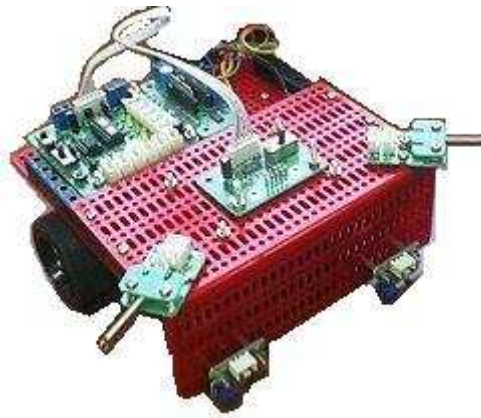


自分で作ったプログラムで



ロボットを動かそう!

！ 目次 ！

『TiColla』の準備

はじめに

ロボットを動かしてみよう

ダウンロードの方法

繰り返し動作をさせよう

タッチセンサーを使ってみよう

ライトセンサーを使ってみよう

ライントレースをしよう

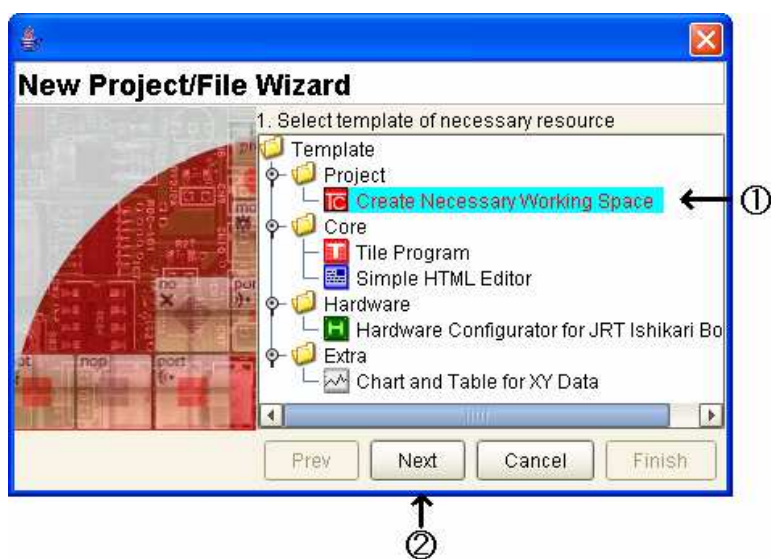
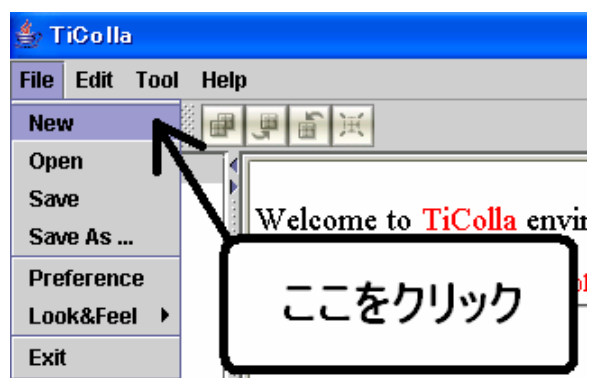
ロボットを使ってサッカーをしよう

グレースケールを使った陣地の判断法

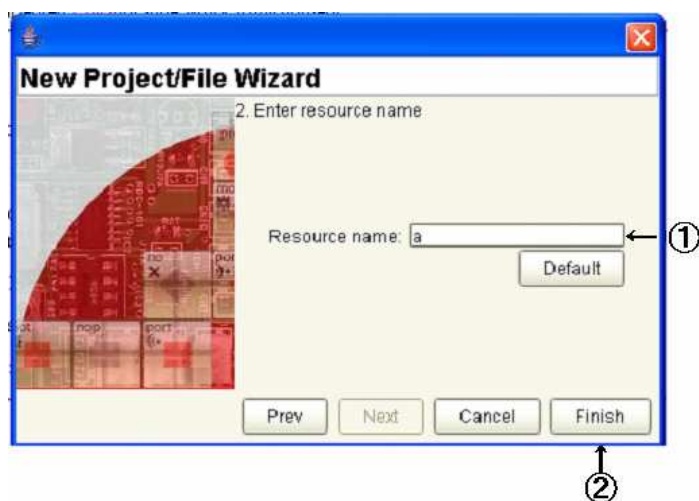
サッカープログラムを作ろう

『TiColla』の準備

プログラムを起動したら、最初に一番左上にある『File』にマウスのアイコンをあわせるとメニューが開くので、その中から『New』を選びクリックしてください。（右図参照）



そうすると上のような画面が現れるので、最初に をクリックして選択し（文字が青く囲まれた状態にする）、次に の『Next』をクリックします。

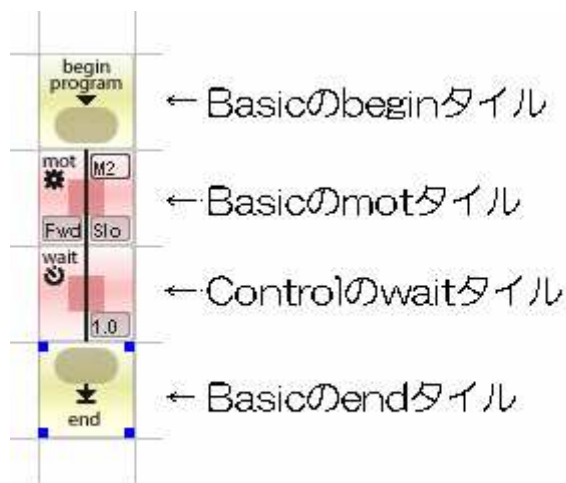


さらに画面が変わるので のテキストボックスに半角英数字でファイル名

ロボットを動かしてみよう

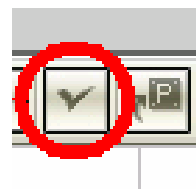
それでは実際にタイルプログラミングをやってみましょう。まず下に示すようにタイルを並べてください。

！ タイルを並べる時にはなるべく縦にまっすぐ並べること ！



上図のようにタイルを置くことができれば、次は作ったプログラムをダウンロードする方法について説明します。

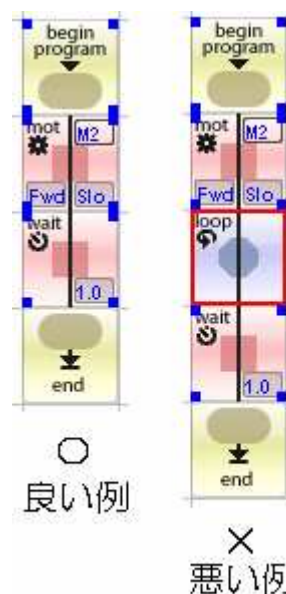
まず、作ったプログラムに間違いがないかチェックする必要があります。プログラムのチェックには作業画面の中央上部にあるチェックボタン（右図参照）をクリックします。



チェックボタンをクリックすると左図のようにend タイル以外のタイル全てにマークが付きます。問題のないタイルの四隅には青いドットが、何か問題のあるタイルは赤線で囲まれます。

end タイル以外の全てのタイルに青いドットが付けば、そのプログラムはダウンロードすることができます。

次のページでダウンロードの方法を示します。



- ・ begin、end タイル
- ・ mov タイル
- ・ wait タイル

実際にこれらを使って問題に挑戦してみましょう。

<問題 1 >

上にある のプログラムを変更（それぞれのタイルの設定を変更）し

後退

90° 右回転

90° 左回転

180° 回転（左右どちらからでも可）

360° 回転（左右どちらかでも可）

するプログラムをそれぞれ作りなさい。

！ ヒント ！

- ・ 回転において最も重要な事は、目標地点までの秒数である
- ・ 秒数をすこしずつ調整しながら、何度も実行して近づけることが大切

！ ポイント ！

回転する方法は2種類あります。

1： 両方のタイヤを動かして回転する方法

2： 片方のタイヤを停止させて、コンパスのように回転する方法

この2つの方法を見たときに円の直径を考えてください。

1の方法で回転を行ったときは、タイヤの軌跡から、2つのタイヤ間の距離が円の直径になります。また、その場で回転することになります。

2の方法で行ったときには、コンパスのように回転するのですから、タイヤ間の距離が円の半径になります。また、回転後のロボットの位置は若干変わることになります。

このように、同じ回転でもスピードや移動距離に違いがでます。

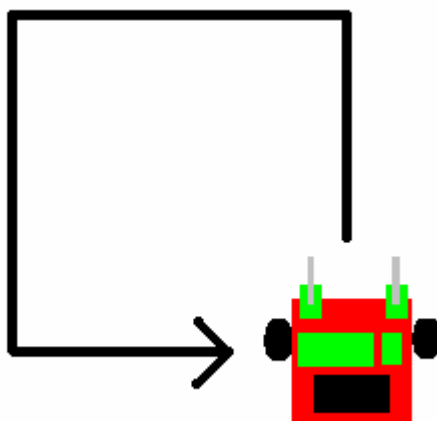
回転方法は用途によって使い分けることで動作効率を上げることができます。

ただし、今回のように回転することだけが目的の場合は、上のどちらの方法を使っても構いません。

繰り返し動作をさせよう

<問題 2 >

下の図のようにロボットが動くようにプログラミングしなさい。
(1周して戻ってくるプログラムで、ターンは90°にすること)

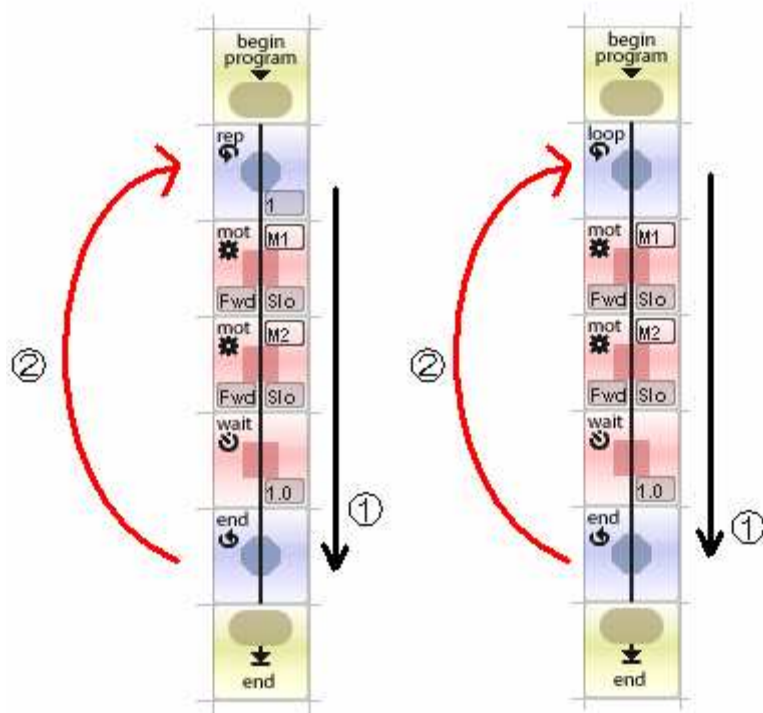


！ ヒ ン ト ！

- ・ ある二つの動作を4回書けばよい

一見複雑に見える問題ですが、前進 左回転を4回繰り返すだけの単純なプログラムで構成できます。しかし全てのタイルを並べるのは大変です。

問題2のように、一定の動作を繰り返すようなプログラムを書く場合には『繰り返し』を命令するタイルを使います。繰り返し命令のタイルには rep タイルと loop タイルの2種類が用意されています。



rep タイル・ loop タイル・ ループ end タイル

rep タイルと loop タイルはどちらもループ end タイル(青い end タイル)との間に挟まれた命令を繰り返すというものです。

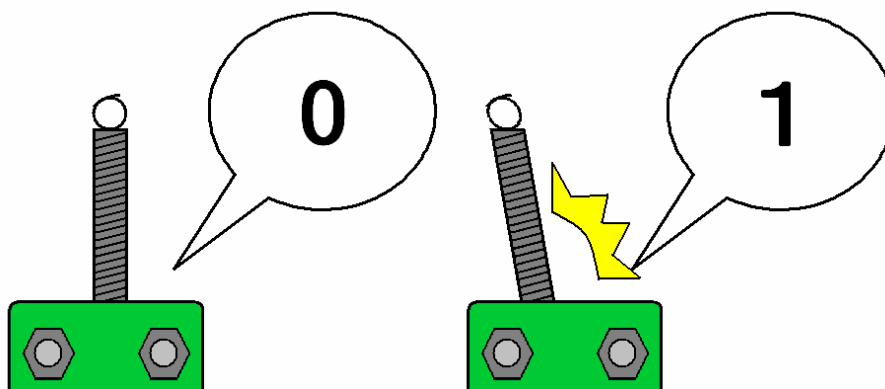
まず rep タイルか loop タイルを見つけたあとは普通に順番どおり命令を実行していきます。(の矢印)そしてループ end タイルを見つけると rep タイルか loop タイルのあった場所まで戻るといふものです。(の矢印)

rep タイルと loop タイルの見てわかる違いは、タイルにある数字です。rep タイルのほうには数字があるのがわかります。

rep タイルは繰り返す回数を自分の任意の回数に設定することができます。一方 loop タイルですが、繰り返す回数の設定が無いので、プログラムを実行したときロボットは止まることなく動き続けます。これを止めるにはロボット本体の電源を切る必要があります。

タッチセンサーを使ってみよう

皆さんのロボットには2種類のセンサーがついています。1つはタッチセンサー、もう1つはI R（赤外線）センサーです。最初にタッチセンサーの学習から始めます。



タッチセンサーとは、その名前のとおり触覚をつかさどるセンサーで、人間でいうと手にあたる部分です。タッチセンサーは前方に突き出ているバネが特徴で、普段は左右にあるネジ部分に当たらないようになっています。

上の図のように左右のネジ部分にバネが当たっていないとき、つまり何にもぶつかっていないときはセンサーの信号は“ 0 ”です。

このバネが衝突などの原因で曲がることで、左右のネジ部分に当たるとセンサーが反応し、ぶつかった事を示す信号“ 1 ”が返ってきます。

この信号を考慮したプログラムを書くことによって障害物を避けるなど、アクティブな動作が可能になります。

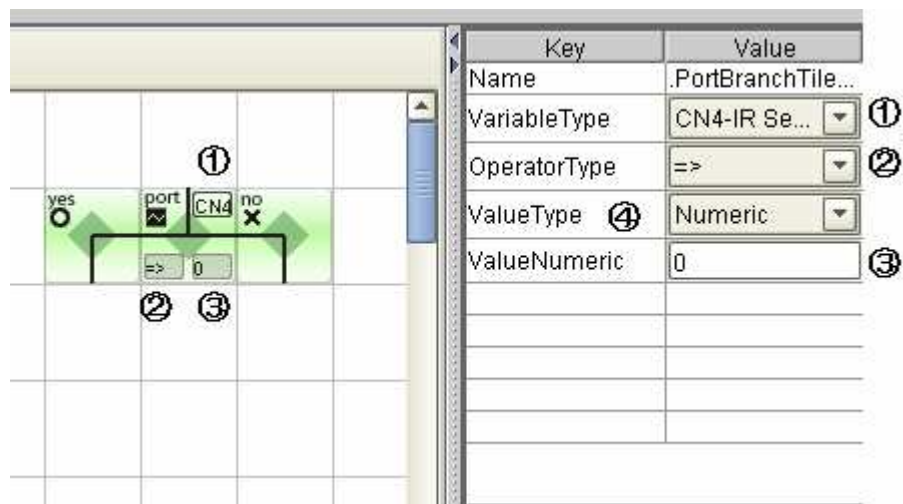
最初の基板設定の事を思い出してください。タッチセンサーが書かれたタイルを基板の『CN1』と『CN2』に配置しました。これには意味があり、CN1に左右どちらのセンサーを繋いだか（CN2についても同様）ということ意識しなければいけません。

実際にロボットを確認してみましょう。

| チャンネル | センサー（左 or 右） |
|-------|--------------|
| CN1 | |
| CN2 | |

ライトセンサーを使ってみよう

つぎはもう一つの『A port タイル』を使ってライトセンサーを使ってみましょう。



A port タイル

この『A port タイル』は0から255までの値をつかって光の強さを表現します。

タイルをクリックすると青くマークされます。

センサーのチャンネルを決定するものです。

ボックス右の をクリックすると選択できます。どちらのセンサーがCN3かCN4かを知るにはセンサーから出ているケーブルをたどると、つないである場所（基板）にCN3かCN4と書いてあります。

センサーから得られた数値データと比較する方法を決定するものです。

ボックス右の をクリックすると『 = < 』（センサーで得られた数値が で指定した値よりも小さい）『 = > 』（センサーで得られた数値が で指定した値よりも大きい）のどちらか選択できます（『 = < = > |』というのも出てきますが、これらのパラメータは使用することができないので選択しないで下さい）。

センサーで得られた数値と比較するための値を指定する場所です。

センサーの数値とは光の強さのことです。（暗い時0～明るい時255）のボックスをクリックすると数字が反転するのでバックスペースキーで削除します。自分の任意の値を半角英数字で入力し、最後に enter キーを押してください。

！ 注 意 ！

のパラメータは変更しないでください

全てのパラメータを設定したら画面中央の作業スペースか別のタイルをクリックしてみてください。A port タイルの が設定したとおりに変化すれば設定は完了です。

(上の画面はセンサーから得られた数値が『0以上かどうか』という分岐)
ライトセンサーというのは赤外線という種類の、目には見えない光を検知します。
そして検知された光は数値

[(何も検知されない) “ 0 ”] ~ [“ 2 5 5 ” 非常に強い赤外線が検知された)]
に置き換えられます。

<問題 5 >

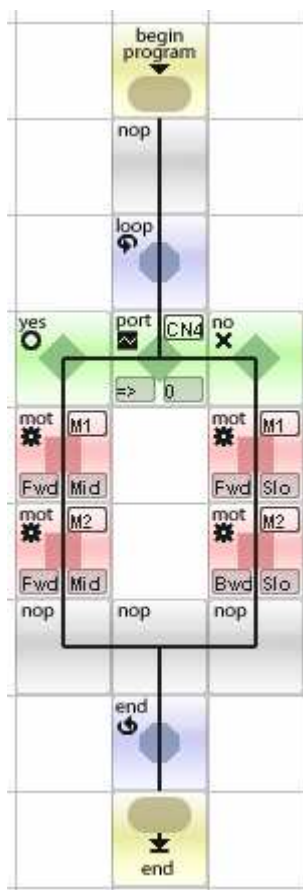
赤外線を発するボールを追いかけるプログラムを作りなさい。

！ ヒ ン ト ！

センサーの値をうまく調整することが肝心です。どの程度の強さの赤外線（どの程度の数値）ならボールとして認識するのかという基準値を、Aport の のボックスで比較する値を変えることにより調整します。基準値が小さいと太陽光や照明にも反応するので、ロボットは反応し続けてしまいます。反対に基準値を大きくしすぎると、赤外線に反応しにくくなるのでロボットは赤外線ボールを見つけることができなくなってしまいます。

ロボットを動作させる環境によってセンサーが感知する値は常に変わります。そのため、少しずつ基準値を変えて何度も試行する必要があります。時間もかかり根気がある作業ですが丁寧に調節することが鍵となります。

下にプログラム例を示します。はじめ Aport の値は0 になっているので、最適な値を探し出して設定しましょう。

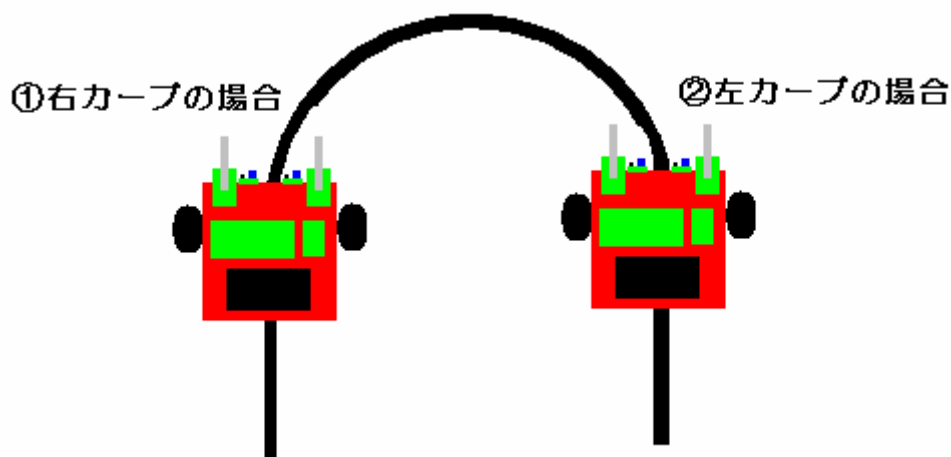


ライトレースをしよう

白と黒を見分けることができれば、次はライトレースです。

ライトレースとは床に引かれた黒い線をロボットにたどらせるというものです。

センサーを左右2つに増やして、先ほど書いた白と黒を見分けるプログラムを書きます。センサーを2つ使うプログラムの書き方（2つの分岐を用いた方法）はタッチセンサーのところを参考にしてください。



右カーブに差し掛かった場合ロボット右側のセンサーが線の黒を検知します。この場合、少し後退してから右にほんの少しだけ回転するようにしてください。

（回転しすぎると左側のセンサーが線よりも外側にきて道を外れてしまいます）

左カーブに差し掛かった場合ロボット左側のセンサーが黒を検知します。この場合、少し後退してから、今度は左にほんの少し回転するようにしてください（回転しすぎると今度は右側のセンサーがラインを外れてしまいます）

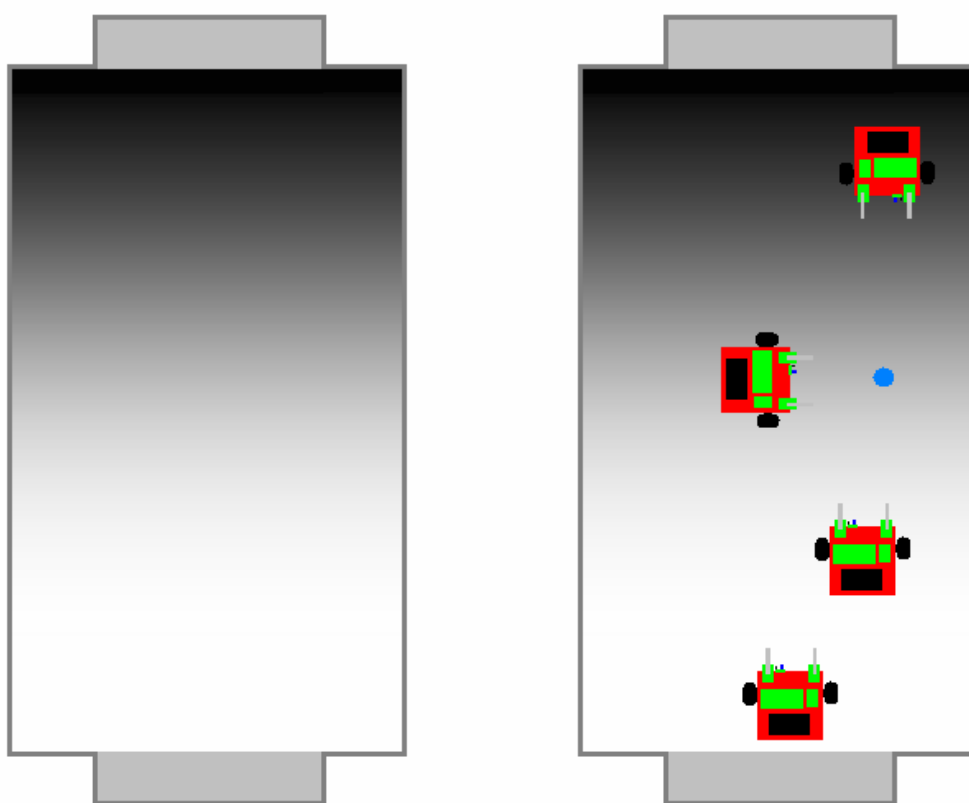
このようにすると、ロボットは線に沿って動き続けます。

それでは、ライトレースを行うプログラムを作ってみましょう。

ロボットを使ってサッカーをしよう

ここまでに学習した全ての要素を総合することで、ロボットによるサッカープログラムができるようになります。

サッカーとは下に示す図のようなコートを使って、ロボット同士2対2で行います。また、サッカーボールの代わりとなるのは皆さんがライトセンサーの最初で使用した赤外線ボールです。



なぜコートに色がついているのかというと、ライトセンサーを使ってロボットが自陣と敵陣を判断するためのものです。以前にもライトセンサーを使って白と黒を判断させるプログラムを作った時にこのようなシートを教科書から切り取って使いました。このように白から徐々に黒に変化するものをグレースケールというので、覚えておいてください。

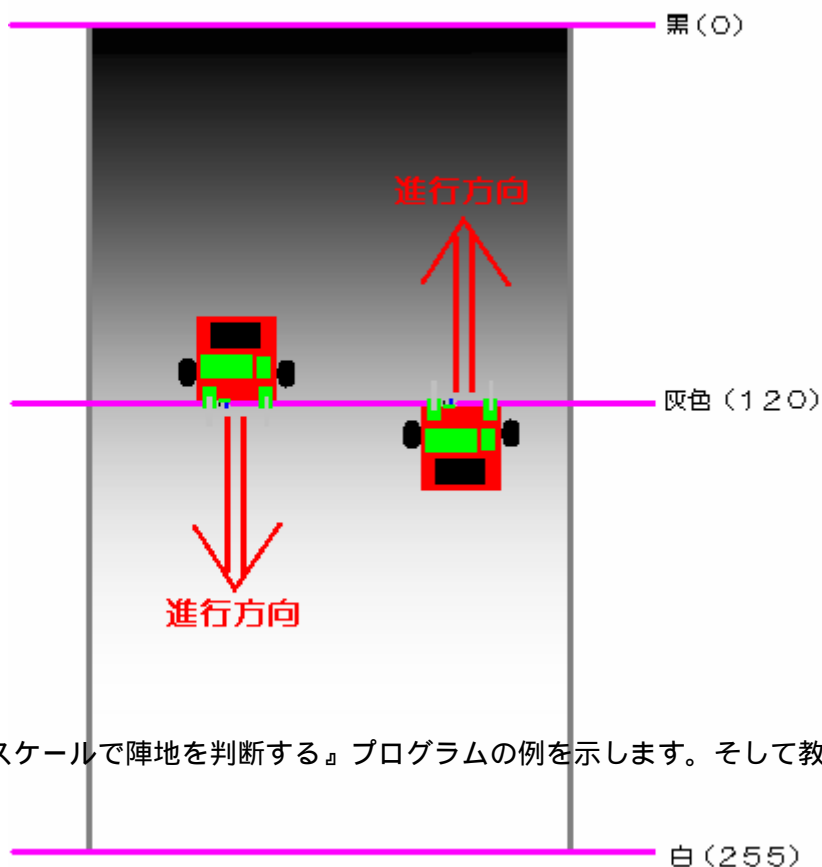
ロボットを使ってサッカーをするのに必要な要素は

- ・ 直進や回転、後退などの基本動作
- ・ 試合中にロボットが停止しないような繰り返しの動作
- ・ 障害物に接触した際のタッチセンサーの動作
- ・ 赤外線ボールを見つけるためのライトセンサーの動作
- ・ グレースケールによって自陣・敵陣を判断するためのライトセンサーの動作

このように、サッカーのプログラムを作るにはいままでに学習してきた全ての要素が必要となることがわかります。皆さんにはこの学習の総仕上げとして自分で最適だと思うサッカープログラムを作ってもらいます。次は上の最後に記述したグレースケールによって陣地を判断するセンサー動作についての説明です。

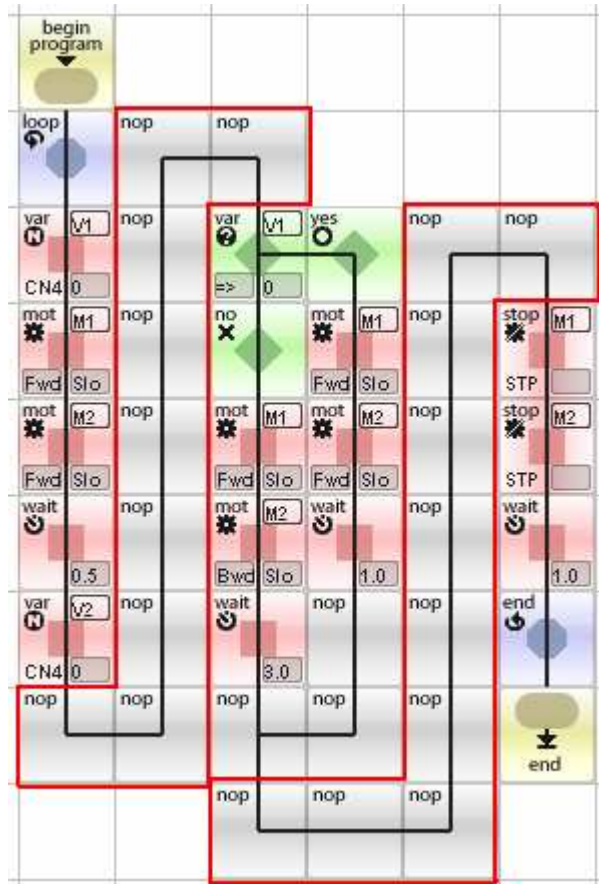
グレースケールを使った陣地の判断法

グレースケールを使用すると何故、自陣と敵陣の区別がつくようになるのでしょうか。まず下の図を見てみましょう。



下に『グレースケールで陣地を判断する』プログラムの例を示します。そして教科書のグ

レーススケールシートを切り取って、実際にどのように動くのか確かめてみてください。



1つ目の var タイルで最初にいる位置の色データを『Variable1』に保存します。
次に0.5秒間直進し、wait タイルの後ろにある2つ目の var タイルで、前進した位置での色データを『Variable2』に保存します。
var 分岐タイルで『Variable1』と『Variable2』を比較します。
var 分岐タイルの意味は『変数1の値が基準となる値(変数2)以上どうか』ということなので、今回の場合『Variable1』が変数1にあたり、『Variable2』が変数2にあたります。つまり、分岐の条件は『Variable1>=Variable2』となります。
グレースケールで白側(0)から黒側(255)に移動すると yes の分岐へ、黒側(255)から白側(0)に移動すると no の分岐へ進みます。
最後の停止命令(1秒間)はループが1回終わった事確かめます。

これでグレースケールを使った陣地の判別方法を、プログラムで実現できます。

！ 注 意 ！

このプログラムは教科書に載せる都合上、本来かなり縦長になるプログラムですが nop タイルを用いて適度な大きさに整えてあります。従って実際にプログラムを作るときは赤線で囲んである nop タイルを書く必要は無いので、今までと同じ形で書いてください。