/100

指導教員氏名　高橋　直樹

情報工学実験報告書

ROBOCODE関する実験

報告者	第4学年　出席番号　27
	氏名　吉田裕明
実験日	1回目　2005年10月11日(火)　9回目　2005年12月20日(火) 2回目　2005年10月25日(火)　10回目　2005年12月21日(火) 3回目　2005年11月 1日(火)　11回目　2006年1月17日(火) 4回目　2005年11月 8日(火)　12回目　2006年1月24日(火) 5回目　2005年11月15日(火)　13回目　2006年2月 7日(火) 6回目　2005年11月22日(火)　14回目　2006年2月 14日(火) 7回目　2005年12月 6日(火) 8回目　2005年12月13日(火)
実験所要時間	56時間00分
実験場所	情報通信実験室
提出期限	2006年3月3日(水) 17:00
提出日	2006年 3月 3日(水)

1．課題実験の目標

強いロボットを作る。

2．開発環境

OS	Fedora Core 1 Linux
使用した言語	J2SDK 1.4.2_04 (Java 言語)
使用したソフト	Robocode

3．Robocode について

（1）Robocode とは

Robocode とは IBM が作成したソフトウェア。砲身・レーダーの２つをもつロボット戦車を作成し、戦わせることによって、ゲームを楽しみながら Java 言語を学ぶことができる。

（2）Robocode の入手方法

Robocode は IBM の Robocode ホーム(http://www-06.ibm.com/jp/event/robocode/home/)からダウンロードすることができる。

Robocode は最低でも次のような環境なら動作すると保障されている。

・Pentium Ⅱ 400MHz、または同等の CPU

・64MB RAM

・約 10MB の空きディスクスペース、およびインストール用の一時スペース

さらに Java™ 2 version 1.3 かそれ以上のバージョンのものがインストールされていることが条件となる。これはサン・マイクロシステムズのサイトから(http://java.sun.com/javaee/downloads/)からダウンロードできる。

（3）ロボット戦車の作成方法

ロボット戦車を作成する手順は

①戦略を決定する。

②戦略の具体的な戦術を決定する。

③戦術を実現するためのアルゴリズムを考える。

④プログラミング作業

⑤コンパイルし、動作確認をする。戦術通りでなければ修正する。

となる。

（4）プログラミング

プログラミングは Robocode の Robot Editor で行う。

Robot Editor は

[Robot]→[Editor]の順にクリックすると起動できる。

新規にロボット戦車を作成する場合は

[File]→[New]→[Robot]とクリックし、作成するロボット戦車の名前、パッケージ名を入力する。

この時ロボット戦車の名前の最初は大文字でなければならない。

そして開かれたエディタにソースコードを入力し、[Compiler]→[Compile]をクリックしてコンパイルし、エラーがあれば、ソースコードを修正し、なければ動作確認をする。

最初エディタにはサンプルコードが記述されているのでそのままコンパイルしてもロボット戦車として動作する。

（5）基本的メソッドの説明

ここでは最初に記述されているサンプルコードについて説明する。

```
public class Test extends Robot
{
	public void run() {
		while(true) {
			ahead(100);
			turnGunRight(360);
			back(100);
			turnGunRight(360);
		}
	}

	public void onScannedRobot(ScannedRobotEvent e) {
		fire(1);
	}

	public void onHitByBullet(HitByBulletEvent e) {
		turnLeft(90 - e.getBearing());
	}
}
```

（i）run メソッド

このメソッドには何もイベントが発生していない時の処理が記述されている。

このロボット戦車では

100 前進
砲身を 360 度右に回転
100 後退
砲身を 360 度右に回転
という動作を繰り返している。

（ii）onScannedRobot メソッド
このメソッドには相手を発見した時の処理が記述されている。
相手を発見することはレーダーを回すことでできる。
このロボット戦車では相手を発見したときに
エネルギーを 1 消費して弾を発射
という動作を行っている。

（iii）onHitByBullet メソッド
このメソッドは相手に接触した時の処理が記述されている。
このロボット戦車では相手に接触したときに
90 度―相手の角度の方向に左回転
という動作を行っている。

4．自機の戦略
ストーキング

5．自機の戦術
相手を追跡しながら攻撃し、相手がこちらに攻撃をしてきたらその場から離脱する。
離脱した後は再び攻撃を再開する。

6．自機の戦術の実現方法
（1）相手がこちらに攻撃してきたらその場から離脱する。
・最初に索敵したときに相手のエネルギー値を保存する。
・次に索敵したときに相手のエネルギーが減少していたなら、攻撃が行われてたということなので、攻撃を避けるために右後方へ移動する。

（2）相手を追跡し攻撃する。
・レーダーを回転させ、相手を捕捉する。
・相手の移動している向き（

角度）を自機の移動する方向に設定し、前進する。

7．プログラム中の各メソッドについての説明

プログラムリストは後半を参照のこと。

大域変数の説明

変数名	型	説明
comp	boolean	エネルギーチェック後の回避動作をするかどうかの判断フラグ
fieldX	double	フィールドの横幅を格納する変数
fieldY	double	フィールドの縦幅を格納する変数
fieldX_min	double	自機が移動できるフィールドの横幅の最小値を格納する変数
fieldX_max	double	自機が移動できるフィールドの横幅の最大値を格納する変数
fieldY_min	double	自機が移動できるフィールドの縦幅の最小値を格納する変数
fieldY_max	double	自機が移動できるフィールドの縦幅の最大値を格納する変数
myHead	double	自機が向いている角度を格納する変数
myGunHead	double	自機の砲身が向いている角度を格納する変数
e1	Edata	索敵したときの相手の情報を格納するクラスのインスタンス
e2	Edata	索敵したときの相手の情報を格納するクラスのインスタンス

（1）run()メソッドの説明

（i）働き

・自機の色設定する。

・変数値を設定する。

・カスタムイベントの呼び出すための状態を設定する

・レーダーを動作させる関数 radar()を呼び出す。

（ii）変数

変数名	内容
e1,e2	Edata クラスのインスタンス

fieldX	フィールドの横幅
fieldY	フィールドの縦幅
fieldX_min	“50” ロボットが移動する範囲の左端になる値
fieldX_max	fieldX から fieldX_min を引いたもの。ロボットが移動する範囲の右端になる値
fieldY_min	“50” ロボットが移動する範囲の下端になる値
fieldY_max	fieldY から fieldY_min を引いたもの。ロボットが移動する範囲の上端になる値

（iii）カスタムイベントの設定

自機の位置で x 座標が fieldX_min より小さいとき、fieldX_max より大きいとき、またＹ座標が fieldY_min より小さいとき、fieldY_max より大きいときに状態”Wallhit1”に true を設定し、そうでないときは false を設定することとした。座標の原点はフィールドの左下となる。

（iv）戻り値

なし

（2）onScanedRobots()メソッドの説明

（i）働き

・相手の情報を格納

・砲身を操作する関数 gun()の呼び出し

・自機の進む方向と距離を設定する関数 move()の呼び出し

・相手のエネルギーの減少を調べる関数 eneCheck()の呼び出し

（ii）変数

変数名	内容
myHead	自機の向き（角度）
myGunHead	自機の砲身の角度

e1.dis	今回索敵した時の相手との距離
e1.bear	今回索敵した時の相手との相対角度（-180 度～180 度）
e1.velo	今回索敵した時の相手の速度
e1.ene	今回索敵した時の相手のエネルギー
e1.head	今回索敵した時の相手の向いている方向
e1.time	今回索敵した時の時間
e2.dis	前回索敵した時の相手との距離
e2.bear	前回索敵した時の相手との相対角度（-180 度～180 度）
e2.velo	前回索敵した時の相手の速度
e2.head	前回索敵した時の相手のエネルギー
e2.ene	前回索敵した時の相手の向いている方向
e2.time	前回索敵した時の時間

（iii）パラメータ

相手のインスタンス

（iv）戻り値

なし

（3）関数 radar()の説明

（i）働き

・レーダーを操作する関数。

・今の時間と前回索敵した時間が 5 以上の時、レーダーを 360 度右回転させる。

・そうでないときは相手の居る方向にレーダーを向ける。

（ii）変数

変数名	内容
absoluteBear	相手の居る方向（0～360 度）。 自機の向いている方向に、相手の居る相対的な方向から自機のレーダーの向いている方向を引いたものを加えて求める。
radarBear	相手との相対角度。

（iii）戻り値

なし

（4）関数 move()の説明

（ｉ）働き

・自機の進む方向と距離を操作する関数。

・角度は相手との相対角度とする。

・エネルギー減少判断フラグ comp が True の時は回避行動をとる。

（ii）変数

変数名	内容
nextBear	相手との相対角度（-180 度～180 度）
distance	相手との距離 ・相手との距離が 100 以上の時はその距離に 0.6 をかけた距離を進む距離とする。 ・そうでないときは距離を 0 とする。

（iii）戻り値

なし

（５）関数 gun()の説明

（ｉ）働き

・砲身を操作し、相手を攻撃するための関数

（ii）変数

変数名	内容
absoluteBear	相手との居る方向（0～360 度）
gunBear	相手との相対角度（-180 度～180 度）

（iii）戻り値

なし

（６）関数 eneCheck の説明

（ｉ）働き

相手のエネルギーの減少を感知する。

（ii）変数

変数名	内容
comp	相手のエネルギーが減少したかどうかを判断するフラグ

（iii）パラメータ

・前回索敵したときの相手のエネルギー（lastene）

・今回索敵した時の相手のエネルギー（ene）

（iv）戻り値

判断フラグ comp（True/False）

（7）関数 normalRelativeAngle()の説明

（i）働き

・相対的な角度を返す。

（ii）変数

変数名	内容
fixedAngle	相対角度。パラメータで与えられた角度を-180～180 度の範囲に変換する。

（iii）パラメータ

・角度（angle）

（iv）戻り値

相対角度 fixedAngle（-180～180 度）

（8）onCustomEvent メソッドの説明

（i）働き

カスタムイベントで壁の近くに来た時に自機の方向を変え、壁に接触しない動きを実現する。

（ii）変数

変数名	内容
nowX	現在の自機の位置（X 座標）
nowY	現在の自機の位置（Y 座標）
nowHead	現在の自分の向き

（iii）パラメータ

CustomEvent のインスタンス

（iv）戻り値

なし

その他使用したクラス

Edata クラス

（ｉ）働き

相手の情報を保存する。

（ii）変数

変数名	内容
dis	相手との距離を格納
bear	相手との相対角度を格納
velo	相手の速度を格納
ene	相手のエネルギーを格納
head	相手の向いている方向を格納
time	相手を発見した時間を格納

プログラムリスト

J02339_7.java

```
package j02339;
import robocode.*;
import java.lang.*;
import java.awt.Color;

public class J02339_7 extends AdvancedRobot
{
   boolean comp;
   double       fieldX;                  //フィールドの横幅
   double       fieldY;                  //フィールドの縦幅
   double       fieldX_min;
   double       fieldX_max;
   double       fieldY_min;
   double       fieldY_max;
   double       myHead;
   double       myGunHead;
   Edata        e1,e2;

   public void run() {
```

```
        e1 = new Edata();
        e2 = new Edata();
        fieldX = getBattleFieldWidth();
        fieldY = getBattleFieldHeight();
        fieldX_min = 50;
        fieldX_max = fieldX - fieldX_min;
        fieldY_min = 50;
        fieldY_max = fieldY - fieldY_min;

        setColors(Color.cyan,Color.cyan,Color.cyan);

        addCustomEvent(
                new Condition("Wallhit1") {
                        public boolean test() {
                                if((getX() <= fieldX_min)
                                        ||(getX() >= fieldX_max)
                                        ||(getY() <= fieldY_min)
                                        ||(getY() >= fieldY_max)) {
                                        return true;
                                }
                                else
                                        return false;
                        };
                }
        );

        while(true) {
                radar();
        }
}

public void onScannedRobot(ScannedRobotEvent event) {
        myHead = getHeading();
        myGunHead = getGunHeading();
        e1.dis  = event.getDistance();
        e1.bear = event.getBearing();
```

```
        e1.velo = event.getVelocity();
        e1.ene  = event.getEnergy();
        e1.head = event.getHeading();
        e1.time = event.getTime();

        gun();
        move();
        eneCheck(e2.ene,e1.ene);
        execute();

        e2.dis  = e1.dis;
        e2.bear = e1.bear;
        e2.velo = e1.velo;
        e2.ene  = e1.ene;
        e2.head = e1.head;
        e2.time = e1.time;
    }

    public void onWin(WinEvent event) {
        setTurnGunRight(360);
    }

    public void radar(){
        double absoluteBear;
        double radarBear;

        if(getTime()-e1.time > 5)
                turnRadarRight(360);

        absoluteBear = getHeading() + e1.bear-getRadarHeading();
        radarBear = normalRelativeAngle(absoluteBear);

        turnRadarRight(radarBear);
    }

    public void move(){
```

```
        double nextBear;
        double distance;

        if(e1.dis > 100)
                distance = e1.dis*.6;
        else
                distance =0;

        nextBear = e1.bear;
        if(comp){
                setBack(150);
                setTurnRight(nextBear+90);
        }
        else{
                setAhead(distance);
                setTurnRight(nextBear);
        }
}

public void gun(){
        double absoluteBear;
        double gunBear;

        absoluteBear = getHeading() + e1.bear-getGunHeading();
        gunBear = normalRelativeAngle(absoluteBear);
        turnGunRight(gunBear);
        fire(1);
}

public boolean eneCheck(double lastene,double ene) {
        if(lastene <= ene) comp = false;
        else    comp = true;
        lastene = ene;
        return comp;
}
```

```
    public double normalRelativeAngle(double angle) {
        if (angle > -180 && angle <= 180)
                return angle;
        double fixedAngle = angle;
        while (fixedAngle <= -180)
                fixedAngle += 360;
        while (fixedAngle > 180)
                fixedAngle -= 360;
        return fixedAngle;
    }

    public void onCustomEvent(CustomEvent event) {
        if(event.getCondition().getName().equals("Wallhit1")) {
                removeCustomEvent(event.getCondition());
            double      nowX = getX();
                double  nowY = getY();
                double  nowHead = getHeading();
                if (nowX <= fieldX_min) {
                        turnLeft(90-nowHead);
                }
                else if(nowX >= fieldX_max) {
                        turnLeft(270-nowHead);
                }
                else if(nowY <= fieldY_min) {
                        turnLeft(0-nowHead);
                }
                else if(nowY >= fieldY_max) {
                        turnLeft(180-nowHead);
                }
                ahead(200);
        }
    }
}
```

Edata.java

```
package j02339;
```

```
public class Edata
{
        double   dis;
        double   bear;
        double   velo;
        double   ene;
        double   head;
        double   time;
}
```

8．模擬線結果

開発者	千葉	早川	吉田	和田	小田	高橋	浅野	土屋	中田	NTx
千葉	＼	○○○	○○○	×××	×××	×××	××○	○×○	××○	×××
早川	×××	＼	×○×	×××	×××	×××	×××	○××	○○×	×××
吉田	×××	○×○	＼	×××	×××	×××	×××	×××	×××	×○×
和田	○○○	○○○	○○○	＼	○○×	○××	○○○	○○○	○○○	×××
小田	○○○	○○○	○○○	××○	＼	○×○	○○○	○○○	○○○	×××
高橋	○○○	○○○	○○○	×○○	×○×	＼	○○○	○○○	○○○	×××

9．感想

実験の目標の「強いロボットをつくる」ということは実現できなかったが、相手を追跡することが実現できた。相手を追跡する動作や離脱する動作にバリエーションを持たせるともう少し勝てるようになったかと思う。

Java 言語に関しては以前どこかで見たときには全く理解できなかったということがあった。しかし、今回の実験で実際に使ってみることによってすこしは理解できるようになった。そして 3 年生のときに授業で習ったオブジェクト指向の考え方を復習するきっかけともなった。

HTML 言語に関しては使用したことがあるので、HP を作る上でほとんど問題は無かった。カスケード・スタイル・シートで H タグのサイズを設定することは初めてだったので参考になった。

発表資料

目次

Robocode ～ 自律制御戦闘機械 ～

高橋研究室 情報工学科4年 27番 吉田 裕明

共同発表者 24番 早川 会理

もどる

目的

目的

強いロボットを作る。

戻る | インデックスへ | 次へ

開発環境

開発環境

OS	Fedora Core 1 Linux
開発言語	Java言語 J2SDK 1.4.2_04-b05
開発ツール	ROBOCODE

戻る | インデックスへ | 次へ

戦略

戦略

ストーキング

戻る | インデックスへ | 次へ

戦術

戦術

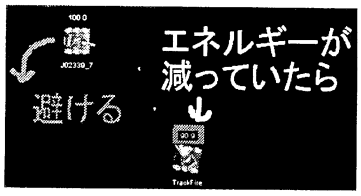
1.相手を追跡 → 攻撃
2.相手に攻撃されたら → 一旦離脱
3.また相手を追跡

戻る | インデックスへ | 次へ

戦術の実現１

戦術の実現1

.「相手に攻撃されたら → 一旦離脱」



.1.相手のエネルギーを見る

.2.攻撃を避ける

戻る | インデックスへ | 次へ

戦術の実現２

戦術の実現2

.「相手を追跡 → 攻撃」

.1.相手をレーダーで捕捉

.2.追跡しながら攻撃

戻る | インデックスへ | 次へ

問題点

問題点

.壁に引っかかってしまう

.↑なかなか抜け出せない時もある

.相手に近づいたときに弾にあたりやすい

戻る | インデックスへ | 次へ

まとめ

まとめ

- 実現できた点
  - 相手を追跡する
    - 相手に向かって移動→攻撃
  - 相手を見失わない
    - レーダーで相手を追う
  - 相手のエネルギーを見て攻撃を避ける
    - 斜めに後退→追跡

戻る | インデックスへ | 次へ

感想

感想

- 戦略のほとんどを実現できた
- java言語を学ぶことができた

戻る | インデックスへ | 次へ

最後に

最後に

課題実験報告用ページ
http://nt.hakodate-ct.ac.jp/~j02339/

戻る | インデックスへ | 次へ

定期報告

第 1 回	2005/10/11 13:00-14:30
活動内容	簡単な課題実験の説明 実験の日程(だいたいの室内発表の日など)
今回の成果	ROBOCODE の開発手順を書いたプリント
現在の問題点	UNIX と Java は初めて使うので知識が無い。
次回の目標	UNIX と Java について覚えてくる。 ROBOCODE をインストール、動作確認をする。

第 2 回	2005/10/25 13:00-17:30
活動内容	課題実験についての具体的な説明 課題実験について具体的な説明を受けた。 実験は UNIX と Java を使用して進めていく。 概要と計画と定期報告が毎週必要で、これらは web ページとして作成する。 それらのページは完成したら、HTTP サーバーにアップロードする。 完成するまでは File サーバーに保存する。 UNIX のコマンド等は高橋研究室の web ページや配布された資料を参考にすること。 Java の実行環境の整備 http://nt.hakodate-ct.ac.jp/~takahasi/java/index.html を参考に整備した。 ROBOCODE の動作確認 上記のページを参考にした。 エディタ等の動作確認をおこなった。 デモをおこなった。 課題実験の web ページの作成 QUANTA を使用して作成した。
今回の成果	課題実験の web ページ UNIX の使いかたが少しわかった。 ROBOCODE の起動方法までの操作など。
現在の問題点	ROBOCODE の起動方法はわかった。 ROBOCODE の概要等を考えていない。次回までに考える。
次回の目標	ROBOCODE の概要等を考える

第 3 回	2005/11/01 13:00-17:30
活動内容	概要ページの作成 戦略は「ストーキング」 戦術は「相手を追跡して攻撃する。相手に攻撃されたら距離をとる。」 というもの ROBOCODE でのロボットの作成 http://www.takahashi.lab/~takahasi/nt/labo/robocode/api_j/index.html を参考に作成した。 まずは壁に当たった時に壁から離れる行動を作成し、試した。 ー onHitWall イベントで前に進む行動をとる。
今回の成果	課題実験の概要ページ ROBOCODE のロボット
現在の問題点	ROBOCODE の概要は決まったが、具体的にどのような行動をとれば実現できるか 壁に当っても動かなくなることがあった。
次回の目標	相手を追跡するために具体的にどうするかを考える。 壁に当って動かなくならないようにすることを考える。 相手のいる位置へ移動する手順を考える

第 4 回	2005/11/08 13:00-17:30
活動内容	ROBOCODE でのロボットの作成 http://www.takahashi.lab/~takahasi/nt/labo/robocode/api_j/index.html を参考に作成した。 onScannedRobot イベント発生時に turnRight(e.getBearing()); ahead(e.getDistance() - 70); の文を実行して、相手に接近する動きを作成した。 基本の動きを sample.wall にした。
今回の成果	ROBOCODE のロボット
現在の問題点	相手に後ろから接近できない。 解決法 ー 相手の軌跡をたどることができればいいかもしれない。
次回の目標	相手の軌跡をたどれるならその方法を考える。 まず止っている相手に後ろから接近する方法を考える。

第 5 回	2005/11/15 13:00-17:30

活動内容	ROBOCODE でのロボットの作成
	http://www.takahashi.lab/~takahasi/nt/labo/robocode/api_j/index.html を参考に作成した。
	基本の動きを wall ではなく、turnGunRight()の無限ループにし、 常時相手をスキャンする事にした。
	静止している相手に後ろから近づく動きを作成した。
	相手との距離をスキャンして、相手の後ろまで直角三角形の辺の動きで 近づく事ができた。
	turnRight(bear[0] + 30);
	ahead(dis[0] + 100);
	turnLeft(120);
	ahead(dis[0] / Math.sqrt(3));
	turnLeft(90);
	相手との距離が 200 以上の時は上のコードを実行する。
	相手との距離が 200 以下で 80 以上の場合は距離が 80 になるまで 徐々に近づく。
	動作を終えた後に弾を射つ。

今回の成果	ROBOCODE のロボット(相手の後ろへ近づく動き)
現在の問題点	壁に当った時におかしな動きになる 解決法 ― 相手に近づく方法が一つだからだと思う。 壁に当った時に方向を切換えられる様にする。 近づく動きが直線的過ぎる。 解決法 ― 直線的でない動きで近づく様にする。例えば円の弧の様に。 動いている相手には通用しない。 解決法 ― 動いている相手に通用するものを考える。 壁に当った時の対処を考える。
次回の目標	直線的に近づかない方法を考える。 動いている相手への動作を考える。

第 6 回	2005/11/22 13:00-17:30
活動内容	ROBOCODE でのロボットの作成 http://www.takahashi.lab/~takahasi/nt/labo/robocode/api_j/index.html を参考に作成した。 相手に近づく動きに円運動を取り入れた。 最初は前回の三角形の動きを改良しようとしたが、新しく作りなおした。
今回の成果	ROBOCODE のロボット(相手の後ろへ近づく動き)
現在の問題点	円運動の調整が必要。 setTurnRight の角度等の調整が必要。 近づく動きが直線的。 解決法 – 曲線的な動きで近づく様にする。 動いている相手には通用しない。 解決法 – 動いている相手に通用するものを考える。(実現方法等) 円運動の調整。
次回の目標	直線的に近づかない方法を考える。 動いている相手への動作を考える。

第 7 回	2005/12/06 13:00-17:30
活動内容	ROBOCODE でのロボットの作成 http://www.takahashi.lab/~takahasi/nt/labo/robocode/api_j/index.html を参考に作成した。 前回作成した円運動の調整を行った。 動いている相手への対策として直線的な動きを読むアルゴリズムを着手した。
今回の成果	ROBOCODE のロボット(相手の後ろへ近づく動き) 直線的な動きを読むアルゴリズムが未完成
現在の問題点	解決法 – 実現方法を考える。 相手を見失なう事がある。再び見付けるまでに時間がかかる。 解決法 – 相手を見失なわない方法を考える。
次回の目標	直線的な動きを読むアルゴリズムの完成。相手の動きを読む。

第 8 回	2005/12/13 13:00-17:00
活動内容	ROBOCODE でのロボットの作成 http://www.takahashi.lab/~takahasi/nt/labo/robocode/api_j/index.html を参考に作成した。 前回途中までだった直線運動の予測を作成した。 敵の情報を保存する Edata クラスを作成した。
今回の成果	直線運動の予測アルゴリズム 敵の情報を保存する Edata クラス
現在の問題点	直線的な動きを読むアルゴリズムで Wall 相手にそこそこ攻撃できる。 解決法 — 大砲の回転するために必要な時間を計算すれば、より当たりやすいのではないか。 相手を見失なう事がある。再び見付けるまでに時間がかかる。 解決法 — 相手を見失なわない方法を考える。
次回の目標	大砲の回転時間をもとめるアルゴリズムの作成。 相手を見失なわない方法。

第 9 回	2005/12/20 13:00-17:00
活動内容	ROBOCODE でのロボットの作成 http://www.takahashi.lab/~takahasi/nt/labo/robocode/api_j/index.html を参考に作成した。 線形予測をするときと予測をしない時の行動を分けた。 弾にあたった時の行動を変更した。
今回の成果	直線運動の予測するときとそうでないときの動き 被弾したときの動き
現在の問題点	相手を見失なう事がある。再び見付けるまでに時間がかかる。 解決法 — 相手を見失なわない方法を考える。
次回の目標	相手を見失なわない方法。

第 10 回	2006/01/17 13:00-17:00
活動内容	ROBOCODE でのロボットの作成 http://www.takahashi.lab/~takahasi/nt/labo/robocode/api_j/index.html を参考に作成した。 相手をレーダーから外れない動きを作成した。 弾にあたった時の行動を変更した。
今回の成果	相手をレーダーに捉え続ける。 被弾したときの動き
現在の問題点	相手の弾に当ってしまう 解決法 ― 動きながら相手を捉え続ければいいと思う。 被弾した時の相手から逃げる動きで壁に当ってしまう。 解決法 ― 壁に当たらないように動きを作成する。 (フィールドの大きさをもとにギリギリまで動けるような)
次回の目標	動きながらでも相手を見失なわない方法。

第 11 回	2006/01/24 13:00-17:00
活動内容	ROBOCODE でのロボットの作成 http://www.takahashi.lab/~takahasi/nt/labo/robocode/api_j/index.html を参考に作成した。 壁にあたらないようにカスタムイベントを作成した。うごいている相手にも弾があたるアルゴリズムを作成した。
今回の成果	相手をレーダーに捉え続ける。 動きながら相手をねらう
現在の問題点	動きながら攻撃することで相手の弾には当たりにくくなったが、逆に自分の攻撃は当たりにくくなった。 解決法 ― 攻撃が相手に到達する時間を考えて、砲身の角度を決めているが その時間を求めるアルゴリズムをみなおす。 壁にあたった後の動きがおかしい。 解決法 ― カスタムイベント周りをみなおす。
次回の目標	時間を求めるアルゴリズムの見直し カスタムイベントの見直し 発表用資料の作成

第 12 回	2006/01/31 13:00-17:00
活動内容	ROBOCODE でのロボットの作成 http://www.takahashi.lab/~takahasi/nt/labo/robocode/api_j/index.html を参考に作成した。 壁にあたらないようにカスタムイベントを作成した。 発表用ページの作成
今回の成果	壁にあたらない動きをカスタムイベントで作成。
現在の問題点	発表用ページがまだできていない。 解決法 ― 次回までに完成させる。
次回の目標	時間を求めるアルゴリズムの見直し 発表用ページの作成

第 13 回	2006/02/07 13:00-17:00
活動内容	室内発表 模擬戦も行った。
今回の成果	発表用ページの改良点がみつかった。
現在の問題点	発表用ページへ図の追加。 ページの追加。
次回の目標	発表用ページの作成。

第 14 回	2006/02/14 13:05-16:30
活動内容	全体発表
今回の成果	壁にあたらない動きをカスタムイベントで作成。
現在の問題点	発表の準備が不十分だった。 ―しっかりと準備をする。
次回の目標	課題実験レポートの作成。