Nama : Raif Haidar Darmawan

NIM : 1103190133

Technical Report : Micromouse Competition

1. Synopsis dari project Micromouse Competition

Proyek ini merupakan upaya untuk mengembangkan kompetisi Micromouse di Webots, simulator robot 3D, untuk tujuan mempelajari dasar-dasar pemrograman agen otonom dan menerapkan pengetahuan teoretis yang diperoleh ke dalam praktik. Implementasi sebagian besar dibatasi oleh kompleksitas subjek dan terbatasnya waktu untuk penelitian dan pengembangan. Akibatnya, hasil dari upaya ini adalah robot otonom yang disimulasikan yang mencari jalur ke pusat labirin blok 16x16. Untuk mencapai tujuan ini, robot menggunakan empat prinsip dasar: lokalisasi, pemetaan, perencanaan jalur, dan kontrol gerak. Saat robot bergerak di dalam labirin, ia menggunakan serangkaian sensor untuk menghindari rintangan dan merekam posisi mereka di labirin menggunakan referensi posisi awalnya. Secara bersamaan, peta labirin yang direkam digunakan untuk menentukan kemungkinan jalur ke pusat setiap kali masuk ke sel berikutnya.

1. Apa itu Micromouse

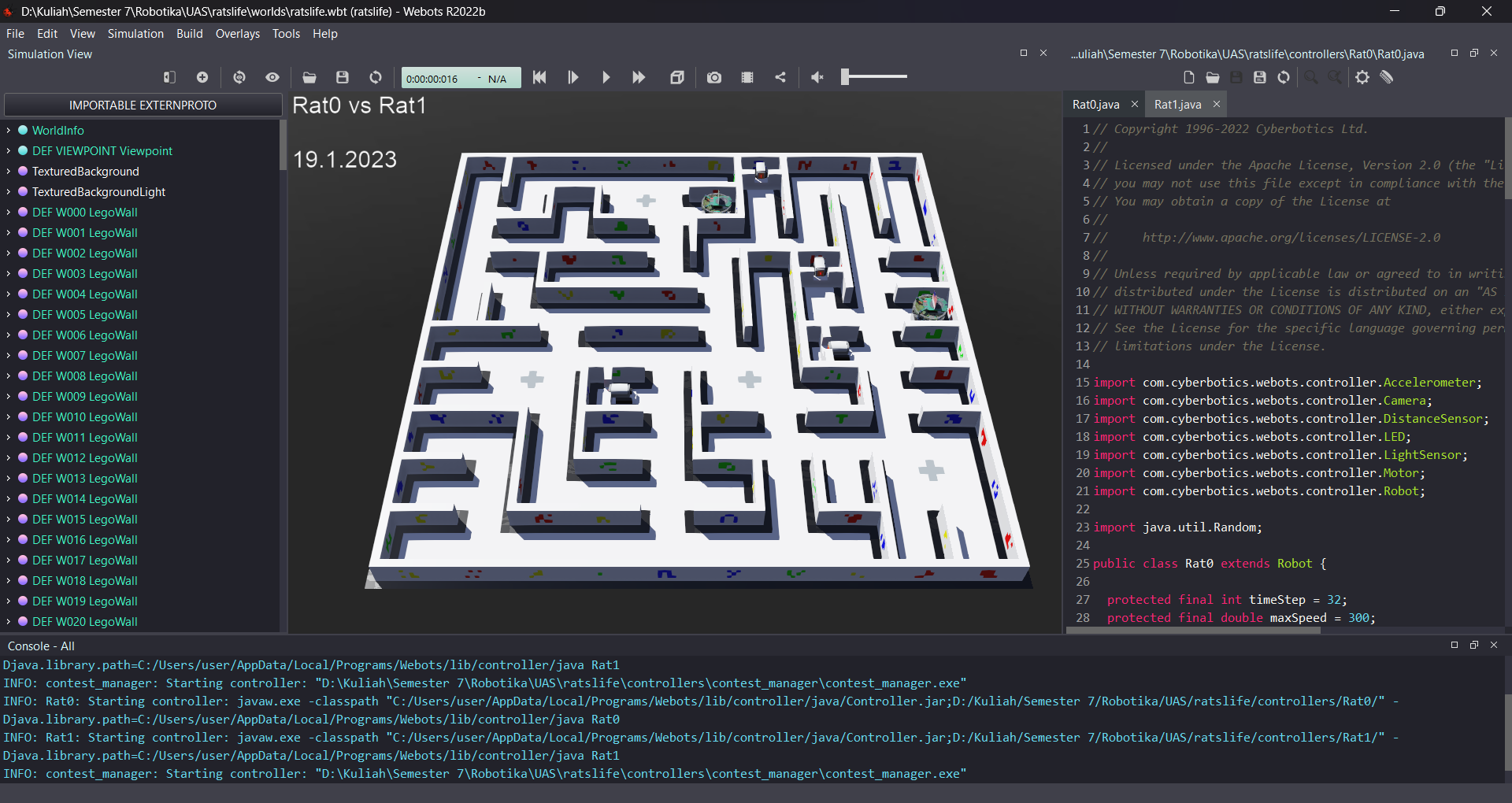
Micromouse merupakan kontes di mana robot otonom kecil akan berlomba ke tengah labirin dan micromouse menyelesaikan 5 lari ke tengah. Lari pertama adalah lari pencarian dan sisanya lari lari. Mouse tidak memiliki informasi lain selain posisi awalnya (yang selalu berada di sudut labirin dengan sisi kirinya menghadap bingkai labirin), ukuran labirin, dan harus mencapai pusatnya. Untuk menghitung jalur terbaik ke pusat, proses pertama digunakan untuk memetakan labirin dan ini disebut pencarian. Saat tikus berlomba lari, ia mungkin mencoba menjelajahi labirin dalam perjalanan kembali ke sel awal. Labirin dirancang dengan cermat untuk menonjolkan agen otonom yang lebih canggih yang mungkin memanfaatkannya untuk meminimalkan waktu kerja mereka. Perhatikan bahwa jalur terbaik ke tengah belum tentu yang terpendek karena mouse dapat melaju lebih cepat saat tidak harus berbelok. Setiap acara Micromouse kurang lebih memiliki seperangkat aturan yang sama. Perbedaan ditemukan dalam sistem penilaian, terutama untuk mempromosikan perilaku otonom yang lebih maju.

1. Pembuatan
2. Trek pada robot akan dibut secara otomatis dan secara random/acak menggunakan ‘Maze\_builder.c’ dan ‘Maze\_generator.c’. Setengahnya dibuat menggunakan Webots World Editor UI (menambahkan dinding, sambungan dinding, dll.) dan setengahnya lagi dengan memprogram pengontrol (dalam C) di sejumlah fungsi penyihir yang paling banyak diubah adalah file ‘maze\_generation.c’ yang berisi fungsi untuk mengimpor labirin dari arsip labirin.

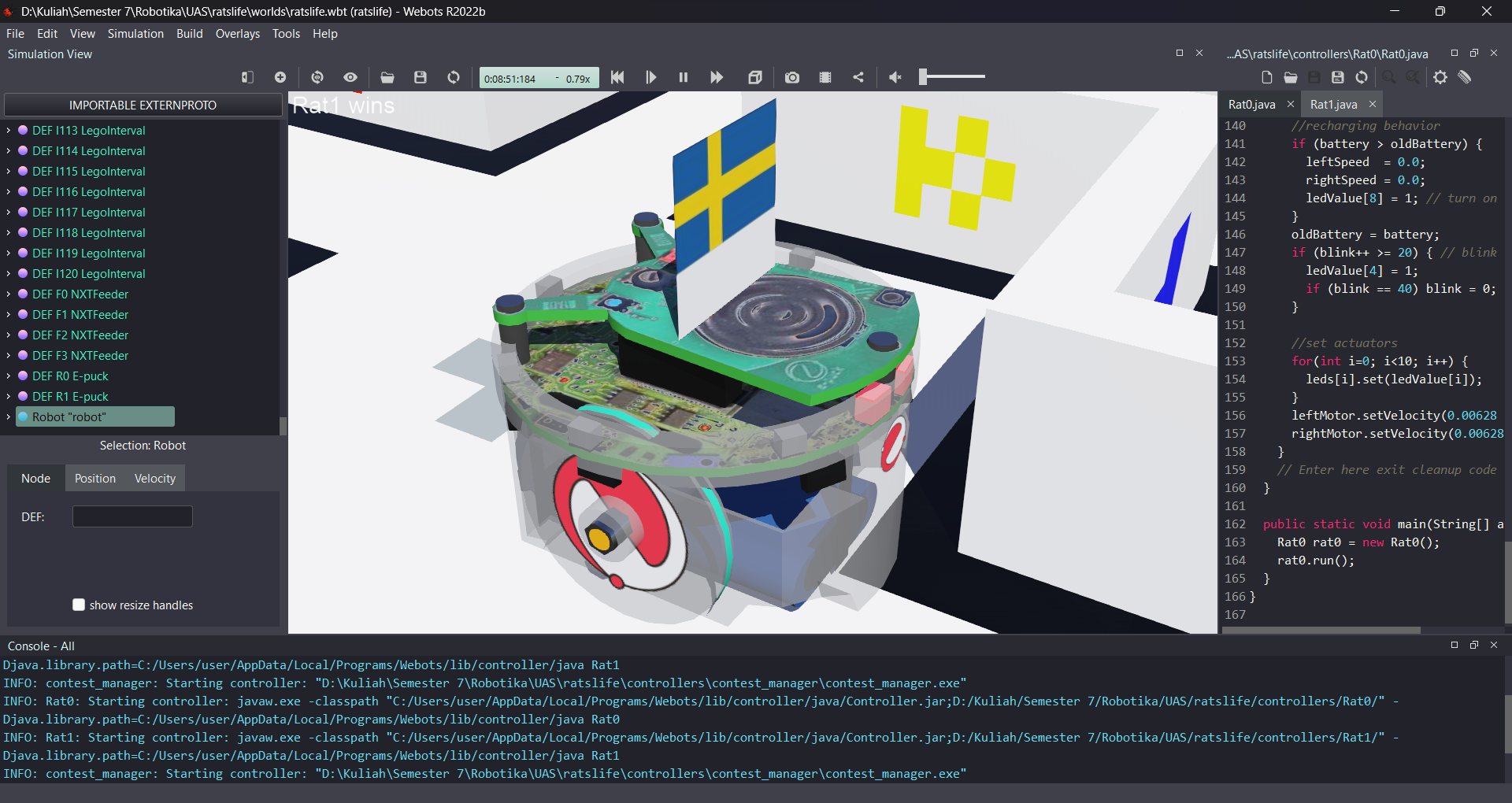
Graphical user interface, website

Description automatically generated

1. Trek yang dibuat secara random dan robot pun ditaruh secara random juga, robot akan mengikuti kodingan yang sudah dibuat. File untuk robot terdapat di 2 folder yaitu ‘rat0’ dan ‘rat1’



1. Untuk robot yang digunakan adalah robot e-punk, alas an menggunakan robot ini ialah robot sudah memiliki sensor yang diperlukan agar robot dapat mengenali apa yang diperlukan untuk mendeteksi tembok, dalam hal ini sensor yang digunakan adalah sensor jarak dan odometer.



1. Dibawah ini merupakan kodingan untuk memanggil library yang diperlukan oleh robot e-punk ini.

import com.cyberbotics.webots.controller.Accelerometer;

import com.cyberbotics.webots.controller.Camera;

import com.cyberbotics.webots.controller.DistanceSensor;

import com.cyberbotics.webots.controller.LED;

import com.cyberbotics.webots.controller.LightSensor;

import com.cyberbotics.webots.controller.Motor;

import com.cyberbotics.webots.controller.Robot;

import java.util.Random;

1. Dibawah ini merupakan kodingan untuk meng-inisialisasikan apa yang diperlukan oleh robot e-punk ini.

public class Rat0 extends Robot {

protected final int timeStep = 32;

protected final double maxSpeed = 300;

protected final double[] collisionAvoidanceWeights = {0.06,0.03,0.015,0.0,0.0,-0.015,-0.03,-0.06};

protected final double[] slowMotionWeights = {0.0125,0.00625,0.0,0.0,0.0,0.0,0.00625,0.0125};

protected Accelerometer accelerometer;

protected Camera camera;

protected int cameraWidth, cameraHeight;

protected Motor leftMotor, rightMotor;

protected DistanceSensor[] distanceSensors = new DistanceSensor[8];

protected LightSensor[] lightSensors = new LightSensor[8];

protected LED[] leds = new LED[10];

1. Dibawah ini merupakan kodingan untuk menjalankan sensor dari accelerometer, kamera, sensor IR, odometer. Dikodingan ini juga diatur kecepatan dari motor dan akan bergerak selama-lamnya (Infinity). Dikodingan ini juga diatur hidupnya lampu LED yang terhubung dengan Sensor jarak (IR).

public Rat0() {

accelerometer = getAccelerometer("accelerometer");

camera = getCamera("camera");

camera.enable(8\*timeStep);

cameraWidth=camera.getWidth();

cameraHeight=camera.getHeight();

leftMotor = getMotor("left wheel motor");

rightMotor = getMotor("right wheel motor");

leftMotor.setPosition(Double.POSITIVE\_INFINITY);

rightMotor.setPosition(Double.POSITIVE\_INFINITY);

leftMotor.setVelocity(0.0);

rightMotor.setVelocity(0.0);

for (int i=0;i<10;i++) {

leds[i]=getLED("led"+i);

};

for (int i=0;i<8;i++) {

distanceSensors[i] = getDistanceSensor("ps"+i);

distanceSensors[i].enable(timeStep);

lightSensors[i] = getLightSensor("ls"+i);

lightSensors[i].enable(timeStep);

}

batterySensorEnable(timeStep);

}

1. Jika sudah terdapat cara kerja robot yang diatur didalam fungsi ‘void run’.

public void run() {

int blink = 0;

int oldDx = 0;

Random r = new Random();

boolean turn = false;

boolean right = false;

boolean seeFeeder = false;

double battery;

double oldBattery = -1.0;

int image[];

double distance[] = new double[8];

int ledValue[] = new int[10];

double leftSpeed, rightSpeed;

while (step(timeStep) != -1) {

// read sensor information

image = camera.getImage();

for(int i=0;i<8;i++) distance[i] = distanceSensors[i].getValue();

battery = batterySensorGetValue();

for(int i=0;i<10;i++) ledValue[i] = 0;

// obstacle avoidance behavior

leftSpeed = maxSpeed;

rightSpeed = maxSpeed;

for (int i=0;i<8;i++) {

leftSpeed -= (slowMotionWeights[i]+collisionAvoidanceWeights[i])\*distance[i];

rightSpeed -= (slowMotionWeights[i]-collisionAvoidanceWeights[i])\*distance[i];

}

// return either to left or to right when there is an obstacle

if (distance[6]+distance[7] > 1800 || distance[0]+distance[1] > 1800) {

if (!turn) {

turn = true;

right = r.nextBoolean();

}

if (right) {

ledValue[2] = 1;

leftSpeed = maxSpeed;

rightSpeed = -maxSpeed;

} else {

ledValue[6] = 1;

leftSpeed = -maxSpeed;

rightSpeed = maxSpeed;

}

} else {

turn=false;

}

// vision

int blobX=0,blobY=0,blobCounter=0;

// looking for an alight feeder

for(int x=0; x<cameraWidth; x++) for(int y=cameraWidth/3; y<2\*cameraWidth/3; y++) {

int pixel = image[y \* cameraWidth + x];

if (Camera.pixelGetGreen(pixel) >= 248 &&

Camera.pixelGetBlue(pixel) >= 248) {

blobX += x;

blobY += y;

blobCounter++;

}

}

if (blobCounter > 2) { // significant enough

seeFeeder = true;

blobX /= blobCounter;

blobY /= blobCounter;

int dx = (blobX-cameraWidth/2)\*10;

if (dx > 0) ledValue[1] = 1;

else ledValue[7] = 1;

if (oldDx != dx){

leftSpeed = 2\*dx;

rightSpeed = -2\*dx;

oldDx = dx;

} else {

leftSpeed = maxSpeed/3;

rightSpeed = maxSpeed/3;

}

}

//recharging behavior

if (battery > oldBattery) {

leftSpeed = 0.0;

rightSpeed = 0.0;

ledValue[8] = 1; // turn on the body led

}

oldBattery = battery;

if (blink++ >= 20) { // blink the back LEDs

ledValue[4] = 1;

if (blink == 40) blink = 0;

}

//set actuators

for(int i=0; i<10; i++) {

leds[i].set(ledValue[i]);

}

leftMotor.setVelocity(0.00628 \* leftSpeed);

rightMotor.setVelocity(0.00628 \* rightSpeed);

}

// Enter here exit cleanup code

}

public static void main(String[] args) {

Rat0 rat0 = new Rat0();

rat0.run();

}

1. Robot akan melakukan Algoritma Flood Fill yang mana sebuah algoritma pencarian yang sangat berguna untuk perencanaan jalur di labirin. Dengan menetapkan nilai nol ke bobot sel tujuan, kami kemudian dapat memprogram robot untuk mengikuti jalur terpendek ke pusat.

Background pattern

Description automatically generated >> Background pattern

Description automatically generated

