ChatGPT/LLM Log - Document all prompts and output

Lawrence Winters (lawrence.winters@digipen.edu)

Do your best to copy and paste all prompts and output into this document.

IMPORTANT NOTE:

- I also have GitHub copilot on my visual studio and I use the auto completions from time to time, I'm unsure of exactly how to list these except for one example of autocompletion that I copied line for line:

// Remove the current agent from the list to avoid selecting itself

std::vector<Agent\*> avaliableAgents = allAgents;

auto itr = std::remove(avaliableAgents.begin(), avaliableAgents.end(), agent);

avaliableAgents.erase(itr, avaliableAgents.end());

This is very basic c++ code however, so I'm unsure if this should be included or not.

**Chat\_GPT Prompt:**

REGISTER\_CONTROLFLOW(C\_ParallelSequencer, "C\_ParallelSequencer")

REGISTER\_CONTROLFLOW(C\_RandomSelector, "C\_RandomSelector")

REGISTER\_CONTROLFLOW(C\_Selector, "C\_Selector")

REGISTER\_CONTROLFLOW(C\_Sequencer, "C\_Sequencer")

REGISTER\_DECORATOR(D\_Delay, "D\_Delay")

REGISTER\_DECORATOR(D\_InvertedRepeater, "D\_InvertedRepeater")

REGISTER\_DECORATOR(D\_RepeatFourTimes, "D\_RepeatFourTimes")

REGISTER\_LEAF(L\_CheckMouseClick, "L\_CheckMouseClick")

REGISTER\_LEAF(L\_Idle, "L\_Idle")

REGISTER\_LEAF(L\_MoveToFurthestAgent, "L\_MoveToFurthestAgent")

REGISTER\_LEAF(L\_MoveToMouseClick, "L\_MoveToMouseClick")

REGISTER\_LEAF(L\_MoveToRandomPosition, "L\_MoveToRandomPosition")

REGISTER\_LEAF(L\_PlaySound, "L\_PlaySound")

These are all the nodes in the engine currently. I need ideas for control flow and decorator nodes. I'm trying to keep things very simple

**ChatGPT said:**

Perfect — thanks for sharing your existing nodes. Since your setup already includes fundamental control flows like C\_Selector, C\_Sequencer, and even C\_RandomSelector, and your decorators are simple (like D\_Delay, D\_InvertedRepeater, etc.), we can design 3 custom control flow/decorator nodes that stay lightweight, yet still feel distinct.

✅ Suggested Simple and Unique Control Flow / Decorator Nodes

1. C\_ThresholdSelector

✅ Control Flow

Idea: Like a Selector, but it needs at least N children to succeed before returning Success. Otherwise returns Failure.

Parameters: threshold (e.g., 2)

Example: Out of 5 children, if 2 or more succeed, return Success.

2. D\_SkipFirstTime

✅ Decorator

Idea: Skips the first time it's evaluated, then allows its child to run normally on future ticks.

Use Case: Delay execution until second evaluation pass (good for staggered startup behavior).

Behavior:

First tick: return Success or Failure immediately (without running child).

Subsequent ticks: run the child normally.

3. C\_FlipFlop

✅ Control Flow

Idea: Alternates between two children every tick.

Use Case: Toggle between two behaviors (e.g., patrol points, idle vs scan).

Behavior:

Tick 1: run child 0

Tick 2: run child 1

Tick 3: run child 0...

**Chat\_GPT Prompt:**

I have the direction vector towards an object but want my object to face towards it by setting the yaw but am unsure of the equation to use to get the radians for that.

**ChatGPT said:**

If you already have a **direction vector** toward an object (let’s say dir), and you want your object to **rotate to face that direction**, here’s how to think about it:

**🎯 Goal:**

1. Make the object **face** the dir direction (look-at heading).

**✅ Step-by-Step (Assuming Y-Up)**

**1. Normalize the direction vector:**

glm::vec3 forward = glm::normalize(dir);

**2. Compute the yaw (horizontal rotation):**

float yaw = atan2(forward.x, forward.z); // radians **THIS WAS PART I USED**

**3. Set a fixed pitch:**

Say you want the object to tilt **downward** 30 degrees:

float pitch = glm::radians(-30.0f); // convert to radians

**4. Build the final rotation:**

Using glm::yawPitchRoll or similar:

glm::quat rotation = glm::yawPitchRoll(yaw, pitch, 0.0f); // yaw, pitch, roll

Or use Euler angles with a matrix:

glm::mat4 rot = glm::rotate(glm::mat4(1.0f), yaw, glm::vec3(0, 1, 0));

rot = glm::rotate(rot, pitch, glm::vec3(1, 0, 0));

**🔁 Summary:**

* **Yaw** comes from direction vector (atan2(x, z)).
* **Pitch** is what you define manually (in degrees).
* You can now apply that rotation to your object.