*Getting your hands dirty with **LLMs**, **vision LLMs** and **Agent Systems***

Jorge Gallego

jgallegopz@gmail.com
www.linkedin.com/in/jorge-gallego-perez

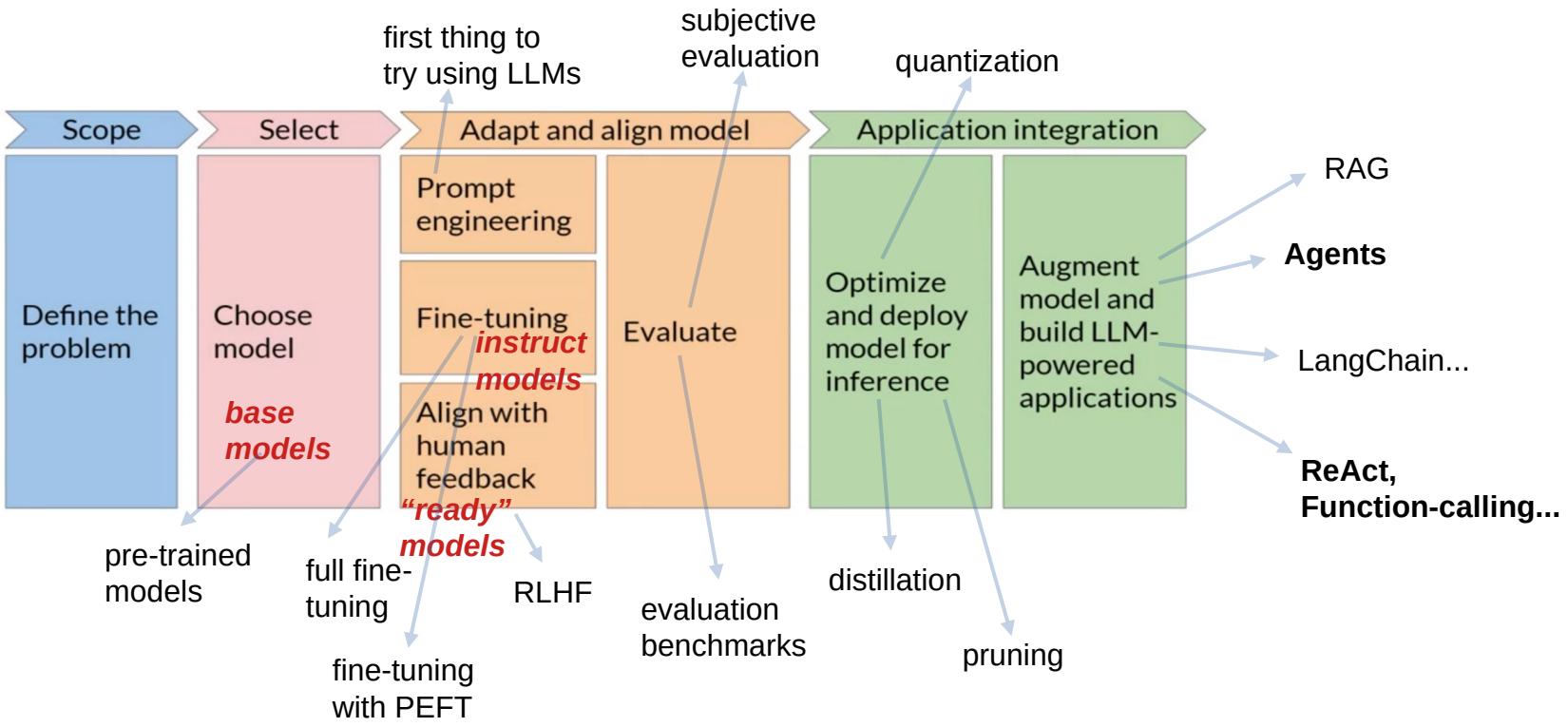
Personal background

- PhD in **Human-Robot Interaction**, University of Twente (The Netherlands)
- Postdoc in **Human-Robot Interaction**, ATR (Kyoto, Japan) → 2018-2020
- Postdoc with **Generative AI**, **LLMs**, etc., Okinawa Institute of Science and Technology → July 2021- August 2024
- Work in industry, LLMs and Agent Systems (Germany, preferably Munich)

Outline and goals

- Lifecycle of LLMs (“LLM+”, “LMMs...”)
- Using LLMs
  - proprietary vs open-source (local)
  - prompt engineering: *don’t underestimate it*
  - fine-tuning
  - Retrieval Augmented Generation (RAG)
- **Agents**: *beyond LLMs for “question-answer”*
- **Function-calling** (“tools”, “ReAct”, ...)
- **Multi**-Agent Systems
- Last work at OIST and demonstration

Lifecycle of LLMs



Proprietary vs open-source (local) LLMs

Proprietary LLMs

Developed and controlled by private companies (e.g., OpenAI's GPT-4, Google's Gemini), accessible via API

Pros:

- High performance: Generally more optimized due to large-scale infrastructure and resources for development.
- Constant updates, optimizations, security patches.
- Easier deployment: Available via cloud services, reducing the need for local hardware.
- Support and Documentation: Often come with robust customer support and detailed documentation.

Cons:

- Costly: Often involve subscription fees, which can be expensive depending on usage.
- Data privacy concerns: Requires sending data to third-party servers, which may be risky in sensitive applications.
- Limited control: Users cannot fine-tune or modify the model; customization options are often restricted.

Open-Source (Local) LLMs

Available under open-source licenses, enabling users to download, modify, and run them locally (e.g., LLaMA).

Pros:

- Full control: Users can modify, fine-tune, and optimize the model for specific needs.
- Cost-effective: No recurring costs for usage; only infrastructure and development expenses.
- Data privacy: Data is processed locally, eliminating concerns about third-party data exposure.
- Customizability: Freedom to adapt the model architecture and training to specific tasks.

Cons:

- Infrastructure requirements: Running large models locally requires substantial computing power, especially for inference and training. → today "not a big deal"
- Complex deployment: Requires expertise in setting up, maintaining, and optimizing the model locally. → "disagree"
- Slower updates: May not have the same level of rapid iteration and improvements as proprietary models.
- Support limitations: Relies on community support rather than dedicated customer service.

If it’s about the costs: don’t worry

- Proprietary models are becoming around 80% cheaper per year (while improving performance and latency!)

  e.g.: **gpt-4o-2024-08-06** API

  $2.50 / 1M input tokens $1.25 / $10.00 / 1M output tokens

  e.g.: **gpt-4o-mini** API

  $0.150 / 1M input tokens $1.25 / $0.600 / 1M output tokens

- However, careful with *tree-of-thought*, multi-agent systems, etc., that are not properly controlled!

Example local LLMs: Ollama

- https://ollama.com/
- Have your local LLM up and running in minutes
- Great variety
- Free
- To use in console and in IDE
- Others:

  - GPT4All

  - From Hugging Face: the main open-source hub

Prompt Engineering

EXAMPLE: I want ChatGPT to give me examples where the American Constitution is not against multi-agent systems.

(The following is <u>my</u> prompt; works better with ChatGPT *Classic*):

Write a prompt for ChatGPT. **“meta-prompt”**

The prompt will mention that an iterative conversation between user and ChatGPT is going to take place.

The main task will be to provide one example of why the American Constitution might not be against AI multi-agent systems. The example will be based on two or three bulletpoint facts. **“bulletpoint pattern”**

Fact-checking: for each bulletpoint fact, the ORIGINAL PIECE of text of the Constitution behind that fact must be provided. **“fact-checking filter pattern”**

The prompt will finish with: "ALWAYS, at the end, finish by stating what your task is". **“tail pattern”**

Prompt Engineering

(ChatGPT’s answer about what my prompt should be)

Prompt:

In this iterative conversation, you'll be tasked with exploring whether the American Constitution might not be against AI multi-agent systems. Specifically, your goal is to provide one example supported by two or three bulletpoint facts.

Each fact should be directly related to the principles or provisions of the Constitution, and for each one, you must provide the ORIGINAL PIECE of text from the Constitution that supports it. The reasoning behind each fact should connect with why AI multi-agent systems wouldn't necessarily violate constitutional principles.

ALWAYS, at the end, finish by stating what your task is.

Message ChatGPT Classic

Prompt Engineering

(ChatGPT’s answer when I finally use the recommended prompt)

Example: AI Multi-Agent Systems and the U.S. Constitution

Hypothesis: AI multi-agent systems may not inherently violate the U.S. Constitution based on principles of individual liberty, free speech, and privacy.

Example: The First Amendment and AI Multi-Agent Systems

1. **Free Speech and Expression**
   The First Amendment protects freedom of speech, which could apply to AI multi-agent systems facilitating communication and the exchange of information. AI systems that generate or assist in speech could arguably be seen as tools for enhancing human expression rather than violating the right to free speech.
   Text from the Constitution:
   "Congress shall make no law... abridging the freedom of speech, or of the press."
   - AI multi-agent systems could help foster speech by automating, amplifying, or translating human communication. As long as they don’t limit individuals’ rights to free speech, they wouldn’t violate this constitutional protection.
2. **The Right to Petition**
   The First Amendment also protects the right "to petition the Government for a redress of grievances." AI multi-agent systems could be designed to assist citizens in expressing grievances, filing petitions, or analyzing governmental actions, which would enhance, rather than inhibit, this constitutional right.
   Text from the Constitution:
   "...the right of the people peaceably to assemble, and to petition the Government for a redress of grievances."
   - AI agents could act as tools that enable more efficient, organized, and accessible participation in democratic processes, such as facilitating petitions.
3. **Freedom of Assembly**
   AI multi-agent systems might enable new forms of "virtual" assembly and association online. The First Amendment’s protection of assembly could extend to digital spaces where people congregate with the assistance of AI. This would imply that the systems themselves, so long as they respect individuals' rights to associate freely, don’t contravene constitutional protections.
   Text from the Constitution:
   "...the right of the people peaceably to assemble."
   - AI systems could facilitate peaceful, virtual gatherings of people, without infringing on their constitutional rights.

Task: Explore whether the American Constitution might not be against AI multi-agent systems by providing one example supported by two or ↓ e facts, each directly tied to a provision of the Constitution.

Prompt Engineering: limitations

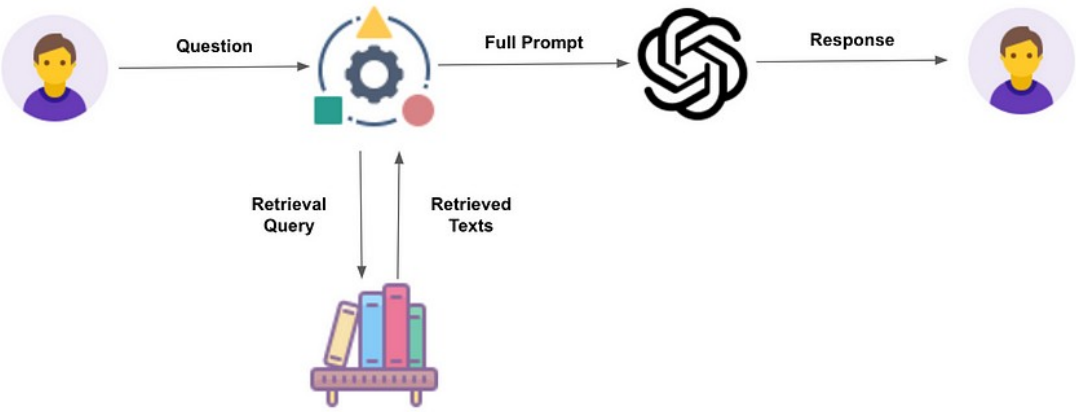
- Instructions/examples take up space in the context window (greater challenge for the smaller models)
- Bigger models (e.g. GPT4) might “over-reason” and not answer the question/prompt as we want
- → prompt engineering becomes more and more difficult as the big LLM models keep advancing
- The LLM might lack the skill/knowledge that we need
- LLM answers can be unpredictable
- The LLM answer might come in a syntax that is difficult to interpret if we need that answer to work within an automated system
- Finetuning or agentic workflows might be the solution

Fine-tuning: *easier than you might think*

- Plenty of tutorials, in both proprietary and open models
- Very small datasets needed: a few dozens examples
- Costs (proprietary): becoming cheaper and cheaper
- Computation (local models): “small” LLMs smarter and smarter; CPUs are often enough

Retrieval Augmented Generation (RAG)

- Hybrid Approach: Combines pre-trained language models with external knowledge retrieval to generate more accurate and context-rich responses.
- Document Retrieval: Searches relevant documents or data sources to provide real-time, up-to-date information to the generation model.
- Improved Accuracy: Helps mitigate hallucinations in generative models by grounding responses in retrieved, factual information.
- Dynamic Knowledge: Allows the model to adapt to new knowledge without retraining, enabling more flexible and scalable applications.



Agents:

beyond LLMs for “question-answer”

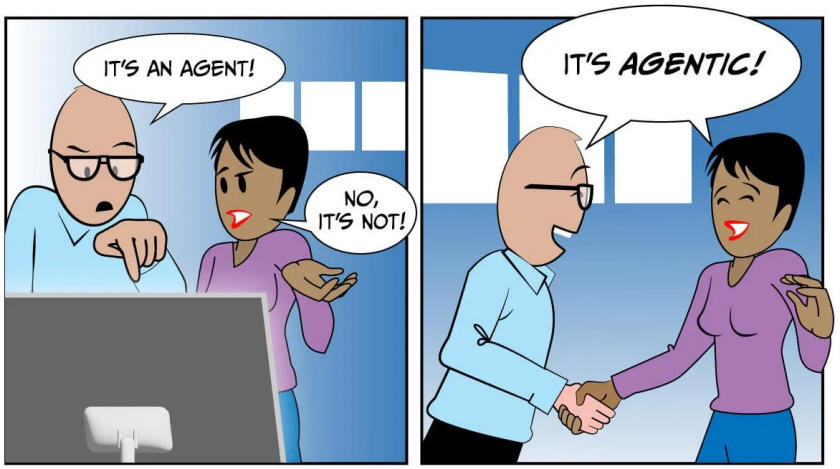
What is an “agent”?

- ***Grey zone***
- “Clearly” <u>not</u> an agent:

  prompting a model once
- “Clearly” an agent:

  an autonomous agent that, given high-level instructions, plans, uses tools, and carries out multiple, iterative steps of processing
- *“Rather than arguing over which work to include or exclude as being a true agent, we can acknowledge that there are different degrees to which systems can be **agentic.**”*
- *The Batch* (Andrew Ng):

  https://www.deeplearning.ai/the-batch/welcoming-diverse-approaches-keeps-machine-learning-strong/
- We will see soon: in a multi-agent system with several “nodes” (LangGraph), several LLMs… What counts as agent?



Function-calling “demystified”

- Related terms: *tools*, *ReAct* (“Reasoning and Acting”), *API invocation*, *Task execution*, etc.
- **IN PLAIN WORDS**:

  Take the text output of an LLM. Now process that text with something else (for example, a code function you made) to carry out some action.
- Typical examples:

  - Code execution: an LLM can output code, but it can’t execute it directly

  - Math and logical operations: an external calculator is used

  - Coordinating tasks between multiple agents in a system, where different agents handle various subtasks.
- Frameworks that facilitate function-calling (and agents, agentic workflows…)

  - LangChain, LangGraph

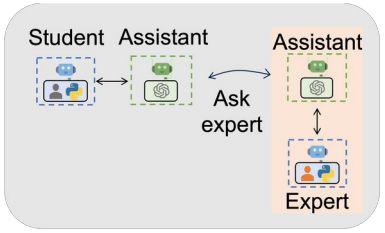
  - AutoGen (Microsoft)

  - LlamaIndex

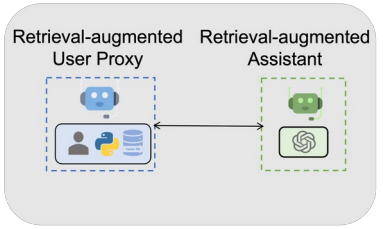
  - OpenAI API, etc.
- Big effort of LLM providers (e.g. OpenAI) training the models to excel at function-calling.

Multi-agent Systems

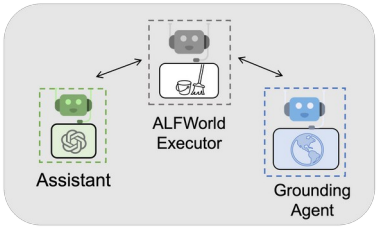
- Let us decompose a complex task in smaller subtasks, which can be carried out by different “agents”.
- We can enforce various degrees of control:
  - establish the information flow sequentially
  - the agents decide by themselves who is going to do what



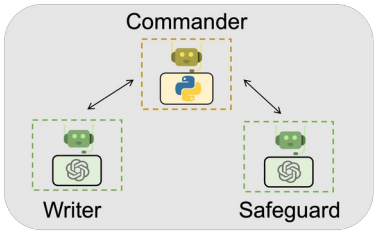
A1. Math Problem Solving



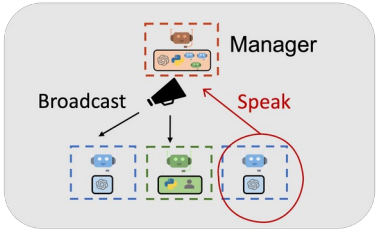
A2. Retrieval-augmented Chat



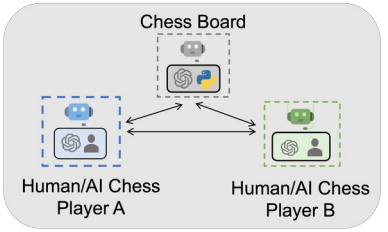
A3. Decision Making



A4. Multi-agent Coding

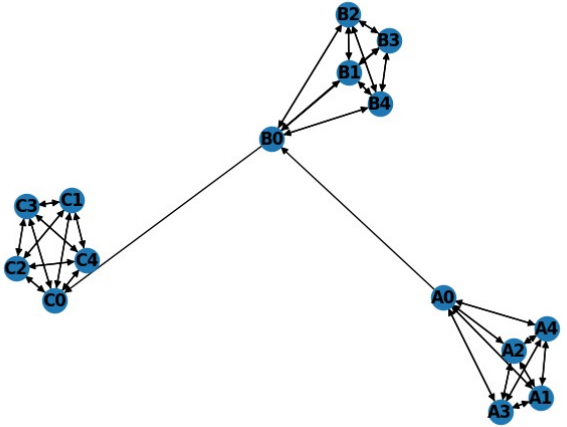


A5. Dynamic Group Chat



A6. Conversational Chess

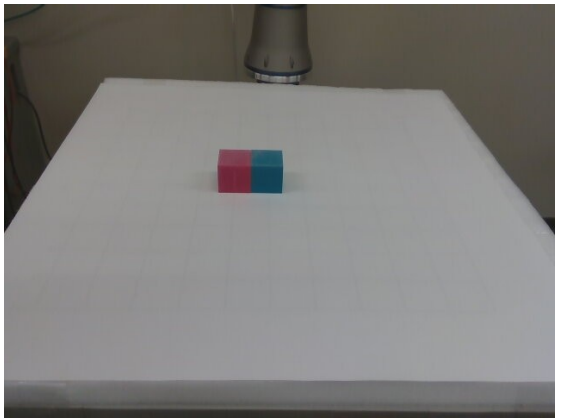
Automatic setup with X number of agents

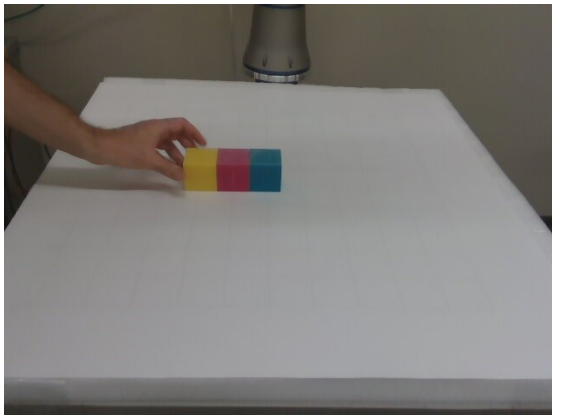


Child-tutoring system at OIST

- Develop a system to support future child-robot interaction research
- The system took the role of a mother interactively teaching a child how to move colored blocks around on a table, **spatial relationships between the blocks**, etc.
- Able to “understand” the scene and the intentions, correct the child, memory, improvisation, etc.
- (Extra) RAG about general knowledge, robot arm to point at the blocks, etc.

can’t fail





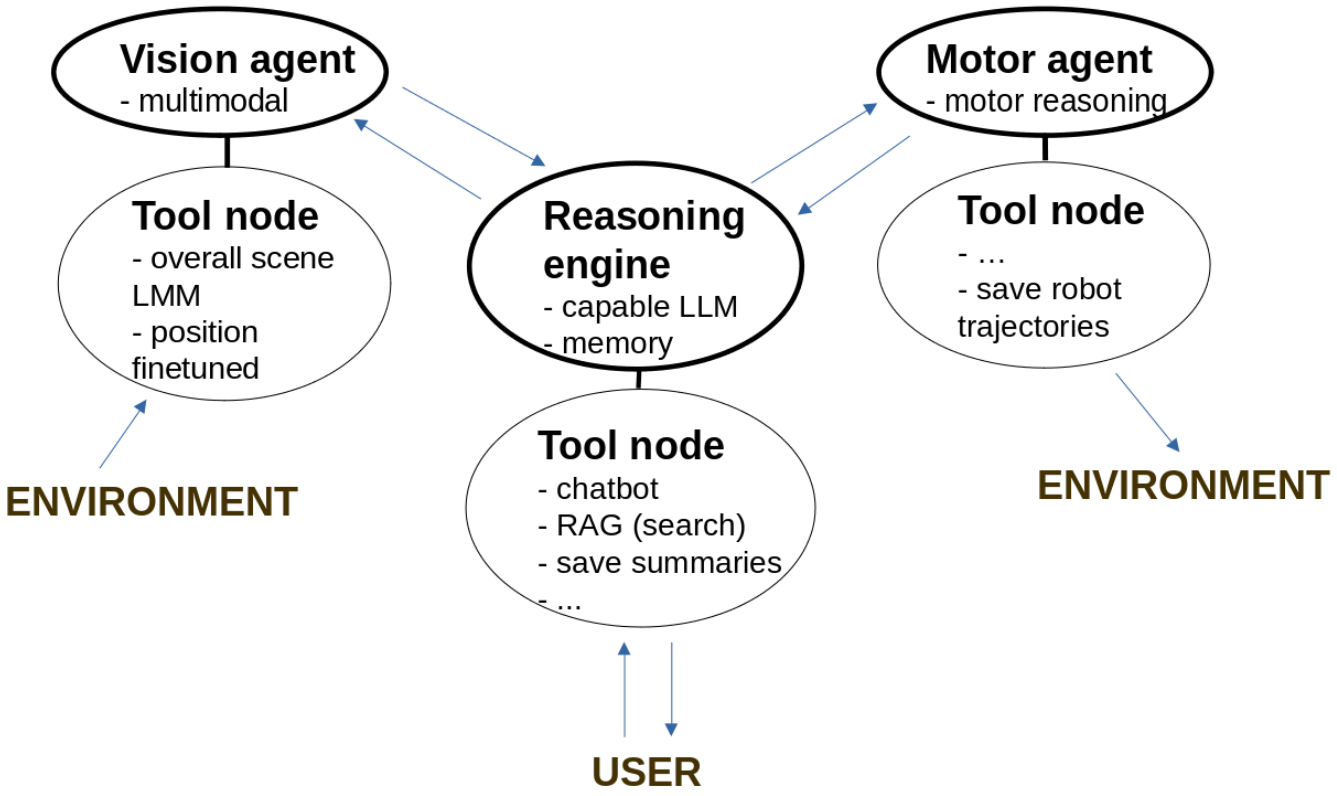
Challenge: *LMMs and spatial orientation*

USER: "*Describe what is in the image and the positions of the objects with respect to each other.*"

ChatGPT4: "In the image, there is a selection of chess pieces arranged on what appears to be a wooden surface. There are three white chess pieces and three black chess pieces. From left to right:

- The first piece is a white king, identifiable by its cross-shaped top.
- Next to the white king, there is a black rook (castle), recognized by its crenellated top.
- The black rook is followed by a black bishop, which has a slit on top of its miter-shaped head.
- Opposite the black bishop, on its right, stands a white bishop, mirroring the position of the black bishop.
- Next to the white bishop is a black knight, which resembles a horse's head and neck.
- Lastly, to the right of the black knight is a white pawn, the smallest piece, with its round head and simplistic shape.

Multi-Agent System architecture (LangGraph)



Child-tutoring system at OIST

VIDEO

Demo

Thank you for your attention!

Any questions?

jgallegopz@gmail.com
www.linkedin.com/in/jorge-gallego-perez