Build an OpenStreetMap Route Planner

src/main.cpp 1

```cpp
#include <optional>
#include <fstream>
#include <iostream>
#include <vector>
#include <string>
#include <io2d.h>
#include "route_model.h"
#include "render.h"
#include "route_planner.h"

using namespace std::experimental;

static std::optional<std::vector<std::byte>> ReadFile(const std::string &path)
{
    std::ifstream is{path, std::ios::binary | std::ios::ate};
    if( !is )
        return std::nullopt;

    auto size = is.tellg();
    std::vector<std::byte> contents(size);

    is.seekg(0);
    is.read((char*)contents.data(), size);

    if( contents.empty() )
        return std::nullopt;
    return std::move(contents);
}

int main(int argc, const char **argv)
{
    std::string osm_data_file = "";
    if( argc > 1 ) {
        for( int i = 1; i < argc; ++i )
            if( std::string_view{argv[i]} == "-f" && ++i < argc )
                osm_data_file = argv[i];
    }
    else {
        std::cout << "To specify a map file use the following format: " << std::endl;
        std::cout << "Usage: [executable] [-f filename.osm]" << std::endl;
        osm_data_file = "../map.osm";
    }

    std::vector<std::byte> osm_data;

    if( osm_data.empty() && !osm_data_file.empty() ) {
        std::cout << "Reading OpenStreetMap data from the following file: " <<  osm_data_file
        auto data = ReadFile(osm_data_file);
        if( !data )
            std::cout << "Failed to read." << std::endl;
        else
            osm_data = std::move(*data);
    }

    // TODO 1: Declare floats `start_x`, `start_y`, `end_x`, and `end_y` and get
    // user input for these values using std::cin. Pass the user input to the
    // RoutePlanner object below in place of 10, 10, 90, 90.

    float start_x, start_y, end_x, end_y ;

    std::cin >> start_x >> start_y >>  end_x >>  end_y ;

    // Build Model.
    RouteModel model{osm_data};

    // Create RoutePlanner object and perform A* search.
    RoutePlanner route_planner{model, start_x, start_y, end_x, end_y };
```

AWESOME

Nice work, Adding user input. You can also validate whether input entered by user is within `<0, 100>` range.

```cpp
    route_planner.AStarSearch();

    std::cout << "Distance: " << route_planner.GetDistance() << " meters. \n";

    // Render results of search.
    Render render{model};

    auto display = io2d::output_surface{400, 400, io2d::format::argb32, io2d::scaling::none,
    display.size_change_callback([](io2d::output_surface& surface){
        surface.dimensions(surface.display_dimensions());
    });
    display.draw_callback([&](io2d::output_surface& surface){
        render.Display(surface);
    });
    display.begin_show();
}

```

src/route_planner.cpp 1

thirdparty/pugixml/src/pugixml.hpp

thirdparty/pugixml/src/pugiconfig.hpp

thirdparty/googletest/googletest/src/gtest_main.cc

thirdparty/googletest/googletest/src/gtest.cc

thirdparty/googletest/googletest/src/gtest-typed-test.cc

thirdparty/googletest/googletest/src/gtest-test-part.cc

thirdparty/googletest/googletest/src/gtest-printers.cc

thirdparty/googletest/googletest/src/gtest-port.cc

thirdparty/googletest/googletest/src/gtest-matchers.cc

thirdparty/googletest/googletest/src/gtest-internal-inl.h

thirdparty/googletest/googletest/src/gtest-filepath.cc

thirdparty/googletest/googletest/src/gtest-death-test.cc

thirdparty/googletest/googletest/src/gtest-all.cc

thirdparty/googletest/googlemock/src/gmock_main.cc

thirdparty/googletest/googlemock/src/gmock.cc

thirdparty/googletest/googlemock/src/gmock-spec-builders.cc

thirdparty/googletest/googlemock/src/gmock-matchers.cc

thirdparty/googletest/googlemock/src/gmock-internal-utils.cc

thirdparty/googletest/googlemock/src/gmock-cardinalities.cc

thirdparty/googletest/googlemock/src/gmock-all.cc

src/route_planner.h

src/route_model.h

src/route_model.cpp

src/render.h

src/render.cpp

src/model.h

src/model.cpp

build/thirdparty/googletest/googlemock/gtest/CMakeFiles/gtest_main.dir/src/gtest_main.cc.o

▸ build/thirdparty/googletest/googlemock/CMakeFiles/gmock_main.dir/src/gmock_main.cc.o

▸ build/CMakeFiles/route_planner.dir/src/route_planner.cpp.o

▸ build/CMakeFiles/OSM_A_star_search.dir/src/route_planner.cpp.o

RETURN TO PATH