#include "fl/Headers.h"

int main(int argc, char\* argv[]){

using namespace fl;

//Code automatically generated with fuzzylite 6.0.

using namespace fl;

//定义要模糊化的工程

Engine\* engine = new Engine;

engine->setName("ObstacleAvoidance");

engine->setDescription("");

//定义输入参数以及对应的模糊化规则

InputVariable\* obstacle = new InputVariable;

obstacle->setName("obstacle");

obstacle->setDescription("");

obstacle->setEnabled(true);

obstacle->setRange(0.000, 1.000);

obstacle->setLockValueInRange(false);

obstacle->addTerm(new Ramp("left", 1.000, 0.000));

obstacle->addTerm(new Ramp("right", 0.000, 1.000));

engine->addInputVariable(obstacle);

//定义输出参数以及对应的解模糊规则

OutputVariable\* mSteer = new OutputVariable;

mSteer->setName("mSteer");

mSteer->setDescription("");

mSteer->setEnabled(true);

mSteer->setRange(0.000, 1.000);

mSteer->setLockValueInRange(false);

mSteer->setAggregation(new Maximum);

mSteer->setDefuzzifier(new Centroid(100));

mSteer->setDefaultValue(fl::nan);

mSteer->setLockPreviousValue(false);

mSteer->addTerm(new Ramp("left", 1.000, 0.000));

mSteer->addTerm(new Ramp("right", 0.000, 1.000));

engine->addOutputVariable(mSteer);

//定义模糊规则库，方便进行模糊推理

RuleBlock\* mamdani = new RuleBlock;

mamdani->setName("mamdani");

mamdani->setDescription("");

mamdani->setEnabled(true);

mamdani->setConjunction(fl::null);//定义交运算的规则，如minium代表取小

mamdani->setDisjunction(fl::null);//定义并运算的规则，如maxium代表取大

mamdani->setImplication(new AlgebraicProduct);

mamdani->setActivation(new General);

mamdani->addRule(Rule::parse("if obstacle is left then mSteer is right", engine));

mamdani->addRule(Rule::parse("if obstacle is right then mSteer is left", engine));

engine->addRuleBlock(mamdani);

std::string status;

if (not engine->isReady(&status))

throw Exception("[engine error] engine is not ready:\n" + status, FL\_AT);

for (int i = 0; i <= 50; ++i){

scalar location = obstacle->getMinimum() + i \* (obstacle->range() / 50);

obstacle->setValue(location);

engine->process();

FL\_LOG("obstacle.input = " << Op::str(location) <<

" => " << "steer.output = " << Op::str(steer->getValue()));

}

}