UDACITY

‹ Return to "C++" in the classroom

DISCUSS ON STUDENT HUB

Program a Concurrent Traffic Simulation

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Hello there,

Congratulations on finishing the project 🎉
Keep doing the great work and all the best for future project.

FP.1 Create a TrafficLight class

A `TrafficLight` class is is defined which is a child class of `TrafficObject`.

Excellent work!

The class shall have the public methods `void waitForGreen()` and `void simulate()` as well as `TrafficLightPhase getCurrentPhase()`, where `TrafficLightPhase` is an enum that can be either `red` or `green`.

Also, there should be a private method `void cycleThroughPhases()` and a private member `_currentPhase` which can take `red` or `green` as its value.

Excellent work! All of the correct methods and members are implemented in your TrafficLight class.

FP.2: Implement a cycleThroughPhases method

Implement the function with an infinite loop that measures the time between two loop cycles and toggles the current phase of the traffic light between red and green.

The cycle duration should be a random value between 4 and 6 seconds, and the while-loop should use `std::this_thread::sleep_for` to wait 1ms between two cycles.

Well done! Your class has a cycleThroughPhases method implemented, and this method has a cycle duration between 4 and 6 seconds with a while loop that sleeps for 1ms between cycles.

Note that for improved randomness, you can also use a random device and a Mersenne Twister pseudo-random generator as follows:

```
std::random_device rd;
std::mt19937 eng(rd());
std::uniform_int_distribution<> distr(4000, 6000);
double cycleDuration = distr(eng); // duration of a single simulation cycle in ms
```

The private `cycleThroughPhases()` method should be started in a thread when the public method `simulate` is called. To do this, a thread queue should be used in the base class.

Great work!

FP.3 Define class MessageQueue

A `MessageQueue` class is defined in the header of class `TrafficLight` which has the public methods `send` and `receive`.

Perfect here as well.

`send` should take an `rvalue` reference of type `TrafficLightPhase` whereas `receive` should return this type.

Also, the `MessageQueue` class should define a `std::dequeue` called `_queue`, which stores objects of type `TrafficLightPhase`.

Also, there should be a `std::condition_variable` as well as an `std::mutex` as private members.

Great implementation here as well

FP.4 Implement the method `send`

The method `send` should use the mechanisms `std::lock_guard<std::mutex>` as well as `_condition.notify_one()` to add a new message to the queue and afterwards send a notification.

In the class `TrafficLight`, a private member of type `MessageQueue` should be created and used within the infinite loop to push each new `TrafficLightPhase` into it by calling `send` in conjunction with move semantics.

This is well implemented.

FP.5 Implement the methods `receive` and `waitForGreen`

The method receive should use `std::unique_lock<std::mutex>` and `_condition.wait()` to wait for and receive new messages and pull them from the queue using move semantics. The received object should then be returned by the `receive` function.

This works like a charm!

The method `waitForGreen` is completed, in which an infinite while loop runs and repeatedly calls the `receive` function on the message queue. Once it receives `TrafficLightPhase::green`, the method returns.

This option is included as well.

FP.6 Implement message exchange

In class Intersection, a private member `_trafficLight` of type `TrafficLight` should exist.

The method `Intersection::simulate()`, should start the simulation of `_trafficLight`.

The method `Intersection::addVehicleToQueue`, should use the methods `TrafficLight::getCurrentPhase` and `TrafficLight::waitForGreen` to block the execution until the traffic light turns green.

This looks great!

DOWNLOAD PROJECT

RETURN TO PATH

Rate this project