Chloe Tu
ITAI 2373 – Natural Language Processing
Professor Anna Devarakonda
November 20, 2025

# Lab 03 AI Game Master - Designing Training Grounds for Intelligent Systems

## 1. Game Characteristics: The Challenge of Dota 2 - OpenAI Five

I chose to study OpenAI Five. This AI plays the video game Dota 2. I learned that Dota 2 is one of the hardest challenges for computers. It is much harder than Chess or Go. It mimics the messiness of the real world.

- **State-Space Complexity:** The "board" in Dota 2 is huge. The number of possible situations is massive. In Chess, there are 64 square meters. In Dota 2, there is a large, continuous map. There are ten heroes playing at once. There are dozens of other units. There are over 100 unique heroes to choose from. Each hero has different abilities. There are hundreds of items. This creates a state space that is almost infinite. The AI has to track the exact location of every unit. It has to track their health. It has to track their mana. It has to do this every single millisecond.
- **Decision Complexity:** The AI has to make choices constantly. These are not simple turns like in a board game. The AI plays in real-time. It has to choose where to move its hero. It has to choose which enemy to attack. It has to decide when to use a spell. It has to decide when to use an item. I found out that the AI makes these decisions every four frames. That goes by extremely fast. A single game lasts about 45 minutes. This means the AI makes tens of thousands of decisions in one game. It has to plan for the future while fighting in the present.
- **Information Availability:** This is the biggest problem for the AI. In Chess, I can see all the pieces. In Dota 2, there is a "fog of war." This fog hides the enemy. The AI cannot see the whole map. It only sees what its team sees. This is called "imperfect information." The AI has to guess where the enemy is. It has to predict attacks that it cannot see. This forces the AI to make risky choices. This is very similar to the real world. We rarely know anything in real life.

## 2. Learning Approach of OpenAI Five

I found the learning method of OpenAI Five very interesting. It used a method called Reinforcement Learning. It did not learn by watching humans. It learned by playing against itself.

- **Core Algorithm:** The system uses a specific math formula. This is called Proximal Policy Optimization (PPO). The team at OpenAI created this. It is very stable. It helps the AI learn without crashing. It works well when you have many computers working together. The algorithm helps the AI build a "policy." A policy is a strategy for what to do in a specific situation. The AI improves this policy slowly. It takes small steps to get better.

- **Training Method:** The most important part was "self-play." I think this is a brilliant idea. The AI played games against copies of itself. It started to know nothing. It was played randomly. Over time, one copy would find a winning move. Then the other copies would learn to defend against it. Then they would find a new attack. This created an "arms race." The AI kept getting smarter to beat itself. It did not need human teachers. It only needed the rules of the game. It needed a score to know if it won or lost.

- **Neural Network Architecture:** The brain of the AI is a neural network. Each of the five heroes had its own brain. They used a specific type of network called Long Short-Term Memory (LSTM). This gives the AI a memory. It can remember things from the past. This is crucial for the fog of war. If the AI sees an enemy for one second, it remembers them. Even if the enemy walks into the fog, the AI knows they are there. The LSTM helps the AI make plans based on old information.

3. **Training Process & Scale**

The size of this project shocked me. The amount of computer power was enormous.

- **Hardware:** The training ran on a massive system. It used 128,000 processor cores. It used 256 very powerful graphics cards. This is more powerful than a standard supercomputer. This power allowed the AI to play thousands of games at the same time.

- **Experience Volume:** The speed of learning was incredible. The AI played 180 years of gameplay every single day. It did this for ten months. By the end, it had played for 45,000 years. No human can live that long. This massive experience allowed it to see every possible situation. It learned strategies that humans have never seen.

- **Progression:** The team tested the AI in stages. First, it played against amateur humans. They won easily. Then it played against semi-pros. It won again. Finally, it played against the world champions, team OG. It beat them in a live match. This proved that the training worked.

4. **Breakthrough Moments & Emergent Strategies**

The AI did things that surprised everyone. Because it taught itself, it did not follow human rules.

- **Hyper-Rational Prioritization:** Humans play with emotion. We want to stay alive. OpenAI Five played math. It calculated the value of everything. It realized that "staying alive" is not the goal. Winning is the goal. Sometimes, it would let a hero die on purpose. If dying helped the team destroy a tower, it did it. It was cold and calculating. It prioritized the long-term win over short-term safety.

- **Unconventional & Sacrificial Tactics:** One famous move was "buyback." In the game, you can pay gold to come back to life. Humans save this for emergencies. The AI used it constantly. It realized that being on the map was worth more than gold. It bought back instantly to keep fighting. It also used "instant crowd control." It could disable an enemy from the exact moment they appeared. This was faster than human reflexes.

### 5. Real-World Transferability

I believe the skills from Dota 2 are useful for real jobs. The game is just a simulation of a complex system.

1. **Automated Logistics & Supply Chains:** A delivery company is like a Dota team. You have trucks (heroes). You have packages (resources). You have traffic (enemies). An AI could manage a whole fleet of trucks. It could reroute them in real-time. It could work together to deliver packages faster.

2. **Smart Grid Management:** The electric grid is very complex. It has many parts. An AI could manage the flow of power. It could balance the load to prevent blackouts. It could handle unpredictable energy from solar or wind.

3. **Large-Scale Robotics:** In a factory, robots need to work together. They need to avoid hitting each other. They need to coordinate tasks. The teamwork from OpenAI Five is perfect for this. Robots could learn to build complex things without human help.

### 6. Connecting the Case Study to City Flow Orchestrator

My study of OpenAI Five proves that my game idea will work. I am designing a traffic training game called "City Flow Orchestrator." The problems with traffic are the same as in Dota 2. Traffic is messy. It happens in real time. It involves many agents.

The success of OpenAI Five shows that "self-play" works. I can train a traffic AI by letting it play against traffic simulations. The AI will learn to coordinate traffic lights. It will teach you to manage lanes. It will discover new ways to stop traffic jams. OpenAI Five proves that a computer can master a chaotic system if it is practiced enough. This is the foundation for my project.

## Game Design Document: City Flow Orchestrator

**Project: AI Game Master - The Perfect Training Game**

### 1. Introduction & Core Concept

City Flow Orchestrator is a simulation game. It is a training sandbox for AI. It teaches an intelligent system for how to manage city traffic. The game simulates a digital version of a major city. It presents the AI with hard traffic problems.

The player does not drive cars. The player acts as the "City Brain." The player controls the traffic lights. The player controls the speed limits. The player controls the lane directions. The goal is to fix traffic jams. The game starts simply and gets very hard. It mimics real-world chaos.

### 2. AI Learning Objectives

The purpose of the City Flow Orchestrator is to train an AI for the real world. The learning objectives are:

1. **Master Signal Timing:** The AI must learn to time traffic lights perfectly. It must learn to read about the flow of cars. It must minimize the time cars spend waiting at red lights.
2. **To Develop Adaptive Policies:** The AI must learn to change its plans. It cannot use the same timing for rush hour and midnight. It must observe the history of traffic. It must predict jams before they happen.
3. **To Execute Crisis Management:** The AI must learn to handle emergencies. It must learn what to do during a crash. It must learn how to clear a path for an ambulance. It must react instantly to unexpected events.

## 3. Design Requirements Analysis

This section explains how my game meets the requirements for training an AI.

### 3.1. Clear Objectives & Winning Conditions

The goal of the AI is to get a high "Flow Score." This score is based on math. It measures how well the city is moving.

- **Primary Objective (Efficiency):** The AI must minimize "Wait Time." This is the total time cars sit still. A lower number is better. This is the most important metric.
- **Secondary Objective (Safety):** The AI must keep the "Accident Rate" at zero. If the AI makes cars go too fast, they crash. This lowers the score.
- **Tertiary Objective (Emergency Response):** The AI must maximize the "Ambulance Speed." When an ambulance appears, it must be taken to the hospital fast.

**Winning Condition:** The AI wins the level if the Flow Score stays high for a full simulated day. It must handle the morning rush and the evening rush without gridlock.

### 3.2. Meaningful Choices

The game gives AI specific tools. These tools are the choices it makes. These choices teach AI important skills.

- **Phase Timing vs. Fixed Timers:** The AI can choose how long a light stays green. It learns that fixed timers are bad. It teaches that dynamic timing is better. It learns to give more green time to the busy road.
- **Lane Reversal vs. Static Lanes:** The AI can switch the direction of the middle lane. It learns to use this for rush hour. In the morning, it opens more lanes into the city. In the evening, it opens more lanes out of the city.
- **Rerouting vs. Direct Path:** The AI can suggest new routes to cars. It learns to spread the traffic out. If the highway is full, it sends cars to the side of the streets. This teaches load balancing.

### 3.3. Escalating Complexity

I designed the game as a curriculum. It starts easily. It gets harder step by step. This ensures that the AI learns the basics first.

- **Stage 1 (Simple):** The scenario is "Midnight." There are very few cars. There are no accidents. The AI focuses on learning how traffic lights work.

- **Stage 2 (Intermediate):** The scenario is "Rush Hour." There are thousands of cars. The AI must use adaptive signals. It must manage the volume.

- **Stage 3 (Complex):** The scenario is "The Crash." A car breaks down in the main lane. This creates a bottleneck. The AI must use rerouting to fix it.

- **Stage 4 (Crisis Simulation):** The scenario is "The Blizzard." The roads are slippery. Cars move slowly. Accidents happen easily. The AI must lower speed limits. It must be very careful. It must coordinate the whole city to prevent a total freeze.

### 3.4. Measurable Progress

I can track the learning of AI easily. I use a dashboard to show the numbers.

- **Quantitative Metrics:** The dashboard shows the "Average Speed" of all cars. It shows "Total Idling Time." It shows "Flow Score." I can see a graph. If the line goes up, the AI is learning. I can compare the AI's score to a human's score.

- **Qualitative Analysis:** The game keeps a log. It writes down every decision the AI makes. I can read the log. It says things like "Extended green light because of the bus." This helps me understand why the AI made a choice. It helps me see the "thought process."

### 3.5. Safe Failure Space

City Flow Orchestrator is a "digital twin." It is a safe place to fail. This is vital for training.

- **Catastrophic Failure is a Learning Tool:** The AI can cause a massive traffic jam. It can gridlock the whole city. In the real world, this would be a disaster. In the game, it is just data. The AI learns that its strategy failed. It gets a negative score. It adjusts its weight for the next game.

- **High-Risk Experimentation:** The AI can try crazy things. It might turn all lights red to see what happens. It might open all lanes one way. This freedom allows it to discover new tricks. It might find a pattern that humans missed. It can do this without hurting anyone.

4. **Prototype vs. Vision: From Implementation to Intention**

My current design is a prototype. The final vision is much bigger. This table shows the difference.

| **Feature** | **What The Prototype Implements** | **The Intended AI Vision (Full-Scale System)** |
|---|---|---|
| **Pathfinding** | A simple algorithm on a small map. It calculates the shortest distance. | A massive neural network. It calculates routes for millions of cars. It uses real-time data from the whole city. |
| **Adaptive Signals** | A basic rule-based agent. It changes lights based on how many cars are waiting. | A Multi-Agent Reinforcement Learning (MARL) system. Every traffic light is an AI agent. They talk to each other. They create a "Green Wave" across the city. |
| **City Score** | A simulated score based on game rules. | A complex Key Performance Indicator (KPI). It uses real GPS data. It uses pollution sensors. It uses accident reports. |

## 5. Roadmap to Real-World Deployment

I have a plan to move this from a game to the real world. This will happen in three phases.

Phase 1: Data Integration & Digital Twin Fidelity

- Action: I will import real maps from OpenStreetMap. I will use real traffic data from city cameras.
- Goal: To create a perfect copy of the city. The simulation must look exactly like reality. If it rains in the real city, it rains in the game.

Phase 2: Scaling and Training Advanced AI Models

- Action: I will upgrade the AI brain. I will replace simple rules with Deep Reinforcement Learning.
- Goal: To run millions of simulated days. This is "self-play." The AI will experience every possible disaster. It will learn to handle hurricanes. It will learn to handle sporting events. It will become an expert before it controls a real light.

Phase 3: Hardware-in-the-Loop (HIL) Simulation & Safety Validation

- Action: I will connect the AI to a real traffic light controller in a lab.

- Goal: To test safety. I need to make sure digital commands work on real hardware. I need to verify that the AI never creates a dangerous situation.

## 6. Conclusion

City Flow Orchestrator is more than a game. It is a scientific tool. It demonstrates a safe way to train complex AI systems. Managing a city is too dangerous for trial and error. By providing a safe sandbox, I can train an AI to be perfect. The prototype proves that the core principles work. The game teaches AI to prioritize efficiency and safety. This paves the way for smarter, cleaner, and faster cities in the future.

**Visual Representation: Gameplay & AI Learning Loop**

This flowchart outlines how the game works. It shows the loop of the AI making decisions.

```
+-----------------------------+
|      START OF GAME        |
+-----------------------------+
           |
           v
+-----------------------------+
|   PLAYER CONFIGURATION     |
| - Select City Map          |
| - Set Weather (Rain/Snow)   |
| - Choose Difficulty Level   |
+-----------------------------+
           |
           v
+-----------------------------+
|  START SIMULATION CYCLE    |
+-----------------------------+
           |
           v
/-----------------------------\
|     REAL-TIME AI LOOP      |
|                      |
| 1. Observe Traffic State    |
|    (Car positions, Speeds)  |
|                      |
| 2. Calculate Congestion     |
|    (Identify Bottlenecks)   |
|                      |
| 3. AI Systems Execute:      |
|    - Change Signal Timing   |
|    - Reroute Traffic        |
|                      |
| 4. Log Decision to Memory   |
|                           |
\-----------------------------/
            |
            v
+-----------------------------+
| [Is There an Incident?]     | --(No)--> [Back to Loop]
+-----------------------------+
            | (Yes)
            v
+-----------------------------+
|    CRITICAL RESPONSE MODE   |
| (e.g., Clear Ambulance Path)|
+-----------------------------+
            |
            v
+-----------------------------+
| [Simulation Timer Ends?]    | --(No)--> [Back to Loop]
+-----------------------------+
            | (Yes)
            v
+-----------------------------+
|   END OF SIMULATION CYCLE   |
|  - Calculate Final Flow     |
|    Score                    |
+-----------------------------+
            |
            v
+-----------------------------+
|    EVALUATE PERFORMANCE     |
|  - Analyze Wait Times       |
|  - Count Accidents          |
|  - Review Logs              |
+-----------------------------+
            |
            v
+-----------------------------+
|     LEARNING & UPDATE       |
| (AI adjusts its brain based |
|  on the score)              |
+-----------------------------+
            |
            v
+-----------------------------+
|    Ready for Next Cycle     |
+-----------------------------+
```

7. **Comparative Analysis & Ethical Considerations**

**Effective Game Designs** After looking at the project, I realized that the best training games have clear math goals. Games that are too vague do not work for AI. My traffic game works because time is a number. Less time in traffic is a clear win.

**Entertainment vs. Training** I found a big difference between fun games and training games.

- **Fun Games:** They try to be fair. They want the player to have a good time.
- **Training Games:** They are not fair. In my traffic game, I can create a situation that is impossible to solve. This is good for training. It teaches the AI to minimize damage when it cannot win.

**Ethical Considerations** This is the most serious part. We are training AI in a fake world.

- **The Simulation Gap:** If my traffic AI learns that a car stops instantly in the game, it might try that in real life. But real cars need time to stop. If the AI trusts the game physics too much, it could cause a crash. We must make the game realistic.
- **Reward Hacking:** The AI might find a cheat. If the goal is "Zero Accidents," the AI might turn all the traffic lights red forever. If no cars move, no cars crash. The AI thinks it won, but the city is broken. We must be very careful with the rules we give it.

## Transfer Learning Map: From Game to Reality

This diagram shows how game skills translate to real life. I want to show that playing the game makes the AI ready for work.

| Game Skill / AI Competency | Real-World Application | Example Implementation |
|---|---|---|
| *A Pathfinding with Live Cost** | **Real-Time Navigation & Logistics** | Apps like Google Maps or Waze use this. They reroute users around traffic. Delivery fleets use this to save gas. |
| **Adaptive Signal Control** | **Smart City Traffic Grids** | Traffic lights that use cameras. They see the cars waiting. They adjust the green light time instantly to clear the intersection. |
| **Dynamic Rerouting** | **Emergency Vehicle Preemption (EVP)** | Systems that clear the road for fire trucks. The traffic lights turn green for the fire truck and red for everyone else. |

| | | |
|---|---|---|
| **Scenario Simulation** | **Urban Planning & Disaster Preparedness** | City planners use "Digital Twins." They test what happens if they build a new stadium. They test evacuation routes for hurricanes. |
| **Mission-Based Prioritization** | **Hierarchical Task Management** | An AI learns to prioritize tasks. It learns that saving a life (ambulance) is more important than speed (commuter). |
| **AI Decision Logging** | **Explainable AI (XAI) & Auditing** | In real infrastructure, we need to know *why* an AI made a choice. The logs help engineers trust the system. |

## City Flow Orchestrator: Training AI for Real-World Traffic Management

Presentation Write-Up

### Opening

Imagine your city during rush hour. Imagine thousands of cars stuck in places. Imagine the pollution. Imagine the wasted time. Now imagine an ambulance trying to get through that mess. It is a nightmare. Now, imagine an AI system that sees everything. It sees every car. It controls every light. It clears the road before the jam happens. How do we build this AI? We cannot test it on real roads. That would be dangerous. The answer is City Flow Orchestrator. This is a real-time strategy game. It is the perfect training ground for intelligent traffic systems.

### The Problem: Why Traditional Training Fails

Training AI for traffic is very hard. I identified three main problems:

1. **Real-world testing is too risky.** I cannot experiment with real traffic lights. If the AI makes a mistake, people crash.

2. **The complexity is massive.** There are millions of variables. A single rainstorm changes everything.

3. **Traditional simulations are boring.** They do not push the AI. They do not teach the AI to handle crises.

City Flow Orchestrator solves these problems. It creates a sandbox where failure is okay. It turns traffic management into a high-score game.

**The OpenAI Five Connection**

My research on OpenAI Five proves this works. OpenAI Five mastered the game Dota 2. Dota 2 is just like traffic.

- It has a huge map.
- It has hidden information.
- It requires real-time decisions.
- It requires teamwork.

OpenAI Five succeeded by playing against itself for thousands of years. My project uses the same method. I will let the Traffic AI play thousands of years of traffic simulations. It will teach strategies that humans cannot find.

**How City Flow Orchestrator Trains AI**

My game meets all the requirements for a good training environment.

**1. Clear Objectives** The AI has a simple goal: Maximize the "Flow Score." It must minimize travel time. It must minimize accidents.

**2. Meaningful Choices** the AI has real tools.

- It can toggle adaptive signals.
- It can use dynamic rerouting.
- It can reverse lanes. Each choice teaches a fundamental principle of traffic flow.

**3. Escalating Complexity** I do not throw the AI into the deep end.

- Stage 1 is just learning the lights.
- Stage 2 is rushing hour.
- Stage 3 is an accident scene.
- Stage 4 is a hurricane evacuation.

**4. Measurable Progress** I have a dashboard. I can see the AI getting smarter. I can see the wait times going down. I can read the logs to see its "thoughts."

**5. Safe Failure Space** In my game, the AI can cause a total gridlock. It can fail miserably. This is good! It teaches what you do not do. It teaches boundaries.

**Real-World Transfer**

The skills learned in the game transfer directly to reality.

- **Pathfinding** becomes Google Maps navigation.
- **Adaptive Signals** become Smart City Grids.
- **Dynamic Rerouting** becomes an Emergency Vehicle Preemption.
- **Scenario Simulation** becomes Urban Planning tools.

**Why This Works: The Proof**

OpenAI Five proved that massive self-play works. It developed a "superhuman" performance. It learned to sacrifice small things for the big win. My game applies this to traffic. The AI learns to sacrifice one car's speed to save the whole city from a jam. The core innovation is simple: **Treating traffic management as a game makes it solvable through reinforcement learning.**

**Closing: The Vision**

City Flow Orchestrator is not just a game. It is a blueprint. It is a way to train the systems that will run our future cities. By providing a safe and hard training ground, I can develop an AI that:

- Reduces commute times by 20-30%.
- Clears paths for ambulances in seconds.
- Prevents gridlock during disasters.
- Makes cities breathable again.

The question isn't whether AI can manage traffic. The question is: are we ready to build the gym to train it? **City Flow Orchestrator is that gym.**

**References:**

"OpenAI Five Intro • Greg Brockman." Greg Brockman on Svbtle, 6 Aug. 2018, https://blog.gregbrockman.com/openai-five-benchmark-intro.

Bexley Automotive. "Why Is Speeding in Rain More Dangerous than It Seems? - Bexley Automotive." Bexley Automotive, 2025, www.bexleyauto.com/about-us/blog/why-is-speeding-in-rain-more-dangerous-than-it-seems.

"Gold." Liquipedia Dota 2 Wiki, 2024, https://liquipedia.net/dota2/Gold#:~:text=%E2%80%94%20Alchemist,heroes%2C%20creeps%2C%20or%20buildings.

OpenAI, et al. "Dota 2 with Large Scale Deep Reinforcement Learning." ArXiv:1912.06680 [Cs, Stat], 13 Dec. 2019, https://arxiv.org/abs/1912.06680.

OpenAI. “OpenAI Five.” Openai.com, 2018, https://openai.com/index/openai-five/.

“OpenAI Five Finals.” Openai.com, 20 June 2020, https://openai.com/index/openai-five-finals/.