総合実験１

LEGO 回転型倒⽴振⼦

⽒名 佐々木佑 (1935098T)

提出⽇ 2021 年 6 ⽉ 23 ⽇

１．実験の目的

　「センサ－情報処理－アクチュエータ」系による外界とのインタラクションを伴う知能行動体の設計と 製作を体験する．実験を通し、関連講義科目の重要性，必要性，意義を体感する．そして，自分で 考え工夫すること，設計製作すること，グループで協力することの楽しさを体験する．

２．ロボットの説明

　他の生徒のロボットと異なると思われる特徴は５つ。

１．光センサを前方に２つ、後方に１つ備える。

２．距離センサを前方に２つ備える。

３．角度センサ（フィールドに対するロボットの角度を取得）を備える。

４．アームを前方に2つ備える。

以下がロボットの定義である。

Robot {

translation -0.40235 0.0398085 0.448802

rotation 0.0017674855636001065 0.9999780244747116 -0.006389566686054809 2.8557611165062147

children [

InertialUnit {

translation 0 0.042 0

children [

DEF INNER Shape {

appearance PBRAppearance {

baseColor 0 1 0.498039

roughness 1

metalness 0

}

geometry Cylinder {

height 0.01

radius 0.01

}

}

]

name "inner"

boundingObject USE INNER

physics Physics {

}

}

Solid {

translation -0.0682849 0.012 0.0975769

rotation 0.5000000690830143 0.7071080976983041 -0.499998069082738 -3.1415853071795863

children [

Shape {

appearance PBRAppearance {

}

geometry DEF ARM Cylinder {

height 0.15

radius 0.005

}

}

]

name "solid(5)"

boundingObject USE ARM

physics Physics {

}

}

DistanceSensor {

translation 0.03 -0.015 0.05

rotation 0 1 0 -1.2708

children [

DEF DS\_SENSOR Shape {

appearance PBRAppearance {

baseColor 0 0 1

roughness 1

metalness 0

}

geometry Box {

size 0.01 0.01 0.01

}

}

]

name "ds\_left"

boundingObject USE DS\_SENSOR

physics Physics {

}

}

DistanceSensor {

translation -0.03 -0.015 0.05

rotation 0 1 0 -1.8708

children [

DEF DS\_SENSOR Shape {

appearance PBRAppearance {

baseColor 0 0 1

roughness 1

metalness 0

}

geometry Box {

size 0.01 0.01 0.01

}

}

]

name "ds\_right"

boundingObject USE DS\_SENSOR

physics Physics {

}

}

LightSensor {

translation 0 0.042 -0.042

rotation 0 1 0 1.59

name "back"

lookupTable [

0 0 0

4 1024 0

]

}

Gyro {

translation 0 0.042 0

lookupTable [

4 1024 0

]

}

Solid {

translation 0.0824266 0.0120003 0.097574

rotation 0.28108510281603044 -0.6785992482197749 0.6785972482190434 2.59357

children [

Shape {

appearance PBRAppearance {

}

geometry DEF ARM Cylinder {

height 0.15

radius 0.005

}

}

]

name "solid(4)"

boundingObject USE ARM

physics Physics {

}

}

LightSensor {

translation 0.02 0.042 0.042

rotation 0 1 0 -1.07

name "ls1"

lookupTable [

0 0 0

4 1024 0

]

}

LightSensor {

translation -0.02 0.045 0.042

rotation 0 1 0 -2.07

name "ls0"

lookupTable [

0 0 0

4 1024 0

]

}

DEF BODY Shape {

appearance PBRAppearance {

baseColor 1 0 0

roughness 1

metalness 0

}

geometry Box {

size 0.1 0.06 0.1

}

}

HingeJoint {

jointParameters HingeJointParameters {

position 2409.756664438725

anchor -0.606 0 0

}

device [

RotationalMotor {

name "wheel2"

}

]

endPoint DEF WHEEL2 Solid {

translation -0.06013416313823039 0.0005747212824638923 0.00013172678965499735

rotation -0.12102501339737204 0.11899381816315739 0.9854914598164305 1.586472604814353

children [

DEF WHEEL Shape {

appearance PBRAppearance {

baseColor 0 0 0

roughness 1

metalness 0

}

geometry Cylinder {

height 0.02

radius 0.04

}

}

]

name "solid(1)"

boundingObject USE WHEEL

physics Physics {

}

}

}

HingeJoint {

jointParameters HingeJointParameters {

position 2654.1212897863497

anchor 0.06 0 0

}

device [

RotationalMotor {

name "wheel1"

}

]

endPoint DEF WHEEL1 Solid {

translation 0.06010467964266675 6.807536285244513e-05 3.6080659947722033e-05

rotation -0.33704546135062097 0.35755036894334413 0.8709524043548105 1.77240237713873

children [

DEF WHEEL Shape {

appearance PBRAppearance {

baseColor 0 0 0

roughness 1

metalness 0

}

geometry Cylinder {

height 0.02

radius 0.04

}

}

]

boundingObject USE WHEEL

physics Physics {

}

}

}

Solid {

translation 0 -0.03 -0.04

children [

DEF CASTER Shape {

appearance PBRAppearance {

baseColor 0 0 0

metalness 0

}

geometry Sphere {

radius 0.01

}

}

]

name "solid(2)"

boundingObject USE CASTER

physics Physics {

}

}

Solid {

translation 0 -0.03 0.04

children [

DEF CASTER Shape {

appearance PBRAppearance {

baseColor 0 0 0

metalness 0

}

geometry Sphere {

radius 0.01

}

}

]

name "solid(3)"

boundingObject USE CASTER

physics Physics {

}

}

]

boundingObject USE BODY

physics Physics {

}

controller "test\_go\_light"

controllerArgs [

""

]

}

３．フロチャート

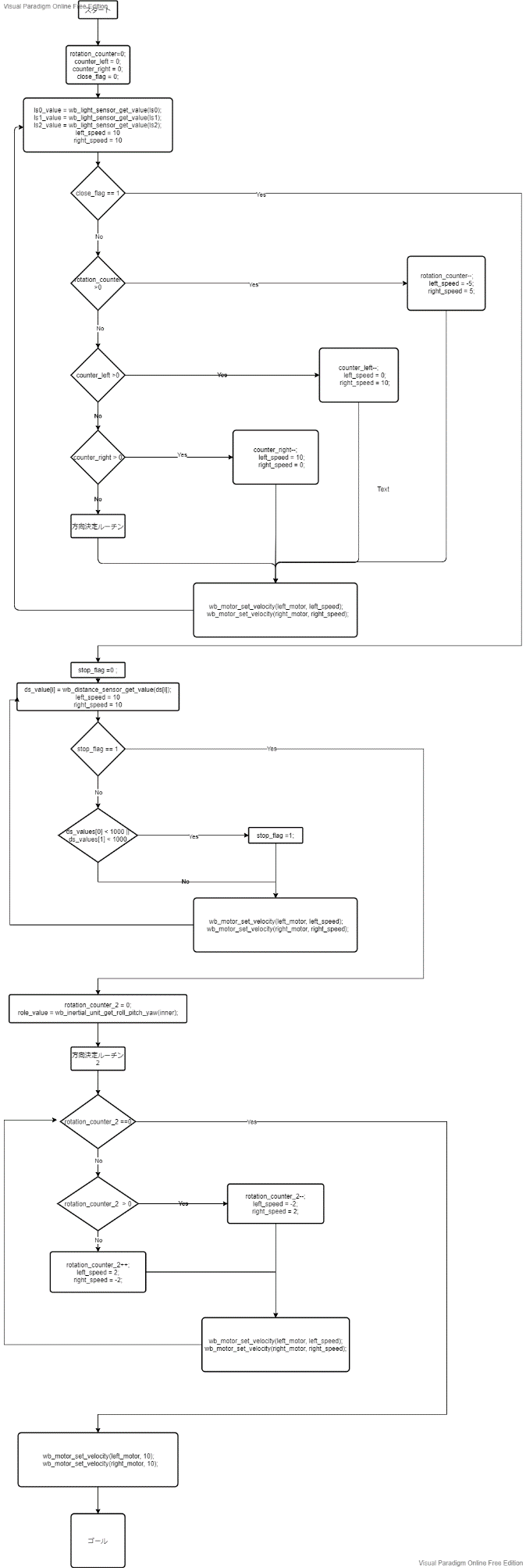
フローは大きく４つに分かれる

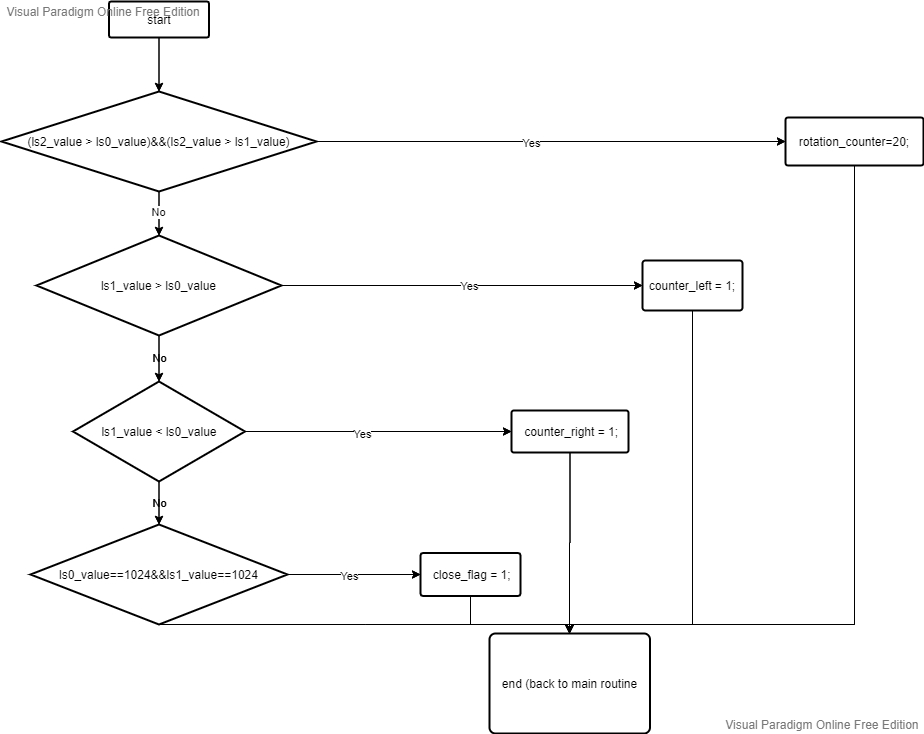
１．光センサを使ってボールにある程度近づく

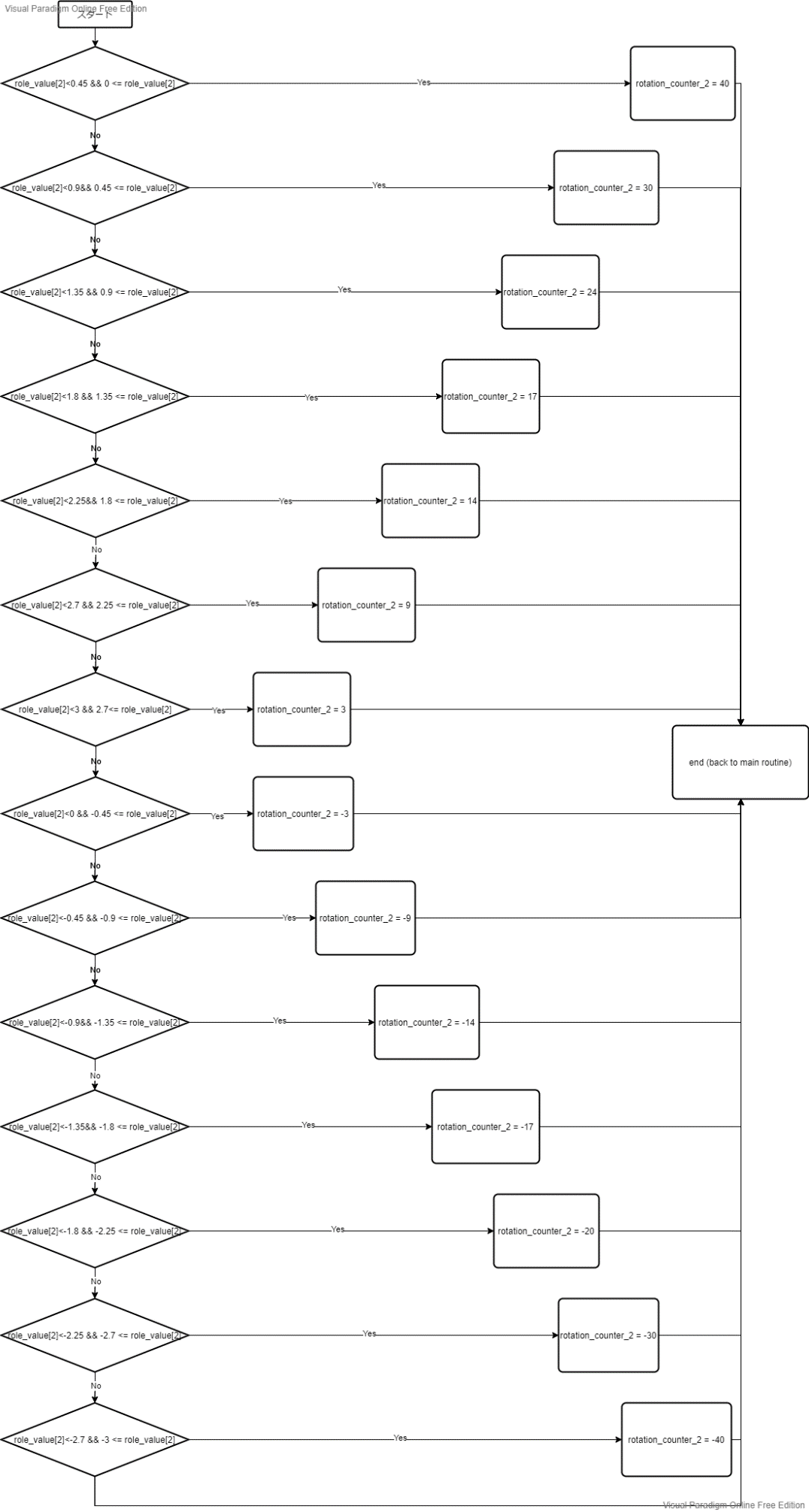
２．距離センサを使ってボールをアームの内側に置く

３．角度センサを使ってロボットをゴールの方向へ回転させる

４．ゴールまでボールを運ぶ。

　図１　メインルーチン

図２　方向決定ルーチン

図３　方向決定ルーチン２

４．プログラムの説明

プログラムは以下の通り。フロチャートを忠実に実装している。

#include <stdio.h>

#include<unistd.h>

#include <webots/light\_sensor.h>

#include <webots/distance\_sensor.h>

#include <webots/motor.h>

#include <webots/robot.h>

#include <webots/inertial\_unit.h>

#define MAX\_SPEED 10

#define SPEED 6

#define TIME\_STEP 64

int main() {

WbDeviceTag ls0, ls1,ls2, left\_motor, right\_motor,inner;

int i;

wb\_robot\_init();

/\* get a handler to the distance sensors. \*/

ls0 = wb\_robot\_get\_device("ls0");

ls1 = wb\_robot\_get\_device("ls1");//left

ls2 = wb\_robot\_get\_device("back");

wb\_light\_sensor\_enable(ls0, TIME\_STEP);

wb\_light\_sensor\_enable(ls1, TIME\_STEP);

wb\_light\_sensor\_enable(ls2, TIME\_STEP);

WbDeviceTag ds[2];

char ds\_names[2][10] = {"ds\_left", "ds\_right"};

for (i = 0; i < 2; i++) {

ds[i] = wb\_robot\_get\_device(ds\_names[i]);

wb\_distance\_sensor\_enable(ds[i], TIME\_STEP);

}

inner = wb\_robot\_get\_device("inner");

wb\_inertial\_unit\_enable(inner,TIME\_STEP);

int rotation\_counter = 0;

int counter\_left = 0;

int counter\_right = 0;

int stop\_flag = 0;

int close\_flag = 0;

//int close\_left = 0;

//int close\_right = 0;

/\* get a handler to the motors and set target position to infinity (speed control). \*/

left\_motor = wb\_robot\_get\_device("wheel1");

right\_motor = wb\_robot\_get\_device("wheel2");

wb\_motor\_set\_position(left\_motor, INFINITY);

wb\_motor\_set\_position(right\_motor, INFINITY);

/\*

wb\_motor\_set\_velocity(left\_motor, 0.0);

wb\_motor\_set\_velocity(right\_motor, 0.0);

\*/

while (wb\_robot\_step(TIME\_STEP) != 1) {

/\* read sensor values \*/

const double ls0\_value = wb\_light\_sensor\_get\_value(ls0);

const double ls1\_value = wb\_light\_sensor\_get\_value(ls1);

const double ls2\_value = wb\_light\_sensor\_get\_value(ls2);

printf("%f %f %f \n", ls1\_value,ls0\_value,ls2\_value);

double left\_speed = 10;

double right\_speed = 10;

if(rotation\_counter >0){

//真後ろにあるので回転中

rotation\_counter--;

left\_speed = -5;

right\_speed = 5;

}else if(close\_flag == 1 ){

//近づいて止まる

break;

}else if(counter\_left >0){

counter\_left--;

left\_speed = 0;

right\_speed = 10;

printf("a\n");

}else if(counter\_right > 0){

counter\_right--;

left\_speed = 10;

right\_speed = 0;

printf("b\n");

}else{

if((ls2\_value > ls0\_value)&&(ls2\_value > ls1\_value)){

//ボールがロボットの真後ろにある時　回転する

rotation\_counter=20;

}else if(ls1\_value > ls0\_value){

//向かって左にボールがある

counter\_left = 1;

}else if(ls1\_value < ls0\_value){

//向かって右にボールがある

counter\_right = 1;

}

else if(ls0\_value==1024&&ls1\_value==1024){

//真正面にあるとき

close\_flag = 1;

}

}

/\* Set the motor speeds. \*/

wb\_motor\_set\_velocity(left\_motor, left\_speed);

wb\_motor\_set\_velocity(right\_motor, right\_speed);

}

while (wb\_robot\_step(TIME\_STEP) != 1) {

double ds\_values[2];

for (i = 0; i < 2; i++)

ds\_values[i] = wb\_distance\_sensor\_get\_value(ds[i]);

printf("%f %f \n",ds\_values[0],ds\_values[1]);

double left\_speed = 10;

double right\_speed = 10;

if(stop\_flag == 1){

break;

}else {

if(ds\_values[0] < 1000 || ds\_values[1] < 1000 ){

stop\_flag =1;

}

}

wb\_motor\_set\_velocity(left\_motor, left\_speed);

wb\_motor\_set\_velocity(right\_motor, right\_speed);

}

wb\_motor\_set\_velocity(left\_motor, 0);

wb\_motor\_set\_velocity(right\_motor, 0);

int rotation\_counter\_2 = 0;

double \*role\_value;

role\_value = wb\_inertial\_unit\_get\_roll\_pitch\_yaw(inner);

printf("現在の角度　%f\n",role\_value[2]);

//黄色のゴールだと　3.14の角度の方向に

//青色のゴールだと０の方向に回転する

//今回は黄色のゴールに向かう

if(role\_value[2]<0.45 && 0 <=role\_value[2]){

rotation\_counter\_2 = 40;

}else if(role\_value[2]<0.9 && 0.45 <=role\_value[2]){

rotation\_counter\_2 = 30;

}

else if(role\_value[2]<1.35 && 0.9 <=role\_value[2]){

rotation\_counter\_2 = 24;

}

else if(role\_value[2]<1.8 && 1.35 <=role\_value[2]){

rotation\_counter\_2 = 17;

}

else if(role\_value[2]<2.25 && 1.8 <=role\_value[2]){

rotation\_counter\_2 = 14;

}

else if(role\_value[2]<2.7 && 2.25 <=role\_value[2]){

rotation\_counter\_2 = 9;

}else if(role\_value[2]<3 && 2.7 <=role\_value[2]){

rotation\_counter\_2 = 3;

}

else if(role\_value[2]>= -0.45 && 0 > role\_value[2]){

rotation\_counter\_2 = -40;

}

else if(role\_value[2]>= -0.9 && -0.45 > role\_value[2]){

rotation\_counter\_2 = -30;

}else if(role\_value[2]>= -1.35 && -0.9 > role\_value[2]){

rotation\_counter\_2 = -24;

}else if(role\_value[2]>= -1.8 && -1.35 > role\_value[2]){

rotation\_counter\_2 = -17;

}

else if(role\_value[2]>= -2.25 && -1.6 > role\_value[2]){

rotation\_counter\_2 = -14;

}

else if(role\_value[2]>= -2.7 && -2.25 > role\_value[2]){

rotation\_counter\_2 = -9;

}else if(role\_value[2]>= -3 && -2.7 > role\_value[2]){

rotation\_counter\_2 = -3;

}

while(wb\_robot\_step(TIME\_STEP) != 1){

double left\_speed = 10;

double right\_speed = 10;

if(rotation\_counter\_2 == 0 ){

//回転する必要なし

printf("a\n");

break;

}else if(rotation\_counter\_2 > 0){

//左周りに回転してゴールに正対する

rotation\_counter\_2--;

left\_speed = -2;

right\_speed = 2;

printf("b\n");

}else {

//右周りに回転してゴールに正対する

rotation\_counter\_2++;

left\_speed = 2;

right\_speed = -2;

printf("c\n");

}

wb\_motor\_set\_velocity(left\_motor, left\_speed);

wb\_motor\_set\_velocity(right\_motor, right\_speed);

}

//ゴールまで運ぶ

wb\_motor\_set\_velocity(left\_motor, 10);

wb\_motor\_set\_velocity(right\_motor, 10);

wb\_robot\_cleanup();

return 0;

}

５．実行結果

成功例のスナップショットを連続でのせる。

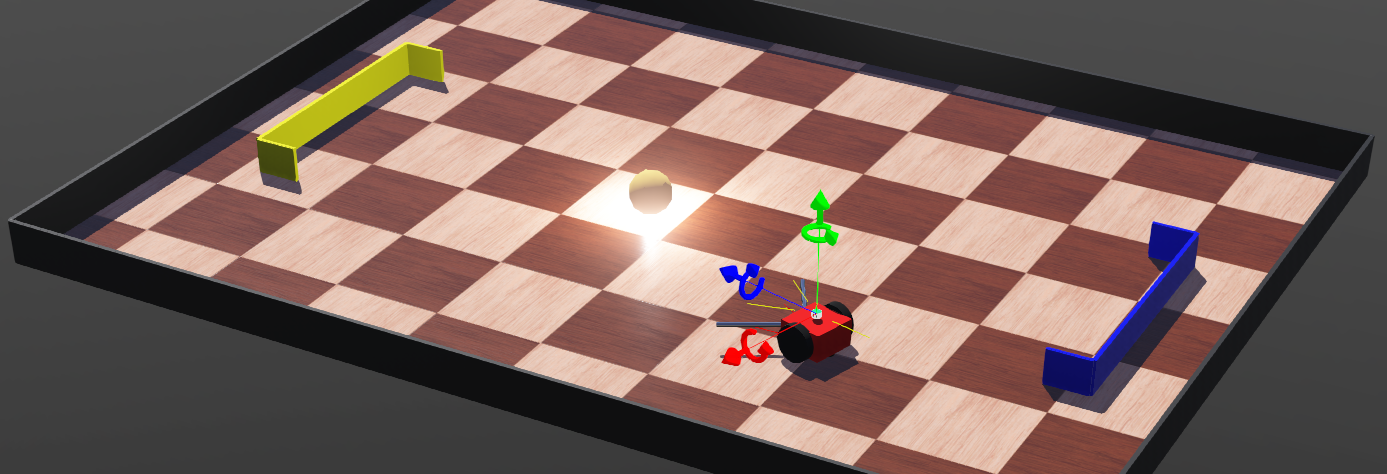
状態１：初期状態

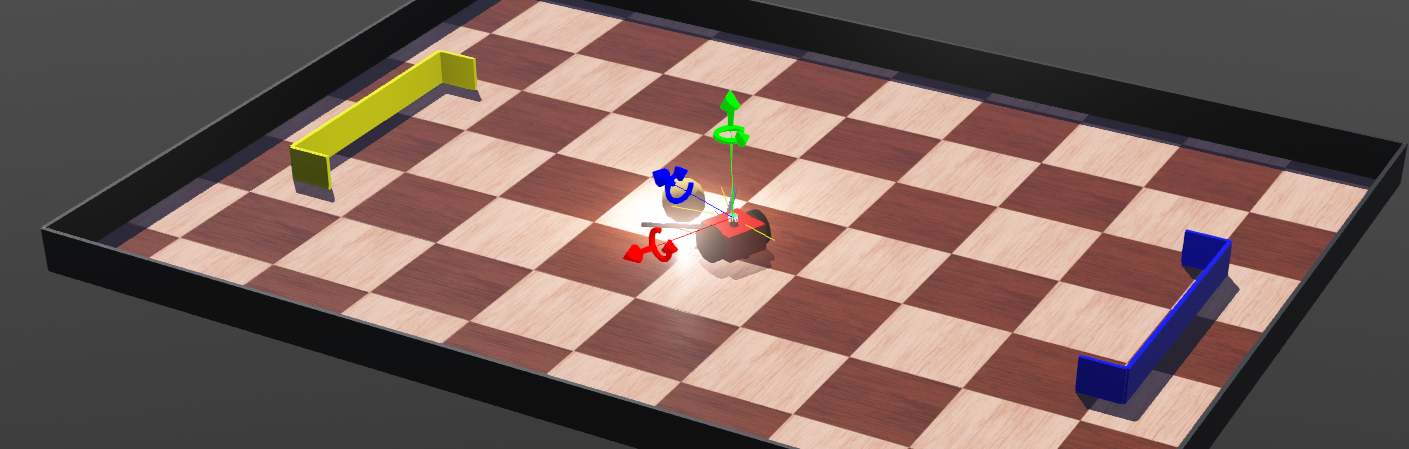
状態２：光センサを使ってボールにある程度近づいた状態

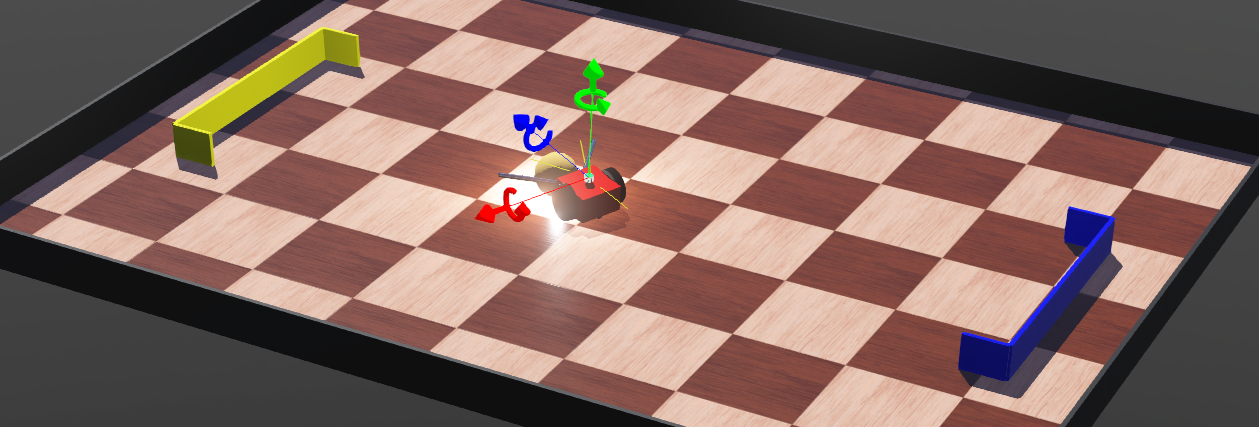
状態３：距離センサを使ってアームの内側にボールを置いた状態

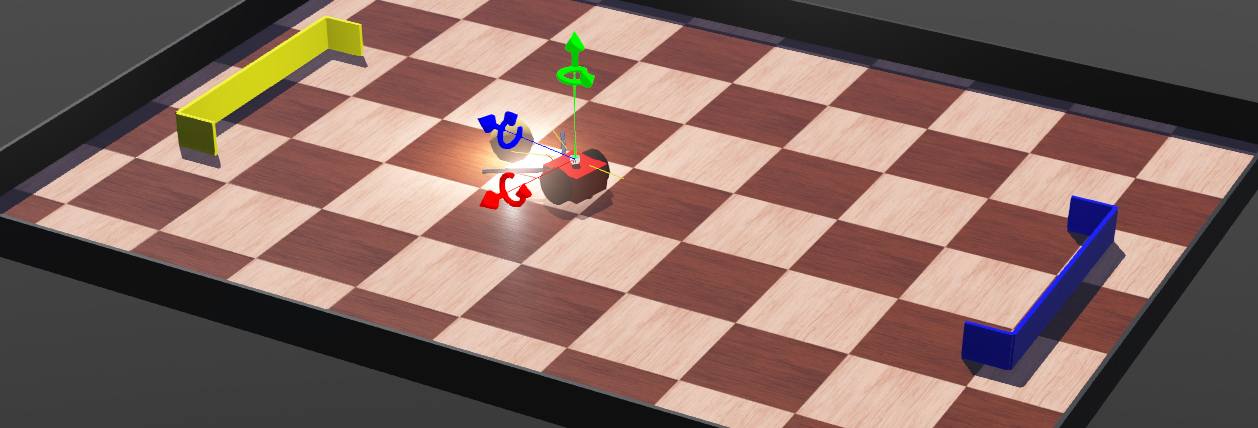
状態４：角度センサを使ってゴール方向に回転した状態

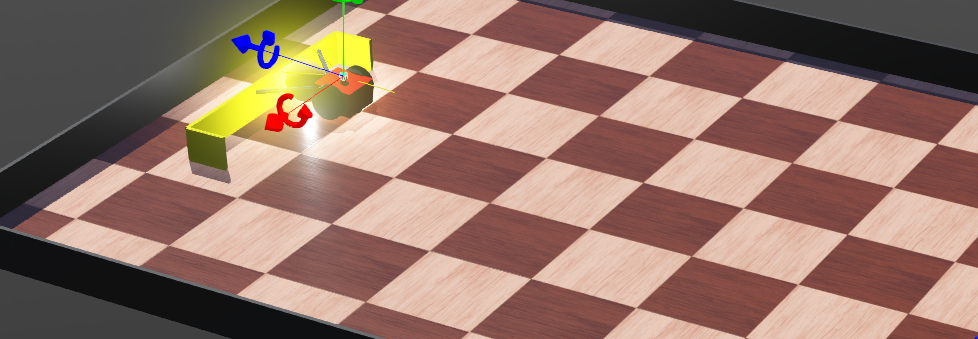
状態５：ゴールした状態

写真１　状態１

写真２　状態２

写真３　状態３

写真４　状態４

写真５　状態５

６．今後の課題

今回、解決しきれなかった点は以下の４つ。

１．ボールへの近づき方の効率化を図る。

現在はライトセンサーの値が異なると、すぐにロボットが進む方向を修正している。方向修正の頻度が多くて、ロボットのスピードがでない。

「ライトセンサーの左右の値の差が小さいときは、方向修正せずにとりあえず直進する。」みたいなアルゴリズムを組むと、ロボットのスピードが損なわれることもなく、もっと早い時間でボールにたどり着けると考える。

２．ロボットが回転したときに、アームの内側にセットしたボールが、飛び出る。

ロボットが回転したときに、アームがボールを外側に押し出してしまうことがある。

アームの形を工夫する。例えばアームの形を楕円にしてその焦点をアームの内側に設定したら、回転したときにボールがアームにあたっても、アームの内側にボールは跳ね返るのではないか。

３．壁際でアームが壁に引っかかり、身動きが取れなくなる。

アームが大きすぎると壁に引っかかりやすい。

４任意の位置のボールをゴールに運ぶ。

現在はボールがフィールド中央にあるときだけゴールに運べる。

角度センサとGPSセンサを組みわせるとゴールへの向きを計算できる。