I utilised Matplotlib for data visualisation. This allowed me to track trends in resource prices, population levels, and other metrics over time. Visualising the data was extremely helpful in debugging and refining the simulation.

However, tuning the simulation proved to be one of the most time-consuming tasks. Initially, the economy faced severe instabilities: either production rates lagged, leading to economic crashes, or excessive cash flow caused hyperinflation. Achieving a stable simulation required lots of adjustments to the parameters and logic until a balanced and functional economy was working.

Once the simulation was stable, I added an interactive component, enabling users to play the game by making decisions within the economy. However, this gameplay proved to be less engaging than I had hope, and as a result I decided not to prioritise its further development. Ultimately, the majority of the development time was spent stabilising the simulation, which formed the core of the base deliverable.

3.3 Gym Environment Setup

Before the AI could interact with the simulation, I needed to create a Gym environment to serve as the interface. Using the Gym documentation, I structured the environment to include all necessary elements, such as observations, actions, and rewards.

The Gym environment was designed to provide observations based on the state of the economy, including details such as resource prices, market supply and demand. The action space allowed the AI to make decisions such as buying, selling, or advancing the simulation.

3.4 Stable Baselines Development

The initial AI was implemented using Stable Baselines3, focusing on a single agent interacting with the simulation. The reward scheme was straightforward: the agent received positive rewards if its net worth increased compared to the previous day. While this approach worked to some extent, it had limitations due to the simple nature of the rewards scheme. However the AI achieved impressive results, reaching net worths of up to $180,000 in some runs. However, these results were inconsistent, and performance varied significantly depending on the simulation's initial conditions and random factors.

To monitor and evaluate the agent's training progress, I implemented a system to log data into a CSV file after each training session. This file recorded metrics such as the agent's net worth, total cash, and inventory levels for each resource, providing a detailed view of its performance over time. Additionally, a separate text file was used to track the best-performing run, capturing key metrics from sessions where the agent achieved the highest net worth. These tools were greatly helpful in identifying trends and diagnosing issues during training, making it easier to refine the AI's behaviour.

Training the agent involved balancing the complexity of the simulation with the AI's learning capacity. The agent faced challenges in adapting to the dynamic and unpredictable nature of the simulated economy. The reward scheme, while simple, sometimes failed to guide the AI toward long-term optimal strategies, highlighting the need for a more better approach. Training sessions were computationally intensive, requiring hours of training and careful tuning of hyperparameters such as learning rates and batch sizes to achieve good results.

3.5 Conclusion

The development of the simulation and the integration of Stable Baselines3 proved to be both complex and time-intensive. While the AI demonstrated the potential to achieve impressive outcomes, its inconsistent performance underscored the challenges of creating a robust reward system. The scale and complexity of the simulation, combined with the need for fine-tuning, consumed a significant portion of the project timeline, far more than initially expected.

Given these challenges, I decided to shift my future focus to other aspects of the project, such as enhancing the simulation, improving the AI's capabilities, and incorporating extensions like multi-agent interactions and customisable agents.

Chapter 4

Extensions

4.1 Improvement of the Simulation

The simulation underwent enhancements to improve both stability, scale and AI compatibility. One major update was the introduction of water as a new resource, complementing the existing ones (food, fuel, and clothes). This addition added a realistic dimension to the simulation, as individuals now required water daily alongside other resources. Water consumption and production rates were again carefully calibrated to integrate seamlessly with the overall resource dynamics. Factories producing water followed similar economic rules, including wages and production outputs, ensuring the new resource aligned with the simulation's existing structure.

To address challenges in AI training, I eliminated randomness from the simulation, which had previously impeded the AI's learning process. Features such as reproduction rates, laziness factors, and resource consumption fluctuations were stabilised. For example, reproduction rates were fixed at zero, and consumption rates were standardised for consistency across all simulations. These changes created a predictable and stable environment, enabling the AI to identify patterns and develop strategies more effectively.

4.2 Improvement of the AI

The AI component of the project saw further refinement, particularly in its reward system. The original reward structure, which primarily focused on net worth increases, was replaced with a more well rounded approach. The new system incorporated several factors: penalties for excessive inventory accumulation, transaction costs for buying and selling resources, and bonuses for maintaining a diverse portfolio. These changes encouraged the AI to adopt efficient and realistic resource management strategies, aligning its behaviour more closely with real world trading strategies.

Additionally, the AI was rewarded for achieving balance in its inventory. A portfolio diversity bonus incentivised the AI to manage resources dynamically, ensuring it avoided specialising in just one resource and instead adapted to the changing market conditions. These updates also included penalties for exceeding inventory thresholds, ensuring that the AI's strategies did not focus solely on hoarding.

The training process was significantly improved through careful tuning of hyperparameters such as learning rates, batch sizes, and reward weighting. Using these changes, this allowed the AI to achieve dramatically improved results, with one session resulting in a peak net worth of over \$700,000—far exceeding the initial performance of around \$180,000. These results highlighted the effectiveness of the new reward system and demonstrated the AI's capability to adapt and succeed in a competitive economic environment.

4.3 Integrating Multiple Agents

One of the more complex extensions was the integration of multiple agents into the simulation. Initially, this posed a technical challenge, as Stable Baselines3 does not natively support simultaneous multi-agent interactions, which was something I missed in the research phase of this project. To address this limitation, I implemented a sequential interaction mechanism. In this approach, each agent took turns interacting with the environment, and their states were saved and reloaded between actions. While agents did not operate concurrently, they still influenced the shared environment, effectively simulating multi-agent dynamics.

This setup enabled agents to compete for resources and adapt to a dynamic market influenced by the actions of others. Although the sequential design introduced constraints, it provided valuable insights into multi-agent behaviour, such as competition and market saturation. The interaction between agents added a strategic layer to the simulation, making it more complex and engaging. The results of each agent were saved to their own respective CSV and txt files, with the winner of the run being marked in t he CSV file.

4.4 Customisable Agents

To further enhance the AI, I implemented customisable agents by allowing parameterisation of their behaviour. Each agent was configured with three key attributes: risk-taking levels, portfolio diversity preferences, and investment horizons. These parameters were incorporated into the environment by passing them as inputs to the agent's decision-making logic.

The implementation began with the modification of the action and reward systems to accommodate these attributes. For risk-taking, I adjusted the agent's willingness to spend or sell based on its parameter value, scaling its aggressiveness in actions such as resource purchases and sales. For instance, high-risk agents allocated larger portions of their cash to purchasing resources, whereas low-risk agents retained higher reserves for stability. This behaviour was implemented by dynamically weighting the agent's decision factors based on its risk profile.

Portfolio diversity preferences were encoded as bonuses or penalties in the reward function. Agents with a higher diversity preference received additional rewards for maintaining balanced inventories, encouraging them to interact with a variety of markets. To implement this, I computed the number of unique resources held by the agent and factored this into the final reward calculation.

Investment horizons influenced the reward system by adjusting the weighting of short-term versus long-term gains. Short-horizon agents focused on immediate profitability, with rewards scaled to prioritise rapid returns, while long-horizon agents were rewarded for strategies that built wealth gradually over time. This was achieved by tuning the decay rate of rewards based on the horizon parameter, which was integrated into the episode's reward aggregation.

The implementation of these features required careful balancing of the parameter ranges to ensure the agents could operate effectively within the simulation. By incorporating these attributes directly into the decision-making and reward systems, I successfully enabled agents to adopt distinct strategies.

4.5 Conclusion of Extensions

The extensions significantly enhanced the simulation and AI, transforming the project from a basic economy model into a more complex platform model supporting multi-agent interactions. The improvements in simulation stability, reward systems, and agent customisation not only increased the AI's performance but also expanded the depth and realism of the economic environment.

Chapter 5

Conclusion

Reflecting on any project is vital for understanding its successes and areas for improvement. In this section, I will discuss what went well during the development of this project and what I would approach differently if given another opportunity.

5.1 Things I Think Went Well

Overall, I consider this project a success. The simulation, in particular, was a major achievement. It required significant effort to implement and refine, but the end result was a robust system capable of producing realistic economic trends. The resource price movements and market dynamics were comparable to graphs one might observe in a stock exchange, reflecting real-world behaviours.

The AI component exceeded my expectations. As my first attempt into developing a learning AI, I am proud of the results it achieved, particularly in navigating the complexities of the simulation and achieving impressive economic performance. This project now serves as a valuable addition to my portfolio, showcasing my ability to design and implement an interactive system combining simulation and AI. Beyond this, it provided me with invaluable insights into the development process, laying a strong foundation for future potential projects in this field.

5.2 Things I Would Do Differently

Looking back, there are several aspects I would approach differently with the knowledge I have now. The primary consideration would be narrowing the scope of the project. The economy simulation proved highly complex, making it challenging for the AI to achieve consistent results without extensive training and refinement.

If I were to start over, I would focus on one aspect of the project rather than attempting both simulation and AI development. For instance, I might train the AI on a simpler game, such as Snake or Space Invaders, or use a premade example. This approach would have reduced the scale of the problem, enabling quicker training and more opportunities to explore advanced AI techniques. Alternatively, if I had chosen to concentrate solely on the simulation, I could have developed a more detailed and scalable model without the constraints of integrating AI. This focus would have allowed for a deeper exploration of economic mechanics, creating an even more realistic and comprehensive simulation.

5.3 Final Thoughts

In conclusion, I am highly satisfied with the outcome of this project. The process of designing and implementing both the simulation and the AI required a significant investment of time and effort. While there is room for further refinement in both components, I am proud of what I achieved. This project has not only provided a strong portfolio piece but has also expanded my understanding of AI and simulation development.

Bibliography

[1] Farama Foundation. Gymnasium Documentation, n.d. [Accessed: 6-Dec-2024].

[2] Pygame Development Team. Pygame Front Page — pygame v2.0.0.dev15 documentation, n.d. [Accessed: 6-Dec-2024].

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[4] Stable Baselines Team. Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations — Stable Baselines3 1.2.0a2 documentation, n.d. [Accessed: 6-Dec-2024].