Project: Memory Management Chatbot

Submission Results

Submission Date: September 16, 2024

Submission Passed

Download Submission

Feedback Details

Specification Review | Code Review

Reviewer Note

Congurtutation! You meet all requirements 🥳🥳

Very impressive work Udacian!

It was nice going through your work, you have implemented code in the required files nicely.

- 🎯 Is the code functional?
- 🎯 `_chatLogic` is an exclusive resource of `ChatbotPanelDialog`
- 🎯 Class design meets the Rule of Five guidelines.
- 🎯 The `GraphNodes` in the vector `_nodes` are exclusively owned by the class `ChatLogic`
- 🎯 `GraphNode` ownership is not transferred when passing instances.
- 🎯 `GraphNodes` exclusively own the outgoing `GraphEdges` and hold non-owning references to incoming `GraphEdges`.
- 🎯 Move semantics are used when transferring ownership from class `ChatLogic` into instances of `GraphNode`.
- 🎯 Move semantics are used correctly with `ChatBot`.
- 🎯 `ChatLogic` has no ownership relation to the `ChatBot` instance.
- 🎯 The `Chatbot` prints output to indicate Rule of Five components.

In this project, you mainly learned how to use the `Smart Pointer`. Because that method is currently being actively used in various industrial fields, the knowledge learned this time will be of great help in your career in the future.

- Unreal Smart Pointer Library
- ROS2: Efficient intra-process communication

I wish you luck and many new successes with future projects. I hope you get much more knowledge in future learning and that will surely result in more successful projects too!

Smart Pointers

What is difference between `unique_ptr` and `shared_ptr` and `weak_ptr` in C++?

`std::unique_ptr` and `std::shared_ptr` are both smart pointers in C++ that provide automatic memory management, but they have different ownership and reference counting semantics. Here are the main differences between `std::unique_ptr` and `std::shared_ptr`:

- Differences between unique_ptr and shared_ptr
- std::unique_ptr vs std::shared_ptr vs std::weak_ptr vs std::auto_ptr vs raw pointers

Quality of Code

✓ Is the code functional?

Task 1: Exclusive Ownership 1

✓ `_chatLogic` is an exclusive resource of `ChatbotPanelDialog`

Task 2: The Rule of Five

✓ Class design meets the Rule of Five guidelines.

Task 3: Exclusive Ownership 2

✓ The `GraphNode`s in the vector `_nodes` are exclusively owned by the class `ChatLogic`.

✓ `GraphNode` ownership is not transferred when passing instances.

Task 4: Moving Smart Pointers

✓ `GraphNode`s exclusively own the outgoing `GraphEdges` and hold non-owning references to incoming `GraphEdges`.

✓ Move semantics are used when transferring ownership from class `ChatLogic` into instances of `GraphNode`.

Task 5: Moving the ChatBot

✓ Move semantics are used correctly with `ChatBot`.

✓ `ChatLogic` has no ownership relation to the `ChatBot` instance.

✓ The `Chatbot` prints output to indicate Rule of Five components.

Submission History

Choose a different project submission

Rate & Review

How clear and easy is it to understand the project review?

☆ ☆ ☆

← Previous | Next →

Click next to continue to the next lesson. You may return and submit or view a project at any time.