# Tutorials and Tools

Christiaan Verhoef

2024-10-08

# Table of contents

# Preface

In the world of logistics and serious game development, we often find ourselves juggling complexity and creativity. Here's where OpenWebUI comes in—our trusty tool to bring order and insight to that delightful chaos.

We're not going to bother with installation today (you're clever enough to have that sorted!). Instead, let's jump straight into what matters most: **How to use OpenWebUI to get things done**.

This book is your guide to mastering OpenWebUI, built step by step, with a focus on practical usage. Whether you're optimizing logistics routes or crafting engaging in-game interactions, you'll find ways to streamline your processes, automate tasks, and make your work both efficient and fun.

Ready? Breathe, get comfortable, and let's embark on this journey together. It's going to be an exciting ride.

---

# 1 Introduction

Welcome to the world of OpenWebUI, where creativity meets efficiency! Whether you're a **Serious Game Developer** or a **Logistics Researcher**, OpenWebUI is your new secret weapon. It allows you to build smart agents that automate complex tasks, freeing you to focus on innovation and strategy.

But this isn't just about creating bots. It's about crafting intelligent assistants that help streamline your workflow, optimize processes, and reduce the repetitive, mundane tasks that often bog us down. Imagine a team of digital helpers at your disposal, handling everything from resource management in your games to route optimization in logistics.

### 1.0.1 Who Is This Book For?

This book is designed for **Serious Game Developers** and **Logistics and Supply Chain Researchers**—fields that demand innovative, flexible solutions. Whether you're designing interactive NPCs for your latest game or testing complex logistics scenarios, OpenWebUI offers a set of tools to make your job easier and more efficient.

Here's what you'll learn:

- How to **set up and configure your agents**.
- How to **create custom workflows** for both game development and logistics.
- How to **test and refine your agents** to make them more responsive and capable.
- How to **integrate OpenWebUI** into your daily processes for seamless automation.

### 1.0.2 Why OpenWebUI?

OpenWebUI is a modular, flexible platform that grows with your needs. Whether you're working on a small game prototype or managing a massive supply chain, this tool adapts to your projects. The best part? You don't need to be a coding expert to get started. OpenWebUI makes it easy for you to spend less time wrestling with the tech and more time doing what matters most: creating, optimizing, and innovating.

**Reminder**: This book focuses solely on usage. You won't find long installation guides here—we assume you're ready to dive straight into the good stuff!

Let's get started. By the end of this book, you'll have the skills to build agents that work for you, leaving you more time for creativity and strategic thinking. Ready? Let's dive in!

See Knuth (1984) for additional discussion of literate programming.

# 2 The Interface: Your Dashboard of Possibilities

# 3 The Interface: Your Dashboard of Possibilities

When you launch OpenWebUI, the interface is designed to give you full control over the interaction between your agents and their tasks. The layout consists of three main panels: the **left panel** for managing active sessions and templates, the **center panel** for live agent interaction, and the **right panel** for adjusting chat controls and system parameters.

---

### 3.0.1 The Left Panel: Your Workspace and Navigation Hub

The **left panel** organizes your active and archived agent interactions. This section allows you to quickly move between different agent conversations and manage templates.

- **Active Sessions**: All active agent conversations are listed here. Click on a session to open and interact with the agent in real time.
- **Archived Conversations**: A space to revisit old conversations with agents, useful for refining agent performance based on previous tasks.
- **Templates**: Pre-configured agents that you can quickly launch, such as a logistics optimizer or NPC for games.

#### 3.0.1.1 Example: Managing Sessions

1. Click on **Active Sessions** to view any agents you have running. For instance, if you have an agent performing a logistics task, you'll find its current progress and can issue commands directly.
2. To review a past interaction, click on **Archived Conversations** and load up the history of any agent's previous task.
3. If you want to quickly create an agent, click **Templates**, select **Logistics Optimizer**, and watch as the agent template is loaded and ready for interaction.

---

### 3.0.2 The Right Panel: Adjusting Chat Controls and System Parameters

The **right panel** is where you can modify the behavior of your agents in real time by setting the **system prompt**, adjusting various **advanced parameters**, and fine-tuning other controls such as temperature or token limits. These settings directly influence how your agent behaves and responds during conversations.

Here's a breakdown of what each option does:

## 3.1 System Prompt

The **System Prompt** is the initial instruction or context that guides the agent's overall behavior. It sets the tone for the agent's interactions and is one of the most critical settings.

- **Example**: `"You are a logistics optimization agent. Your goal is to minimize delivery time by adjusting routes based on real-time traffic data."`

  **Effect**: This ensures the agent's responses stay focused on optimizing routes. Changing the system prompt to include more detailed instructions, such as considering fuel efficiency or avoiding toll roads, will shift the agent's decision-making process.

---

## 3.2 Temperature

**Temperature** controls the randomness or creativity of the agent's responses. It affects how likely the agent is to deviate from predictable answers.

- **Range**: `0.0` to `1.0`
  - A **lower value** (closer to `0.0`) makes the agent more deterministic, meaning it will choose the most predictable and straightforward responses.
  - A **higher value** (closer to `1.0`) increases the variability in responses, allowing the agent to explore less obvious possibilities.
- **Example**:
  - At `0.2`, the agent will give highly predictable and consistent answers. Ideal for tasks like logistics optimization where precision is key.
  - At `0.8`, the agent will provide more creative, varied responses, useful when testing dynamic NPC dialogue in games or brainstorming creative solutions.

**Effect**: Higher temperatures introduce more randomness. For a logistics agent, using a high temperature could lead to more unconventional route suggestions, which might not always be optimal. Lowering the temperature ensures the agent sticks to tried-and-tested routes.

---

## 3.3 Top K

**Top K** limits how many options the agent considers at each step of its response generation. It controls how wide the agent's "vocabulary" is during interactions.

- **Range**: `1` to infinity (higher values mean more possible words to choose from)
  - A **lower value** (e.g., `10`) restricts the agent to a smaller, more focused set of choices.
  - A **higher value** (e.g., `50`) allows the agent to consider a broader range of possible next words or actions.
- **Example**:
  - Set **Top K** to `10` for a logistics agent. This ensures the agent sticks to the most reliable, often-used routes.
  - Set **Top K** to `100` for an NPC in a game to allow for more varied dialogue options, creating a more dynamic conversation.

**Effect**: Lower **Top K** values make the agent more predictable by narrowing its choices. In logistics, a low **Top K** ensures the agent only considers the most efficient routes. Higher **Top K** values allow the agent to explore more creative options, useful in contexts where variety is desirable.

---

## 3.4 Top P

**Top P** (nucleus sampling) controls how the agent selects from the top `P%` of most likely responses. This method can reduce reliance on overly common responses.

- **Range**: `0.0` to `1.0`
  - A **lower value** (e.g., `0.1`) means the agent will only select from the top 10% of the most likely responses.
  - A **higher value** (e.g., `0.9`) allows the agent to consider a much broader range of responses.

- **Example**:
  - **Top P** of `0.1` forces the agent to choose the safest, most common responses. Great for logistics where efficiency and reliability are key.
  - **Top P** of `0.8` allows for more variability, letting the agent experiment with routes or responses outside the norm.

**Effect**: Lower **Top P** values restrict the agent to the safest options, which works well for practical applications like logistics. In creative tasks, like generating dialogue for NPCs, a higher **Top P** value might produce more interesting and varied results.

---

## 3.5 Frequency Penalty

**Frequency Penalty** controls how much the agent penalizes words or phrases that it has already used. This prevents repetitive responses and encourages more varied output.

- **Range**: `0.0` to `2.0`
  - A **higher value** (e.g., `1.5`) makes the agent less likely to repeat words, encouraging it to find new ways of responding.
  - A **lower value** (e.g., `0.2`) allows the agent to repeat words more freely.
- **Example**:
  - Set **Frequency Penalty** to `1.2` for an NPC agent to avoid repetitive dialogue in-game conversations.
  - Set **Frequency Penalty** to `0.5` for a logistics agent, so it can repeat key terms like route names or instructions if necessary for clarity.

**Effect**: A higher penalty is useful in creative tasks to avoid repetitive answers, like generating dynamic NPC dialogue. In logistics or similar applications, setting a lower penalty ensures that important details (like road names) are repeated when necessary.

---

## 3.6 Max Tokens

**Max Tokens** limits how long the agent's responses can be by setting a cap on the number of tokens (which typically represent words or parts of words).

- **Range**: Any positive integer
  - A **lower value** (e.g., `50`) ensures the agent's responses are short and concise.
  - A **higher value** (e.g., `200`) allows for more detailed and elaborate responses.
- **Example**:
  - Set **Max Tokens** to `100` for logistics tasks to ensure the agent provides concise but detailed route recommendations.
  - Set **Max Tokens** to `150` for NPC agents in games to allow for more detailed explanations or conversations.

**Effect**: A lower **Max Tokens** value ensures responses remain short and focused, useful for tasks where brevity is important, like logistics. Higher values allow for longer, more elaborate explanations, great for creative writing or character dialogue.

---

## 3.7 Mirostat (Eta and Tau)

**Mirostat** is an advanced control mechanism that adjusts creativity in real-time as the agent generates text.

- **Eta**: Controls how aggressive the adjustment is.
  - Higher values mean more aggressive changes in creativity.
- **Tau**: The target perplexity, or how unpredictable the agent should be.
- **Example**:
  - Set **Mirostat Eta** to `1.0` and **Mirostat Tau** to `5` for a logistics agent to keep its responses stable and predictable.
  - Increase **Tau** for an NPC agent in a game to make dialogue more varied and unpredictable.

**Effect**: **Mirostat** helps fine-tune the unpredictability in long conversations. In logistics, using a low **Tau** ensures the agent stays consistent over longer interactions, while a higher **Tau** in a game allows for more dynamic and evolving conversations.

---

## 3.8 Use of Advanced Parameters in Specific Scenarios

### 3.8.1 Logistics Use Case:

- **System Prompt**: `"Optimize delivery routes with minimal delays, considering traffic and package urgency."`
- **Temperature**: Set to `0.4` for stable and predictable results.
- **Top K**: Set to `20` to limit route options to the most common choices.
- **Max Tokens**: Set to `100` to keep the responses concise.

### 3.8.2 Game NPC Use Case:

- **System Prompt**: `"You are a game character. Respond to player questions with wit and humor."`
- **Temperature**: Set to `0.8` for more creative and spontaneous replies.
- **Frequency Penalty**: Increase to `1.5` to avoid repetition in dialogue.
- **Max Tokens**: Set to `150` for more in-depth interactions.

#### 3.8.2.1 Example: Fine-Tuning a Logistics Agent

Let's say you have an agent designed to optimize delivery routes, and you want to adjust how it prioritizes efficiency.

1. **System Prompt**: In the **System Prompt** field, type:
    - `"Prioritize optimizing delivery routes based on shortest time while considering real-time traffic delays."`
2. **Temperature**: Set the **Temperature** to `0.5`. This will make the agent's responses balanced—not too random but still adaptable to complex routing decisions.
3. **Top K & Top P**: Set **Top K** to `50` and **Top P** to `0.9` to allow the agent to consider a wide range of potential routes while still being focused on the most likely options.
4. **Max Tokens**: Set **Max Tokens** to `150` to keep the responses concise but still detailed enough for logistics decisions.
5. **Frequency Penalty**: Increase the **Frequency Penalty** to `0.6` to ensure the agent avoids repeating similar route suggestions over and over.

---

### 3.8.3 The Center: Interacting with Agents in Real-Time

The **center panel** is where you have direct conversations with your agents. This panel is where the agent's responses appear, and you can issue new commands or monitor task progress.

#### 3.8.3.1 Example: Running Your Configured Agent

After adjusting the settings on the right panel, it's time to see the results in the center panel.

1. In the **center panel**, type:
   `"Optimize delivery routes for trucks in the downtown area, considering current traffic conditions."`
2. Press **Enter** and observe the agent's response, which should consider your new system prompt and settings.
3. Simulate a traffic delay by typing:
   `"There is a traffic jam on Main Street. Recalculate delivery routes."`
4. The agent will adjust the routes in real-time, providing alternative suggestions based on the adjusted parameters like **Temperature** and **Top K**.

# 4 Crafting Your First Agent: Getting Started with Bots

- Creating a new agent: Step-by-step
  - Naming your agent: Keep it descriptive
  - Assigning roles: Logistics vs. Game development applications
  - Basic agent instructions: Starting simple
  - A walkthrough for a first-time agent

# 5 Refining Agent Behavior: Making Them Smarter

- Adjusting parameters for more effective performance
  - Teaching your agent new tricks: Customization
  - Troubleshooting common agent issues
  - Using feedback loops to improve agent responses

# 6 Testing Agents: Trial and Error for Perfection

- Setting up test cases for logistics scenarios
  - Running in-game simulations with NPC bots
  - Gathering feedback: How to observe agent behavior
  - Tweaking based on test results: Iterative improvements

# 7 Testing Agents: Trial and Error for Perfection

- Setting up test cases for logistics scenarios
- Running in-game simulations with NPC bots
- Gathering feedback: How to observe agent behavior
- Tweaking based on test results: Iterative improvements

# 8 Integrating OpenWebUI into Your Workflow

- Embedding agents into serious games
- Applying OpenWebUI to supply chain and logistics research
- Workflow automation: Using bots to handle repetitive tasks
- Real-world use cases: From theory to practice

# 9 Real-World Examples and Case Studies

- Optimizing game dialogues with NPC agents
  - Using agents to simulate supply chain disruptions
  - Logistics route optimization: Step-by-step case study
  - Applying OpenWebUI in academic research

# 10 Scaling and Maintenance: Keeping Your Agents Up-to-Date

- Scaling agents for larger projects
  - Monitoring agent performance over time
  - Regular updates and maintenance for optimal performance
  - How to retire outdated agents

# 11 Final Thoughts and Future Potential

- The evolving role of AI in logistics and game development
  - Expanding beyond the basics: What's next for you?
  - How to stay updated on new OpenWebUI features
  - Encouragement to keep experimenting and learning

# 12 Summary

In summary, this book has no content whatsoever.

# References

Knuth, Donald E. 1984. "Literate Programming." *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.