目次

はじめに

それぞれの章の概要です。プログラミングの指導に役立てて下さい。

Ⅰ.事前の準備

スクラッチのインストール方法を説明します。

Ⅱ.利用するプログラム言語について

利用するプログラム言語 Scratch（スクラッチ）について解説します。奥の深いプログラム言語であることを紹介します。

Ⅲ.テキストを使った指導の方法

指導者の学習方法、児童への接し方、テキストの使い方を解説します。

Ⅳ.スクラッチ・スクリプトガイド

スクラッチの「うごき」「せいぎょ」「みため」「しらべる」「おと」「えんざん」「ペン」「へんすう」８つのカテゴリーにどのようなスクリプトがありどんな機能もっているかを紹介します。

Ⅴ.指導のヒント

それぞれのテキストの「概要」、「学ぶ内容」「困ったときに」を記載しています。

自分で学習する時、児童から質問を受けた時に役立てて下さい。

Ⅵ.プログラミング（制御編）

ＬＥＤ、センサーやサーボモータなどといった制御装置の運用方法について記載しています。

Ⅰ.事前の準備

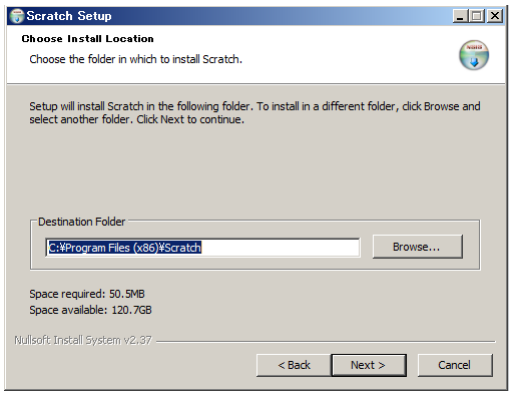
1. スクラッチのインストール方法

1. まずは「小学生ＩＣＴスクール　スタートキット」フォルダ内の「01　事業開始の準備」>「01　インストーラ＆方法」フォルダを開いて「ScratchInstaller1.4.exe」をダブルクリックします。

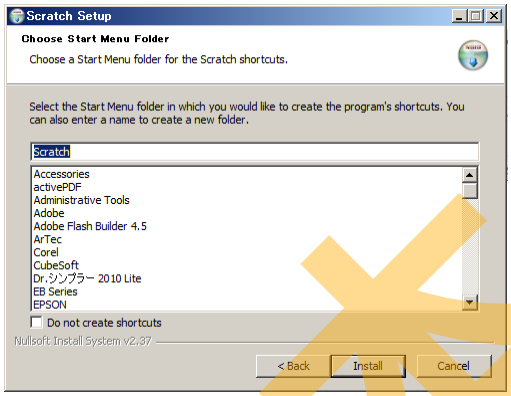
（注）現在プログラムは、Ver.2.0 になっていて、ブラウザから直接サーバーにアクセスしてプログラムの開発から保存、共有などの機能を利用できるようになっています。しかし、プログラムをオフラインで利用できません。パソコン教室での幅広い運営を考えるテキストはオフラインで利用できるVer.１.４を利用します。

2. 最初に右のような画面が表示されます。
「next>」をクリックして次の画面に進みます。

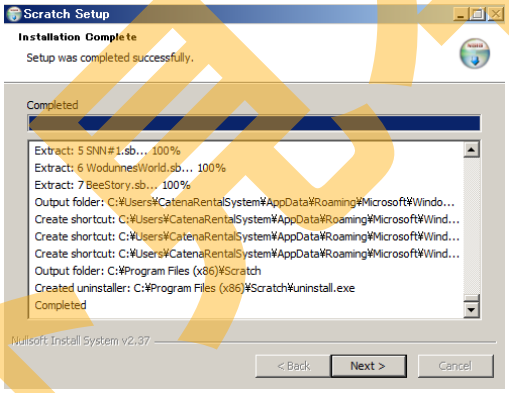
3. インストールするファイルの場所を選択します。
特に指定がない場合はそのまま「next>」をクリックして次の画面に進みます。



4. スタートメニューフォルダの選択と
ショートカットを作成するかを選択します。
特に指定がない場合はそのまま「install」を
クリックします。
これでインストールが始まります。



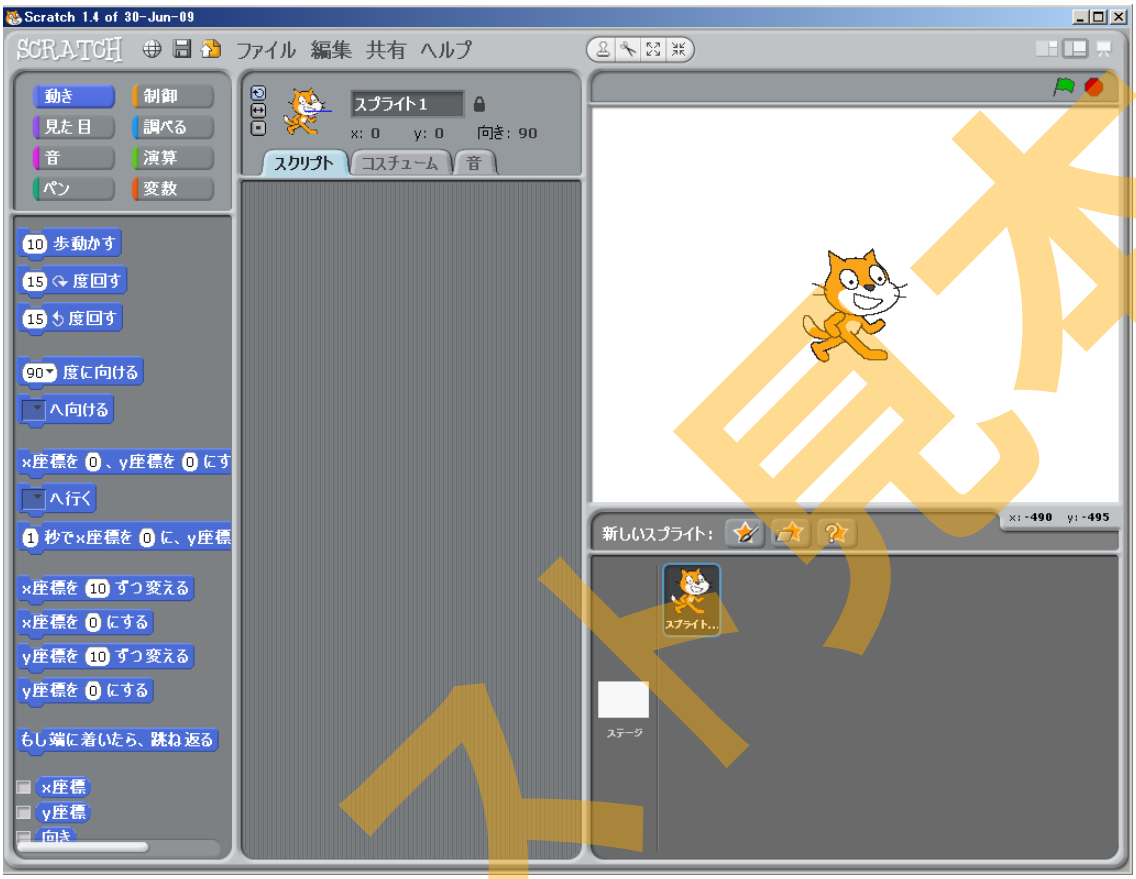
5. インストールが終わると「install」が「next>」
に変わります。
「next>」をクリックします。



6. 最後に早速、スクラッチを始める場合は
「Start Scratch」にチェックを入れます。
デスクトップにショートカットを作成する
場合は「Make a shortcut to Scratch on
the desktop」にチェックを入れます。
「Finish」をクリックすれば完了です。

2. 日本語(ひらがな)表記の設定方法

1．最初にスクラッチを起動すると下の図のような画面になっています。



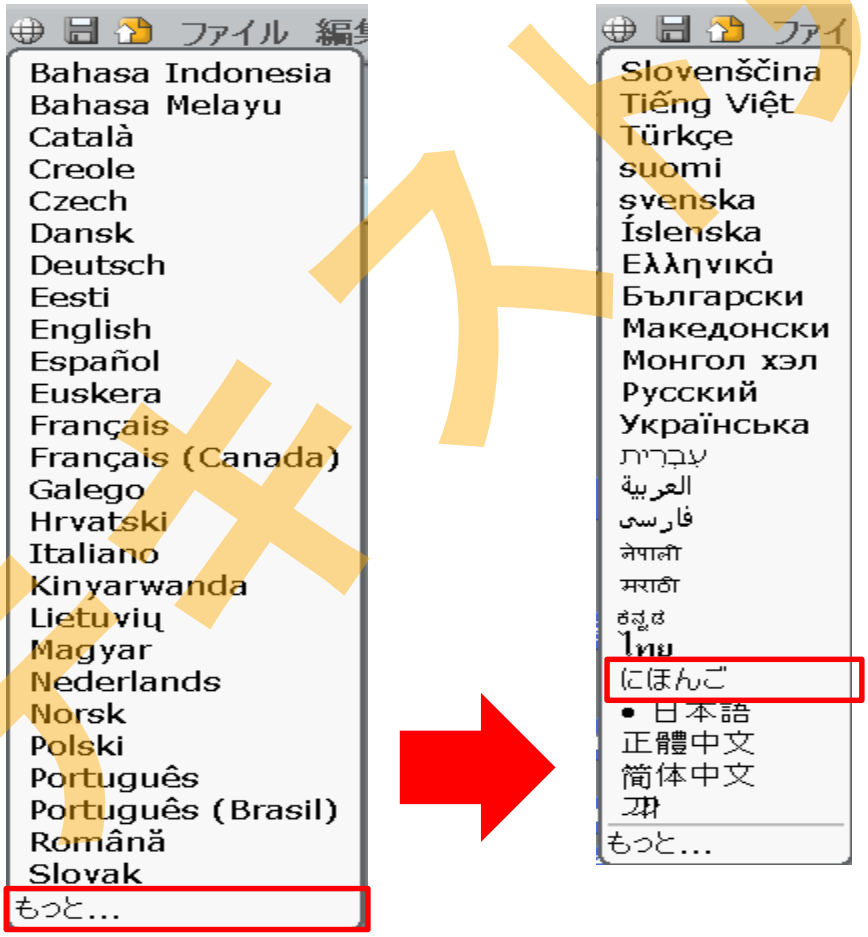
各表記が漢字表記の日本語になっています。漢字表記の日本語をひらがな表記に変更する方法を説明します。

2. メニューバーに  があります。これをクリックします。

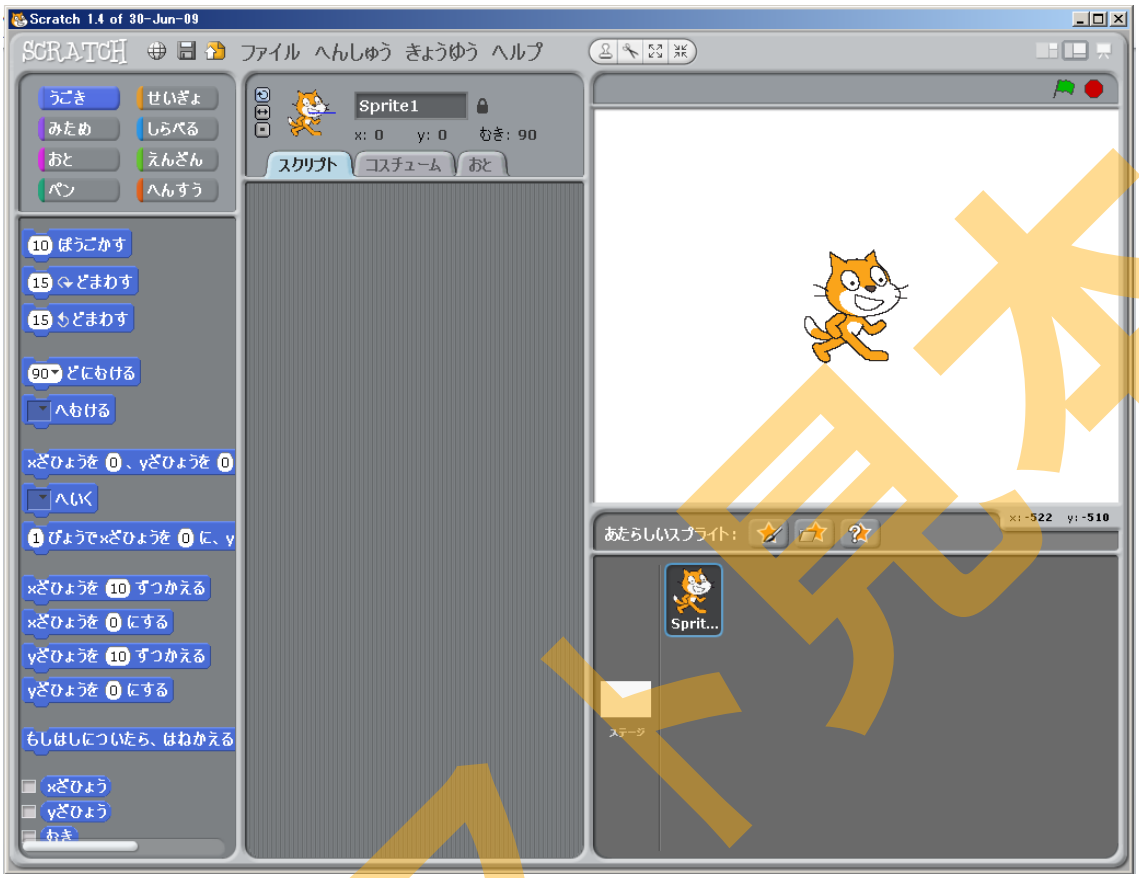
3. 世界各国の言語を選択することができます。

ひらがな表記にする場合は「にほんご」を選択します。

その中から「にほんご」を選択します。リストに「にほんご」がない場合は「もっと…」をクリックすれば別の言語が表示されます。



これでひらがなの日本語に変更されました。



Ⅱ.利用するプログラミング言語について

はじめに

学習に向いたプログラミング言語でスクラッチ（Scratch）という言語を使います。小学生でも使える言語ですが、たいへん奥の深い言語です。

Scratch は、MIT（マサチューセッツ工科大学）メディアラボで開発された命令を視覚的なブロックで表示するプログラミング言語です。

プログラミングに必要な機能が、ブロックの形で用意されています。

アルファベットで記述するソースプログラムを理解していなくても視覚的・直観的な操作でプログラムを作成し、動作させることができます。そのため専門のプログラマでなくとも、手軽にプログラミングを行えるという利点があります。

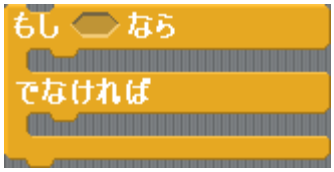
ソースプログラムを記述しないと正式なプログラミング言語ではないと思いがちですが、プログラミングを初めて学ぶ人がプログラムの動作やアルゴリズムを学ぶ最強のツールです。

次がスクラッチのロゴです。 

1. スクラッチとは

Scratch はプログラミングを学ぶ最強のツールで、プログラミングに必要な機能が、ブロックの形で用意されています。

ブロックは次のような形をしています。アルファベットで記載するソースコードと比較すると次のようになります。

Scratch の 

は「if～else」、 

は「while」。

このように、視覚的でわかりやすいので、初心者にも取り組みやすいプログラム言語です。アニメーションやゲーム更に、制御のプログラムなどの作成ができます。

ブロックを組み合わせたプログラムをすぐに実行でき、結果も画面上でアニメーションとして分かりやすく表示されます。

簡単にプログラムが作成できて、プログラムの結果が視覚的に分かるので、初心者でもプログラミングに取組むことができるのです。

誤りが起きにくいプログラミング

誤り（バグ）が起きにくいプログラミング方法として「構造化プログラミング」の考えがあります。Scratch のブロックを使うことで、自然に構造化したプログラムを作成する習慣を身につけることができます。

子供向け＝お遊び　ではなく**プログラミング**を学習する最強の優れた言語です。

「プログラミング的思考」を鍛える

プログラミングを学ぶ良い方法は、実際に動くプログラムを作ることです。工作の勉強では「ノコギリ」「かんな」「きり」の使い方を教えるだけでなく、実際に棚を作ったり、犬小屋を作ったりするのが一番です。

プログラミングの学習も同じです。このシリーズ次のように組み立てています。

テキスト「スクラッチの基本操作」ではスクラッチの基本的なブロックの機能と使い方を学び、続けてＵＦＯ撃退ゲームといった簡単なゲームづくりを体験します。

この学んだ基本を元にして、色々な機能を学習するために実際のゲームを作ります。実際のゲームを作り、少しずつ高度な機能を学んでいきます。

工作で正しく切ったつもりが、上手く本棚にならないということがあります。ゲームづくりも同じです。テキスト通りにやっても、動かしてみるとうまくいかないことがあります。上手くいかない原因は様々です。

解決方法は１つではありません。「これがダメなら、あの方法で」と沢山の解決方法を身につけることが、プログラミングの力につながります。

うまくいかない問題に挑戦し「考える力」を磨くことが、「プログラミング的な思考」を鍛えることにつながります。

いくつものプログラムの作成に挑戦し「プログラミング的思考」を鍛えて下さい。

Scratch のプログラミングの学習では２つの壁があります。

１つは、こんなものかと飽きること。

１つは、解決できない課題に突き当たること。

プログラミングが学校で取り入れられる理由の第一は、答えのない問題に試行錯誤をしながら、粘り強く解決する能力を身につけることです。解決できない課題に試行錯誤して挑戦する。これがプログラミングの勉強です。奥の深いプログラミング言語を教えることができるだろうか？

「教える必要はありません。一緒に考える。いっしょに勉強する」こうした気持ちに切り替えて下さい。　「指導」でなく「支援」することが子供たちを伸ばします。

これまで教育の中心は、答えのある問題の中から答えを掘り出すことが中心で、先生は「答え」を知っていて、その掘り出し方を教えるのが仕事でした。

今育つ子供たちには、答えのない問題に取り組んで未来を切り開いていく力を身につけることが期待されています。その力の１つが「プログラミング的思考」です。

さあ、これから本当のプログラミングの勉強です。

Scratch はどのようなプログラムか見てみましょう。

Scratch にはスプライトという便利な機能があります。
そのスプライトとは何か、どう使うものかを説明していきましょう。

① スプライトとは何か

スプライトとはコンピューターグラフィックスで、複数の画像や図形を合成して表示する技術で、キャラクタ（画像）を表示する（操作する）機能です。他の画像、キャラクタに影響されずに自由に動かすことが可能で、
スプライト単位で絵を動かしたり、見た目を変えたりすることができます。
また、当たり判定やイベント処理も設定できます。

Scratch のステージはスプライトと背景が合わさって作られています。
その様子を次のページに示します。

スプライト

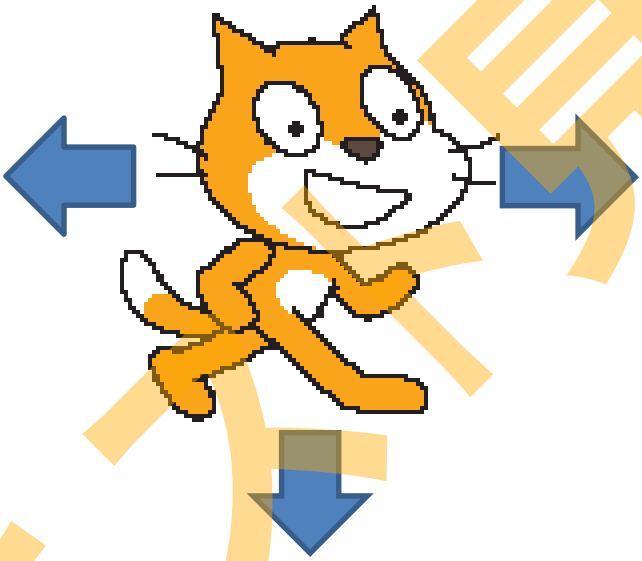
背　景

ステージ

スプライトの向き

0（上を向く）

0度



-90（左を向く）

-90度

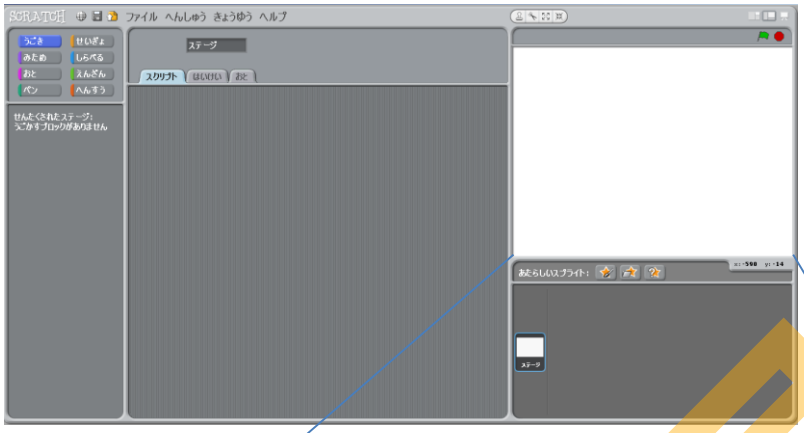
90（右を向く）

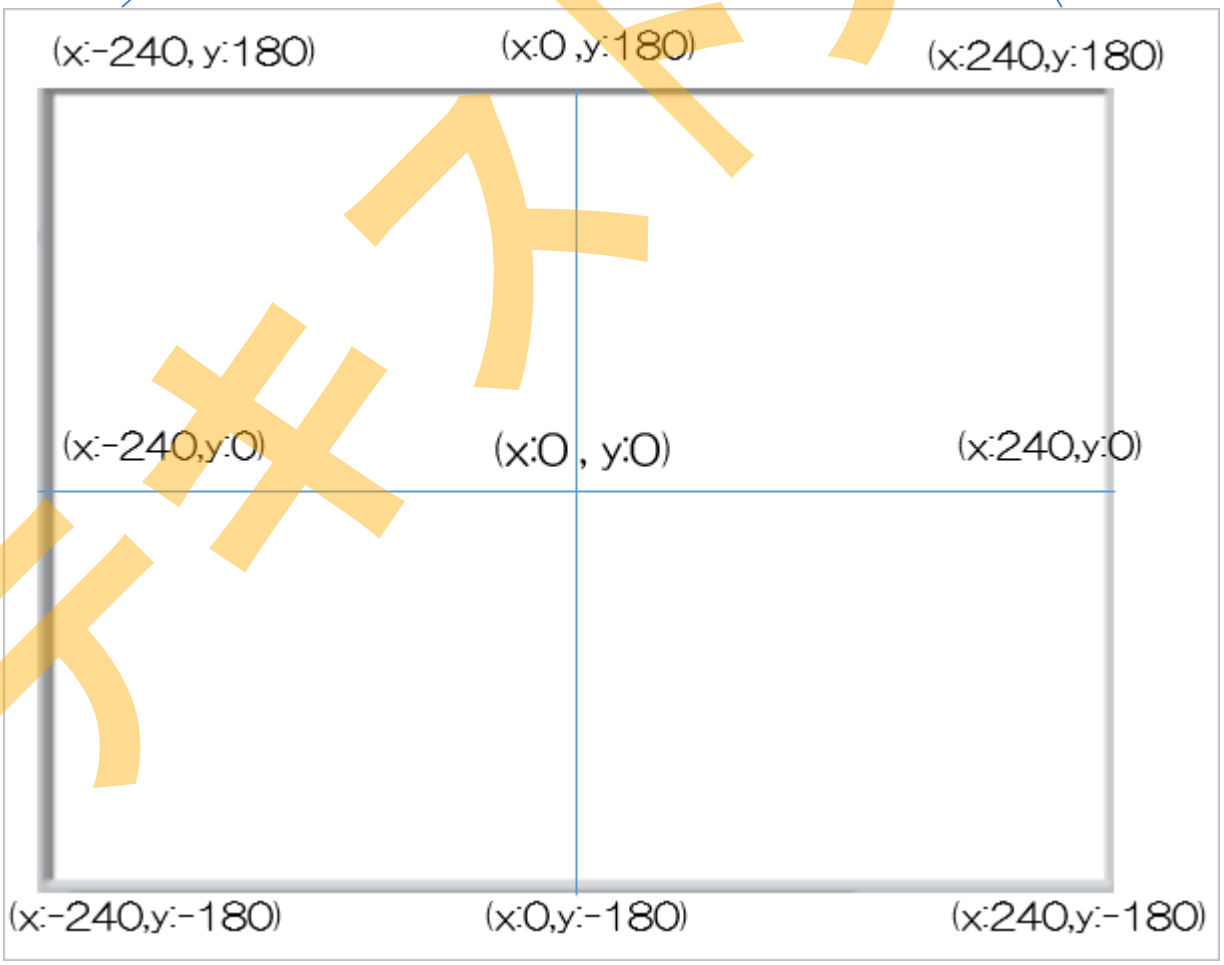
90度

180（下を向く）

180度

ステージの座標



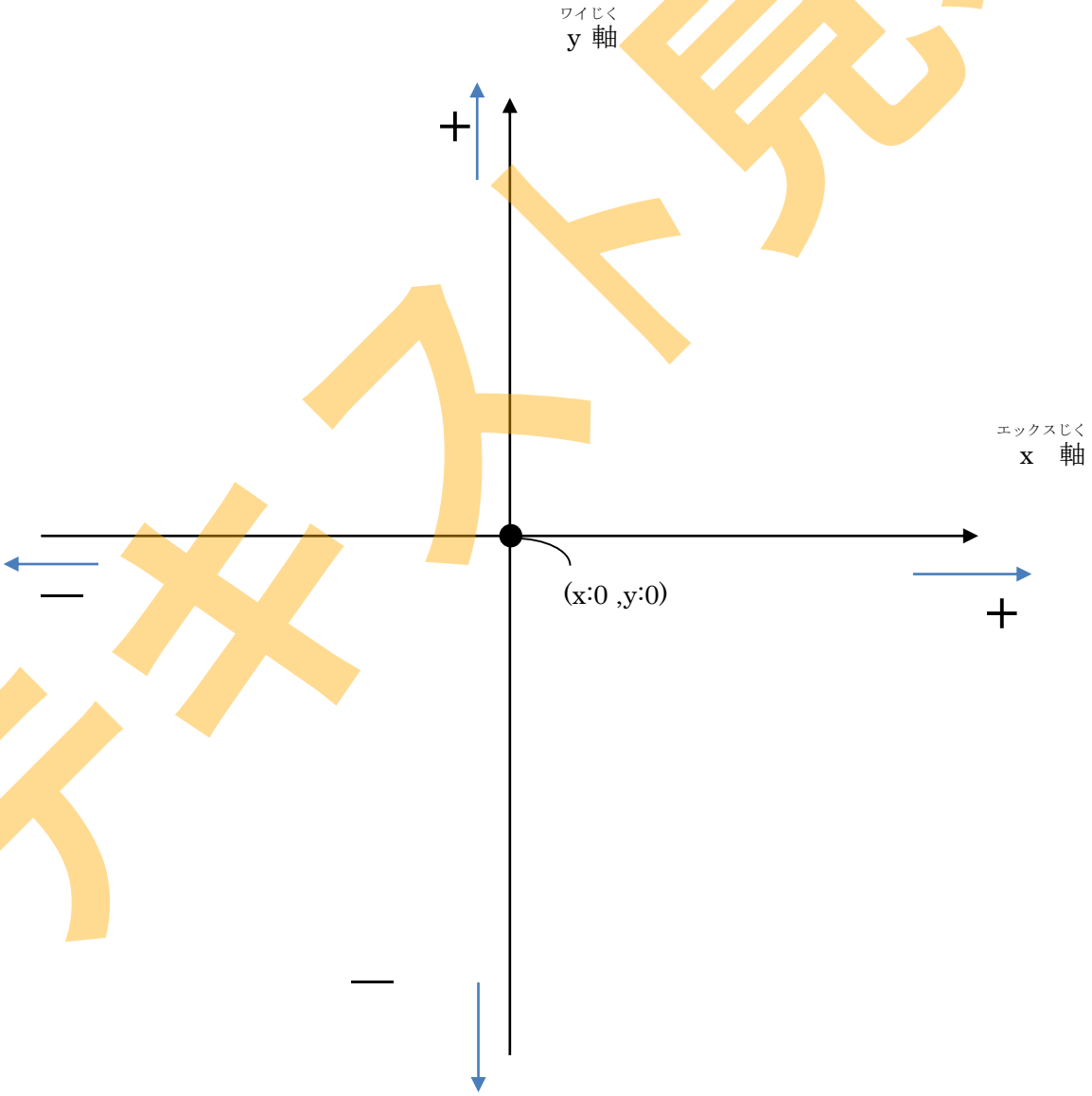


座標

　点の位置を明確にするための数の組を座標と言います。座標を習うのは中学になってからで「ｘとｙの数の組を座標として表す」というように習います。

　ｘが８、ｙが９　というかたちで点の１が決まります。

　スプライトを動かすためには、スプライトの位置を示すことが必要です。座標の考えを小学生３年生に理解できるでしょうか？　心配はいりません。スプライトの位置を示すために座標を使うことで、子どもたちは自然に座標を理解できるようになります。

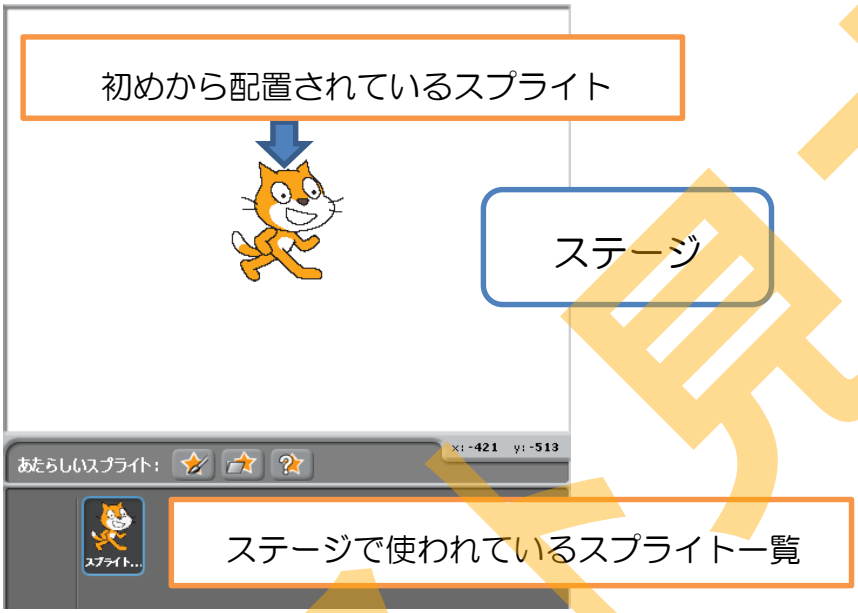


② スプライトを動かしてみよう

まずは Scratch を起動してみましょう。

右の図のアイコンをクリックします。

スクラッチを起動すると、ステージにはネコのスプライトが配置されています。



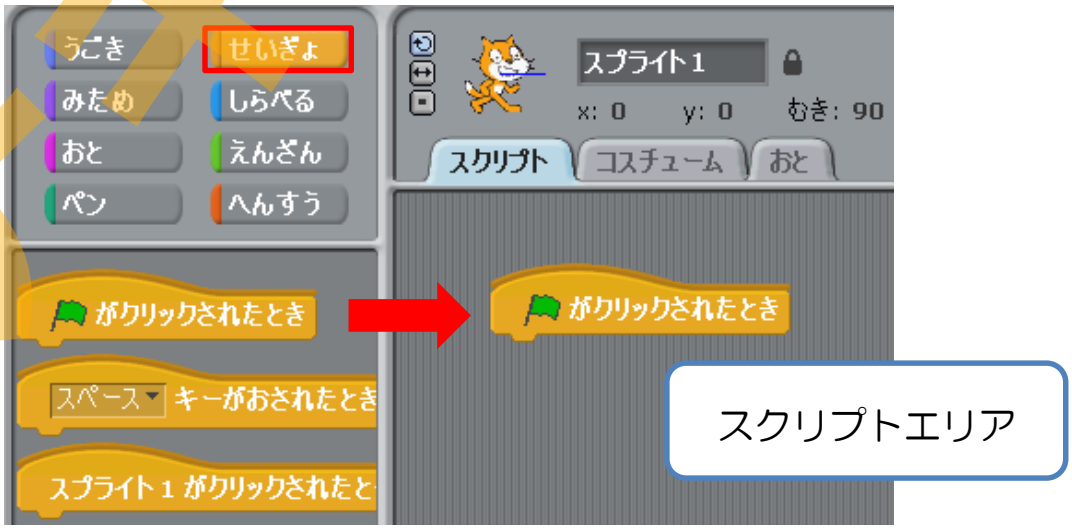
ステージの下に、ステージで使用されるスプライトの一覧が表示されています。

起動した時は、ネコのスプライトが１つだけ表示されます。

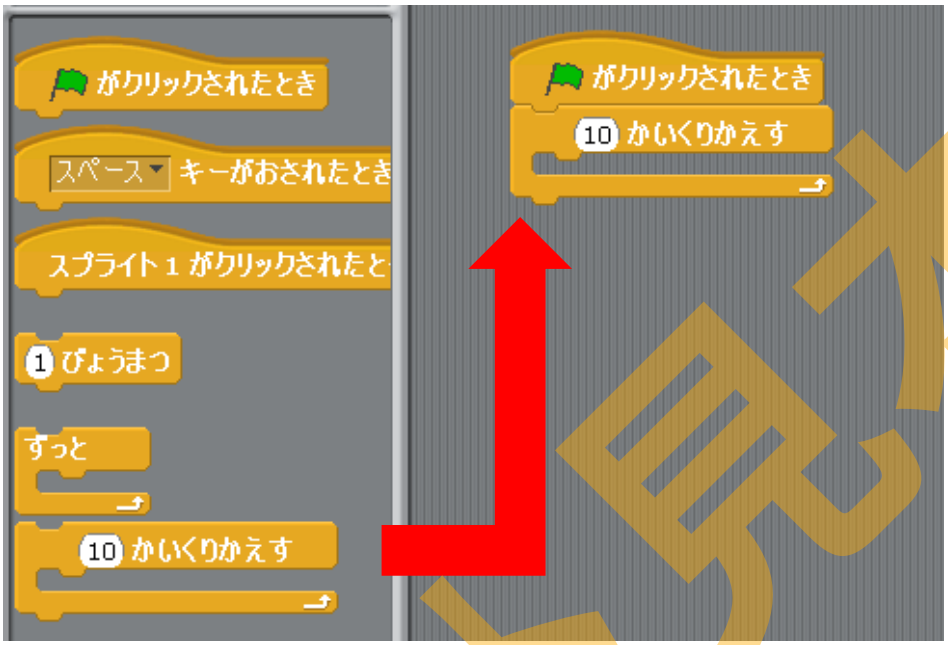
まず、

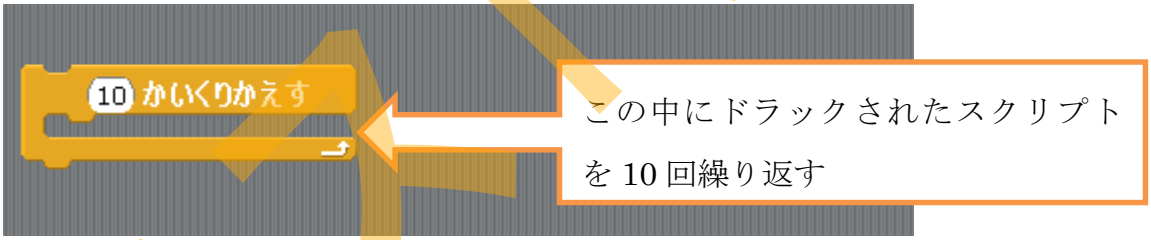
ブロックパレットの「せいぎょ」をクリックして「旗がクリックされたとき」

をドラックしてスクリプトエリアに置きます。



「せいぎょ」から「10 かいくりかえす」を「旗がクリックされたとき」の下にドラックして移動します。

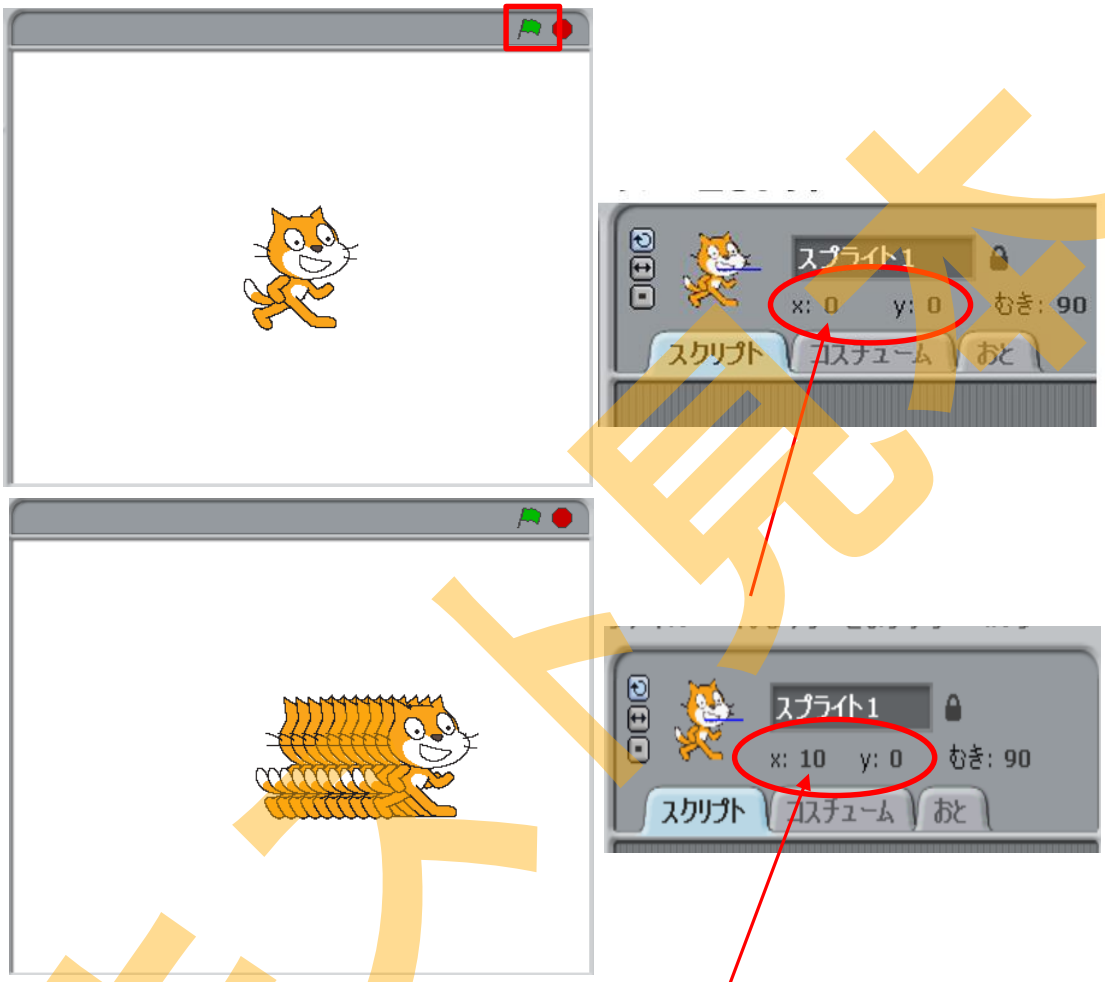




「うごき」をクリックして「10 ぽうごかす」 10 ぽうごかす を「10 かいくりかえす」の中にドラックして入れます。

これで実行してみましょう。

ステージの右上にある旗のアイコンをクリックします。



いかがでしょう？　ネコが動いているのが確認できますね？

しかし、１０歩動いていないのでは？

この命令　10 ぼうごかす　で　どう見ても猫は、１０歩、歩いていません。

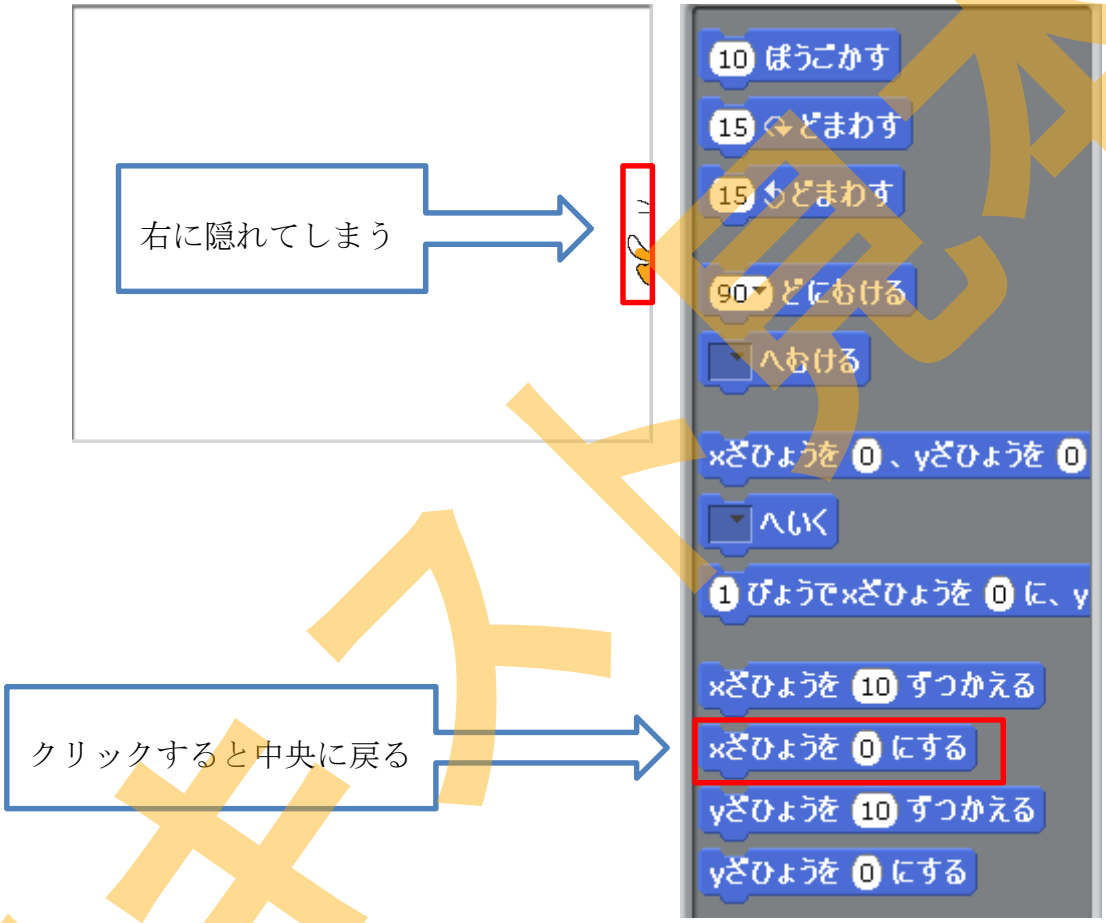
この１０歩の意味は、猫のＸ座標を１０増やすという意味です。

何回か「旗」のアイコンをクリックしてネコを動かしてみましょう。

どんどん、右に進み、右端へ隠れていきます。

その場合は「x ざひょうを 0 にする」をクリックしてみましょう。

中央の位置に戻ります。



2. スクラッチとプログラミング言語の関係

スクラッチではブロックをつなげてプログラムを作成します。「このように動きなさい」や「この条件では止まりなさい」などの命令を与えていきます。

ブロックの命令はプログラミングのソースプログラムの構文に対応させることができます。

スクラッチのブロックと java 言語を対応させるとつぎの様になります。

は

```
while(true) {
    if(条件){
```

もう少し加えると

```
i = 0;
while(i < 100){
    if(i % 2 == 0){
```

のようになります。

これはＪＡＶＡですが、現在よく使われる言語の種類でも１３種類あり（後述）、このソースプログラムとは、表現方法が使う言語がことなりますが、ブロックであれば次のように共通に表現できます。

プログラミング言語は、開発の目的によって様々な言語があるので、これからプログラミングを学ぶ人には、プログラミングの基本を言語の違いにとらわれることなく共通に学ぶ事ができるスクラッチは、非常に向いた言語と言えます。

スクラッチのブロックとＪＡＶＡとの対応とＪＡＶＡ言語のソースプログラムを参考に示します。

スクラッチ

Java

もし……ならずっと

```
while(true) {
    if(条件){

    }
```

もし……なら

```
if(条件){

}
```

もし……なら、でなければ

```
if(条件){

}
else{
```

までまつ

```
while(true){
    if(条件){
        break;
    }
```

10 かいくりかえす

```
for(i= 0; i < 10; i = i + 1 ){

}    繰り返し回数が決まっている場合は「for」
while を用いることもできる
i = 0;
while(i < 10){
    i = i + 1;
}
```

java のソースプログラム

```
public static void main(String[] args){
    int i;

    i = 0;
    while(i < 100){
        if(i % 2 == 0){
            System.out.println(i + “偶数です”);
        }
        else{
            System.out.println(i + “奇数です”);
        }
    }
}
```

ソースプログラムとして記述するアルファベットが１文字でも違うとエラーが発生してうごきません。スクラッチのようにブロックで示すと、そうした間違いの発生を抑えることができます。そのため処理したい意図をブロックのように表現できるスクラッチが教育用の言語として適しているのです。

3. 代表的なプログラミング言語

現在、よく使用されるプログラム言語は、ＩＢＭ、アップルなど大企業が開発したものより個人のプログラマーが開発したものが多いのですが、よく使われる代表的な１３種類のプログラム言語を紹介します。

1. 主要な 13 のプログラミング言語

1 C	**登場時期**	**開発**	**概要**
	1972 年	ケン・トンプソン氏 デニス・リッチー氏	応用範囲が広いが初心者には高度。 ハードウェアの組み込み機器向け（OS や家電など）
	言語タイプ		
	コンパイラ言語、静的型向け、手続き型		
	特徴		
	応用範囲が広くハードウェア制御に活躍		
2 C++	登場時期	開発	概要
	1982 年	ストロヴストルップ氏	「C 言語+オブジェクト指向」 言語仕様が多く複雑だがその分、大規模開発が可能 実行速度が速く、ゲーム業界では人気。
	言語タイプ		
	コンパイラ言語、オブジェクト指向		
	特徴		
	ゲーム開発向けの言語で根強い人気		
3 C#	登場時期	開発	概要
	2002 年	マイクロソフト社	フレームワーク開発の主力言語 さまざまな言語の強みを吸収。 ゲーム開発、Windows アプリ開発、web アプリ開発で使われている。
	言語タイプ		
	コンパイラ言語、オブジェクト指向（+α）		
	特徴		
	マイクロソフト製品と相性がよい多機能言語		
4 Java	登場時期	開発	概要
	1992 年	サン・マイクロシステムズ社	オブジェクト指向の代表的言語。 ライブラリ言語が豊富、ネットワーク処理がしやすい。 きちんと作成しないと動かないため、入門者には鬼門。
	言語タイプ		
	コンパイラ言語、静的型向け		
	特徴		
	やさしくないが最多のプログラム人口を誇る		

5 Perl	**登場時期**	**開発**	**概要**
	1987 年	ラリー・ウォール氏	「軽量化言語」「スクリプト言語」の父的存在。 文字列処理が得意。 Web サーバーのプログラムで活躍。
	言語タイプ		
	インタプリタ言語、手続き型（+オブジェクト指向+α）、スクリプト言語		
	特徴		
	初心者が Web サーバー側で活躍しやすい		
6 Ruby	登場時期	開発	概要
	1993 年	まつもとゆきひろ氏	オブジェクト指向を学ぶのに最適 「ストレスなくプログラミングを楽しむ」ことを目的に開発。 いろいろな書き方ができる。
	言語タイプ		
	インタプリタ言語、オブジェクト指向(+α)、スクリプト言語		
	特徴		
	日本初の愛され言語で Web アプリを開発		
7 PHP	登場時期	開発	概要
	1995 年	ラスマス・ラードフ氏	HTML を動的に作成する用途で開発。 WEB サービスが簡単に作成できる。 データベースへのアクセスが簡単 文字列処理が得意 ライブラリが豊富
	言語タイプ		
	インタプリタ言語、手続き型（+オブジェクト指向+α）、スクリプト言語		
	特徴		
	Web サーバー側プログラミングで役立つ		
8 Python	登場時期	開発	概要
	1991 年	グイド・ヴァン・ロッサム氏	言語仕様が少なくシンプルな文法 大きなデータを扱いたい人、沢山計算をさせたい人向け インタプリタ言語の中では最速の処理速度
	言語タイプ		
	インタプリタ言語、オブジェクト指向(+α)、スクリプト言語		
	特徴		
	データサイエンティスト御用達で数字に強い		

9 Java Script	**登場時期**	**開発**	**概要**
	1996 年	ネットスケープコミュニケーションズ社	Web ブラウザ向け言語。 あらゆる分野で活躍（パソコン、スマホ、サーバー、組み込みなど） 結果がすぐわかるので勉強しやすい
	言語タイプ		
	インタプリタ言語（JIT コンパイラ）、オブジェクト指向（+α）		
	特徴		
	Web ブラウザでプログラミング結果を確認		
10 Visual Basic	登場時期	開発	概要
	1991 年	マイクロソフト社	初心者でも windows アプリケーションが作れる。ビジネス向けアプリケーションにも対応。画面処理が簡単なので直感的に取り組める 実行環境や開発環境が制限される。
	言語タイプ		
	インタプリタ言語、手続き型（+α）		
	特徴		
	Windows アプリ開発の初心者に最適		
11 Objective_C	登場時期	開発	概要
	1983 年	ブラット・コックス氏	C 言語とオブジェクト指向のハイブリット　、Mac OS X、iOS アプリの公式開発言語 実行時の柔軟性が良い、癖がある
	言語タイプ		
	コンパイラ言語、オブジェクト指向（+α）		
	特徴		
	iOS アプリの開発で一攫千金が狙える		
12 Swift	登場時期	開発	概要
	2014 年	アップル社	Mac OS X、iOS アプリの公式開発言語　C 言語、Objective_C とソースコードを共存、いろいろな言語の長所を集めてできている
	言語タイプ		
	コンパイラ言語、関数型（+α）		
	特徴		
	アップルの新言語で iOS アプリを開発		
13 HTML	登場時期	開発	概要
	1990 年	ティム・バーナーズ・リー氏	プログラム言語ではなく、文書の構造を記述するマークアップ言語。 処理が制御できない。 Web アプリの開発には必須
	言語タイプ		
	マークアップ言語		
	特徴		
	Web エンジニアに必須の技術		

2. 使う用途とプログラミング言語

多くのプログラミング言語がありますが、どの言語を学べば良いのでしょうか？
プログラミングで何を作りたいかによって決めることになります。
iPhone アプリの制作なら「Objective_C」や「swift」、Android アプリの開発をしたいなら「Java」、word や excel など Microsoft Office で効率よく作業しいなら Micro Office には VBA（visual basic for applications）があり「visual basic」を身に付けるのが良いと言われます。
目的に応じた言語は、次のとおりです。

例 1 iPhone を開発したい人

→ Swift、Java

例 2 事務の仕事で役立てたい人

→ Visual Basic

例 3 ゲームプログラマーを目指す人

→ C 言語、C++

例 4 基礎からしっかり学びたい人

→ C 言語、Java

例 5 Web デザイナーとして成長したい人

→ JavaScript

例 6 プログラミングに憧れるインフラエンジニア

→ Perl

Ⅲ.テキストを使った指導の方法

1. 指導者の学習方法

本シリーズのテキストを使って指導する人の条件は次の通りです。

プログラミングの経験　　不要です。

プログラミングについての予備知識　　不要です。

テキスト「スクラッチの基本操作」を最後まで自分で作成する。

ＵＦＯ撃退ゲームを完成させて下さい。（学習時間　４時間から６時間です。）

うまくいかない時は、

本書「Ⅴ.指導のヒント　１．共通に注意すること　2.　スクラッチの基本操作」

を参考にして下さい。簡単ではないかもしれませんが、乗り越えてください。

児童と同じように１ページ、１ページ、丁寧に進めて下さい。

完全に攻略してください。これだけで十分です。

次は**「児童への接し方」**です。それを考える準備として、次の文を読んで下さい。

TRON プロジェクトで有名な日本のコンピューター先駆者坂村健氏が毎日新聞のコラムに書かれた文章です。

内容は、日本ビーコム戸所の責任で一部省略しました。

坂村健の目　未来の学び　「プログラミング教育」　毎日新聞　２０１７年3月１７日

２０２０年から小学校でプログラミングの授業が始まることとなった。文部科学省と総務省、経済産業省は協力して、「未来の学びコンソーシアム」を設立し支援活動を始めた。このこと自体は望ましい動きだ。 ただ、プログラミング教育に対して誤解もあるので、このタイミングで要点をまとめておきたい。

まず、２０２０年では遅いくらいだが、プログラミング教育を始めた国はどこも教育人材で苦労している。ただでさえ負担が大きい小学校の先生だけでどうにかしろ、は不可能だろう。専科教員を養成する時間もない。なら退職した技術者と先生のセットで授業をできないか。育児のため休職・退職した女性技術者に自宅からテレワークで教えてもらう手もある。教員免許の考え方を変えるとか、対面教育の縛りを緩くするなど制度上の変更もしなければならない。産業界との連携も必要だ。

次にプログラムの基礎「アルゴリズム」が数学の分野になったのは１９２０～３０年代と新しい。アルゴリズムが他の数学分野と異なるのは、「解」でなく「手順が基本」ということだ。他分野では問題の中に初めから答えが内包されていて、解くというのは問題の中から解を「掘り出す」イメージだ。

「手順を踏んでやってみない限り、どうやっても解けない問題がある」という概念が確立し、アルゴリズムがクローズアップされてきた。

他国に比べホワイトカラーの生産性が低い日本。全ての人が手順の効率化を明示的に考える姿勢は、今後ますます重要になる。　（東大教授）

次の２点が指摘されています。

■負担が大きい小学校の先生だけでどうにかしろ、は不可能だろう。

⇒多忙な小学校の先生を支援する体制が必要、産業界との連携も必要

■「手順を踏んでやってみない限り、どうやっても解けない問題がある」

⇒小学校でプログラミング教育が必須化される理由

■多忙な小学校の先生を支援する体制が必要、産業界との連携も必要

ＩＣＴを使う授業も同じなのでプログラミング＋ＩＣＴ活用で必要な先生を支援する体制について考えます。新指導要領では、ＩＣＴを使う授業が毎日１コマ予定されています。ＩＣＴ支援員が児童１０人に１人必要ならクラスに３人。学年２クラスで１２クラスの学校では、どのくらいの支援員が必要になるのでしょうか？

ここに地域に密着したパソコン教室が貢献できることがたくさん考えられます。

産業界との連携も必要

昭和３０年代からの学校でのそろばん指導とそろばん塾が思い出されます。

日本ビーコムは、ＩＣＴ教室で児童を教えると同時に、ＩＣＴ支援員としてパソコン教室のスタッフや受講者を育てることを想定してテキストを開発しています。

経験のないスタッフや受講者をＩＣＴ支援員に育て児童にも使えるテキストです。

（数年先のパソコン教室として避けて通れないテーマと考えます。）

■小学校でプログラミング教育が必須化される理由

ＡＩの休息な進捗、Ｉｏｔ社会、人口減少・高齢化　・・・・・・

これから社会に出る子どもたちは、解けない問題にも、逃げずに取り組む力が必要。

また、手順を踏まないと解けない問題に取り組む力も必要。

⇒正解がなく手順を踏んで解決する問題に取り組む力を育てる必要があります。

手順を踏む必要の有る問題にプログラミングがあり、プログラミングに取り組むことで正解のない問題に取り組む力を持つ子どもを育てます。

例えば、テキストに「ＵＦＯ撃退ゲーム」を作る手順が書かれています。しかし、作成に取り掛かると、テキスト通りにしているつもりなのに、うまくいかないことが起こります。ＵＦＯの大きさがテキストの想定より少し大きいことが原因です。

この時、指導者に求められるのは、困っていることに気づいてあげること。そして、ヒントを出してあげること。ヒントを出してあげることが出来ないなら、一緒に考えてあげること。そして、課題が解決したらそれを認めて褒めてあげることです。

2. 児童への接し方

「ＵＦＯ撃退ゲーム」の例を続けます。ＵＦＯの大きさがテキストのＵＦＯより少し大きいことが原因でテキストのままでは動かないことが起こります。

児童からうまくいかないと質問された時、スタッフは、「ＵＦＯの大きさがテキストと違うと、上手くいかないことが起こるよ」「ＵＦＯを配置した座標は大丈夫？」などとヒントを出してあげます。何が原因かすぐわからないことも少なくありません。

スタッフに求められるのは、「ここが、このようにマズイので、ここを直しなさい」と上手くいかない原因を教えることではありません。カリキュラムを前に進めるのが役目と考える人がいますが、プログラミング教育がめざす指導として誤っています。児童が自分の力で原因に気づき「自分で問題を解決できた」という達成感を味わってくれることが大切です。意欲を持って学習に取り組む心が育つからです。

正解の有る問題で正解を出す学習も大切ですが、手順を踏まないと解決しない課題に取り組む力を育てることこそが求められています。１０数年後に社会に出る児童には、小学校の時から「手順を踏まない限り、解けない問題」に取り組む力を身につけることが求められています。

仕事の効率化、新商品の開発、お店の運営、まちづくりなど、社会には答えのない課題ばかりですが、「手順を踏むことで」上手く行っていなかったことの改善方法が生まれる可能性があります。坂村健氏の言う「他国に比べホワイトカラーの生産性が低い日本。全ての人が手順の効率化を明示的に考える姿勢は、今後ますます重要になる」と言うのはこの点を指しています。

また、人と力を合わせることも大切です。仕事も、まちづくりも、１人ではできません。力を合わせる力も、未来を担う子どもたちに求められています。

テキストを使って、子どもに接する時に心がけたいのは、次の２点です。

「自分でできた達成感を感じ、難しくても課題に取り組む心を育てること」

「人と力を合わせる力を育てること」

3. テキストの使い方

初めて学ぶ児童には、「プログラミング入門　スクラッチの基本操作」を使用します。

テキストの利用例：（タッチタイプの練習とプログラミングを組み合わせた例）

週１回、５０分の練習時間、２年間で９６回を標準とした２年のカリキュラム

最初の４ヵ月はタッチタイプの練習の時間　３０分

プログラミングの練習時間　２０分

児童がマウスを使うことに慣れているかを確認します。

マウスのクリック、ドラック&ドロップを使うので。マウスの練習を行います。

マウス練習（２回から４回）が進んだら「スクラッチの基本操作」で学習を始めます。

「スクラッチの基本操作」の学習は、毎回２０分を目度に学習を進めます。

作成したプログラムを保存するためにＵＳＢを準備して下さい。（２Ｇ程度）

プログラミングのテキストの開発の狙いは次の通りです。

ねらい　・子どもの「プログラミング的思考」や論理的な考え方を育てる。

・シニアの教材としても使える

進め方　完成度の高いプログラムを作りプログラミングを理解する。

前提　　１人のスタッフで５名に対応する。補助スタッフをつけて８名に対応する。

随時の入会を受け付けることができる

パソコン教室のスタッフが対応する

プログラミングのテキストの種類を次ページに示します。

テキストの通りにしている（と思っている）のに上手くいかないことが良くあります。

その際、手助けになるヒントをこの指導の手引きにもまとめています。

指導の手引きのテキストは下記の版に対応しています。

プログラミングテキストの種類と内容

ゲームのテキストの種類	内容
スクラッチの基本操作	スクラッチを初めて学ぶ初心者向け教材。 エリアの説明、簡単なシューティングゲームのプログラム作成。 ペイントエディタの使い方、画像の移動、繰り返し、条件分岐、待機処理、演算、座標の理解、当たり判定処理等を学びます。
ゲーム①　ピンポンゲーム	ラケットで来たボールを打ち返すテニスゲーム。最初の初心者向け。 変数、音の発生、メッセージを送る等を学びます。
ゲーム②　ブロック崩し	ブロックをボールで消していくブロック崩しゲームです。 ステージレベル変化、サブルーチンの考え方等を学びます。
ゲーム③　インベーダーゲーム	インベーダーゲームを作成します。 敵インベーダーの乱数、リストの作成等を学びます。
ゲーム④　スーパーキャッツ　１	地上、水中、謎解きステージの３ステージに分けて横スクロールアクションゲームを作成します。横スクロールの基本、キャラクターの落下判定、敵の当たり判定、ゴール判定等を学びます。
ゲーム④　スーパーキャッツ　２	
ゲーム⑤　自動車ゲーム　１	縦スクロールゲームを学びます。 縦スクロールの基本、ゴールの長さ判定、速度変更、スピードメーター表示等を学びます。
ゲーム⑤　自動車ゲーム　２	
ゲーム⑥　倉庫番	樽を出口まで持っていくパズルゲームです。 スタンプ機能を使って問題を作成します。
制御①　ＣＰＵ基盤・信号機	CPU 基盤で LED 点灯、ブザーを鳴らす制御プログラムを作成します。 押しボタン式の信号機の制御プログラムを作成します。
制御②　扇風機・温度人感センサー	モーターの回る速度をスイッチで制御するプログラムを作成します。 センサーを付けてモーターを制御するプログラムを作成します。
制御③　パクパクパニック	パクパクパニックを開閉させるプログラムを作成します。 パクパクパニックを使ってドキドキゲームを作成します。
制御④　自動車	自動車にセンサーを付けて障害物を避けるプログラムを追加します。自動車の動きを制御するプログラムを作成します。

指導の手引のテキストは下記の版に対応しています。

プログラミング入門	スクラッチの基本操作	２０１９年　７月３日　３版
プログラミング入門	ゲーム①　ピンポンゲーム	２０１９年　６月３日　３版
プログラミング入門	ゲーム②　ブロック崩し	２０１９年　６月３日　３版
プログラミング入門	ゲーム③　インベーダーゲーム	２０１９年　６月３日　３版
プログラミング入門	ゲーム④　スーパーキャッツ　１	２０１９年　８月５日　３版
プログラミング入門	ゲーム④　スーパーキャッツ　２	２０１９年　８月５日　３版
プログラミング入門	ゲーム⑤　自動車ゲーム　１	２０１９年　８月５日　３版
プログラミング入門	ゲーム⑤　自動車ゲーム　２	２０１９年　８月５日　３版
プログラミング入門	ゲーム⑥　倉庫番	２０１９年　８月５日　３版
プログラミング入門	制御①　ＣＰＵ基盤・信号機	２０１９年　８月７日　３版
プログラミング入門	制御②　扇風機・温度人感センサー	２０１９年　８月７日　３版
プログラミング入門	制御③　パクパクパニック	２０１９年　８月７日　３版
プログラミング入門	制御④　自動車	２０１９年　８月７日　３版

プログラミング指導の手引きのテキストは下記の本テキスト

「スクラッチの基本操作　ゲーム編　制御編」と「完成例」、「解答」の３種類です。

プログラミング指導の手引き１	スクラッチの基本操作 ゲーム編 制御編	２０１９年　8月１６日　3版
プログラミング指導の手引き２	完成例	２０１９年　8月１６日　3版
プログラミング指導の手引き３	解答	２０１９年　8月１６日　3版

・プログラミング指導の手引き２「完成例」テキスト

ここでは各プログラミングテキストで作成したゲームの完成例を確認することができます。プログラムがこれと同じでもゲーム上で不具合が起こる場合

○スプライトの形、色、中心点が違っていることがあります。

テキストのスプライト作成手順を見直してみましょう。

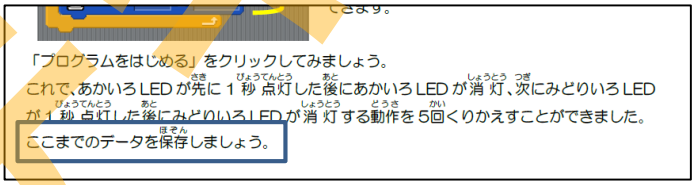
○色の判定設定が見た目上、同じように見えても色がわずかに異なる場合があります。色の判定設定もやり直してみましょう。

○変数のリストを扱う場合、リストを入れる箱が作られているか確認しましょう。

※コスチュームの数が複数ではない場合、完成例に載せておりません。

・プログラミング指導の手引き３「解答」テキスト

プログラミングデータは USB 内で作成したプログラミングデータに上書き保存していきます。各プログラミングテキスト本文を見るとプログラミングデータの保存方法と各テキストページで下図のように保存を促す記載がされています。



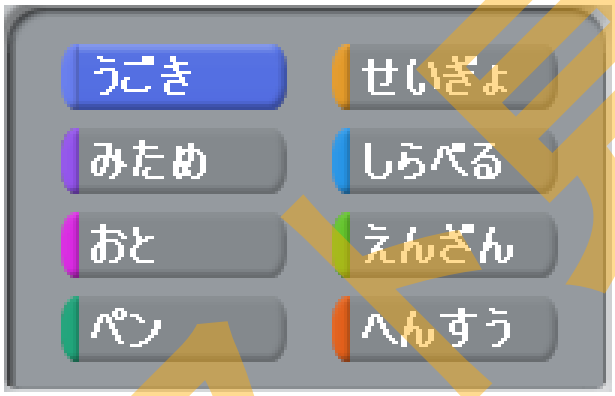
ここで以前保存したプログラミングデータから現在（ここまでのデータを）保存したプログラミングデータまでに着手した部分を本テキスト「プログラミング 指導の手引き３　解答編」で確認することができます。

（プログラミングテキストのｐ４７で保存した場合、以前保存した最終ページがｐ３０の時、ｐ３０～ｐ４７の間に着手した範囲を確認することができます。）

Ⅳ. スクラッチ・スクリプトガイド

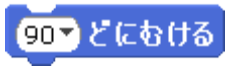
この章は辞書のように使ってください。

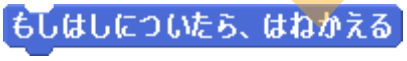
スクラッチにはさまざまなスクリプトがあり、それは「うごき」、「せいぎょ」、「みため」、「しらべる」、「おと」、「えんざん」、「ペン」、「へんすう」の８つのカテゴリーに分けられています。ここではカテゴリーごとにどんなスクリプトがあり、どんな命令を実行するのかを紹介します。



「うごき」のスクリプト	「せいぎょ」のスクリプト
「みため」のスクリプト	「しらべる」スクリプト
「おと」のスクリプト	「えんざん」のスクリプト
「ペン」のスクリプト	「へんすう」のスクリプト

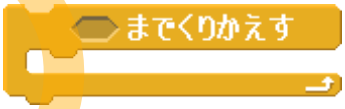
1.「うごき」にあるスクリプト

	スプライトが向いている方向に指定した歩数分動く。負の数(0 より小さい数字)の場合は逆の方向に動く。
	時計回りにスプライトを指定した数だけ回す。0 より小さい数字なら反時計回りになる。
	反時計回りにスプライトを指定した数だけ回す。0 より小さい数字なら時計回りになる。
	指定した方向にスプライトを向ける。
	マウスポインタあるいは指定したスプライト(コスチュームの中心)の方を向く。
	指定の座標にスプライトを移動させる。
	マウスポインタあるいは指定のスプライト(コスチュームの中心)に移動する。

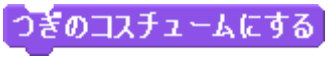
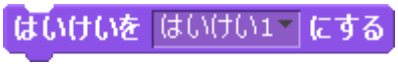
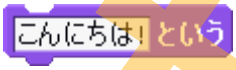
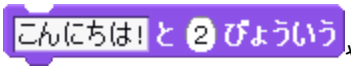
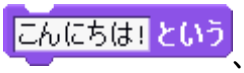
	指定した秒数で、指定した座標に移動する。（指定の秒数で指定の座標に行くようにスライドする）
	x座標に指定の数だけ足して、x 座標を更新する。更新したｘ座標に移動する。
	指定した x 座標に移動する。
	y 座標に指定の数だけ足して、y 座標を更新する。更新したｙ座標に移動する。
	指定した y 座標に移動する。
	ステージの端に触れた時に跳ね返る。
	スプライトのｘ座標
	スプライトのｙ座標
	スプライトの向き

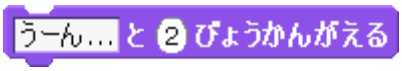
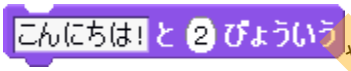
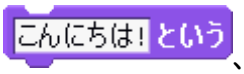
2.「せいぎょ」にあるスクリプト

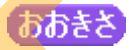
	ステージ上のスタートボタン「緑の旗」がクリックされた時に、このスクリプトに続く命令を実行する。
	指定のキー（左の図ではスペースキー）が押された時に、このスクリプトに続く命令を実行する。
	「スプライト 1」は各スプライトの名前になる。スプライトがクリックされた時にこのスクリプトに続く命令を実行する。
	ステージ（スプライトがないところ）をクリックした時にこのスクリプトに続く命令を実行する。
	指定の秒数が経つまでこのスクリプトに続く命令を実行せずに待つ。
	ブロック内の命令をずっと繰り返し実行する。
	指定の回数だけブロック内の命令を繰り返し実行する。
	全てのスプライトとステージに指定したメッセージを送る。
	全てのスプライトとステージに指定したメッセージを送った後、このメッセージを受け取ったスプライト、あるいはステージのスクリプトの実行が終わるのを待つ。

	指定したメッセージを受け取った時にこのスクリプトに続く命令を実行する。
	指定した条件を満たしている場合に限りブロック内の命令をずっと繰り返し実行する。条件を満たさない（満たさなくなった）場合には繰り返し実行はしなくなる。（一回でも条件を満たせばずっと繰り返し実行をするという意味ではないので注意） 下の図のスクリプト構成と同じと考えれば良い。 
	指定した条件を満たした場合にブロック内の命令を実行する。
	指定した条件を満たした場合に「もし……なら」のブロック内の命令を実行する。そうでなければ「でなければ」のブロック内の命令を実行する。
	指定された条件が満たされるまで、このスクリプトに続く命令は実行せずに待つ。
	指定された条件が満たされるまで、ブロック内の命令を繰り返し実行する。
	実行しているスクリプトを停止する。
	全てのスプライト、ステージのスクリプトを停止する。

3.「みため」にあるスクリプト

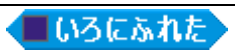
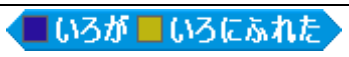
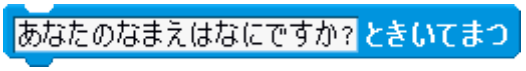
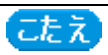
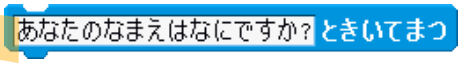
	「コスチューム」に登録したコスチュームの中から指定したコスチュームに変更する。
	次のコスチューム番号のコスチュームに変更する。一番下の場合は一番上のコスチュームに変更する。
	コスチュームの番号は「コスチューム」の表の上から 1 になる
	指定した背景に変更する。
	次の背景番号の背景に変更する。一番下の場合は一番上の背景に変更する。
	背景の番号は「はいけい」の表の上から 1 になる。
	指定したメッセージを指定した秒数間吹き出しの台詞を表示する。
	指定したメッセージを吹き出しの台詞で表示する。新しく  や  、  、  を実行するか、スクリプトを止めるまで吹き出しは消えない。

	指定したメッセージを指定した秒数間考え中の吹き出しの台詞を表示する。
	指定したメッセージを考え中の吹き出しの台詞で表示する。新しく  や  、  、  を実行するか、スクリプトを止めるまで吹き出しは消えない。
	指定した効果を指定した数だけ足して変化させる。初期の効果は０ 効果の数値の範囲は-100～100 指定できる効果： いろ：スプライトの色が変化する。（数が増えていくにつれて色相の時計回りに変化する） ぎょがん（魚眼）レンズ：コスチュームの中心を軸にレンズで拡大したように変化。 うずまき：コスチュームの中心を軸にスプライトが時計回りの渦巻きを描く。効果の数字を大きくするほどより渦を巻くようになる。 ピクセルか：スプライトをピクセルのように正方形の構成で描く。効果の数字を大きくすると正方形の一つの大きさが大きくなるが原型を留めなくなる。 モザイク：コスチュームの幅と縦の間にスプライトが小さくなっていくつも増える。効果を大きくするとスプライトは

	小さくなりより沢山になる。 あかるさ：スプライトの色の明るさ(輝度)が変化する。-100 から 100 で設定する。0 がほどよい色になり、0 より大きくすると明るくなる。0 より小さくすると暗くなる。-100 になるとシルエットのようになる。 ゆうれい：スプライトが透けるようになり、後ろに隠れている背景やスプライトが見えるようになる。効果の数字が大きいとより透けて見え、100 以上に設定すると見えなくなる。
	指定した効果を指定した数字に設定する。指定できる効果は  と同じ
	それぞれの画像の効果がない状態（効果が 0）にする。
	指定した数字の分ずつスプライトの大きさを変化する。
	スプライトの大きさを指定した数字に設定する。
	スプライトの大きさ　オリジナルの大きさが 100 である。
	スプライトをステージ上で見えるようにする。
	スプライトをステージ上で見えないようにする。

	スプライトを最前面に移動する。階層で言うと一番上になる。例えば、別のスプライトに隠れているスプライトにこのスクリプトが実行されると実行したスプライトが前に出て別のスプライトが隠れる形になる。
	スプライトの階層の上げ下げをする。階層が高いほど前に出る。指定した数字が 0 より大きければ階層は下がり（隠れる）、小さければ階層は上がる。（前に出る）

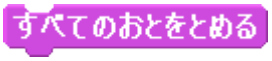
4.「しらべる」にあるスクリプト

	スプライトが指定したスプライト、あるいはマウスポインタやステージの端に当たったか。
	スプライトが指定色に当たったか。
	左側の色が右側の色に当たったか。両方とも色は指定できる。
	指定したメッセージとキーボードで入力するスペースを表示する。スプライトの場合はスプライトが吹き出しでメッセージを表示して、入力スペースは下に表示する。ステージの場合はメッセージと入力スペースが一体化して表示する。
	 で入力した内容。
	マウスポインタのｘ座標
	マウスポインタのｙ座標
	マウスがクリックされたか。
	指定されたキーが入力されたか。
	マウスポインタあるいは指定したスプライトまでの距離
	タイマーを０に戻す。なお、スクラッチの起動時ではタイマーは０であり、リセットしない限り時間をカウントし続ける。

タイマー	タイマー。秒単位で格納している。スクラッチ起動時、またはタイマーをリセットした時からの時間。
スプライト1 の xざひょう	指定したスプライト、ステージの情報。左側ではスプライトやステージを指定できる。 右側で指定できる情報 スプライトの場合（他のスプライトの情報を知りたい場合には特に有効） ・X 座標 ・Y 座標 ・向き ・コスチュームの番号 ・大きさ ・音量 ステージの場合 ・背景の番号 ・音量
おんりょう	外部から接続するマイクの音の大きさ。
うるさい	外部から接続するマイクの音量が 20 より上であるか、 下の図のスクリプト構成と同じと考えてよい。 

	外部から取り付けるセンサーの情報 ・スライダー ・明るさ ・音 ・抵抗 A ・抵抗 B ・抵抗 C ・抵抗 D ・傾き ・距離 このスクリプトを使う場合は遠隔センサー接続を有効にする必要がある。スクリプトを右クリックすれば設定できる。
	外部からどのような処理が行われたか ・ボタンが押された ・A がつながれた ・B がつながれた ・C がつながれた ・D がつながれた このスクリプトを使う場合は遠隔センサー接続を有効にする必要がある。スクリプトを右クリックすれば設定できる。

5.「おと」にあるスクリプト

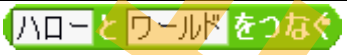
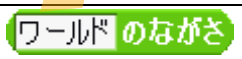
	「おと」に登録した音声の中から指定した音を鳴らす。
	最後まで指定した音声が鳴ってから、このスクリプトに続く命令を実行する。
	全ての音声を停止する。
	選択したドラムの音を指定した拍数分鳴らす。数字をクリックすれば種類が表示される。 拍数については以下の通り 4 分音符：1 8 分音符：0.5 2 分音符：2 全音符：4 16 分音符：0.25 付点 4 分音符：1.5 付点 8 分音符：0.75 付点 2 分音符：3 付点 16 分音符：0.375

	指定した拍数分休止。 拍数については以下の通り 4 分休符：1 8 分休符：0.5 2 分休符：2 全休符：4 16 分休符：0.25 付点 4 分休符：1.5 付点 8 分休符：0.75 付点 2 分休符：3 付点 16 分休符：0.375
	指定した数字の音符（ノート番号という。ノート：note とは英語で音符）を指定した拍数分鳴らす。0 から 127 まで指定できる。60 はピアノの真ん中のドの音。  ある音階は 12 ピアノで見(み)ると白い鍵盤（全音）が 7 つ、黒い鍵盤（半音）が 5 つである。 ノート番号はその音階が高くなることに大きくなる。 つまり、真ん中のド、ノート番号 60 を基準にすると音符は以下の通り ・ド：60 ・ド＃またはレ♭：61 ・レ：62

	• レ＃またはミ♭：63 • ミ：64 • ファ：65 • ファ＃またはソ♭：66 • ソ：67 • ソ＃またはラ♭：68 • ラ：69 • ラ＃またはシ♭：70 • シ：71 • ド：72 音を 1 オクターブ上げたい場合は 12 足す。逆に下げたい場合は 12 減らす。 拍数については以下の通り 4 分音符：1 8 分音符：0.5 2 分音符：2 全音符：4 16 分音符：0.25 付点 4 分音符：1.5 付点 8 分音符：0.75 付点 2 分音符：3 付点 16 分音符：0.375

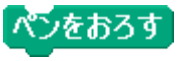
がっきを 1 にする	指定した楽器にする。楽器は数字で指定できる、このような数字をプログラム番号という。数字をクリックすれば楽器の名前が表示される。 楽器の種類は 128 種類あり、ピアノやオルガンといった楽器から銃声や鳥のさえずりまである。
おんりょうを -10 ずつかえる	音量を指定した数字分ずつ変える。
おんりょうを 100 % にする	音量を指定した数字に設定する。
おんりょう	スプライトの音の大きさ 「しらべる」にも同じようなスクリプトがあるがこちらはスクラッチ内部で音量が変わる。
テンポを 20 ずつかえる	テンポを指定した数字分ずつ変える。
テンポを 60 BPMにする	テンポを指定した数字に設定する。
テンポ	現在のテンポ

6.「えんざん」にあるスクリプト

	足し算
	引き算
	掛け算
	割り算、答えは小数点まで出る。
	指定した範囲の数字をランダムに発生させる。整数で指定すると整数のみ乱数が発生する。小数で指定すると小数第１位まで乱数が発生する。
	右の数字が左の数字より大きい場合
	右の数字と左の数字が同じ場合
	左の数字が右の数字より大きい場合
	２ つの条件がともに満たされている場合
	２つの条件のうち、どちらかが満たされている場合
	条件が満たされていない場合
	左側の文字の後ろに右側の文字をつなぐ。左の図の場合は「ハロー」の後ろに「ワールド」をつないで「ハローワールド」
	左側の文字の指定された番目の文字を検出する。左の図では「ワールド」の１番目の文字なので「ワ」
	指定の言葉がいくつ文字があるか。左の図の「ワールド」は４文字なので「4」
	指定された数字を割った余り

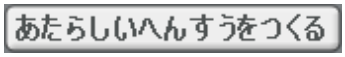
	指定した数字を小数第１位で四捨五入。
	いろいろな関数の数字を検出できる。 検出できるもの ・平方根（√） ・絶対値 この２つは中学校で習う。 ・Sin cos ｔａｎ（サイン　コサイン　タンジェント） ・Asin Acos Ａｔａｎ（アークサイン　アークコサイン　アークタンジェント） 三角関数と呼ばれるもので高校で習う。 ・ln 自然対数と呼ばれるもので高校で習う。 ・Log 常用対数と呼ばれるもので高校で習う。 ・e 指数関数と呼ばれるもので高校で習う。 10^{3} 10 を何乗したか 例えば 3 の数字を入力すると 10 の 3 乗の 1000 になる。

7.「ペン」にあるスクリプト

	ステージにある全ての描画（スタンプや軌跡）を消去する。
	軌跡の描画を開始する。軌跡はコスチュームの中心から描かれる。
	軌跡の描画を終了する。
	ペンの色を指定する。軌跡の描画の際は指定された色で描かれる。
	ペンの色を指定された数の分だけ変える。（色相の時計回りで 200 で一周する） 赤：0 緑：70 青：130
	ペンの色を指定された数に設定する。
	ペンの色の明るさ(輝度)を指定された数だけ変える。0 から 100 では大きいほど色が明るくなり、100 から 200 では大きいほど色が暗くなる。200 から 300 では大きいほど色が明るくなる。つまり、100 単位で色が明るくなったり暗くなったりを繰り返している。 なお、0 から 100 の真ん中の 50、 100 から 200 の真ん中の 150、 200 から 300 の真ん中の 250 は輝度は 50%（暗くもなく明るくもなく丁度いい色）になる。 このことから色の濃さの下 2 桁が 50 であれば丁度いい色合いになる。
	ペンの色の明るさ（輝度）を指定された数字に設定する。

	ペンの太さを指定された数だけ変える。大きくすると太くなる。
	ペンの太さを指定された数字に設定する。
	スプライトのイメージをハンコのようにスタンプする。

8.「へんすう」にあるスクリプト

	クリックすると変数の名前を入力して変数を作成。
	クリックすると変数の名前を入力してリストを作成。
	クリックすると変数名の表が表示され、選択した変数を削除する。
	クリックするとリスト名の表が表示され、選択した変数を削除する。
	現在の変数の値
	変数を指定した値に設定する。
	変数を指定した数字分だけ変化させる。
	変数の値をステージに表示する。
	変数の値をステージに表示しない。
	現在のリストの値
	指定した数字あるいは文字をリストに追加する。
	指定した番号、あるいは最初や全部のリストを削除する。
	指定した番号に数字あるいは文字を格納したものをリストに挿入する。

	指定したリスト番号の値を指定した数字あるいは文字に置き換える。
	指定したリスト番号の値
	リストの数
	リストの中に指定した文字や数字が含まれている項目があるか。

9. 色と濃さの関係

「みため」や「ペン」のスクリプトにある「いろ」や「ペン」のスクリプトにある「いろのこさ」の関係について説明します。

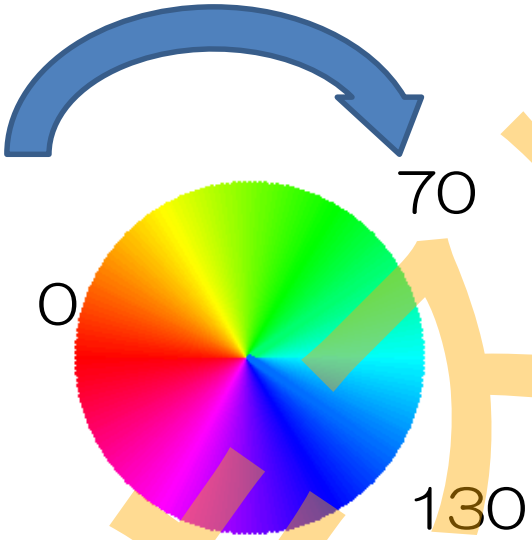
色で説明した色相と、色の濃さで説明した輝度は HSL 表色系の要素になっています。

H が色相、S が彩度、L が輝度です。

色相

下の図は輝度が 50%の時です。この場合は暗くなく、明るくもない丁度いい色合いになります。下の図の円を色相環といいます。色相では赤を基準に 0 度としています。

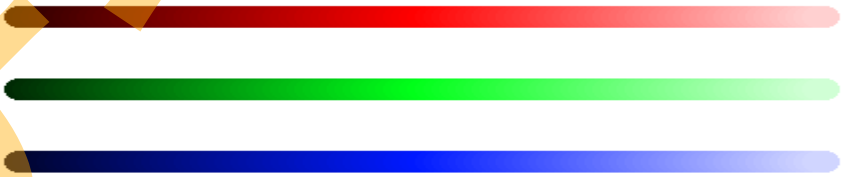
スクラッチでは色は 200 で一周するようになっています。



つまり、0 で赤色になりますが 200 でも 400 でも赤になります。

因みに角度なら赤が 0 度、緑が 120 度、青が 240 度です。

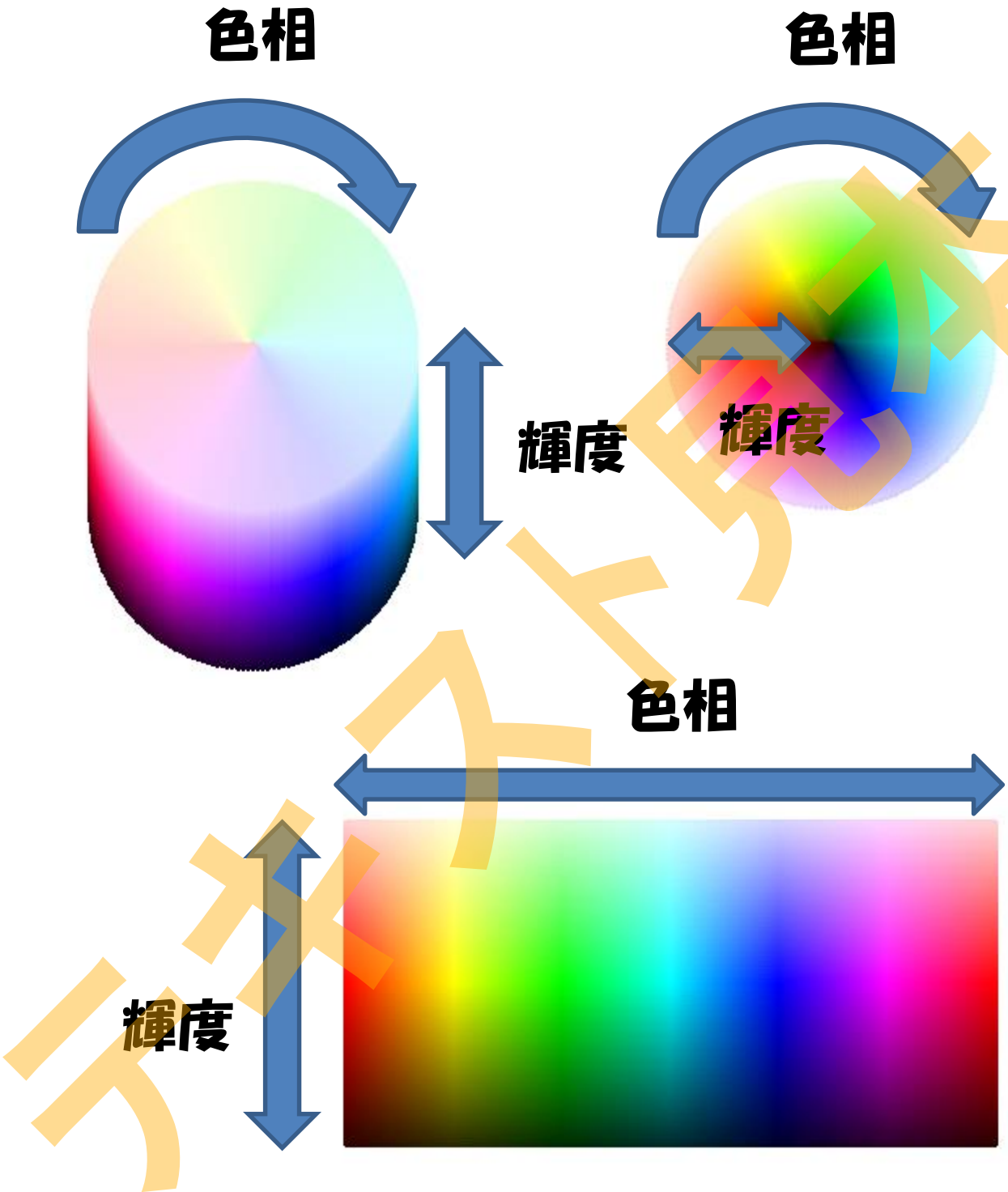
輝度



左端が 0%、右端が 100%になります。50%が真ん中であり、明るくもなく暗くもない丁度いい色合いになります。輝度は 0%の場合は黒になります。

100%の場合は白になります。

スクラッチの場合では白には近づいていきますが厳密に白にはなりません。


色相
色相
輝度
輝度
色相
輝度

V.指導のヒント

テキストの概要、テキストで学ぶ内容、困った時の参考を記載します。

困ったときの参考はテキストに沿って記載しますが、どのテキストでも起こる上手くいかない共通に注意が必要なことがあります。

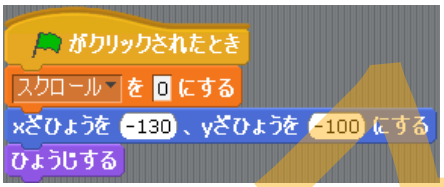
その内容については次の「共通に注意すること」にリストアップします。

1. 共通に注意すること

・パーツの付け忘れ

→スクラッチでゲームを作成するために色んなパーツを組み立てていきますが、もし一つでも欠けているとプログラムが正常に動作しません。

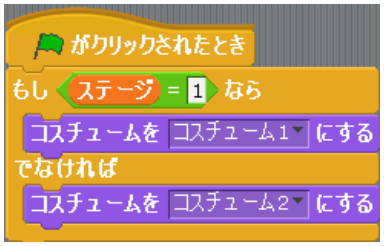
例えば下の図のように組み立てられたパーツがあります。



このパーツの下にある「ひょうじする」のパーツがないと、キャラクターやステージなどが表示されなくなります。

・パーツの挿入箇所

→パーツの挿入箇所がずれると、プログラムが正常に動作しません。例えば右の図のように組み立てられたパーツがあります。



この「もし～なら　～でなければ」のせいぎょのパーツ内にある「コスチュームを～にする」のパーツが逆に挿入されていると、キャラクターやステージなどが「もし～なら　～でなければ」の条件が真逆に表示されます。

・スプライトにパーツが挿入されていない

→スプライトエリアからどのスプライトにそのパーツを置いているか確認します。 このとき、パーツを使って動かしたいスプライトに置いていないと、プログラムが反映されません。

・数値の挿入ミス

→パーツ内で数値を挿入することがあります。このとき、数値の設定（+や-値）を間違えるとプログラムが正常に動作しないことがあります。

例えば、右の図のパーツにある数値「1000」が「-1000」に設定されていると変数が 1000 以下ではなく-1000 以下の時に条件が変わります。

・演算の挿入ミス

→パーツ内で演算を挿入することがあります。このとき、演算の記号を間違えるとプログラムが正常に動作しないことがあります。

例えば、右の図のパーツにある不等号「>」が「<」に設定されていると変数が 1000 以下ではなく 1000 以上の時に条件が変わります。

・パーツ内にあるメッセージの条件が異なる

→全てのスプライトとステージに指定したメッセージを送るときに、挿入するパーツがあります。

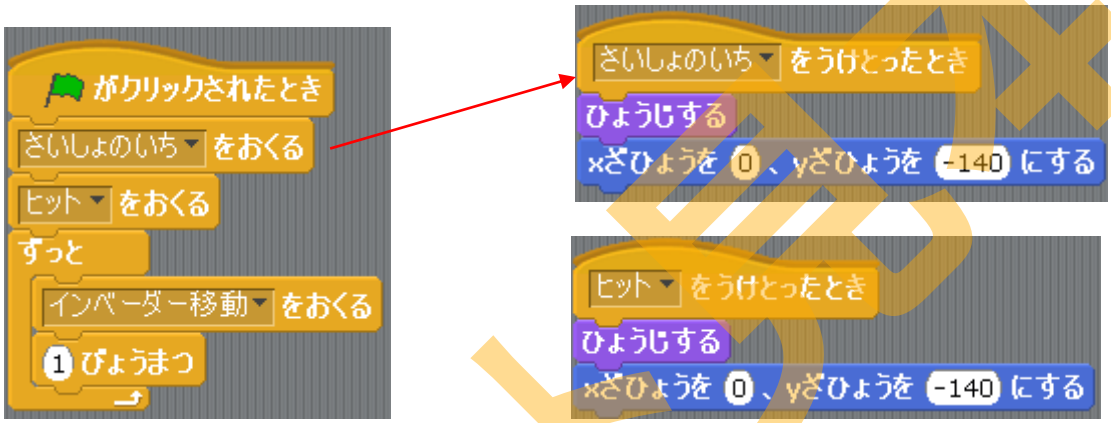
この  をクリックして送るメッセージを「しんき」で作成して、 

でメッセージを送り 

でメッセージを受け取ります。

このようにして、同じメッセージを使ってスプリクトの命令を実行しますが、

二つ以上のメッセージを使用する場合、下の図のように「さいしょのいちをおくる」と「さいしょのいちをうけとったとき」にメッセージが送られますが、「さいしょのいちをうけとったとき」を「ヒットをうけとったとき」に変えるとメッセージは送られなくなり、プログラムが正常に動作しません。



・「いろにふれた」の色の条件が異なる

→ 

の色の条件はこのパーツの  をクリックして、ステージ上の画面から指定したい色をクリックします。

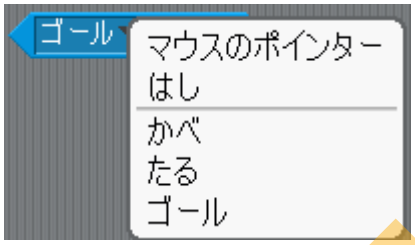
このとき、指定した色を間違えるとプログラムが正常に動作しません。

また、スプライトで細かい部分の色を指定したい時はその見た目の大きさを一時的に大きくして拡大しましょう。

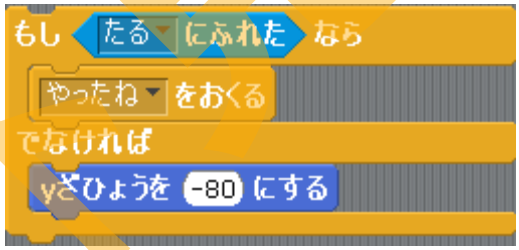
スプライトの見た目の大きさを変更する場合は、見た目の大きさを固定したいスプライトを選択してブロックパレットにある「みため」の「おおきさを…%にする」の「…%」をブロックパレット上で変更してクリックします。これで選択したスプライトの大きさをブロックパレット上で変更した「…%」の大きさに変更できます。

・パーツ内で指定したスプライトが異なる

→複数のスプライトを使用する場合、パーツ内でスプライトを指定するため  をクリックすると下の図のようにパーツ内で指定できるスプライトが増えます。



もし下の図のように間違って別のスプライトを指定してしまうと、プログラムが正常に動作しません。



・作成したスプライトと他のスプライトとの色が同じ場合（似ている場合）

→作成したスプライトと他のスプライトの色が同じ場合（似ている場合）、「いろにふれた」の条件が他のスプライトにも適用されてしまいます。

他のスプライトと色の条件が被らないようにコスチュームの色を変えてみましょう。

・スプライトの見た目の向きを設定する

→スプライトの向きを設定するとき見た目の向きが変わらない、左右にしか反転しないことがあります。スプライトの見た目の向きを設定する場合はスクリプトエリアにある画面左上のボタンを活用しましょう。

を選択するとスプライトの見た目の向きは回転します。

を選択するとスプライトの見た目の向きは左右に回転します。

を選択するとスプライトの見た目の向きは回転しなくなります。

・自分で作成したスプライトの大きさや位置がテキストと異なる

→自分でスプライトを作成した場合、スプライトの大きさや位置がテキストと異なるため、プログラムが正常に動作しないことがあります。

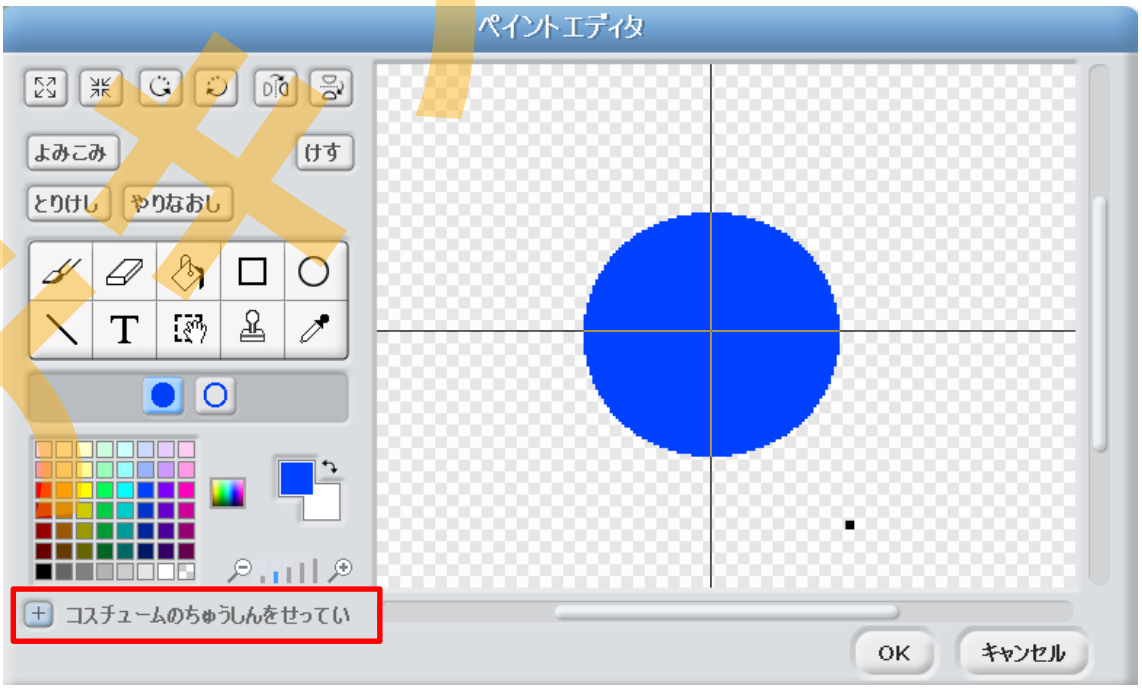
このとき、自分で作成したスプライトの大きさ、位置を変えて調整を行います。

スプライトの見た目の大きさを変更する場合は、見た目の大きさを固定したいスプライトを選択してブロックパレットにある「みため」の「おおきさを…%にする」の「…%」をプロックパレット上で変更してクリックします。これで選択したスプライトの大きさをブロックパレット上で変更した「…%」の大きさに変更できます。

スプライトの位置を調整する際は、ペイントエディタからスプライトの中央を設定することができます。自分で作成したスプライトをペイントエディタで開いて、画面左下にある「コスチュームのちゅうしんをせってい」をクリックします。

すると、下の図のように十字線が表示されます。

これをマウスでドラッグして中央に設定しましょう。



・リストと変数

→ブロックパレットの「へんすう」からリストもしくは変数を作成することができます。

変数は名前を付けて、それにあった値をいれることで必要な時にそれを取得することができます。

リストはいくつもの変数を追加して番号で指定することができます。

このとき、もし変数を作成しようとしたときに間違ってリストの方で作成しないようにしましょう。

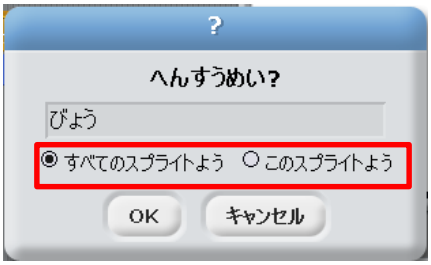
・リストの番号に変数を当てはめる

→リストの番号に変数を当てはめる場合はリストの番号が対応した番号になっているか確認しましょう。

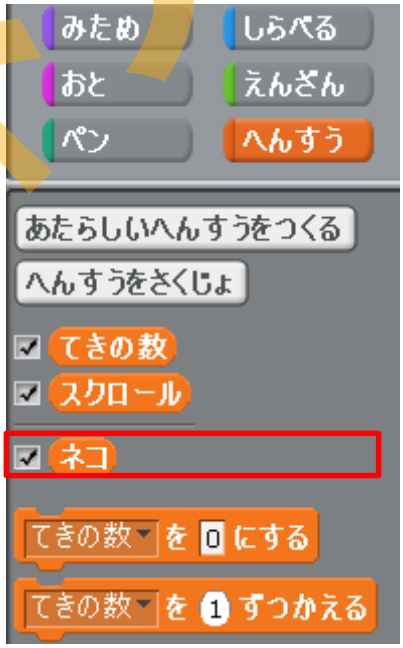
リスト番号	当てはめる変数
1番目	やさい
2番目	にく
3番目	さかな

・変数やリスト作成時に他のスプライトと共有する

→あたらしい変数やリスト作成時、下の図のように「すべてのスプライトよう」か「このスプライトよう」にするか選択する画面があります。



「すべてのスプライトよう」を選択すると他のスプライトからその変数やリストにアクセスでき、逆に「このスプライトよう」を選択すると、その変数やリストを作成したスプライトだけがアクセスできます。「すべてのスプライトよう」で作成すると、意図しないところで変数の値が変更されることがあるため、テキストで「このスプライトよう」で変数やリストを作成する時は「すべてのスプライトよう」を選択しないようにしましょう。変数やリストが「このスプライトよう」で作成されていれば、そのスプライトを選択してブロックパレット上の「へんすう」を確認すると、右の図のように線の下に「このスプライトよう」で作成された変数が表示されます。



・「以上」と「より大きい」の違い

→数字の大きさを比較するのによく使われる言葉に「～以上」、「～より大きい」という言葉がよく使われます。

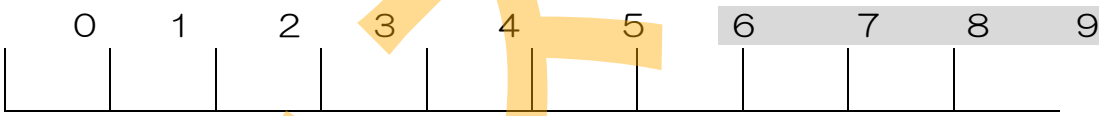
このふたつの言葉の意味はよく似ていますが、プログラミングを作る時には気を付けないといけない大きな違いがあります。

以上は基準となる数字を含む、よりは基準となる数字を含まない！！

例文）「5 以上」の数。

この例文の場合の数字は、5 を含めた数字すなわち上の表では5，6，7，8，9の数字が当てはまります。

例文）5「より大きい」数。



この例文の場合の数字は、5 を含めない数字すなわち上の表では6，7，8，9の数字が当てはまります。

学習しているスクラッチでへんすうは５「以上」という条件が出てきた場合

このままでは条件を満たしていません。

は「より大きい」をあらわす記号です。そのままだとへんすうは５「より大きい」すなわち５を含みません。

５「以上」という条件にする時は５から１引いた数字すなわち４を使います。

これならへんすうは４「より大きい」＝5「以上」条件を満たします。

2. スクラッチの基本操作

①ゲーム概要

スクラッチの画面説明や基本的な操作を学び、
UFO 撃退ゲーム（シューティングゲーム）を作成します。
UFO 撃退ゲームはプレイヤーがショットを発射して UFO を撃退するゲームです。
プレイヤーを左右に動かしてショットボタンを押すとショットが発射され、
ショットが UFO に当たると UFO が消えます。

②学ぶ内容

スクラッチを初めて学ぶ初心者向けのテキストです。
ネコを動かす、動きの切り離し・削除・挿入、
動きの繰り返し、ネコのコスチュームを替える、実行ボタンと中止ボタンを作る。
プレイヤーや背景などの画像作成（ペイントエディタの使い方など）など行なう。
他）動きの向き、条件分岐、コスチュームの追加、待機処理、動きの停止、座標の理解、
当たり判定処理、音の発生などを学びます。
最後に UFO 撃退ゲームの作成、プログラムがうまく動作しないときの対処法などを学びます。

③困りそうなこと

・テキスト通りに行って、プログラムを実行したが反映されない。

→作成したプログラムが合っている場合、そこまでのデータを一旦保存してからスクラッチソフトを再起動してみます。これでプログラムが正常に反映されることがあります。

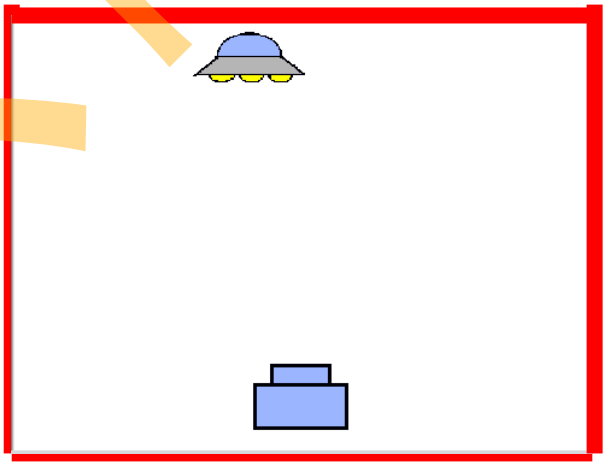
・UFO が出現しない

→UFO を出現させようとして UFO が出現しない時があります。これを解決するためには UFO を動かしている動作、「くりかえし」の条件にある

を正しく理解しましょう。

とは名前の通り、端に触れた場合です。つまり端に触れた時、繰り返しを止めると UFO は

で見えなくなります。この「端」はどこを指すのかといいますと右端、左端、上端、下端になります。

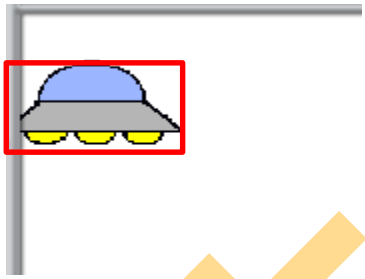


つまり、どの端でも触れれば

の条件が満たされて UFO が見えなくなります。

このことを踏まえた上で UFO のプログラムを動かした時に、最初の UFO の位置を確認してみましょう。

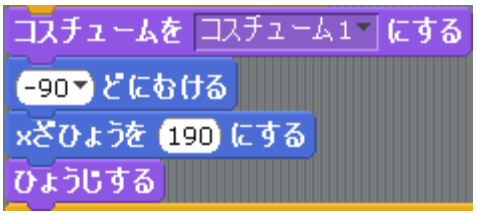
左から右へ動くプログラムで最初の位置を左端に触れる位置に UFO を設定していませんか？
例えば、右の図では左端に触れているので、UFO ははじめから出現することができません。



テキストでは x 座標を-195 に設定しています。テキストの UFO よりも大きな UFO を作成した場合は x 座標が-195 の場合に左端に触れている可能性があります。

これは右から左へ動くプログラムでも同じです。UFO の最初の位置を右端に触れる位置に設定していませんか？
このような場合は、UFO は動かないまま見えなくなってしまいます。

テキストでは x 座標を 190 に設定しています。
これもテキストで作成されているみための UFO より大きい UFO を作成した場合は右端に触れている可能性があります。



これを解決するには２つの方法があります。

1. 最初の位置は端に触れない座標に設定する。x 座標の設定を変えましょう。

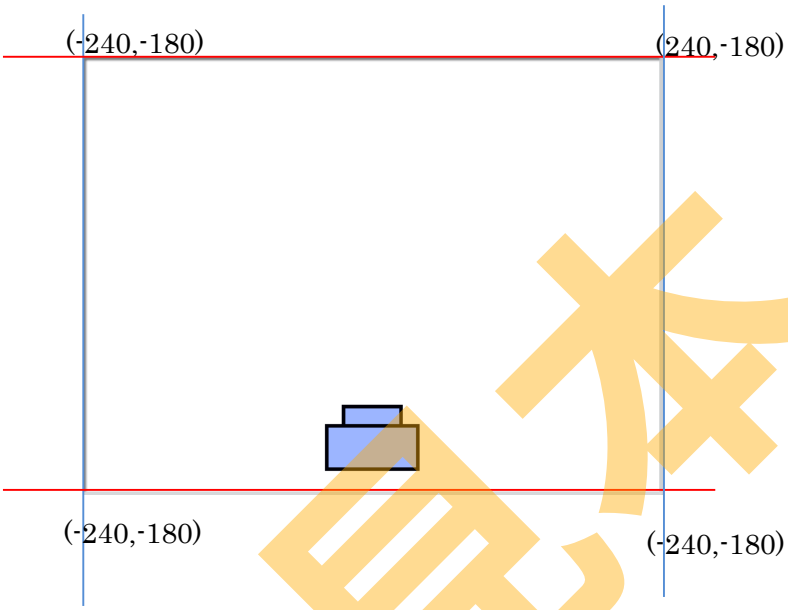
2. 繰り返しの条件に 

を使用せず、座標を用いて「端に触れた」の条件を生成する。（  、  、 

などを用いて条件を生成します）

どちらの場合も UFO の座標とステージ画面端の座標がどのようになっているかを知っておきましょう。

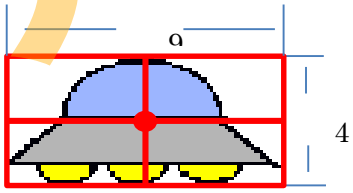
ステージ画面端の座標は以下のようになっています。



左端のｘ座標は-240
右端のｘ座標は 240
上端のｙ座標は 180
下端のｙ座標は-180

UFO の座標について

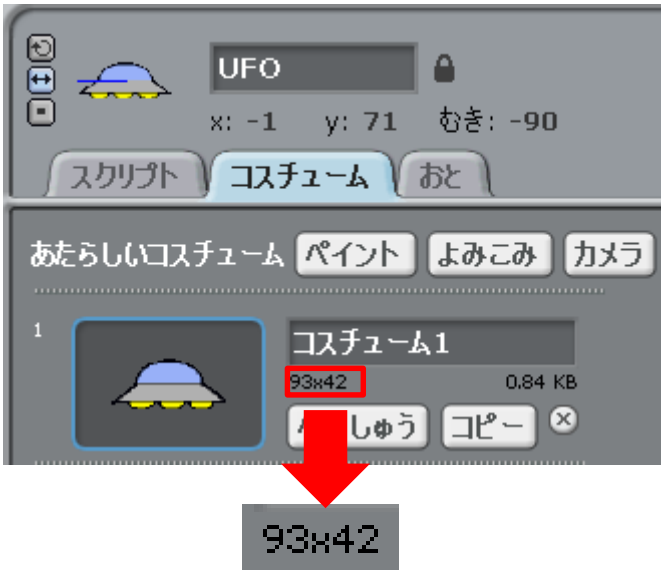
UFO はペイントエディタを用いて作成しました。このとき、UFO のｘ座標とｙ座標は UFO の中心になります。UFO の横幅と縦幅はコスチュームで確かめることができます。

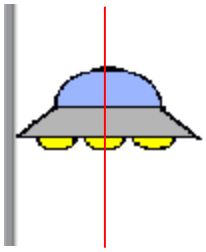


右の図の赤い枠にコスチュームのサイズが表示されています。

「93×42」の前にある数字の 93 が横幅、後ろにある 42 が縦幅になります。

自分で UFO を作成すると、この縦幅と横幅は右の図と違う数字になることがあります。





このことを踏まえて右端に到達する時、左端に到達する時のUFOの座標がどうなっているかを考えてみましょう。これがどちらの解決方法にも大きなヒントになります。

例えば左端についている場合を考えます。

左端の座標は-240、

UFOのx座標は左の図の赤い線です。

この2つの間の幅はUFOの横幅の半分になります。

そして赤い線は左端よりも右にあります。

ではUFOのx座標はどのように表せるでしょう？

UFOのスプライトの横幅は93です。これの半分ですから2で割ってみましょう。

答えは46で余りが1になります。この余り1はそれほど気にしなくても大丈夫です。

左端の座標は-240でUFOの横幅の半分の幅だけ右にあるため

-240+46=-194です。

これが左端についている時のUFOの最も大きな座標になるため、これに1を足せば左端に触れないことになります。

・UFO にショットが当たった時、UFO あるいはショットが消えない

→UFO にショットが当たった時、UFO に変化がなく動き続けている、またはショットに変化がなく動き続けている。

これは前に紹介した端に触れたと同じように何かに当たった場合についてですが大きな違いがあります。

1 つは UFO とショットのスプライト同士が当たっているということです。

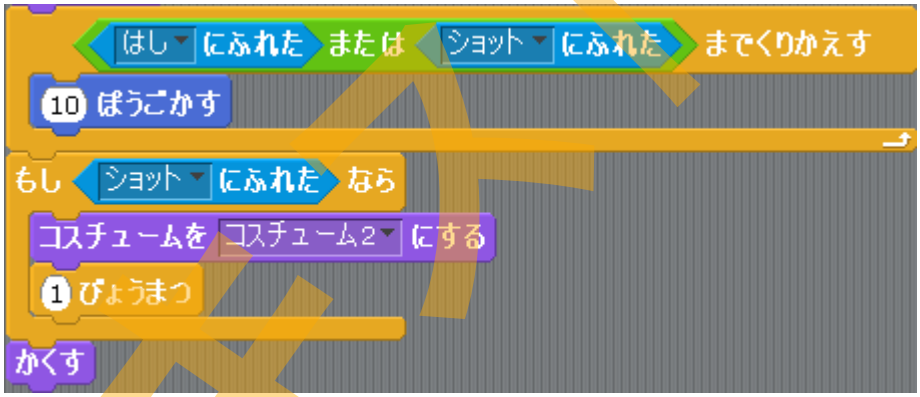
もう 1 つは両方とも動いているということです。

そのため、それぞれ UFO とショットにはそれぞれが当たった時のプログラムがあります。

UFO のプログラムでは

や

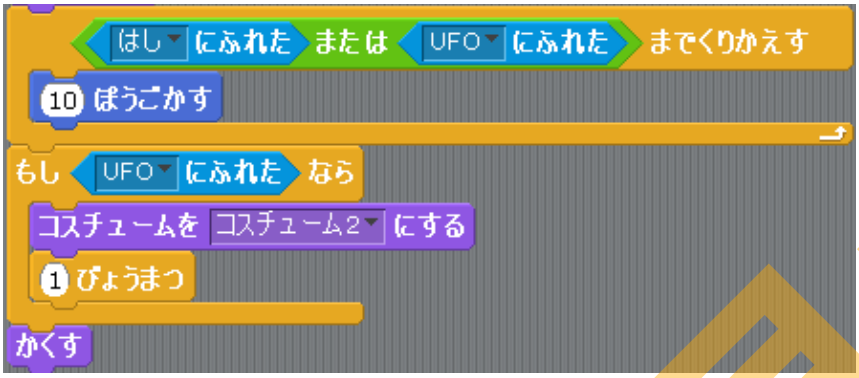

になります。



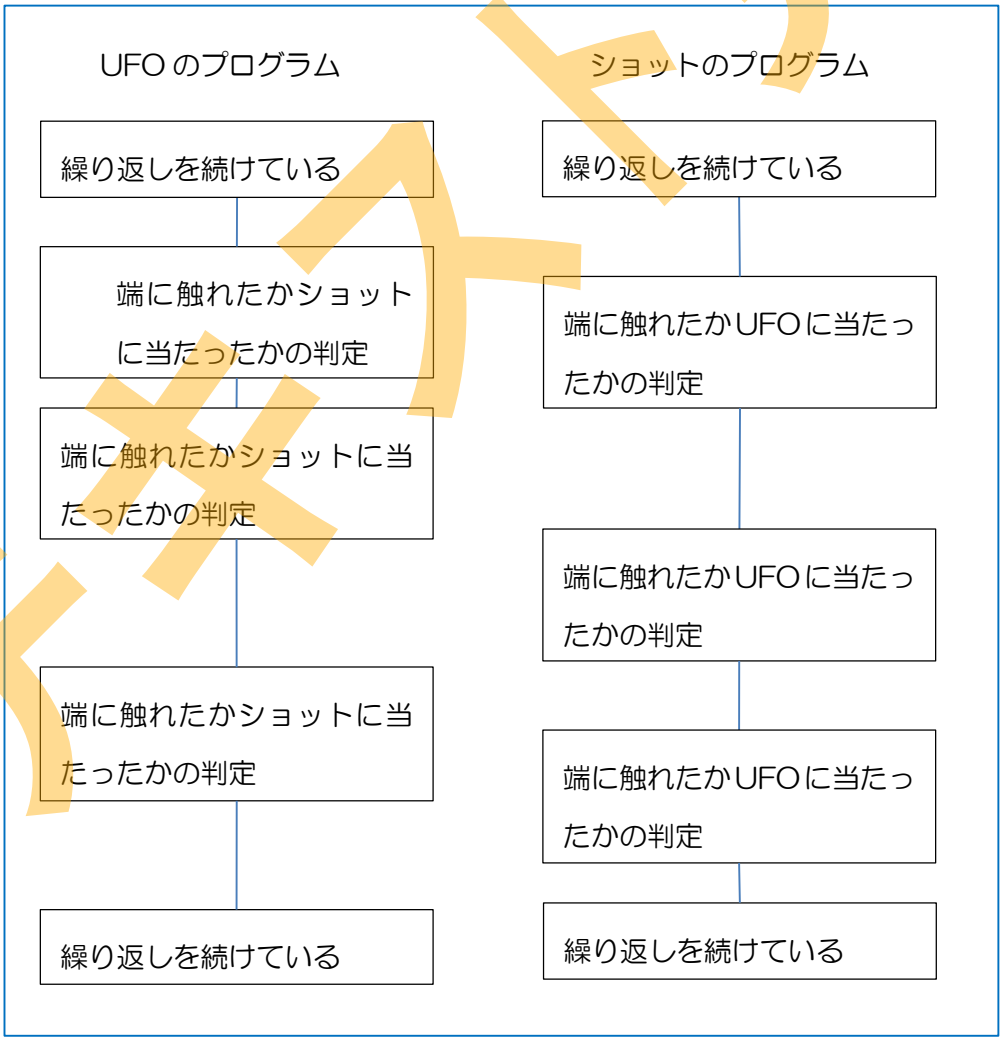
ショットのプログラムでは 

や


になります。



しかし、UFO とショットのそれぞれの判定は同時に行っているでしょうか？UFO を動かすタイミング、ショットを動かすタイミングによってこの２つの判定には 1 秒も満たないほどの差が生まれます。



実際に上の図のプログラムの繰り返しの中に 

を入れると入れないでは動きが全然違いますね。

を入れるとかくかくした動きになってしまいます。数字を 0.5 に変更してもやはりかくかくしています。つまり、1 つの命令の実行に 0.5 秒もかかっていると動きがかくかくになるということです。でも、実際はかくかくにはなっていませんね。

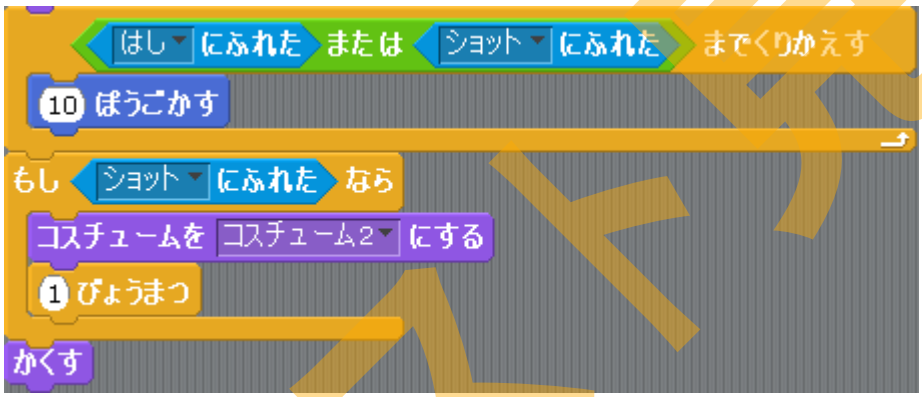
数字をもっと小さくしていくとかくかくした動きがなくなっていきますので試してみるといいかもしれません。

このように上の図の繰り返しは 1 秒よりも速く、ものすごいスピードで動いているのが分かります。ですから、判定のタイミングに 1 秒も満たないほどの差でもスクラッチにとっては大きな差になるのです。

では、実際に UFO とショットの判定のタイミングに差があることでどんな問題が起こり得るでしょう？

ショットのプログラム

UFO のプログラム



ここで UFO とショットが当たった時のことを考えてみましょう。

UFO とショットの判定のタイミングに差がありますから

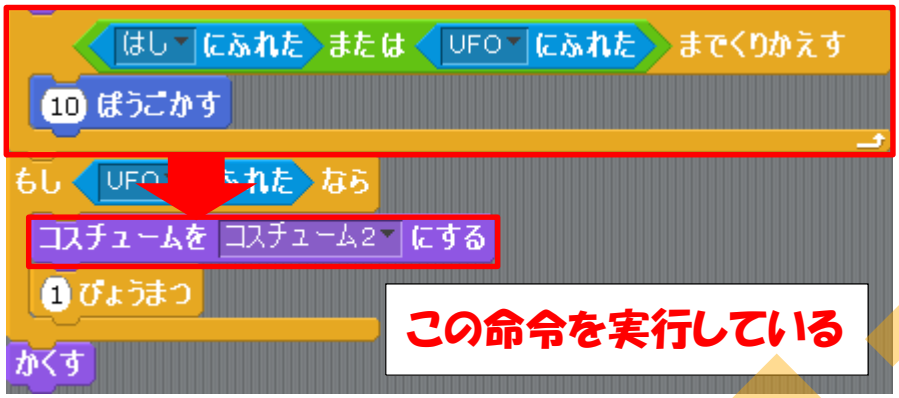
と

の判定にも差があります。

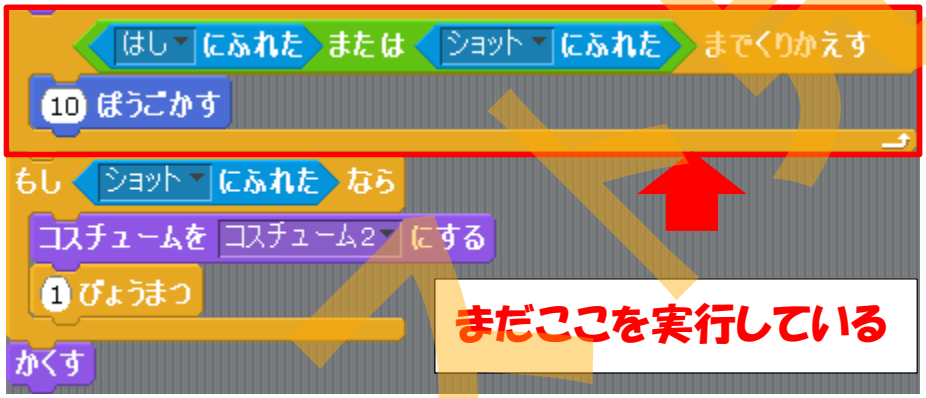
もしも、ショットの判定が速い場合、ショットは UFO に当たった判定して繰り返しを止めて

を実行します。

ショットのプログラム



しかし、この時、UFO はショットよりも遅く、まだショットに当たったかの判定を行っていないため、繰り返しから出ていません。



ここで UFO が判定を行う時、UFO は「コスチューム 1」のままなのに対して、ショットが「コスチューム 2」のときに当たっているかどうかを判定します。

ここでもしも、ショットの「コスチューム 1」と「コスチューム 2」の大きさ、または形が違うとどうなるでしょう？

例えば、ショットの「コスチューム 1」と「コスチューム 2」がこのようになっているとしましょう。

コスチューム 1 とコスチューム 2 の横幅は 20 と 15 でそれほど差はありませんが縦幅は 38 と 20 で差があります。

これでショットの判定が速い場合を考えましょう。

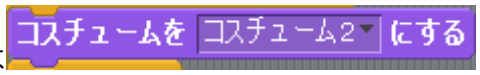
まずはショットの判定が先ですからショットは UFO に当たってコスチュームを変えます。

ここではまだ UFO は判定を行っていませんが、コスチュームが変わったことで上の図では UFO はショットに当たっていないと判定してしまいます。つまり、UFO はコスチュームを変えることなくそのまま移動することになってしまいます。

このように UFO とショット（2 つのスプライト）の判定のタイミングに差があったり、コスチュームの大きさや形の違いが動作に影響を与えることを覚えておきましょう。

判定のタイミングの差は 

を使えば解決します。

実は 

の後にある 

はコスチューム 2 を見せる以外にもの判定のタイミングの差によって起こる不具合を防ぐ役割もあるのです。

この 

をとった状態でプログラムを動かしたらどうなるか考えてみましょう。実際に動かしてみるのもいいでしょう。

の使いすぎて動きがかくかくにならないように注意しましょう。

コスチュームの大きさや形の違いの対策はコスチュームを上手く編集することで解決します。

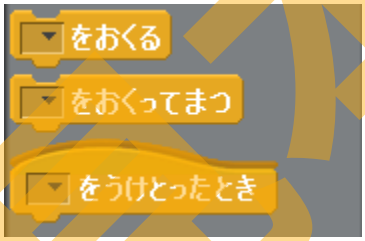
・UFO が出現しない

・UFO のショットが当たった時に UFO あるいはショットが消えない

→もし上記の二種類でまだ解決できない場合はタイミングを合わせるという方法を使いましょう。

UFO とショットの判定には差があるのにタイミングを合わせる方法があります。

の中には右の図のパーツがあります。



は、すべてのスプライトにメッセージを送ることができます。まず、ショットがＵＦＯにふれたとき、UFO にメッセージを送るプログラムを作ります。

ショットのプログラム

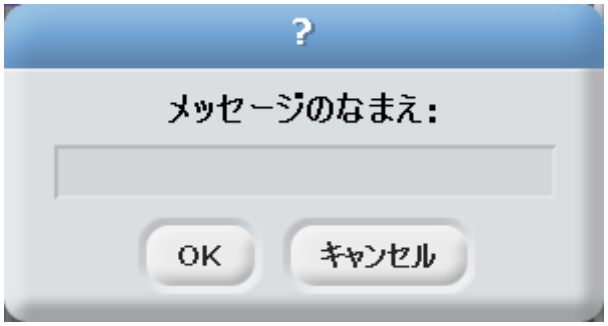
の後に


を挿入します。

に「あたった」を入力します。

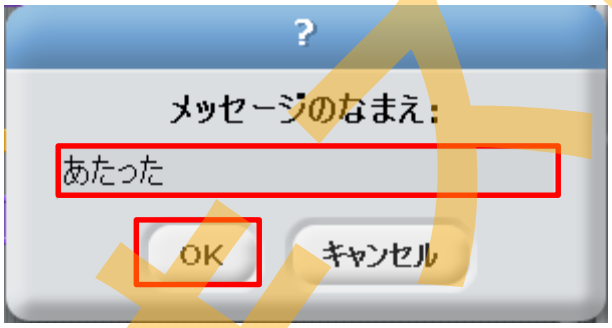
右クリックをして「しんき」を選びます。

すると下のような画面が表示されます。

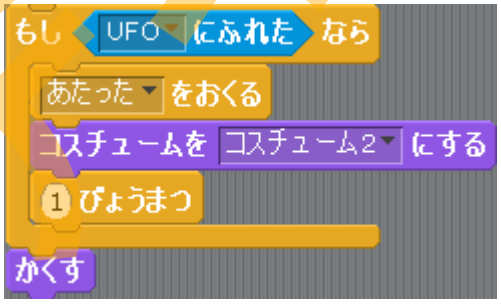


ここにメッセージ「あたった」と入力します。

入力が終われば「OK」をクリックします。



すると「あたった」のメッセージが入力されます。

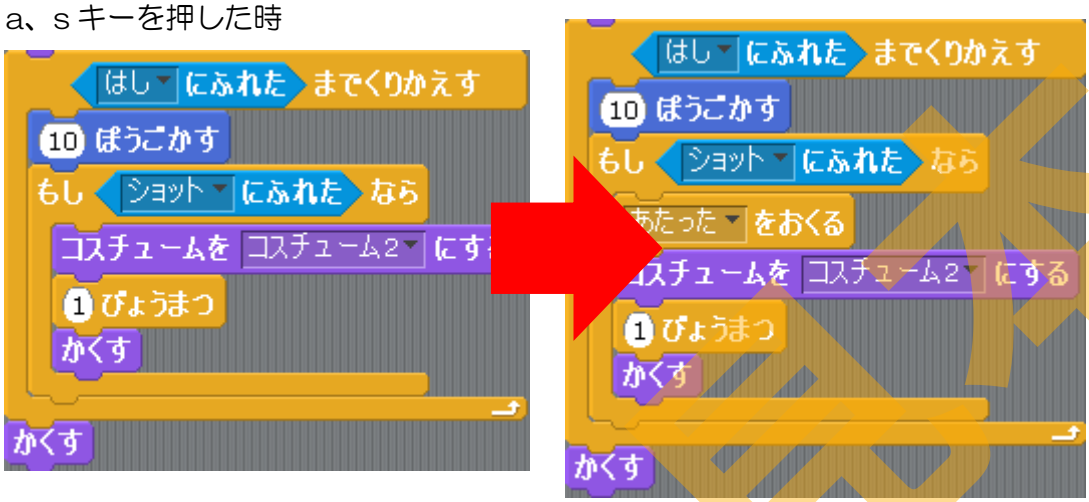


この 

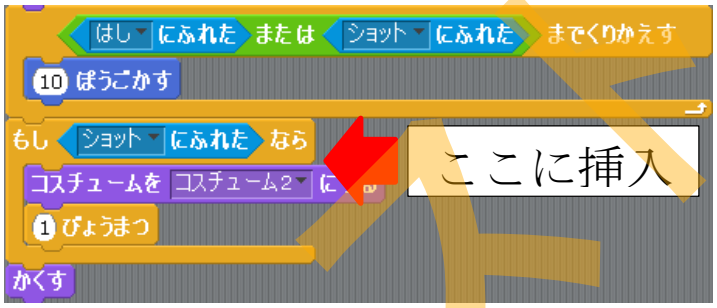
を実行してＵＦＯに「あたった」とメッセージを送ります。

次は、ＵＦＯが先にショットにふれたのを知ったときを考えてUFOのプログラムも同じように変更します。

a、ｓキーを押した時

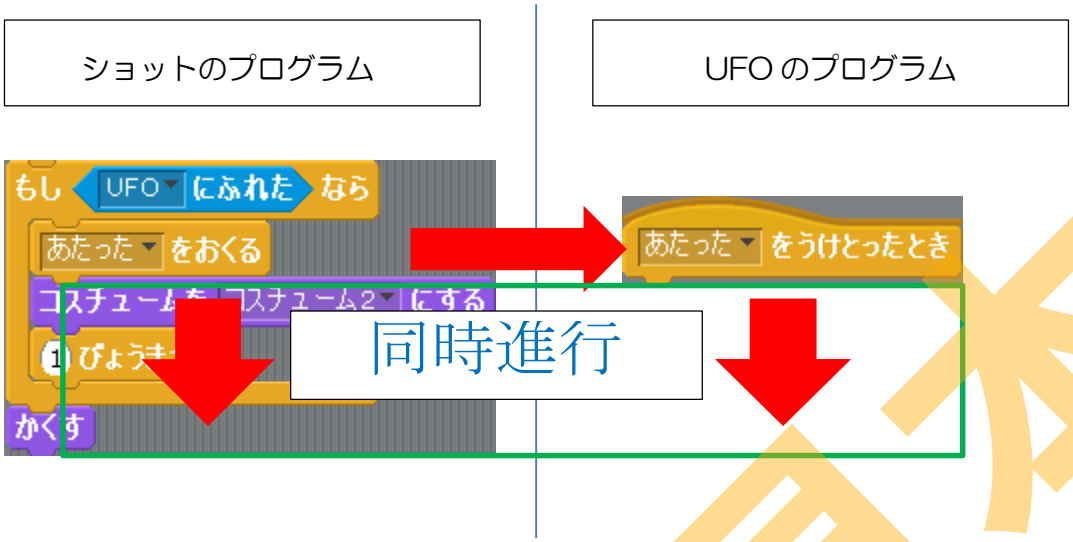


d、ｆキーを押した時



次のようになります。

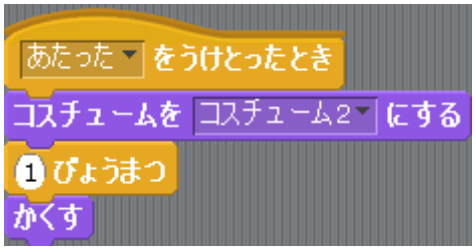
こうしてショットが先にＵＦＯにふれた時、ショットからＵＦＯが先にショットにふれた時にＵＦＯから「あたった」を送ることでお互いに相手にふれたことを知らせることができます。



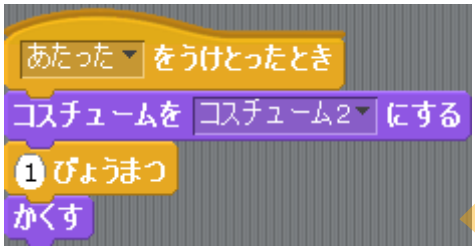
では、UFO とショットのプログラムに 

を挿入した後どのように命令をつなげていけばいいでしょうか。

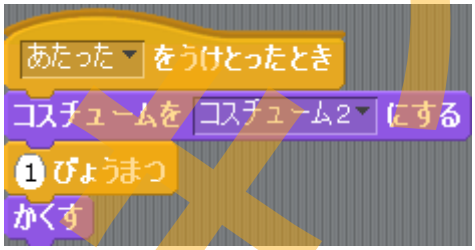
ＵＦＯのプログラムに、ショットから「あたった」と知らされたプログラムを
ａとｓのキーが押された時のプログラムに加えます。



ｄとｆのキーが押された時も同じです。



ショットのプログラムに、UFO から「あたった」とメッセージを受け取るプログラムを作る必要があります。
「あたった」を受けた処理は次のようにおなじになります。



これで、ショットとＵＦＯがあたった時、同時にプログラムを実行することができます。

3. ピンポンゲーム

①ゲーム概要

飛んでくるボールをラケットで打ち返すゲームです。ゲームを開始するとボールが飛んでくるため、ラケットを左右に移動させてボールを打ち返す。ボールが打ち返せなかったらゲーム終了とします。

②学ぶ内容

基本を終了した初心者向け。

ラケットを作成し、プログラミングを使ってそのラケットを動かせるようにします。

続いてボールにも動きを加えてボールが画面端に当たると跳ね返るようにし、ボールが画面下にあたるとゲームオーバーの処理が発生するようにします。

さらに時間が経過するとラケットが小さくなったり、障害物が出現するようにします。

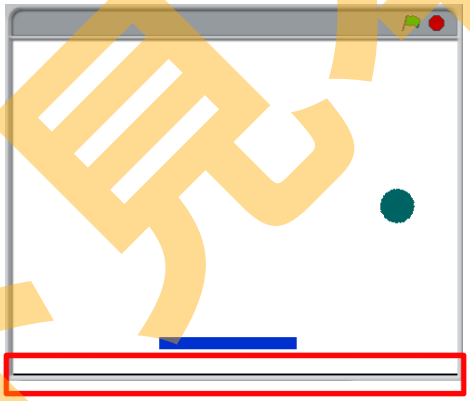
③困りそうなこと

・テキスト通りに行って、プログラムを実行したが反映されない。

→作成したプログラムが合っている場合、そこまでのデータを一旦保存してから
スクラッチソフトを再起動してみます。これでプログラムが正常に反映されること
があります。

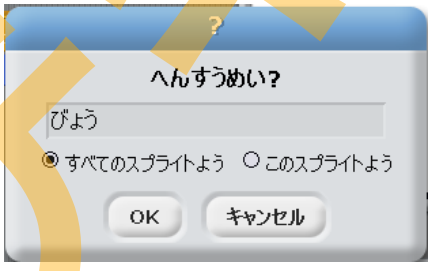
・ボールが画面の下に着いたとき、ゲームオーバーにならない

→p17 からｐ19 までの手順を誤っていない
場合、右の図のように「画面下」のバーが
端に寄りすぎるとボールが当たり判定を
うまく認識しません。画面下の
バーをやや上に移動させてみましょう。



・時間が経過してもラケットが小さくならない。

→p24 で「あたらしいへんすうをつくる」時に「すべてのスプライトよう」ではな
く「このスプライトよう」を選択していると、プログラムが合っていて「びょう」
がカウントされてもラケットは小さくなりません。
変数を作成し直して「あたらしいへんすうをつくる」から「すべてのスプライトよ
う」を選択しましょう。



また時間が経過していない場合、パーツやパーツの並びが合っていても、ステージ
のスプリクトに「びょう」をカウントするパーツが入っていないとプログラムは動
きません。（p25）
「びょう」をカウントするパーツをステージのスプリクトに入れてみましょう。

・割り算の余りから「はち」や「お化け」を表示させるプログラムが難しい

→p52 から割り算のあまりが０になったとき、「はち」が表示されるようにプログラミングを作成していきます。

これは開始から 10 秒経過すると変数の「びょう」は 10 の数字になります。「変数」が 10 の場合、10 割った余りは０になるため、10 割った余りが０のときに「はち」が表示されるようにしています。逆に変数の「びょう」が１や２の場合 10 で割った余りは１や２になり、０ではないため「はち」は表示されません。これで 10 秒、20 秒…のときに「はち」が表示されるようになります。

「びょう」を 10 で割った余りが０のとき

開始からの秒数	10 で割ったあまり		開始からの秒数	10 で割ったあまり	
1	1		11	1	
2	2		12	2	
3	3		13	3	
4	4		14	4	
5	5		15	5	
6	6		16	6	
7	7		17	7	
8	8		18	8	
9	9		19	9	
10	0		20	0	

「お化け」の表示処理も同様です。

4. ブロック崩しゲーム

①ゲーム概要

画面上に置かれているブロックをボールで当てて消していくゲーム。

画面下にあるラケットを使って落ちてくるボールを打ち返します。

ゲームを開始するとブロックの下にあるボールがラケットに向かって落ちてくるため、ラケットを左右に移動させて上方向にあるブロックに向かって落ちてきたボールを打ち返します。

②学ぶ内容

プログラミングを使って画面上に残っているブロックの数をカウントしてゲームのクリア判定を作成します。中級編からはステージの数を増やしてブロックの数や障害物を増やす、障害物に当たるとボールが加速するようにします。

ここでは一定期間記憶し必要なときに利用できるようにするために、データに固有の名前を与える「変数」の使い方や同じ目的の処理を何回か行わせる場合、その部分を一つのプログラムとして独立させ、あとから呼び出して使えるようにする「サブルーチン」の考え方を学びます。

ここからはこういった「変数」や「サブルーチン」を扱うようになります。

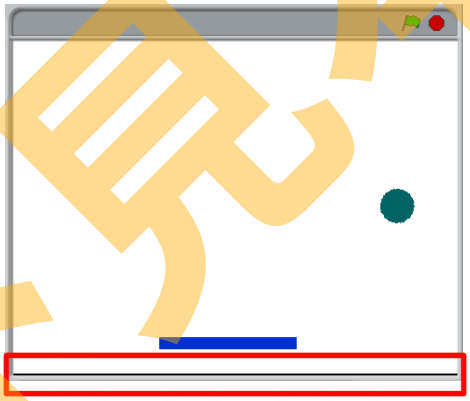
③困りそうなこと

・テキスト通りに行って、プログラムを実行したが反映されない。

→作成したプログラムが合っている場合、そこまでのデータを一旦保存してからスクラッチソフトを再起動してみます。これでプログラムが正常に反映されることがあります。

・ボールが画面の下に着いたとき、ゲームオーバーにならない

→p17 からｐ19 までの手順を誤っていない場合、右の図のように「画面下」のバーが端に寄りすぎるとボールが当たり判定をうまく認識しません。画面下のバーをやや上に移動させてみましょう。



・ボールがブロックに当たっても反応しない

→p22 から「ボールがブロックに当たった時」を「ボールが～色にふれた時」に変更します。このとき 

の■ をブロックと同じ色に変更しますがブロックの色が■ と異なる場合、ブロックにボールが当たっても反応しません。色が異なる場合、 

の■ をクリックしてからステージ上のブロックにマウスのカーソルを合わせてクリックしてみましょう

・ブロックを全部消してもステージ移動されない

→p27 で「あたらしいへんすうをつくる」時に「すべてのスプライトよう」ではなく「このスプライトよう」を選択しているとブロックがうまくカウントされません。変数を作成し直して「あたらしいへんすうをつくる」から「すべてのスプライトよう」を選択しましょう。

「ブロックのかず」の変数が「すべてのスプライトよう」でカウントされていてもステージが移動されない場合は、ｐ36 で 

の数字が

ステージ上に配置されているブロックの数と違っている場合があります。


の数字はステージ上に配置されているブロックの数と同じにしてみましょう。

・正確な座標の位置にブロックを配置したい

→スプライトエリアでブロックを選択してから p18 のようにブロックパレッドの「うごき」の中にある 

と 

を配置したい座標に書き換えてクリックします。これでブロックを正確な座標に配置することができます。

・p63 でボールのスピードを確かめようとしてもステージ２に進まない

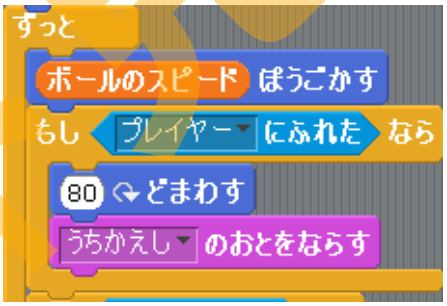
→ステージ２に進むプログラムは上級編でおこなうため、
ステージ１をクリアしてもステージ２に進むことはできません。
そのためステージ２を確認する時
だけステージのスクリプトにある
「ステージナンバーを１にする」を
「ステージナンバーを２にする」
に変更してからプログラムを実行
して確認しましょう。
確認が終わったら「ステージナンバー」を１に戻しましょう。

・障害物が動かない（または出現しない）

→p70 から「えんざん」を用いてｘ座標-200 からｘ座標が 200 以上になるまで「5 ぽうごかす」ように障害物を動かす命令を挿入しますが「えんざん」の  の向きが逆だった場合、障害物が動かない（または出現しない）ことがあります。「えんざん」のパーツを正しい向きに置きかえてみましょう。

・「ボール」がプレイヤーに当たった時に「ボール」がその場で振動する

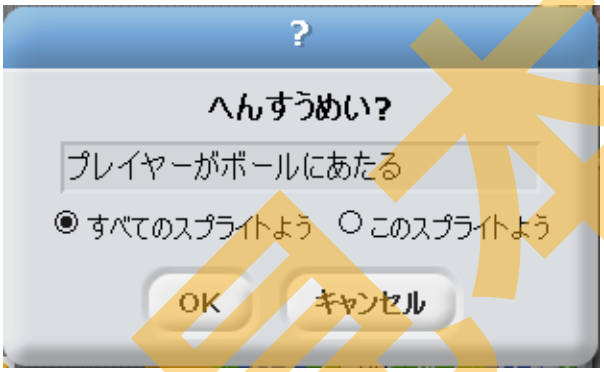
→「ボール」が「プレイヤー」に当たった時のプログラムを見ましょう。右の図がそのプログラムになります。



プレイヤーにふれた時に 80 度右回転するようになっていますが、当たりどころや変える向きによって、「ボール」と「プレイヤー」が当たった状態が続いてしまいます。何回もこの条件が満たされるとボールを何度も向きを変えてしまい、「ボール」が振動しているように見えるのです。

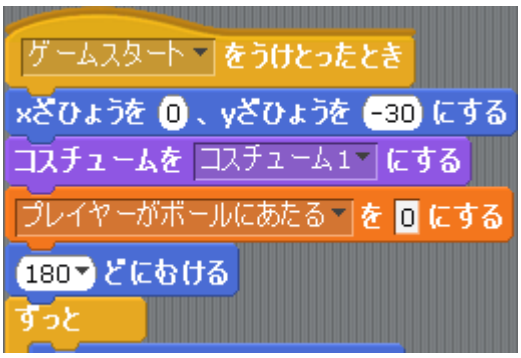
これを解決するためには、「ボール」が「ショット」に当たった場合は一度だけ「80 度右回転する」を実行して、「ボール」が「ショット」から離れるまでは「80 度右回転する」を実行しないようにします。

まず新しい変数を用意しましょう。変数の名前は「プレイヤーがボールにあたる」にします。

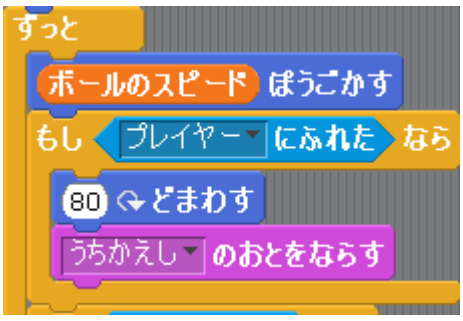


この「プレイヤーがボールにあたる」が 0 の時「プレイヤー」が「ボール」に当たっていない場合、そして 1 の時が「プレイヤー」が「ボール」が当たっている場合、にします。

ゲームスタートをうけとったとき（ゲームが始まったとき）、「ボール」は「プレイヤー」に当たっていないので「プレイヤーがボールにあたる」の変数は 0 にしましょう。



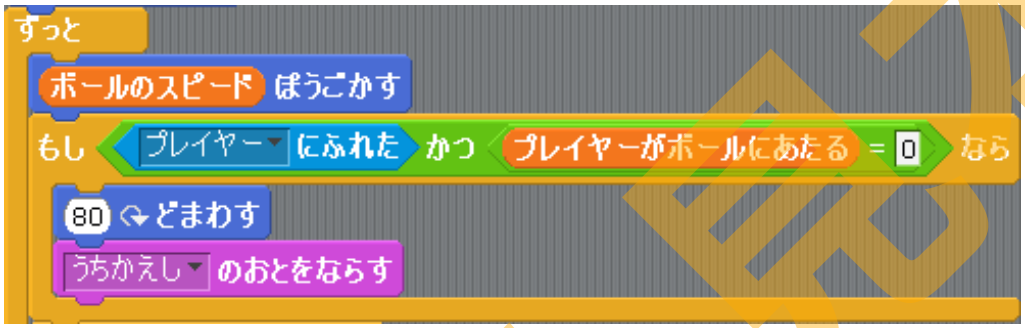
次に「プレイヤーにふれた」ときのプログラムを変更します。最初に「プレイヤーにふれた」場合、80 度右回転するようにします。



最初に「プレイヤーにふれた」時、ゲームが始まったときに「プレイヤーがボールにあたる」の数字は０です。「プレイヤーにふれた」と「プレイヤーがボールにあたる」の数字が０の時を同時に満たす条件にするため、 

を使って２つの条件を挿入します。

これが新しい条件になります。



「プレイヤー」が「ボール」が当たっている時は「プレイヤーがボールにあたる」の数字を１にするため、「プレイヤーがボールにあたるを１にする」を挿入します。

これでもう一度「プレイヤーにふれた」としても「プレイヤーがボールにあたる」は１になるため、右回転の命令は実行されなくなり振動が起こらなくなります。
もちろん、「ボール」が「プレイヤー」にふれなくなった時、「プレイヤーがボールにあたる」を０に戻さなくてはなりません。
そこで「もし……なら」を挿入します。

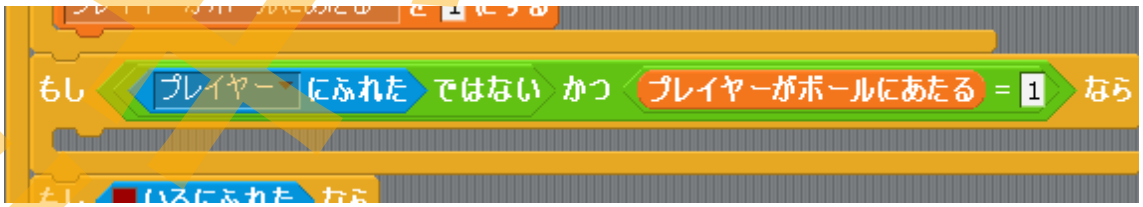
条件は「プレイヤーにふれていない」時かつ「プレイヤーがボールにあたる」が１になる時です。

「えんざん」の中には「ではない」という命令があります。

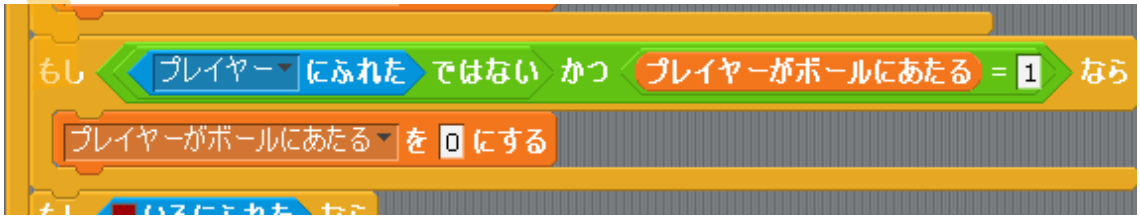
この「ではない」に「プレイヤーにふれた」を挿入しましょう。

これで「プレイヤーにふれていない」という条件が完成しました。これと「プレイヤーがボールにあたる」が１という条件を かつ でつなぎます。

これを「もし……なら」に挿入します。



ここに「プレイヤーがボールにあたる」を０にする命令を挿入します。



これで完了です。

これで振動はしなくなりますが 80 度右回転する命令を実行するため、ボールの向きが下になってしまうことがあります。

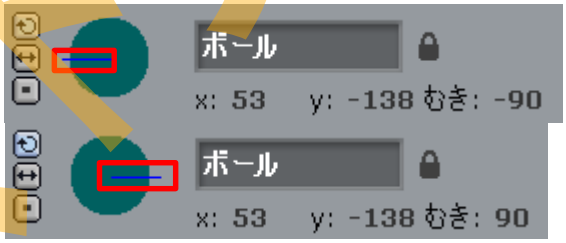
例えば、右の図の状態で下方向にボールが進んでいると「80 度右回転する」の命令を実行しないので、そのまま下に落ちてゲームオーバーになってしまいます。

そこで「ボール」が「プレイヤー」に当たった場合は必ず上の方向に進んでいくように向きを変えるようにします。下の図のプログラムに命令を追加します。

まずは上の方向になる角度の範囲を確認しましょう。

左向きは「むき」が-90 になっています。

右向きは「むき」が 90 になっています。



このことから上方向であるのは「むき」の範囲は-90 より大きく、かつ、90 より小さい場合です。つまり、この範囲内で向きを変えるようにすればいいのです。

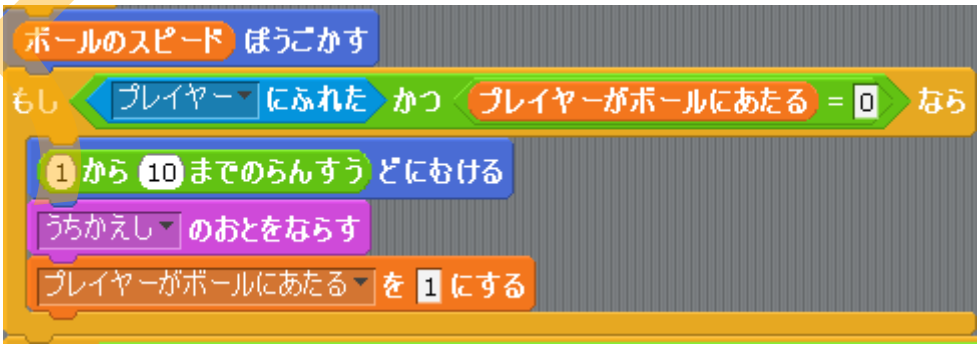
そこでまずは 

を削除して 

を挿入します。

次に「えんざん」から 

を「90▼」に挿入します。



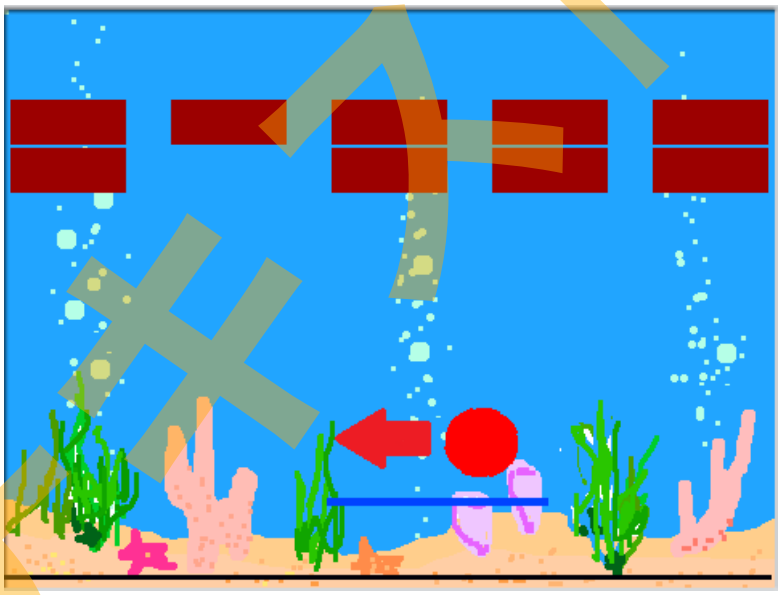
「むき」の範囲は-90 より大きいのですから-89 以上になります。そして、９０より小さいのですから 89 以下になります。

つまり、「-89 から 89 までのらんすう」が発生するようにしましょう。

これでプログラムを動かしてみましょう。

「-89 から 89 までのらんすう」の場合、ボールが大きく左右に動いて、ボールが真っ直ぐブロックの方向へ進まないことがあります。

これは乱数でボールの向きが「-89」や「89」になるためです。



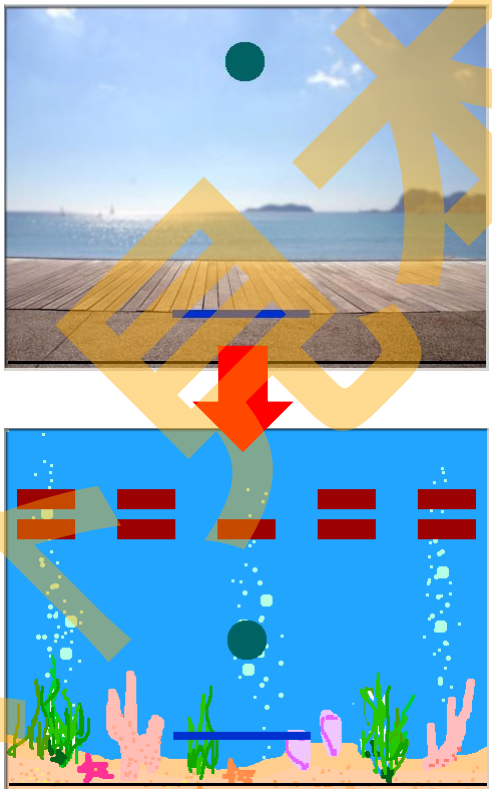
そのため、乱数を「-89 から 89 までのらんすう」から「-60 から 60 までのらんすう」に変えてみましょう。

これでボールはブロックの方向へ進みやすくなります。

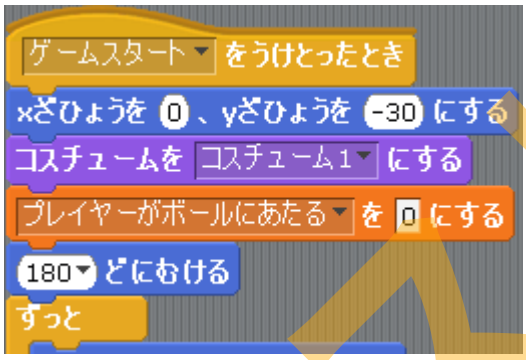
・次のステージに進んだ時、ブロックが一つない場合

→ステージをクリアした時に次のステージに行く時に、ステージをクリアした時に「ボール」の位置にあった「ブロック」が最初からない場合があります。

次のステージに進む時、ボールを開始の位置にする命令とブロックを表示する命令は「ゲームスタートをうけとったとき」に実行しています。

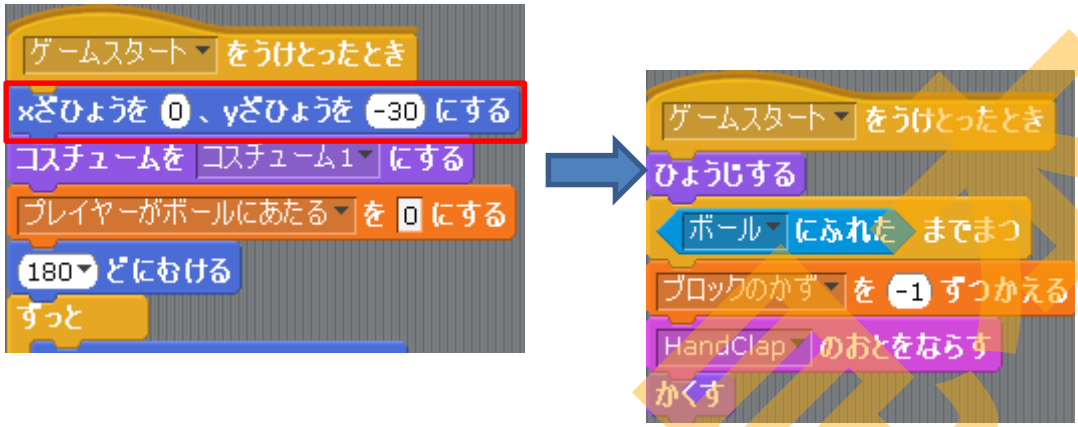


ボールのプログラム



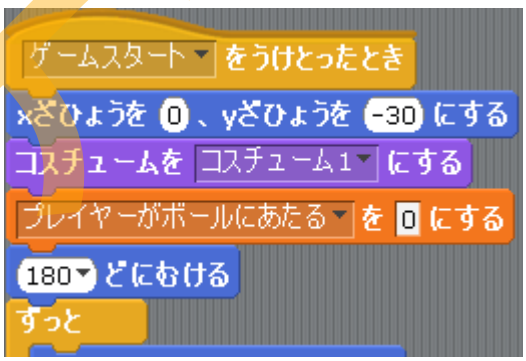
ブロックのプログラム

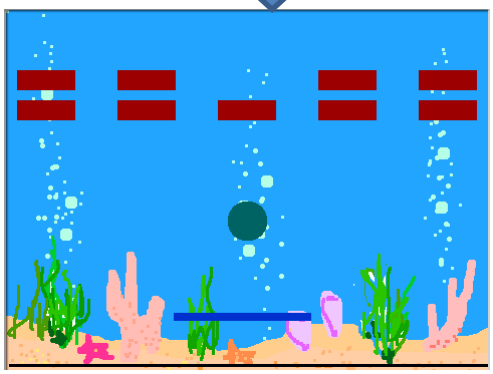
実はボールとブロックの命令には少なからず差があり、ボールのプログラムからボールの座標の設定の命令を先に実行して、その後にブロックの命令が実行された場合（ブロックの配置よりボールの配置が先の場合）は問題がありません。



しかし、先にブロックのプログラムの命令が実行された後にボールのプログラムが開始の時の座標の設定の命令を実行した場合（ボールの配置よりブロックの配置が先の場合）に問題が起こります。

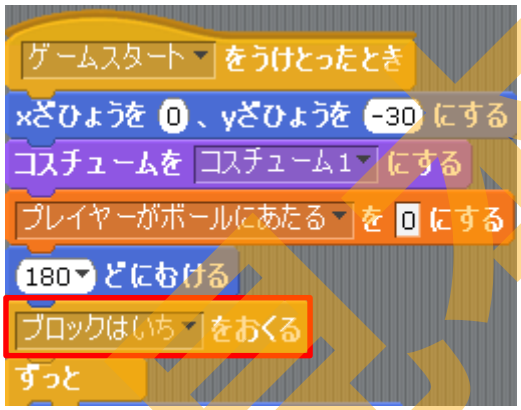
ボールにふれる位置にボールが残っているため（下の図）、「ボールにふれた」の条件を満たしてブロックが消えてしまう。





これを防ぐためにはボールを最初の位置に設定した後にボールのプログラムから全てのブロックを表示するようにメッセージを送ります。

つまり、ボールのプログラムにおいて「ずっと」の前に「せいぎょ」から「……をおくる」を挿入します。メッセージの名前は「ブロックはいち」にします。必ず「x ざひょうを 0、y ざひょうを-30 にする」と「ずっと」の間に挿入しましょう。



次に全てのブロックのプログラムは「ゲームスタートをうけとったとき」に実行しています。これを「ブロックはいちをうけとったとき」に変更します。

これでボールを最初の位置を設定した後にブロックの設定をします。

これでプログラムを動かしてみましょう。

5. インベーダーゲーム

①ゲーム概要

プレイヤーは少しずつ接近してくる複数のインベーダーの攻撃を避けつつ、ショットを打って複数のインベーダーを倒していくシューティングゲームです。

プレイヤーは複数のインベーダーをすべて倒せばクリア、プレイヤーがインベーダーから攻撃を受けたり、インベーダーがプレイヤーのいる場所まで到達した場合はゲームオーバーになります。

インベーダーゲームで用いるプレイヤーやインベーダーなどの素材（画像ファイル）はこの「プログラミング指導の手引き」と同じフォルダ内にある

「ICT」ファイルを開いた場所の「インベーダーゲーム」ファイルにあります。

インベーダーゲームを作成する時はこのフォルダ内にある素材を用います。

「ICT」ファイルをデスクトップにコピーしてからインベーダーゲームの

作成に入ります。プレイヤーやインベーダーなどをペイントエディタで作成することが難しい場合、このフォルダ内にある素材を使用していきます。

②学ぶ内容

プログラミングを使ってプレイヤーからショットを打つだけでなく、敵（インベーダー）からもショットが発射されるようにします。

中級編からはこれらの敵（インベーダー）や敵のショットは複数作成するため、「変数」や「サブルーチン」を扱います。

上級編からは敵のショットを防ぐトーチカ（防壁）や UFO（インベーダーの母船）を作成します。

③困りそうなこと

・テキスト通りに行って、プログラムを実行したが反映されない

→作成したプログラムが合っている場合、そこまでのデータを一旦保存してからスクラッチソフトを再起動してみます。これでプログラムが正常に反映されることがあります。

・インベーダーが増えない

→p41 からスプライトエリアでインベーダーとこうげきのスプライトを「ふくせい」して数を増やしますが、「ふくせい」した全てのインベーダーとこうげきのプログラムの一部を変更していない場合、全てのインベーダーとこうげきのスプライトは同じ座標で重なっていて増えていないようにみえます。p42 のようにすべてのインベーダーとこうげきのスプライトの番号が変更されていて、ステージのスプリクトでそれぞれの番号に座標が設定されているか確認してみましょう。

・インベーダーの動くタイミングがずれる

→p55 で繰り返しの命令を少なくすることでプログラミングがするべき処理を少なくします。これにより処理の負担を減らしてインベーダーの動くタイミングのズレを防ぎます。

・インベーダーを攻撃しても消えない

→P61 で「インベーダーにふれたとき」を 

にかえます。

このとき 

の■ をインベーダーと同じ色に変更しますが、

インベーダーの色が■ と異なる場合、インベーダーにふれても反応しません。

色が異なる場合、 

の■ をクリックしてからステージ上のインベーダーにマウスのカーソルを合わせてクリックしてみましょう

・インベーダーのこうげきしたショットがトーチカをすり抜ける

→p89 でトーチカが壊れていくプログラムを作成します。

ここで 

の■ をインベーダーのショット同じ色に変更しますが、インベーダーの色が■ と異なる場合、インベーダーのショットがトーチカにふれても反応しません。色が異なる場合、 

の■ をクリックしてからステージ上のインベーダーのショットにマウスのカーソルを合わせてクリックしてみましょう

・スプライトの見た目を小さくする

→スプライトの見た目の大きさを固定する場合は、見た目の大きさを固定したいスプライトを選択してブロックパレットにある「みため」の「おおきさを…%にする」の「…%」をプロックパレット上で変更してクリックします。これで選択したスプライトの大きさをブロックパレット上で変更した「…%」の大きさに固定できます。

・インベーダーを倒しても何もないところから攻撃が発射される

→p106 から撃破後のインベーダーの制御を行います。

プレイヤーのショットがインベーダーに当たり、インベーダーのコスチューム番号が「3」になるとそのインベーダーのスプリクトを止めます。

・プレイヤーがインベーダーの攻撃で撃破されてもショットを打つことができてしまう

→p108 から撃破後のプレイヤーの制御を行います。

インベーダーのショットがプレイヤーに当たり、インベーダーのコスチューム番号が「2または3、4」になるとそのプレイヤーのスプリクトを止めます。

6. スーパーキャッツ

①ゲーム概要

スーパーキャッツはキャラクターを操作して、左右に移動させたり、ジャンプしたりして、キャラクターをゴールまで連れて行く横スクロールアクションゲームです。
キャラクターが動くとき、横スクロールのためキャラクターの移動にあわせて画面が左右に移動します。
スーパーキャッツで用いるステージや障害物などの素材（画像ファイル）は
この「プログラミング指導の手引き」と同じフォルダ内にある
「ICT」ファイルを開いた場所の「スーパーキャッツ」ファイルにあります。
スーパーキャッツを作成する時はこのフォルダ内にある素材を用います。
「ICT」ファイルをデスクトップにコピーしてからスーパーキャッツの
作成に入ります。

②学ぶ内容

プログラミングを使ってコースを横スクロールさせてキャラクターを左右に移動させたり、キャラクターを移動させながらジャンプできるようにします。
中級編ではコース上に敵や得点アイテム、障害物を配置したり、水中ステージを作成してステージを切り替えるようにします。
上級編では地上コースや水上コースに加えてもう一つ地上コースを作成してそのコースに謎解き要素を導入していきます。

③困りそうなこと

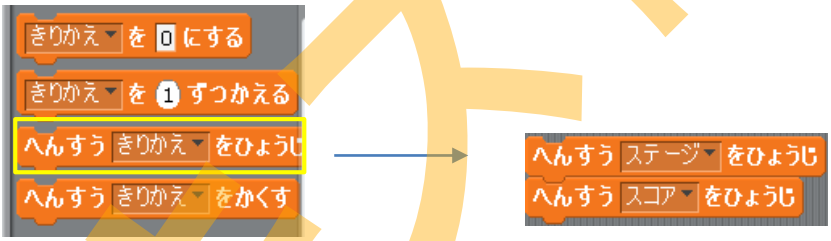
・テキスト通りに行って、プログラムを実行したが反映されない

→作成したプログラムが合っている場合、そこまでのデータを一旦保存し
スクラッチソフトを再起動してみます。これでプログラムが正常に反映されることがあります。

・ゲームを開始すると「ステージ」と「スコア」が表示されなくなる

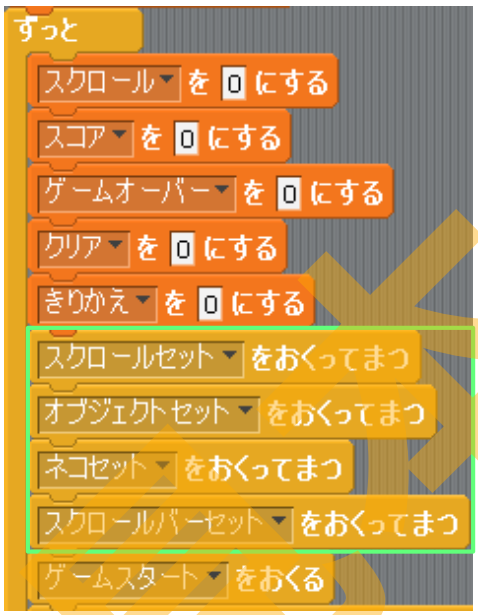
→ゲームを開始してみると「スコア」と「ステージ」が表示されないことがあります。
まず、プログラムのどこかで「スコア」と「ステージ」の変数の中の数を表示する必要があります。

変数を表示する命令がブロックパレットの「へんすう」の中にあります。それが「へんすう……をひょうじ」という命令です。変数名をそれぞれ「スコア」と「ステージ」にした命令を２つ用意します。



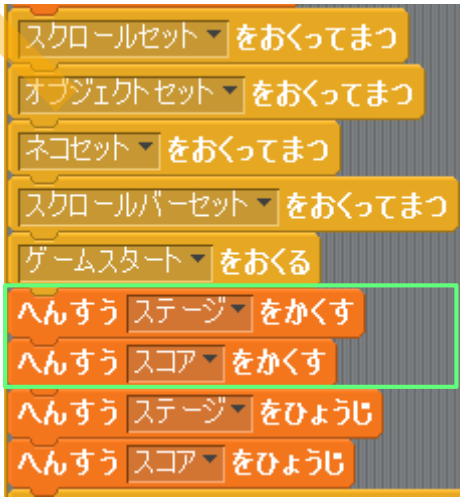
では、これをどこに挿入すればいいでしょう。せっかく「へんすう」を表示しても別の大きなスプライトが「まえにだす」の命令を実行すると隠れて見えなくなってしまいます。そのため、全ての「スクロール」のスプライト、障害物のスプライトが「まえにだす」を実行した後に変数を表示する命令が実行されるようにします。

そこで「ステージ」のスクリプトを見てみましょう。「ステージ」のスクリプトからの「まえにだす」の命令が実行されます。（右の図の緑の枠）



さらにその下に「ゲームスタートをおくる」があります。この後ろに「へんすうステージをひょうじ」と「へんすうスコアをひょうじ」を挿入します。

しかし、これで変数が表示されるのは変数が隠されている場合です。もともと、表示されている変数を「ひょうじする」の命令を出しても「まえにだす」のスプライトに隠れたままになります。そこで「へんすうステージをかくす」と「へんすうスコアをかくす」の命令を「へんすうステージをひょうじする」と「へんすうスコアをひょうじする」の命令の前に挿入します。



これで実行してみましょう。

・マップの端に進むと真っ白な部分に出くわす

→マップの端に移動すると右の図のように左端の方に真っ白な部分に出くわします。ここにはもともと「スクロール」のスプライトが置かれていないため、真っ白になるのです。

これを解決するにはどうすればいいでしょうか？

本来の横スクロールアクションゲームではマップの端に向かうと背景がスクロールするのではなく、キャラクターが端に歩いていきます。つまり、キャラクターのｘ座標が変化するのです。

では、端についたら「スクロール」の座標を変化させずに「ネコ」の座標を変化するようにしてみましょう。

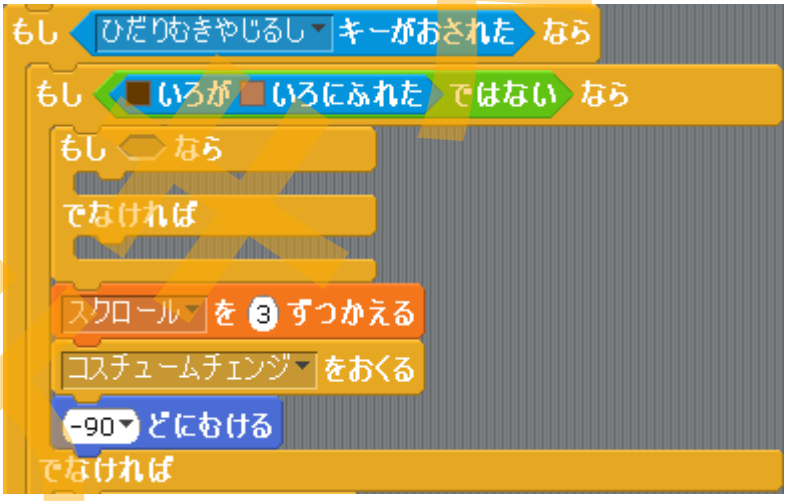
まずはマップが左端あるいは右端まで到達していることを変数「スクロール」で判断しましょう。スクロールの基準は「スクロール 1」の真ん中のｘ座標になります。つまり、「スクロール 1」が画面一杯に表示されている時は変数「スクロール」は 0 になります。右端は「スクロール」が 0 の位置からスプライト 4 枚分の位置になるので-480×4 になります。

スクロール:0

スクロール:-480×4

したがってマップの左端に辿り着いて、かつ、左に移動する場合は「ネコ」のｘ座標を左に寄せます。これを実際にしてみましょう。下の図が「ネコ」の左向き矢印が押された時のプログラムです。

1 まずは「せいぎょ」から「もし……なら、でなければ」を下の図のように挿入します。

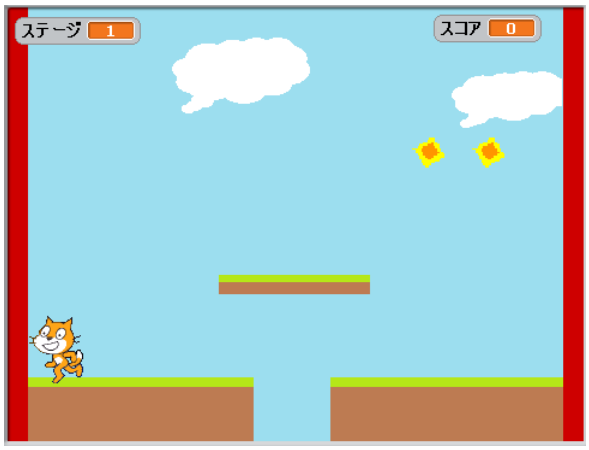


条件はマップの左端なので「スクロール＞-1（スクロールが０以上）」にします。

2 この条件を満たす時に「ネコ」のｘ座標を減らしていきます。

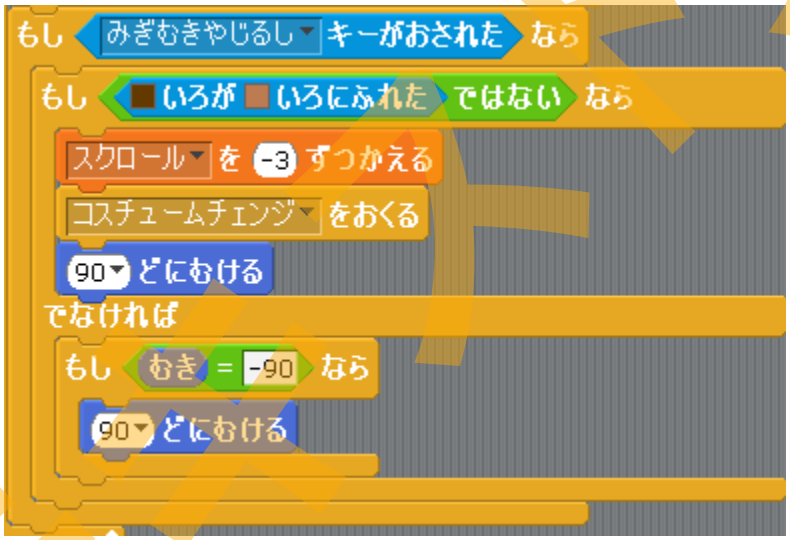
「うごき」から「ｘざひょうを-3 ずつかえる」を挿入します。

満たさない場合は「スクロールを３ずつかえる」にします。「でなければ」に「スクロールを３ずつかえる」を移動します。



3では「ネコ」がどの位置まで左に移動するかを考えます。「スクロールバー」に当たらないこの位置がいいでしょう。この位置の「ネコ」の座標は-199 になります。したがってｘ座標が-199 より大きい場合にｘ座標を減らすようにします。

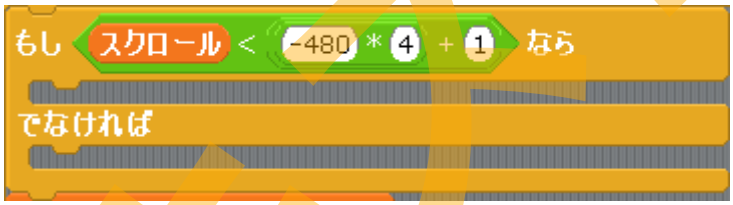
右端に辿り着いて、かつ、右に移動する場合は逆に「ネコ」のｘ座標を増やすようにします。下の図が「ネコ」の左向き矢印が押された時のプログラムです。



まずは「せいぎょ」から

「もし……なら、でなければ」を右の図のように挿入します。

条件はマップの左端なので「スクロール＜-480×4+1（スクロールが 480×4 以下）」にします。



2 この条件を満たす時に「ネコ」のｘ座標を増やしていきます。

「うごき」から「ｘざひょうを３ずつかえる」を挿入します。

満たさない場合は「スクロールを-3 ずつかえる」にします。「でなければ」に「スクロールを-3 ずつかえる」を移動します。

3 どの位置まで右に移動させるかを決めます。

右の図の「ネコ」の位置がちょうどいいでしょう。

この位置のｘ座標は 197 なのでｘ座標が 197 よりも小さい場合にｘ座標を増やすようにします。

さて、これでマップの左端に辿り着いた時に左矢印キーを入力すると「ネコ」が左に移動します。ここで右矢印キーを押した時はスクロールはさせずに「ネコ」を最初の位置に戻るまで右に移動させます。

1 右矢印キーを入力した時のプログラムを見ましょう。「でなければ」に命令を追加していきます。

まず、「せいぎょ」から「もし……なら、でなければ」を挿入します。

条件は「ネコ」が最初の位置より左にある場合です。「ネコ」のスクリプトにある「ネコセットをうけとったとき」に最初の座標が設定されています。

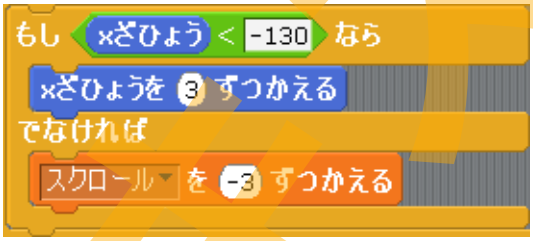
最初はｘ座標：-130 になっているので条件は「ｘ座標が-130 より小さい」になります。

2 この条件が満たす場合は「ネコ」の x 座標を増やします。

「x ざひょうを 3 ずつかえる」を挿入しましょう。

3 条件を満たさない場合は「スクロールを-3 ずつかえる」です。

「スクロールを-3 ずつかえる」を「でなければ」に移動します。



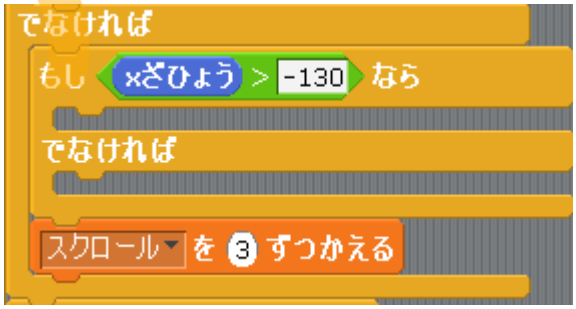
マップの右端に辿り着いた場合も同じです。マップの右端に辿り着いて、なおかつ右矢印キーを入力した場合は「ネコ」を右に移動します。ここで左矢印キーを入力した場合は「ネコ」が最初の x 座標に戻るまでスクロールをしないようにします。

1 左矢印キーを入力した時のプログラムを見ましょう。

「でなければ」に命令を追加していきます。

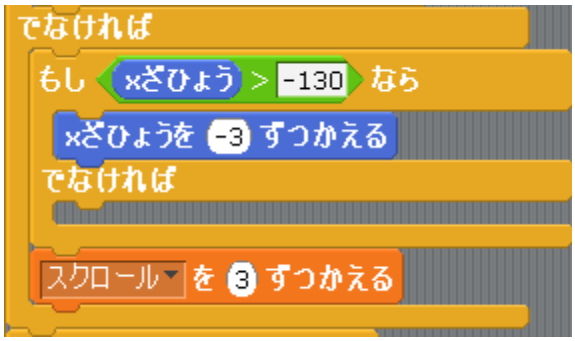
まず、「せいぎょ」から「もし……なら、でなければ」を挿入(そうにゅう)します。

条件は「ネコ」が最初の位置より右にある場合ですから「x 座標が-130 より大きい」になります。



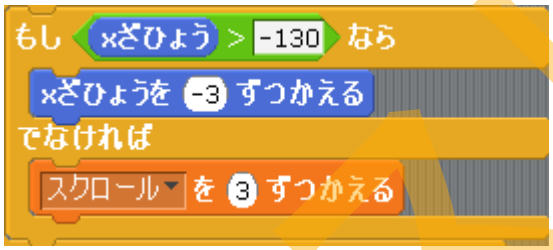
2 この条件が満たす場合は「ネコ」のｘ座標を減らします。

「ｘざひょうを３ずつかえる」を挿入しましょう。



3 条件を満たさない場合は「スクロールを３ずつかえる」です。

「スクロールを３ずつかえる」を「でなければ」に移動します。

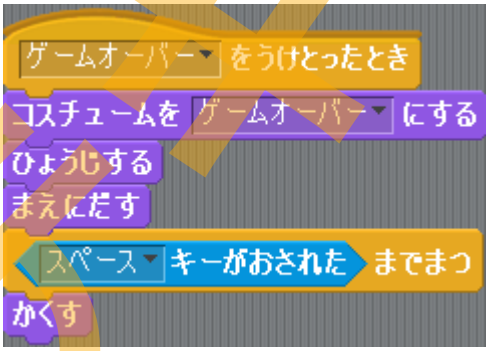


これでプログラムを動かしてみましょう。

・ゲームオーバーの後、再開するとすぐにゲームオーバーになる

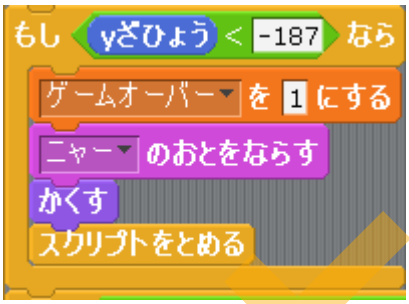
→障害物や敵に当たってゲームオーバーになった後、再開しようとスペースキーを押すとすぐにゲームオーバーになってしまうことがあります。実際に障害物や敵に当たった時、「ネコ」のスクリプトでは右の図のような命令が実行されています。

実はこの命令を実行している時、ステージのプログラムは右の図のようにゲームを終了した後のプログラムを実行しています。この図の「ゲームオーバーをおくってまつ」でメッセージを送ると下の図のような命令を実行されます。



これで「ネコ」の演出が終わる前にスペースキーを押すと、「ネコ」は下降を続けてしまいます。

「ネコ」には落下した時にゲームオーバーになるプログラムがあります。ｙ座標で落下したかを判定してゲームオーバーにしています。



つまり、「ネコ」のゲームオーバーの演出が終わらないうちにスペースキーを押すと、「ネコ」はそのまま落下を続けてゲームオーバーになっていたのです。そこでステージの「ゲームオーバーをおくってまつ」の実行は「ネコ」の演出が終わるまで待ってもらうようにしましょう。

1 「せいぎょ」から「までまつ」を「ゲームオーバーをおくってまつ」の前に挿入します。

2 演出が終わる条件は「ネコ」のｙ座標を使います。

「ネコ」のｙ座標が-187 より小さくなるまで演出は続いています。したがって、「ネコ」のｙ座標が-187 より小さくなるまで待ちます。「ネコ」のｙ座標は「しらべる」にある 

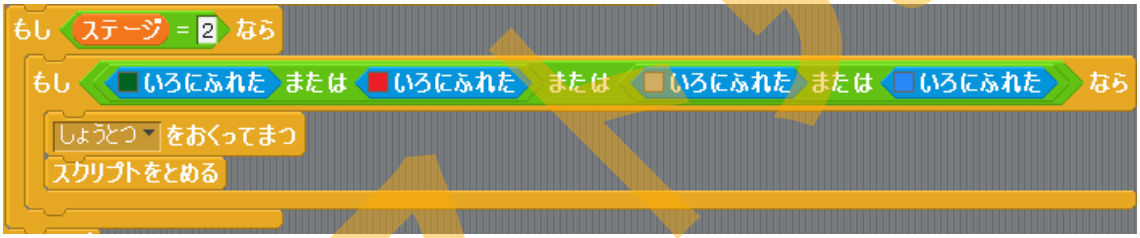
を使います。

・ステージ 3 でスーパーコインを取得したり床が赤くなった「よこバー」に乗るとゲームオーバーになる

→ステージ 2 においてステージ上にあるサンゴに当たった時の判定は色で行っています。ここではその色は赤色になっています。

この色が「スーパーコイン」の赤色 と「よこバー」の赤色 と同じ色になるため、ゲームオーバーになってしまいます。

そこで上の図の命令はステージが 2 の時だけに適用されるようにしましょう。



7. 自動車ゲーム

①ゲーム概要

自動車ゲームとは乗り物を操作し、ライバルと速さを競争して順位を争うゲームです。
コースには障害物が置かれていることがあり、プレイヤーはライバルの車や障害物を避けつつ、より高い順位を目指します。
ここでは縦にスクロールさせる自動車ゲームを作成します。
スーパーキャッツで用いるステージや障害物などの素材（画像ファイル）は
この「プログラミング指導の手引き」と同じフォルダ内にある
「ICT」ファイルを開いた場所の「自動車ゲーム」ファイルにあります。
自動車ゲームを作成する時はこのフォルダ内にある素材を用います。
「ICT」ファイルをデスクトップにコピーしてから自動車ゲームの
作成に入ります。

②学ぶ内容

プログラミングを使ってコースを縦スクロールさせて自動車を前後に移動させたり、プレイヤーのスピードに強弱をつけて、スピードメーターを作ったり、レースの進行を表示するメーターを作成します。そして、レースのゴール位置を決め、プレイヤーが出発してからそのゴール位置に到達した瞬間までのタイムが測定されるようにします。
上級編では競争する相手の車を複数作成して、ゴールした車の順位が表示されるようにします。

③困りそうなこと

・テキスト通りに行って、プログラムを実行したが反映されない

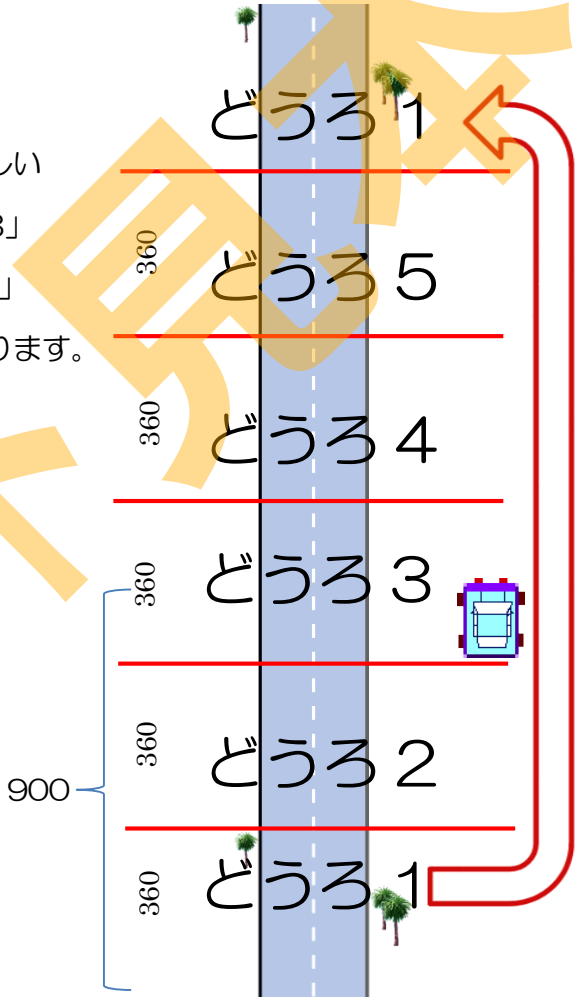
→作成したプログラムが合っている場合、そこまでのデータを一旦保存してからスクラッチソフトを再起動してみます。これでプログラムが正常に反映されることがあります。

初級編

・コースがどのようにして長くなっているのか難しい

→p16 から「どうろ 1」「どうろ 2」「どうろ 3」「どうろ 4」「どうろ 5」からまた「どうろ 1」に戻して…を繰り返すことで長いコースを作ります。

このテキストでは「どうろ 1」の縦の長さは 360 です。「どうろ 2」「どうろ 3」の縦の長さも同じで「どうろ 1」から「どうろ 5」までの長さは合計 360×5=1800 です。

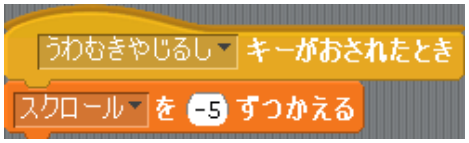


「どうろ 5」に続いて「どうろ 1」「どうろ 2」…を表示させるためプレイヤーが「どうろ 3」を通っている間に「どうろ 1」のスプライトを「どうろ 5」の上に移動させています。そのため、始めは「どうろ 1」から「どうろ 5」までの長さの半分=900 をプレイヤーが通りすぎると「どうろ 1」が移動します。続いてプレイヤーが「どうろ 4」を通っている間に「どうろ 2」のスプライトを先程移動した「どうろ 1」の後に移動させています。「どうろ 3」「どうろ 4」「どうろ 5」も同じように移動していきます。

・p18 の 

のように-1 をかける理由

→プレイヤーが「どうろ」を真っすぐ進ませるとき、プレイヤーは動かさず「どうろ」を下へ移動させていきます。そのため、変数「スクロール」はどうろを y 座標の-方向へ移動させるためマイナスの値になっています。



例えばスクロールの値が 900 になった時に「どうろ１」を移動させたい場合、「-1*-900（スクロールの値）＝900」になるように設定しています（負の数）。

・プレイヤーが「いわ」や「ひと」の障害物を通過した時に「いわ」や「ひと」がステージの下に残る

→p86 からプレイヤーが「いわ」を通過する時以外「いわ」は表示されないようにします。「いわ」の y 座標から「いわ」の表示する範囲（上端～下端）を設定します。これでプレイヤーが「いわ」を通過すると「いわ」はステージの下に表示されなくなります。「ひと」にも同様の処理をおこなうとステージの下に表示されなくなります。

・スプライトの見た目を小さくする

→スプライトの見た目の大きさを固定する場合は、見た目の大きさを固定したいスプライトを選択してブロックパレットにある「みため」の「おおきさを…%にする」の「…%」をプロックパレット上で変更してクリックします。これで選択したスプライトの大きさをブロックパレット上で変更した「…%」の大きさに固定できます。

・プレイヤーがゴールの直前で止まる

→ゴールの y 座標：0 はどうろ 1 の y 座標：0 に対して「プレイヤー」の y 座標：-90 となっています。そのため、P139 から「スクロールの量」が 5000 で「プレヤー」がゴールしたように演出するため「ゴール」の y 座標を-90 ずらしています。

・プレイヤーが「ひと」に当たった後、その「ひと」がずっと動き続ける

→p140 から変数「ゲームオーバー」を作成して、ゲームオーバー＝1 になった時に「ひと」のスクリプトが止まるように設定しています。

・「車」または「いわ」、「ひと」、「メーター」などが表示されなくなった

→ステージ上にある「どうろ」を一度ドラックすると、「どうろ」が前に表示されるようになるため、「車」などのスプライトが見えなくなります。

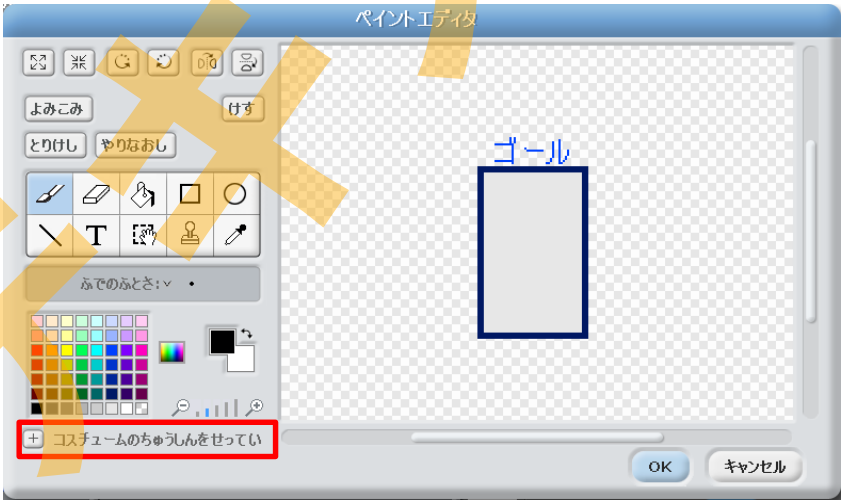
表示されなくなったスプライトを選択した状態で みため から まえにだす をクリックすると「どうろ」の前にスプライトが表示されるようになります。

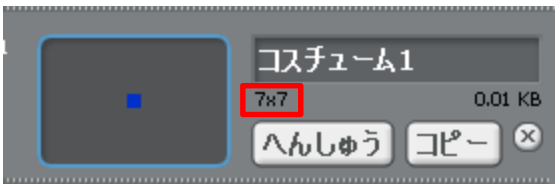
・「レースしんこう」内の「プレイヤーメーター」の位置がずれる

→p148、p150 でペイントエディタから「コスチュームのちゅうしんをせっていい」をクリックして「レースしんこう」と「プレイヤーメーター」のコスチュームの中心をそれぞれ変更しています。

「レースしんこう」内の「プレイヤーメーター」のプレイヤーの位置がずれている場合、コスチュームの中心を変えて調整します。

また、作成した「プレイヤーメーター」の縦幅が7であるかサイズをみます。





・コースの上側または下側がずれた道になっている

→「どうろ」のスプライトが上端または下端に残る場合、ｐ157 から「どうろ」のスプライトを必要な時以外表示させないようにします。

・「どうろ」のスプライトを必要な時以外表示させてから、「どうろ」をスクロールさせると途中からどうろが表示されなくなる。

→p157 から条件式を組み立てていきますが、条件のパーツは

の場合、（「くぎり」*1800）+（360*4）となります。

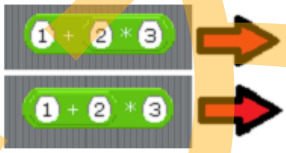
これが

の場合、「くぎり」＊｛1800+（360*4)｝となり

数字の並びが同じでも、パーツの囲いが違うと答えがそれぞれ異なります。

そのため、条件式を組み立てていくときは、条件のパーツの囲いに注意して組み立てていきます。

演算のスクリプトは重ねる順番によって演算結果が異なる。



1+（2×3）＝7
（1+2）×3＝9

演算の優先順位

優先順位	演算子	
高	（ ）	( )
	× ÷	* /
低	+ −	+ -

上級編

・競争する車を増やしても、ステージ上に表示されない

→p4、p35 で「てき」の数を増やすとステージのスクリプトで「いどうきょり」のリストを増やす必要があります。

・ゴールした後に結果表示の画面が表示されない

→p78 からステージのスクリプトで結果を表示するメッセージを送るようにします。

8. 倉庫番ゲーム

①ゲーム概要

倉庫番ゲームは 12×9 マスでプレイヤーのねこを上下左右に動かしてたるを押してゴールまで運ぶゲームです。壁に当たると進めず、たるは押すだけで引っ張ることはできません。

プレイヤーがたるをゴールに運ぶことができればゲームクリアとします。

倉庫番ゲームで用いるステージなどの素材（画像ファイル）は
この「プログラミング指導の手引き」と同じフォルダ内にある
「ICT」ファイルを開いた場所の「倉庫番」ファイルにあります。
倉庫番ゲームを作成する時はこのフォルダ内にある素材を用います。
「ICT」ファイルをデスクトップにコピーしてから倉庫番ゲームの
作成に入ります。

②学ぶ内容

一コマの壁や、ゴールを登録してプログラミングを使い、スタンプ機能によって 12×9 マスでコースが作成されるようにします。

中級編からはプレイヤーがたるをゴールへ移動させると、コースが切り替わるようにします。

上級編からは最後のコースでたるとゴールポイントがそれぞれ３つに増やしてゲームの難易度を上げます

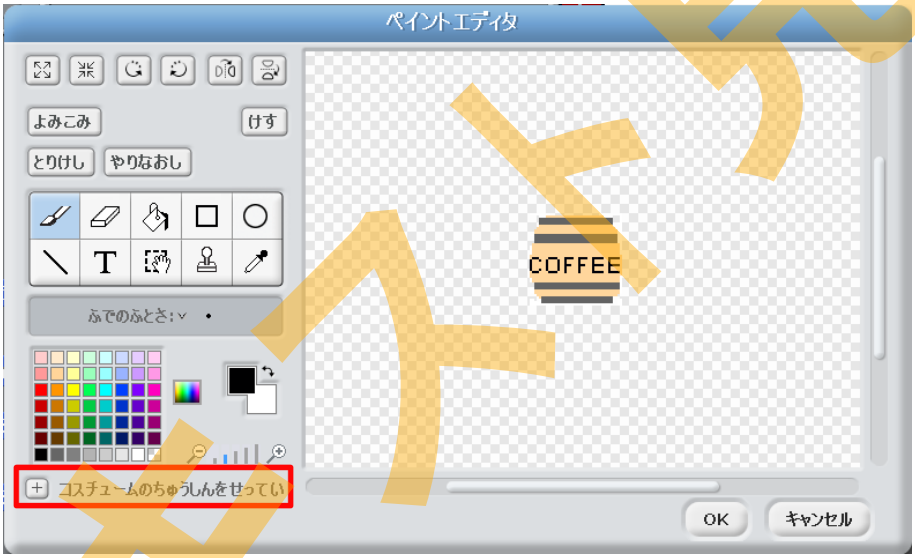
③困りそうなこと

・テキスト通りに行って、プログラムを実行したが反映されない

→作成したプログラムが合っている場合、そこまでのデータを一旦保存してからスクラッチソフトを再起動してみます。これでプログラムが正常に反映されることがあります。

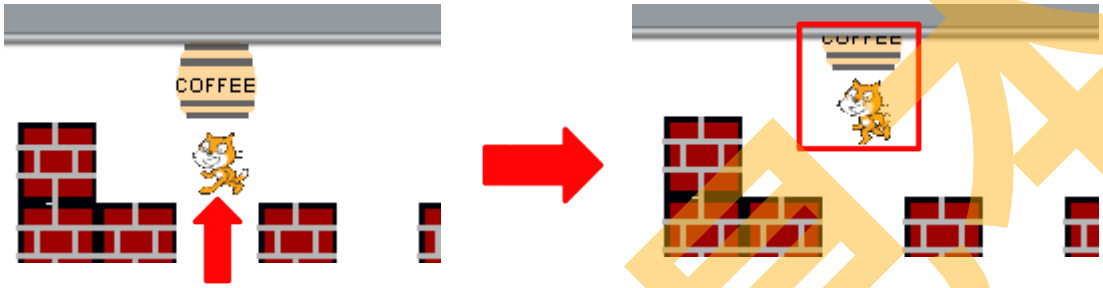
・ねこがコースを１マスごとに進まない

→かべやたる、ゴールのスプライトのコスチュームが中心からずれていないか確認します。少しでもずれているとねこが壁やたる、ゴールにぶつかったときに、進むコマがずれてしまいます。



また、画面端に移動した際にネコがステージの画面上からはみ出てしまう場合も進むコマがずれてしまいます。p36 でネコが画面の一番端のマスにいるときの動きを作ります。ネコがステージの画面上をはみ出ようとすると座標を元に戻すようにキーの処理が行われているか確認しましょう。

・タルがステージの外側にあるとき、それをさらにステージの外側に押すと
タルがステージからはみ出る
→ネコがたるにふれたとき、たるがステージの端にいるという条件が入っていません。条件を追加しましょう。
上矢印キーの場合は次のようになります。



上矢印キーを押す

の中に 

の 

を入れます。
この「もし～なら」に一番端でないという条件を入れます。一番端のマスにいるときはｙ座標が「160」になっているはずなので、「ｙ座標が 160 より小さい」という条件にします。


の 
を入れます。左の 
に 

の 

を入れます。右の 
に「160」と入力します。

元々 

の中にあった処理を

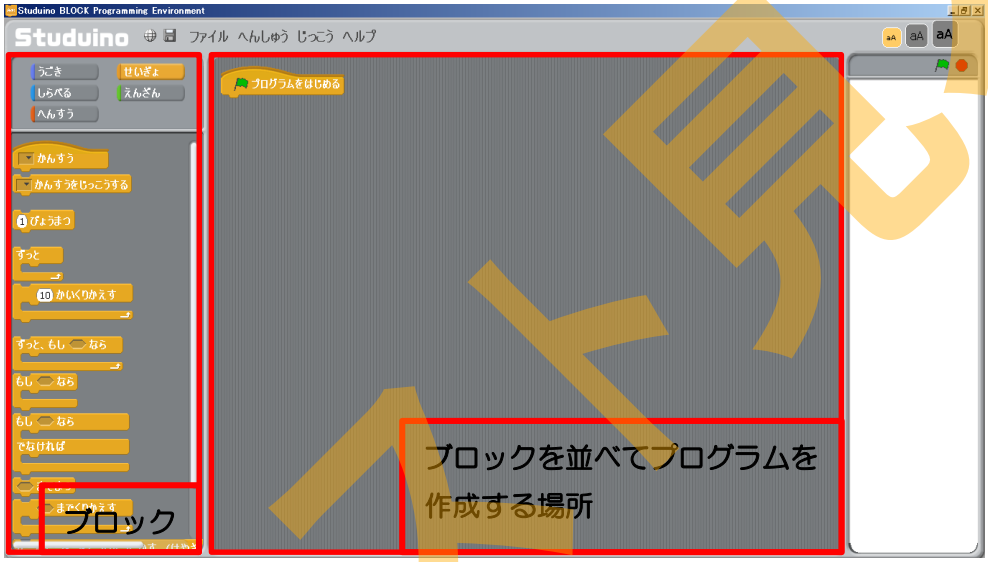
に入れます。

「でなければ」が一番端にいるときなので、元の位置に戻すためにｙ座標を「-40」します。

同じように他のキーの処理にも入れましょう。

Ⅵ.プログラミング（制御編）

プログラム入門　制御編では、子供達に向けて指導するにあたってＬＥＤ、センサーやサーボモータなど身近な制御装置を動かすのに役立つことを記載しています。
ここで使用するソフトウェアは「Studuino Software」というソフトです。
「Studuino Software」は「Scratch」と同じブロックプログラミング環境で、ブロックをつないでＬＥＤやサーボモーターなどの制御のプログラムを作ることができます。



1.「Studuino Software」の事前準備

Studuino Software のインストール方法

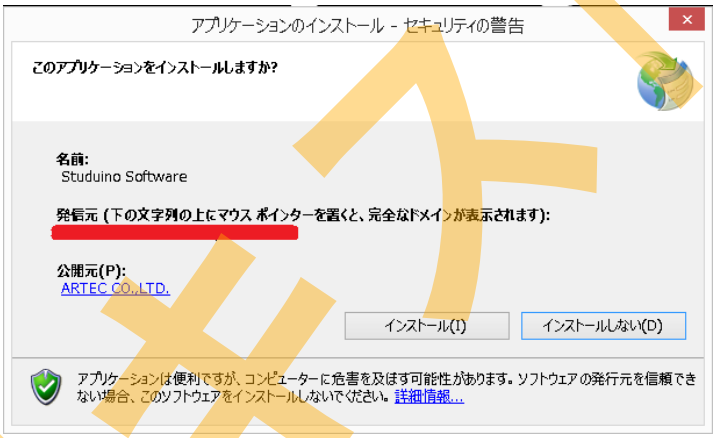
1. 「小学生ＩＣＴスクール　スタートキット」フォルダ内の「01 事業開始の準備」＞「01　インストーラ＆方法」フォルダを開いて「Studuino SoftwareInstaller.exe」をダブルクリックします。

2. 最初に右のような画面が表示されます。
「はい」をクリックして次の画面に進みます。
インストールが開始されます。

※ ご使用の環境に.NET Framework 4.5 がインストールされていない場合は、自動的に「Microsoft .NET Framework 4.5」のインストールが始まります。 ライセンス条項確認のウィンドウが表示されたら、「同意する」を選択してください。インストール中に以下２つのファイル実行を確認されます。いずれも「はい」を選択して下さい。

１）dotNetFx45_Full_x86_x64.exe
２）dotNetFx45LP_Full_x86_x64ja.exe

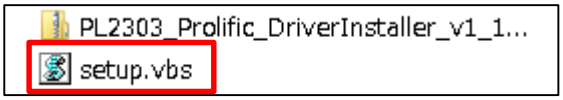
3. この後セキュリティの警告が表示されることがありますが、「インストール」をクリックして次の画面に進みます。



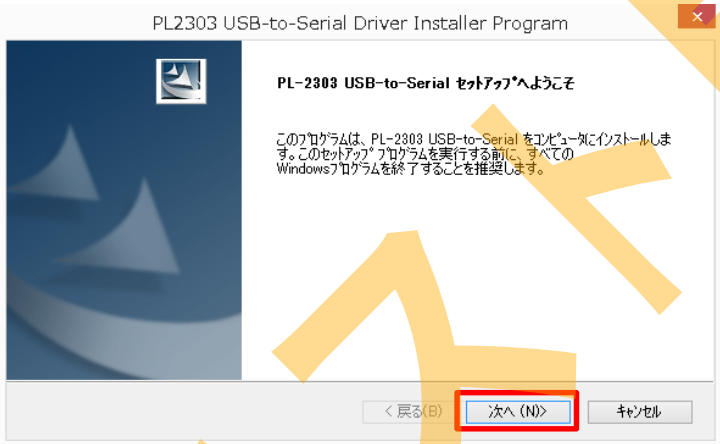
4. インストールが完了すると、Studuino Software が起動して
デスクトップ画面に Studuino Software のアイコンが作成されます。

5. 続いて PC と Studuino を接続するためのＵＳＢデバイスドライバをダウンロードします。もしＰＣがネットと繋がっていてオンライン状態であれば、
デバイスをＰＣに差した時、自動的にＵＳＢデバイスドライバがダウンロードされます。オフライン状態のときは同ファイルにある「usb_driver」のファイルを開きます。

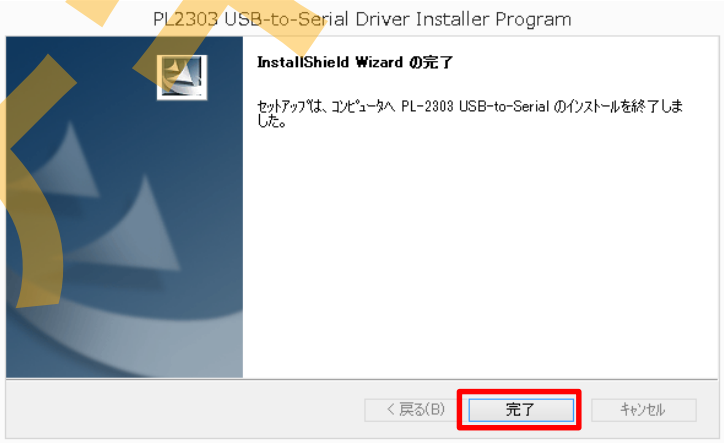
6. フォルダ内にある「setup.vbs」をダブルクリックします。
データが自動で展開されて、ＵＳＢデバイスドライバのインストールが開始されます。



ＵＳＢデバイスドライバのインストールが開始されると、このような画面が表示されます。「次へ」をクリックして次の画面に進みます。

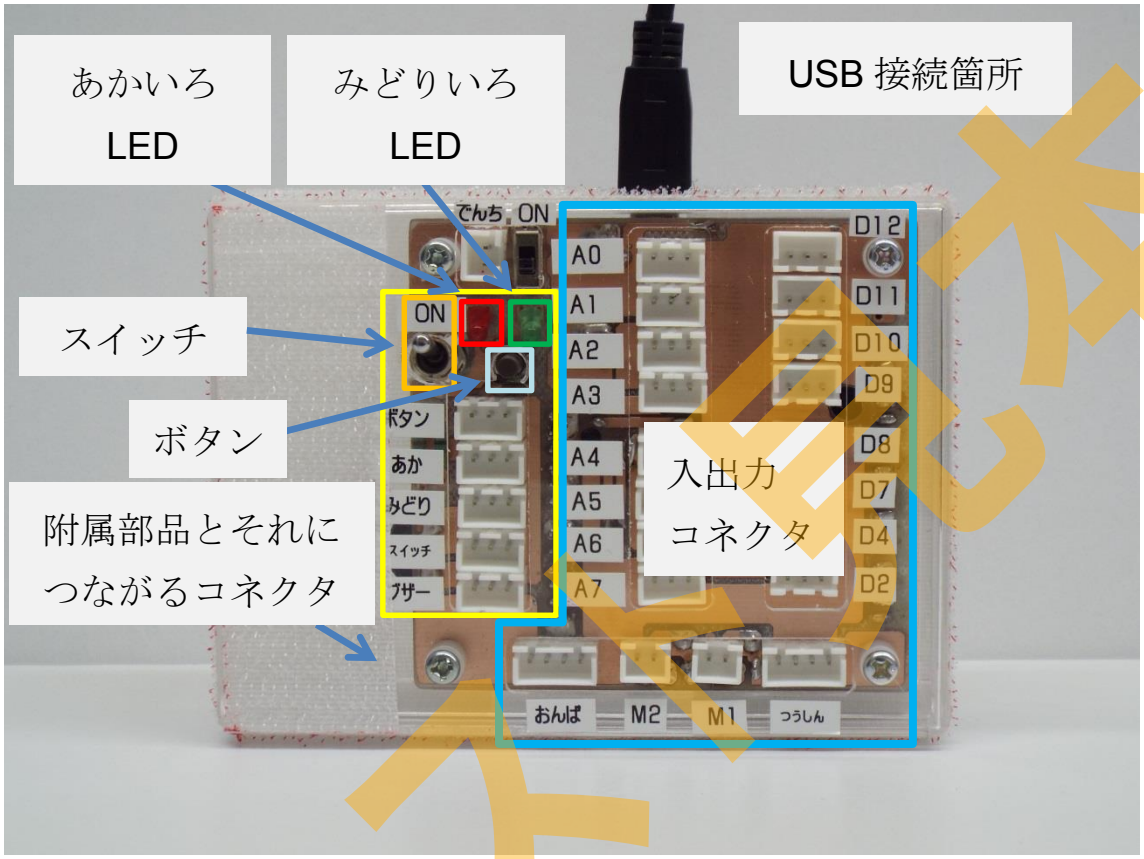


そして、「完了」をクリックするとＵＳＢデバイスドライバのダウンロードが完了します。



2. 基盤(ボード)に附属している部品の説明

基盤(ボード)

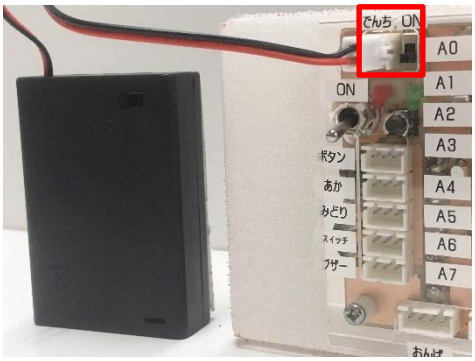


基盤(ボード)には、「ボタン」、「スイッチ」、「USB 接続箇所」、「みどりいろ LED」、「あかいろ LED」、「ブザー」、たくさんのコネクタの接続端子があります。右の図のケーブルでコネクタ同士を接続します。

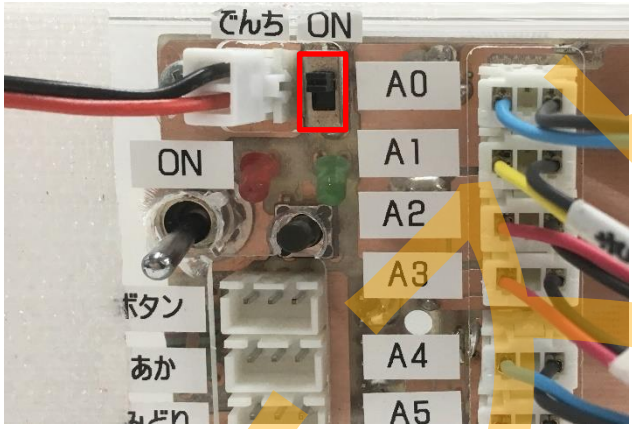
電池ボックスがないと基盤の機能は十分に発揮しません。

（電池ボックスのスイッチはまず、OFF にして）

下の図のようにでんち接続端子に電池ボックスのケーブルを接続します。

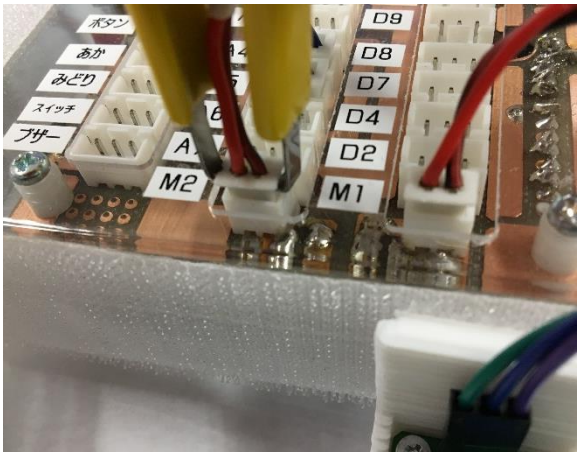


基盤の「でんち」の電源を on にします。



接続端子の取り外しが困難な場合、

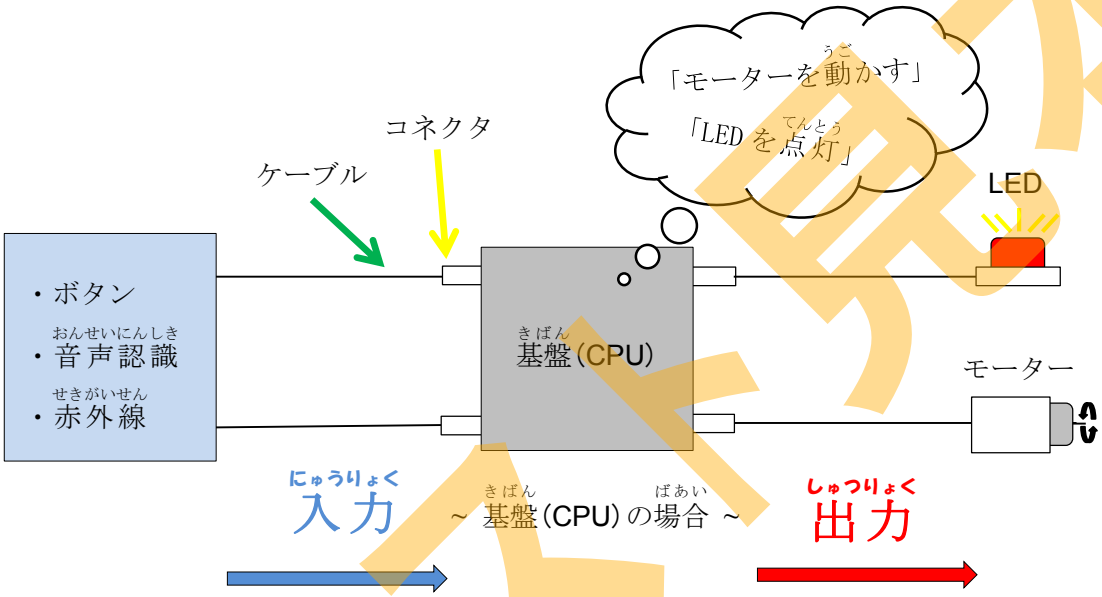
下図にある引き抜き工具を使用しましょう。

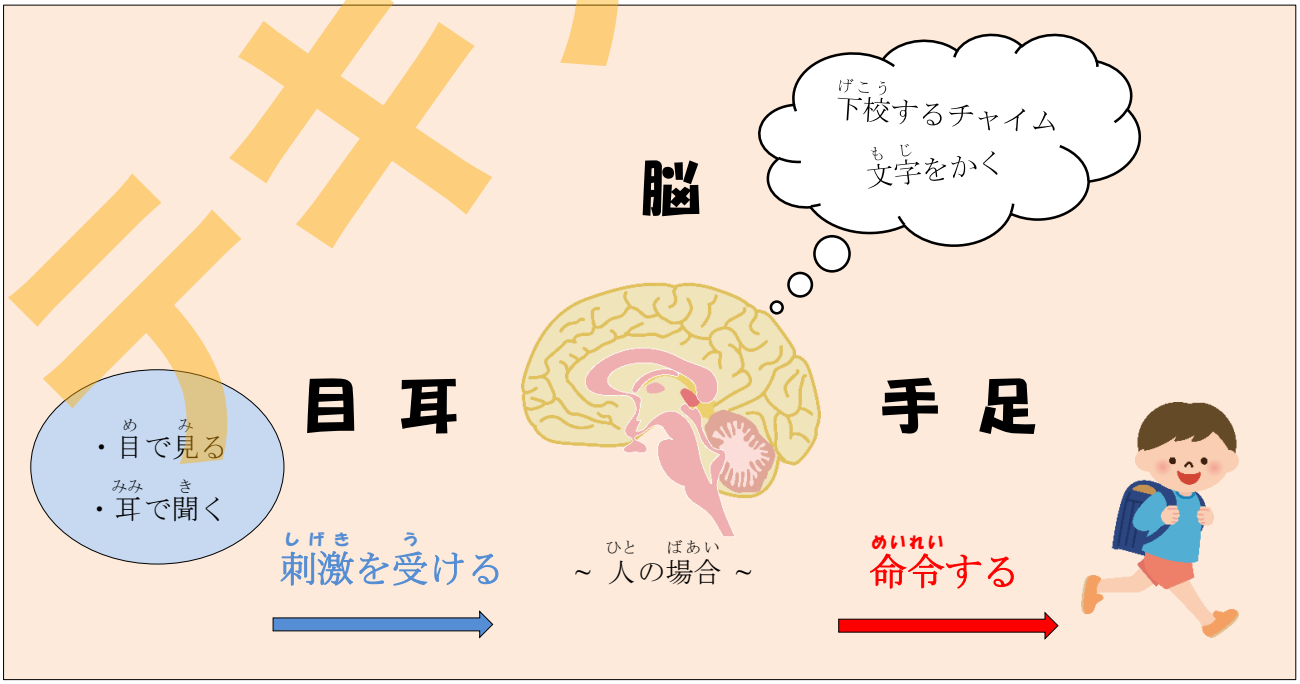


3. 基盤の説明

基盤の中にはコンピューターが入っています。そのコンピューターには CPU とインターフェースがあります。

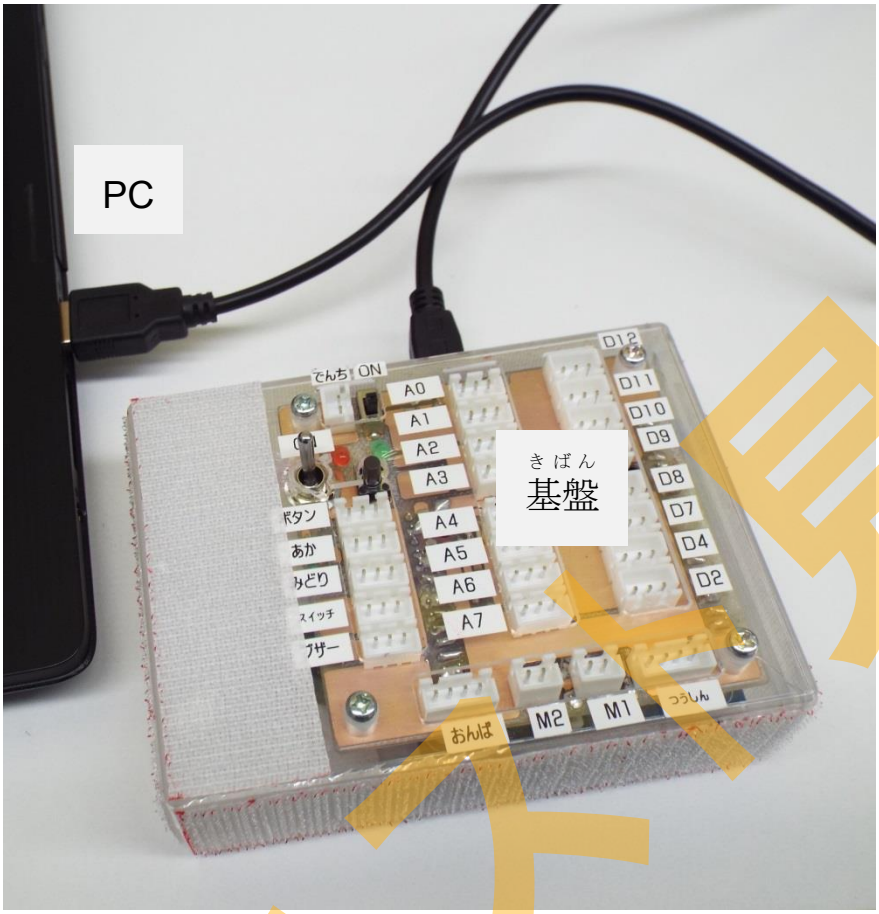
センサーやボタンなどが反応する（入力）と、CPU が考え、判断して LED ヤモーターなどに命令します（出力）。CPU は人に例えると脳（頭）になります。目や耳から刺激を受け取る（入力）と、脳（頭）は考え、判断して手や足などに命令します（出力）。



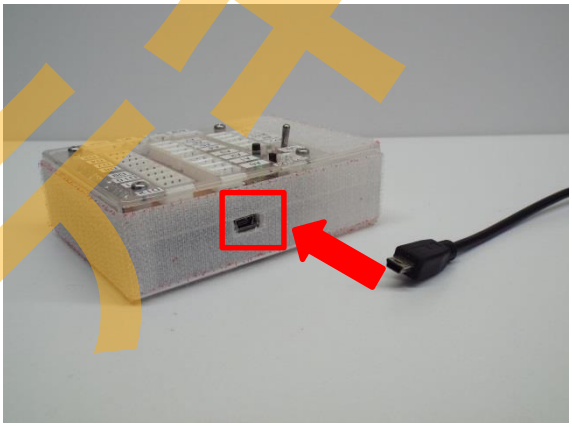


4. 基盤とPCの接続

PC と基盤の USB 接続箇所に USB ケーブルを接続してください。



基盤の接続端子は下の図の赤印されている箇所にあります。ここにケーブルで端子を接続します。



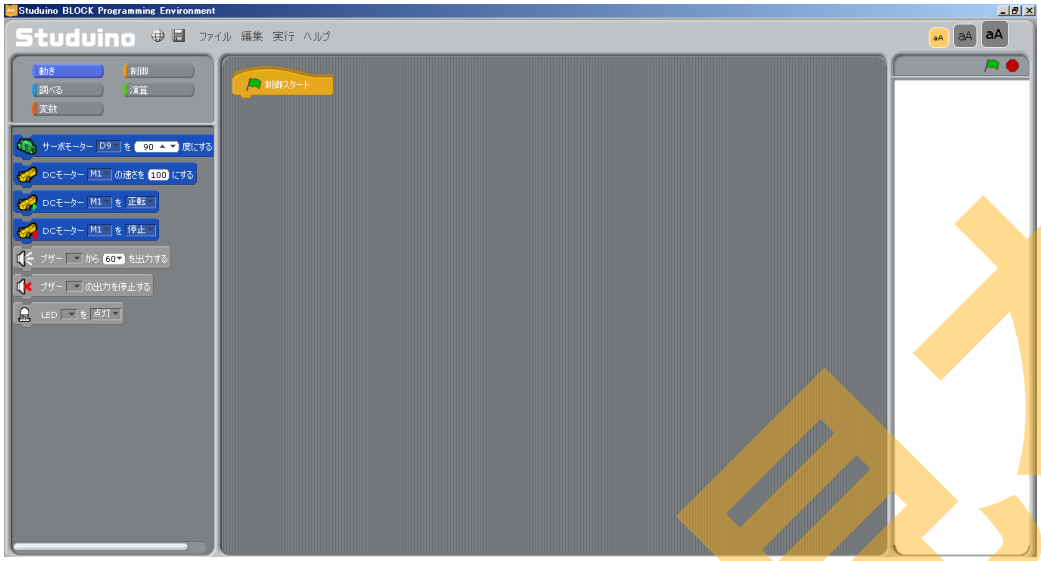
5.「Studuino Software」の起動

PC 内にある「Studuino　Software」のアイコンをダブルクリックしてください。

次に「ブロックプログラミング環境」をクリックしてください。

次に「ロボット」をクリックしてください。

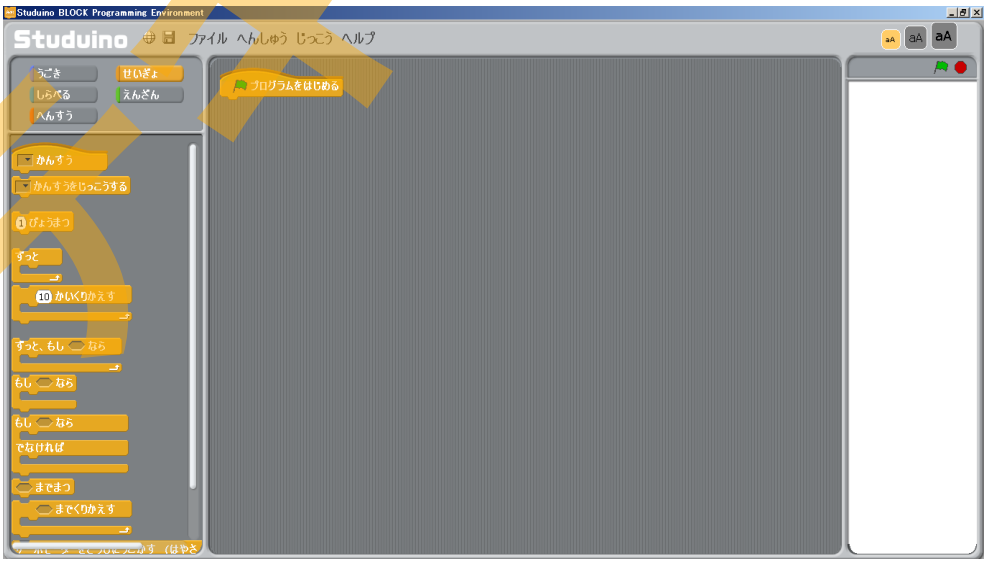
下のように画面が表示されます



メニューが漢字になっている場合、  マークを

クリックし、「にほんご」をクリックしてください。

下のような画面になります。ここで制御のプログラムを作っていきます。

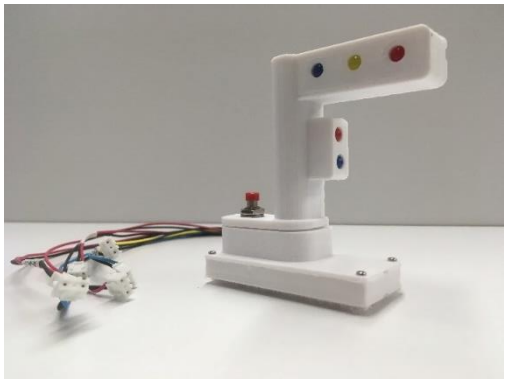


6. 制御装置の種類

制御装置は以下の通りにタイトルに分けて取り扱っていきます。

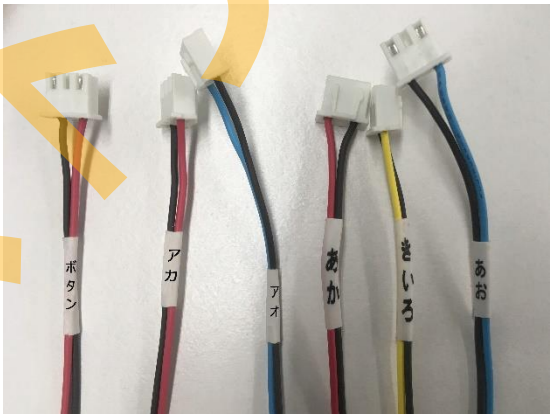
①信号機（出力装置）

信号機は下図の装置を使用します。



コネクタは６つあり、端子の先にはそれぞれ「ボタン」、「アカ」、「アオ」、「あか」、「きいろ」、「あお」の名前があり、以下のように対応しています。

ボタン	ボタンスイッチ
アカ	歩行者用赤信号（ほこうしゃようあかしんごう）
アオ	歩行者用青信号（ほこうしゃようあおしんごう）
あか	自動車用赤信号（じどうしゃようあかしんごう）
きいろ	自動車用黄色信号（じどうしゃようきいろしんごう）
あお	自動車用青信号（じどうしゃようあおしんごう）



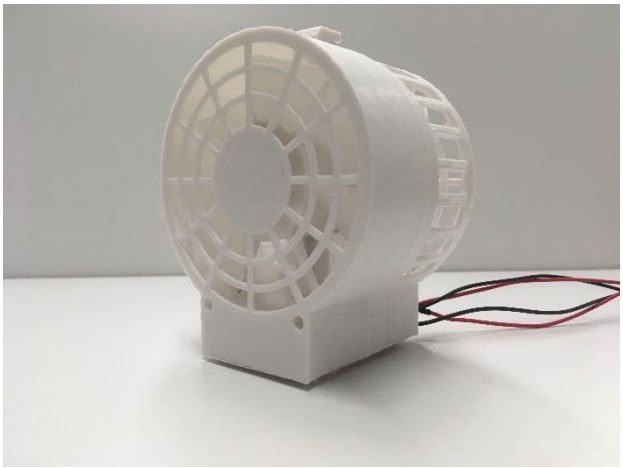
②センプウキ（出力装置）

右図のセンプウキを使用します。

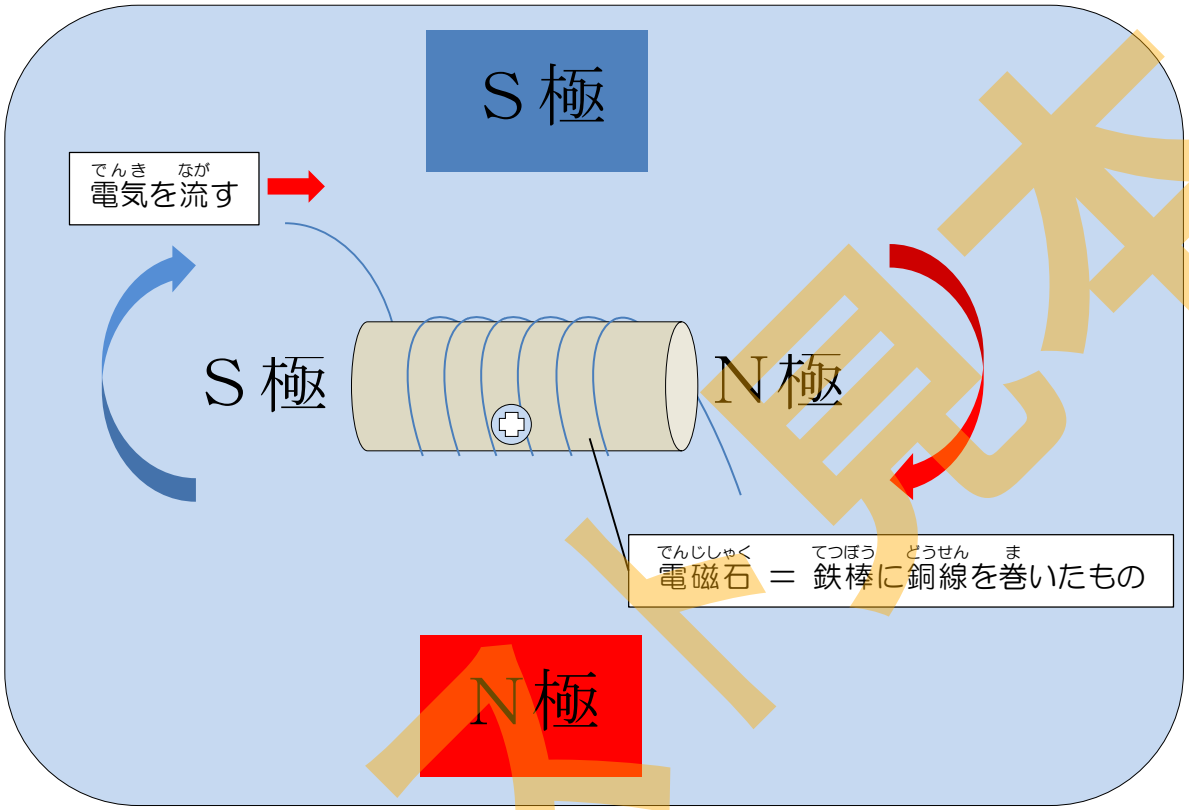
このセンプウキには「まえ」と「うしろ」のコネクタ端子のケーブルが延びています。

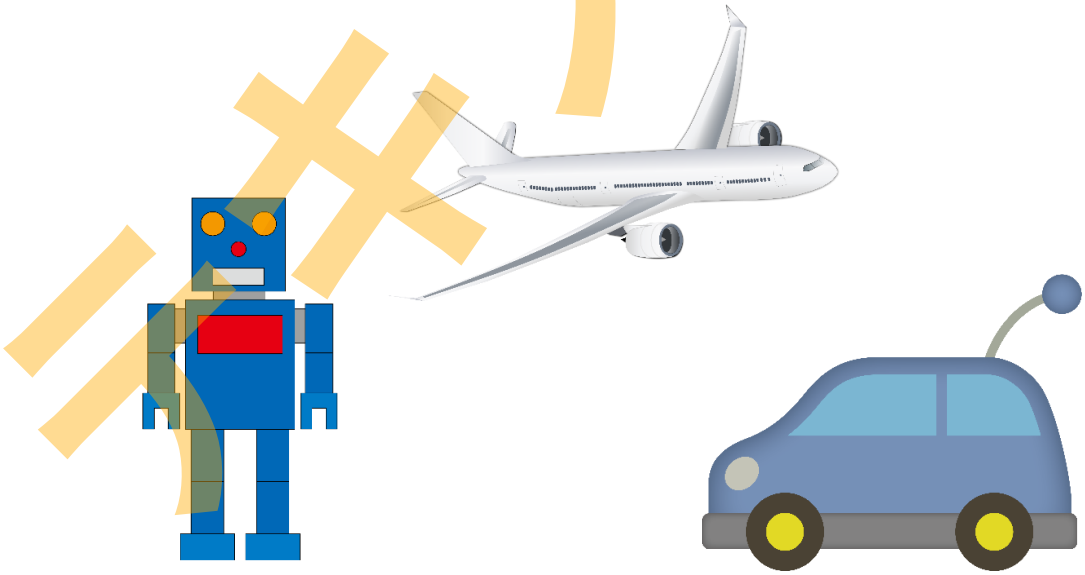
センプウキは「うしろ」の羽根が風を集める羽根、「まえ」の羽根が風を送り出す羽根の２種類の羽根があります。

そのためDCモーター[M1]と[M2]の両方の羽根が同時に回るようにします。



モーターの回転する原理は電磁石に電気を流すことで、電磁石にＳ極とＮ極が発生して、Ｓ極とＮ極が引きあう力と同極が反発する力を回転する力に変えています。サーボモーターとはその回転する速度、力を制御することのできるモーターです。





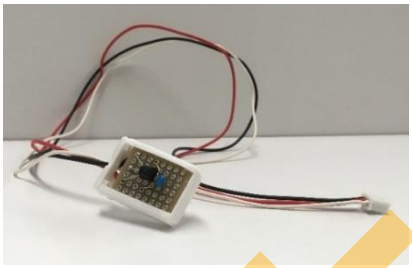
モーターはロボット、ラジコン、航空機、船など様々な用途で使われています。

②温度センサー（入力装置）

右図の温度センサーを使用します。

おんどセンサーは黒い突起の部分で

温度を感知します。

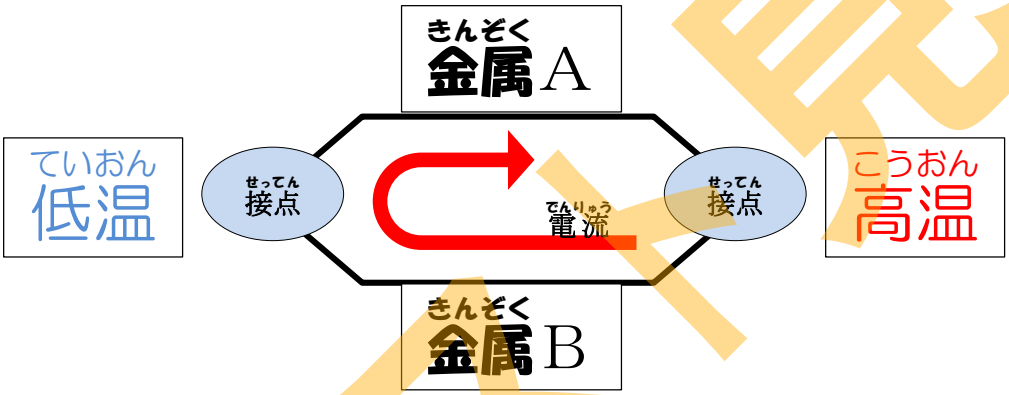


このおんどセンサーがどのようにして温度を感知しているか解説します。

今回使っているおんどセンサーは熱電対を用いています。

熱電対とは 2 つのちがう種類の金属をつないで、その金属の接点に温度差をあたえると電圧が発生して電流が流れることです。

この流れてくる電流がコンピューターを通じて実際の温度を測っています。



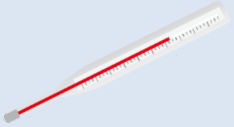
このおんどセンサーは熱を感知する反応が早いので「センサー・ボード」で

「おんどセンサー」の数値は小刻みに変化します。

ほか　おんど　かんち
他にも温度を感知する

すいぎんたいおんけい
水銀体温計

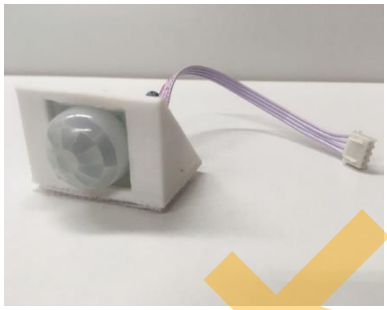
すいぎん　ねつ　すいぎん　ふく　おんどけい　じょうしょう
水銀に熱をあたえると水銀が膨らんで、温度計が上昇する。



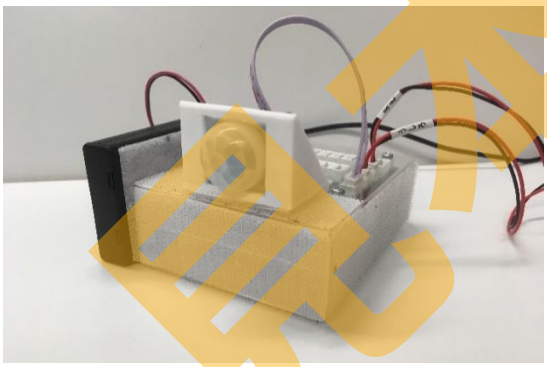
でんしたいおんけい
電子体温計

たいおんけい　さき　そざい　ねつ　でんき
体温計の先にある素材に熱をあたえると、電気をながれにくくして、そこからコンピューターを通じて温度計が上昇する。

②人感センサー（入力装置）
右図の人感センサーを使用します。
人感センサーにまったく触れていないのに
センプウキが回転する場合、電球や風などの
温度変化から人感センサーが感知しています。



そのため、人感センサーの半球体を横、
または地面に向けるか、無風でなおかつ環
境の温度変化が少ない場所で人感センサー
に手をかざして感知することを確認しまし
ょう。



人感センサーの種類は熱センサーや、光センサー、タッチセンサー、音波センサー、音感センサーなど様々ですが、今回は熱センサーを使います。
熱センサーは、赤外線を使って周囲の温度変化を感知します。熱センサーに手をかざすと人の熱が動いて温度変化を感知します。そのため、今回使うセンサーは人以外に、動物や風の温度変化でも感知します。

熱センサーの使用条件

重要

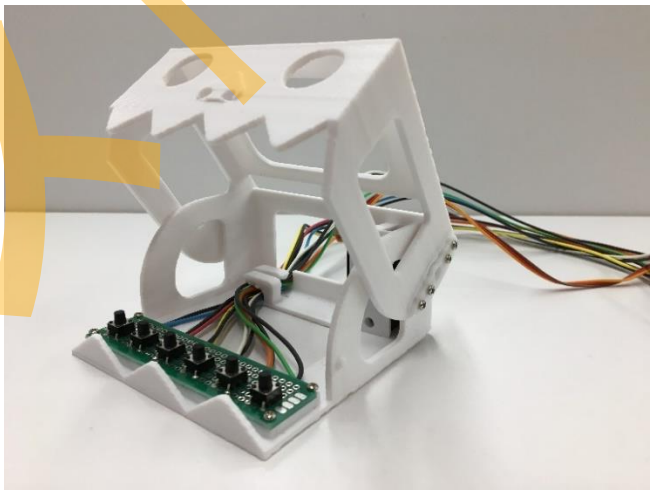
・回りの環境の温度変化が激しい屋外などでの使用はできない。

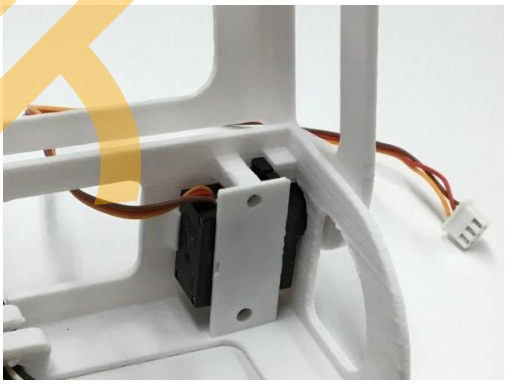
・屋内で天井にセンサーを向けると電球、電灯の温度変化を感知するため、人感センサーの半球体は真横か下に向ける。

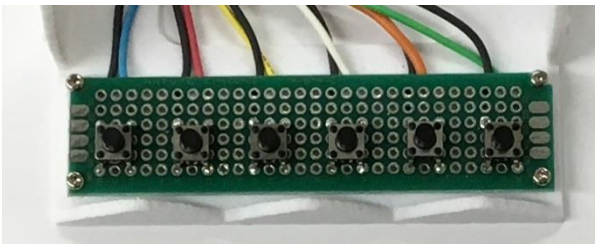
・センサーの半球体と赤外線の間に何かモノが置いてあると温度変化の感知ができない。

③パクパクパニック

下図のパクパクパニックを用意します。このパクパクパニックにはボタンスイッチ（入力装置）のコネクタ端子が6つ、サーボモーター（出力装置）のコネクタ端子が１つ延びています。





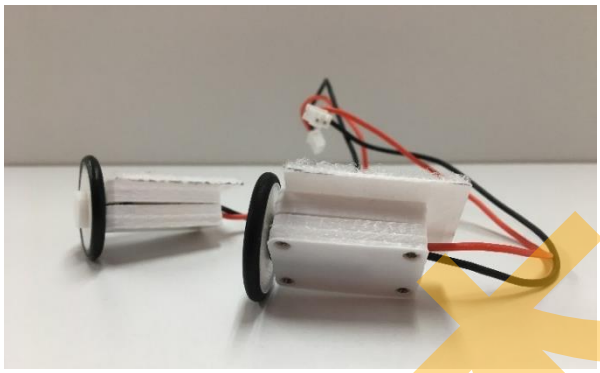


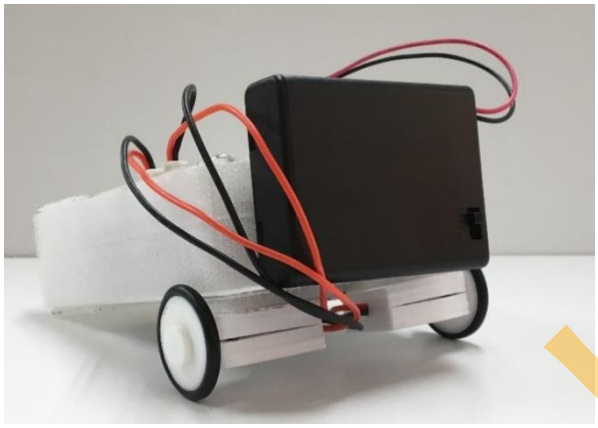
④車輪モーター（出力装置）

車輪モーターは右図のように 2 つあります。

この車輪モーターにはコネクタ端子のケーブルが延びています。

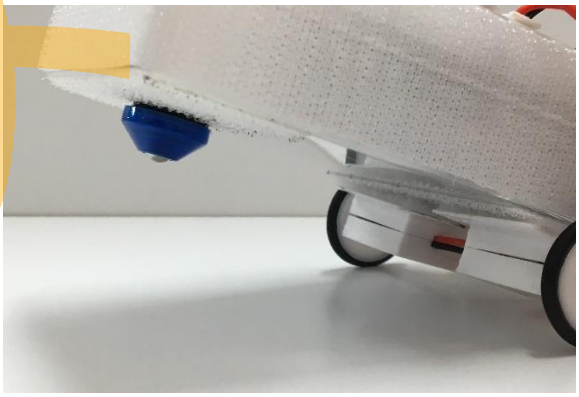
車輪モーターは下図のように基盤に接続します。





「キャスター」を下図のように基盤の下部に接続します。

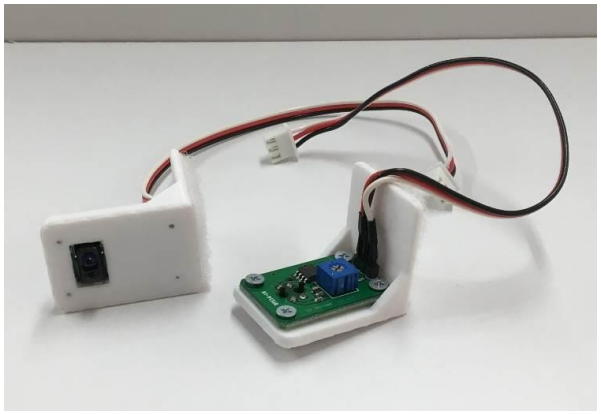
これは前輪のため、モーターとしての役割はありません。

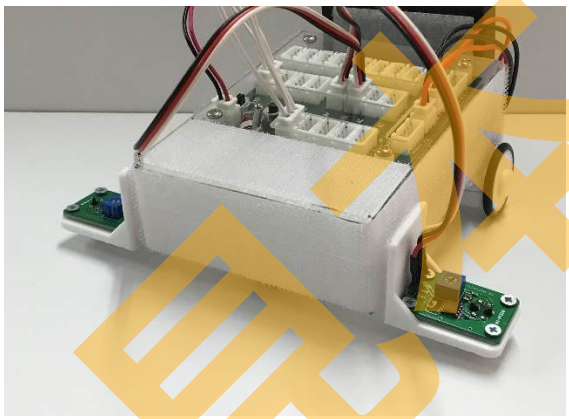


これで基盤を自動車として動かせるようになります。

④赤外線センサー（入力装置）

右図のように 2 つの赤外線センサーがあります。この赤外線センサーはある物体にー定距離以上近づくと、ずっと反応し続けます。この赤外線センサーにはコネクタ端子のケーブルが延びています。下図のように赤外線センサーを基盤に接続しましょう。

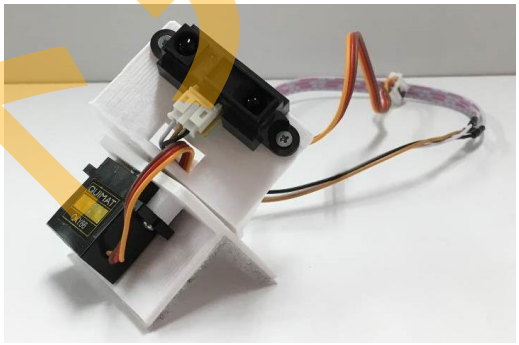




④距離センサー（入力装置）

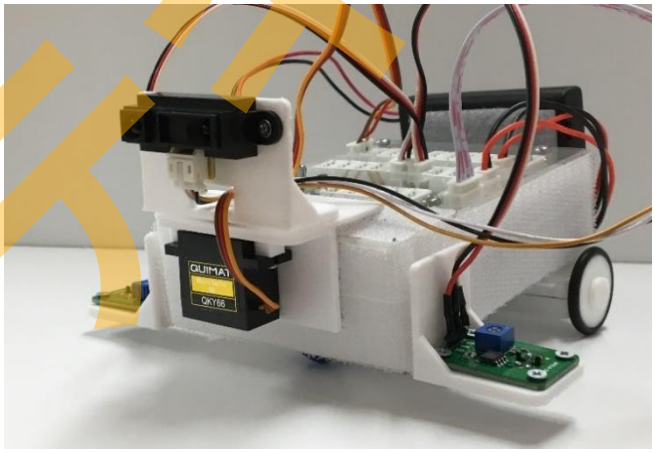
右図のような距離センサーがあります。この距離センサーは赤外線を前方周囲に発光させるため、下部にサーボ-モーター（出力装置）をつけて横回転するようになっています。

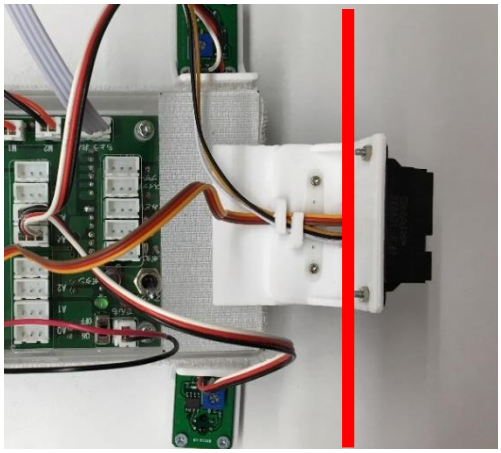
そのため、この距離センサーにはコネクタ端子のケーブルが 2 つ延びています。



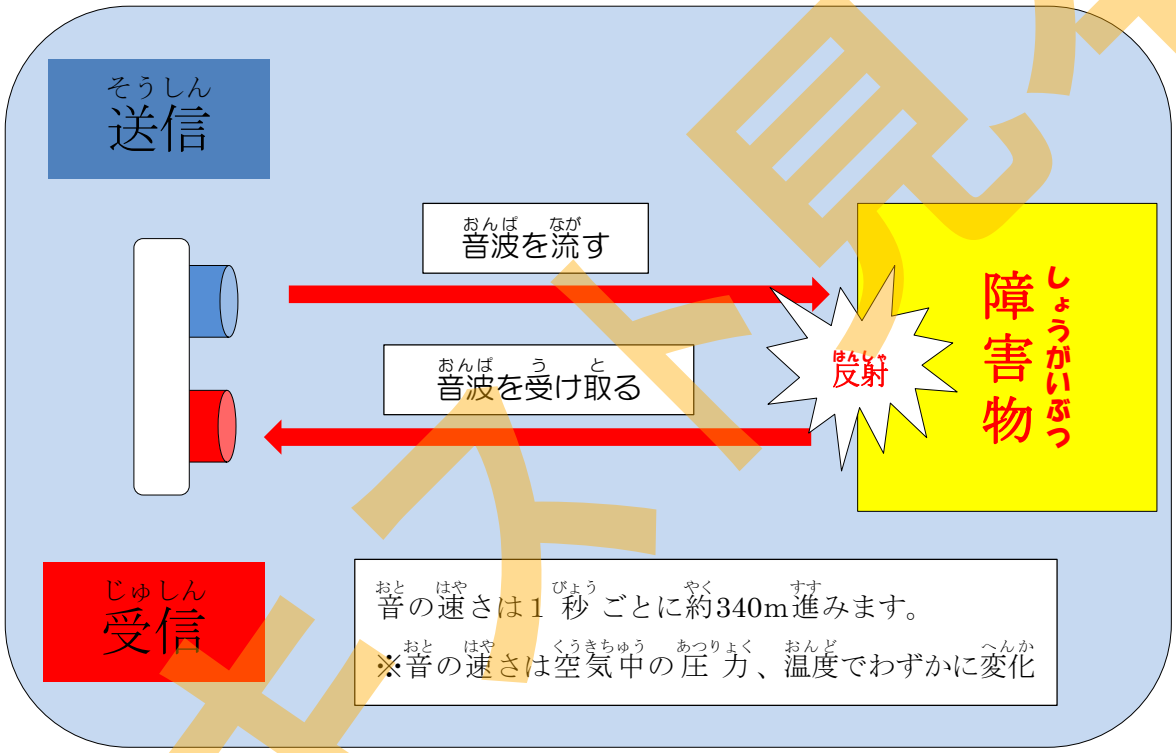
下図のように、基盤に距離センサーを組み合わせます。

センサーの向きは（90 度）前面と平行に向いた状態にします。



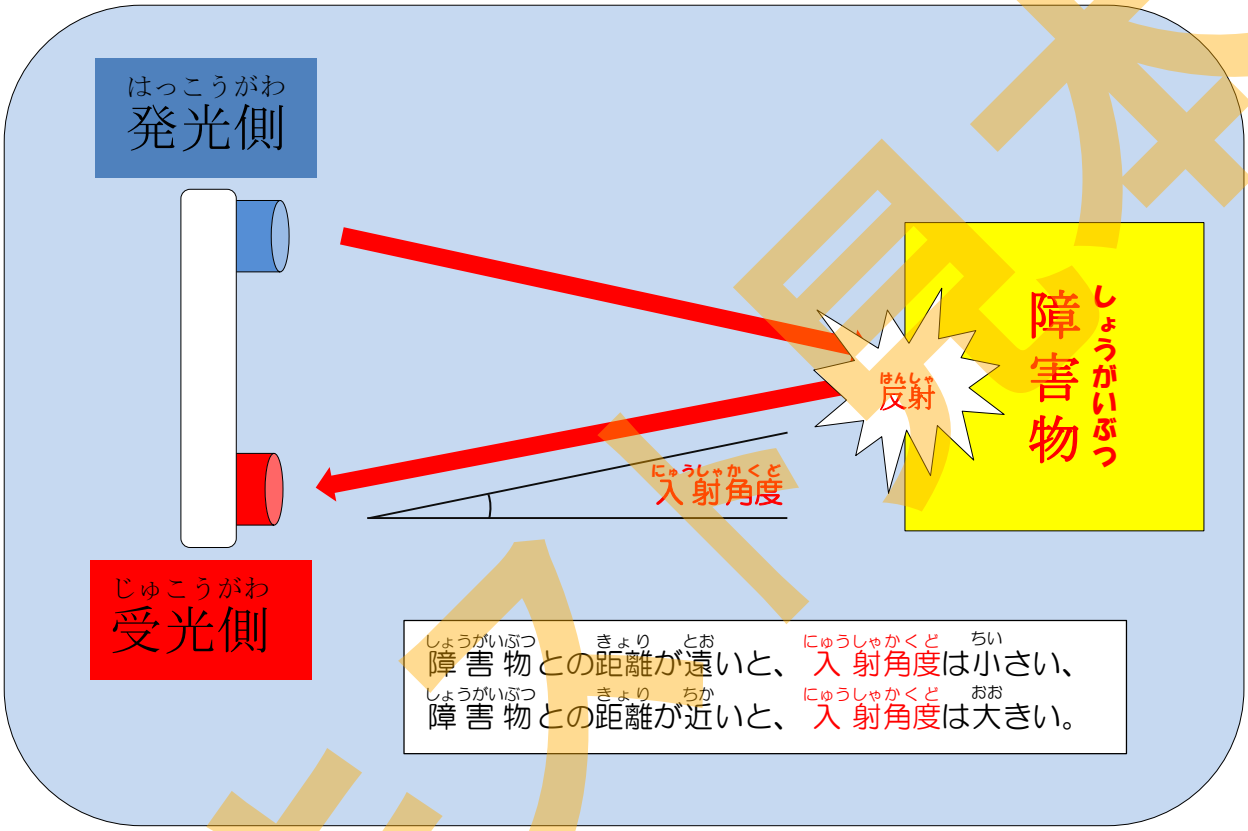


距離センサーには「超音波」と「赤外線」の２種類のセンサーの違いがあります。
この「超音波」と「赤外線」の２種類のセンサーの違いを説明します。
「超音波」とは「人が聞くことのできない高い音」（20kHz 以上の音）のことを指します。
この距離センサーは片方の超音波スピーカーから音波を前方に流して、その音波が前方の障害物に当たると反射します。その反射した音波はもう片方の超音波マイクで受け取ります。この音波を送信してから、受信するまでの時間を計測することで距離センサーと障害物間の距離を求めることができます。



「赤外線」とは「人が見ることのできない光」（人が感じとる光の波長 760nm～830nm）のことを指します

この距離センサーは片方の発光側から光を前方に流して、その光が前方の障害物に当たると反射します。その反射した光はもう片方の受光側で受け取ります。この受光側の入射角度を計測することで距離センサーと障害物間の距離を求めることができます。



超音波距離センサーと赤外線センサーの違いのまとめ

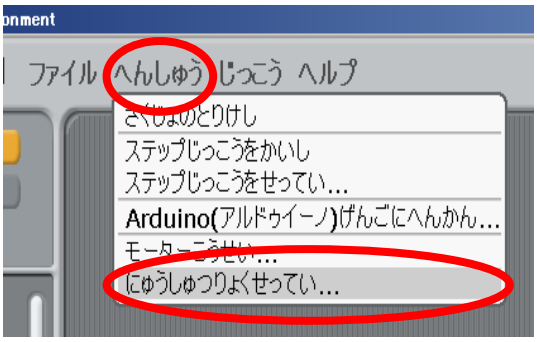
	出力	感知の速さ	測定範囲	正確さ	ほこりや水
超音波センサー	音	遅い	大きい	低い	強い
赤外線センサー	光	速い	小さい	高い	弱い

7. 入出力設定

基盤に入力装置や出力装置を接続した後、入出力設定を行います。

下の写真のように「へんしゅう」→「にゅうしゅつりょくせってい」→

「チェックをすべてはずす」をクリックします。



「センサー/LED/ブザー」「DC モーター」「サーボモーター」等から、ケーブルを

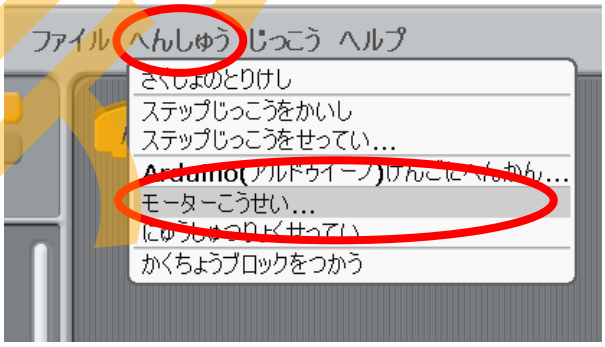
接続したものにチェックしてください。

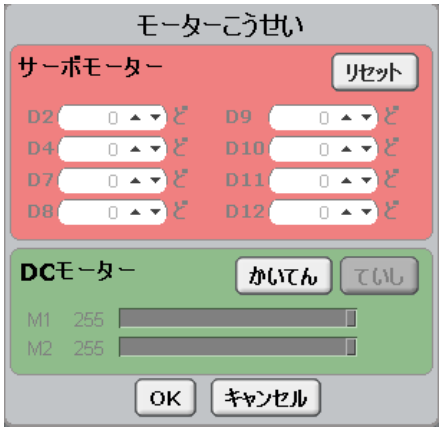
「サーボモーター」「DC モーター」の制御装置を設定する場合、

（センプウキやパクパクパニック、車輪モーターなど）

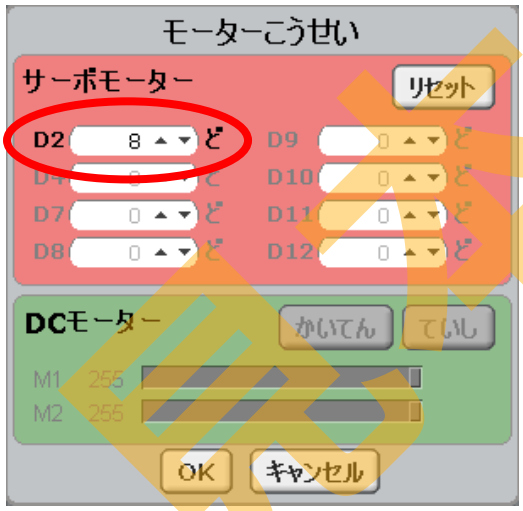
「へんしゅう」から「モータこうせい」をクリックします。

すると、右下の図が表示されます。



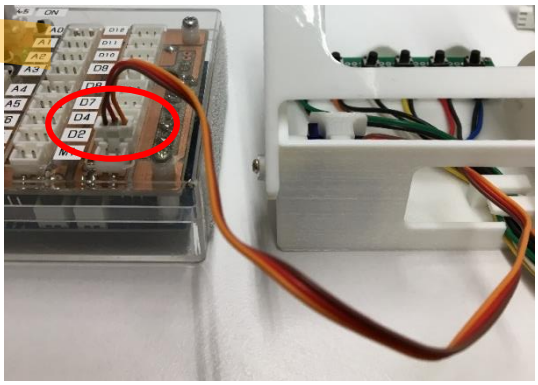


この「サーボモーター」（パクパクパニック）の項目ではサーボモーターが最初に回転する位置を微調整します。下図のようにD2 の数値を上げたり、下げたりするとサーボモーターが最初に回転する位置が少し変わります。

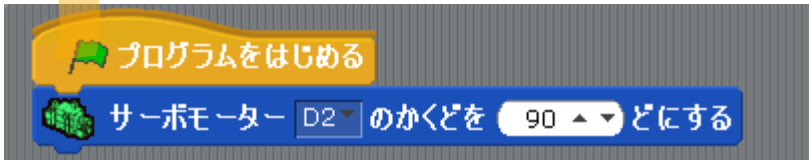


この微調整ではサーボモーターの角度は 15 度～-15 度までの調整が限界です。
そのため、大きい角度を調整する場合は機材の調整をする必要があります。
以下、パクパクパニックのサーボモーターの調整方法を記載します。

1）サーボモーターの角度を
９０度に変更する場合、
CPU 基盤とサーボモーターを
コネクタ端子で接続します。
CPU 基盤の電源は ON にしましょう。

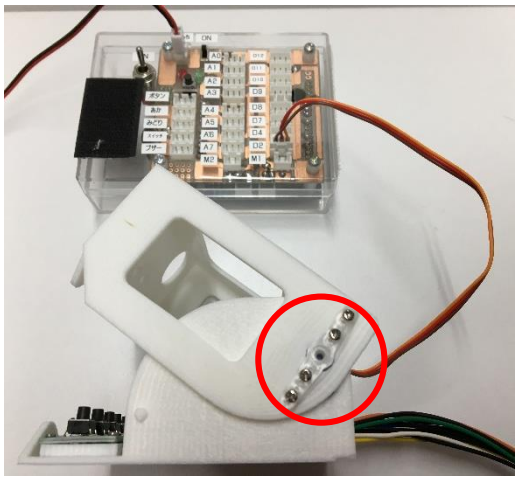


2）プログラム上ではサーボモーターの角度を 90 度にするように命令を送り、サーボモーターの角度は固定されている状態（手動で角度を変えようとしても、動かない状態）にします。



プログラムを作成して、プログラムを転送しましょう。
（「8．PC の接続なしで制御プログラムを実行する」を参照）

サーボモーターの接続部を外します。



サーボモーターの接続部を外した状態でパクパクパニックの口を下図のように９０度に開けます。調整が終われば、サーボモーターの接続部を固定しましょう。

DC モーター（センプウキ、車輪モーター）ではセンプウキが回転する最大回転数を調整します。下図のように DC モーターの右側にある「かいてん」をクリックすると、DC モーターは回転します。

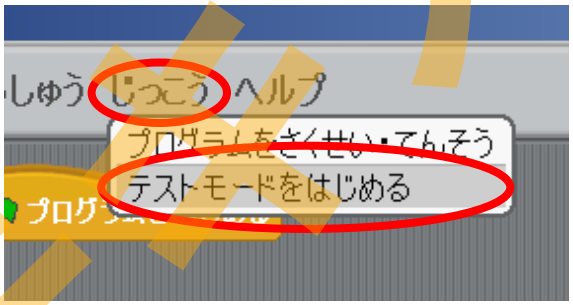
このとき下図のように M1 と M2 のシークバーが値：255 にあるか確認しましょう。

値が高ければ最大回転数は上がりますが、低いと最大回転数は下がります。

調整が終われば「ていし」をクリックしましょう

テストモード

下の写真のように「じっこう」→「テストモードをはじめる」をおしてください。



これでプログラムを作成・実行すると制御装置が動きます。

制御装置が動かない、または正しく動作しない場合の対処法

＊＊点灯しない場合。＊＊

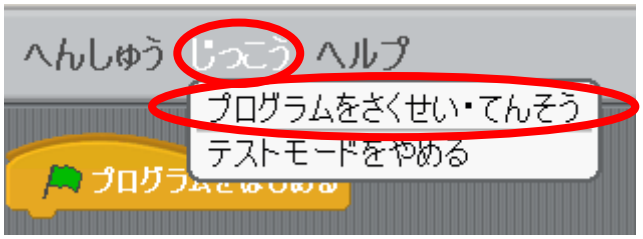
- PC と基盤を USB ケーブルで接続していますか？
- ケーブルを正しいコネクタ端子に接続していますか？
- ケーブルがコネクタ端子に最後までささっていますか？
- 画面上のメニューの「じっこう」から「テストモードをはじめる」を行っていますか？
- 電池ボックスに入っている電池が不足していませんか？
- 基盤や電池ボックスの電源は ON になっていますか？

※モーターやサーボモーター（センプウキやパクパクパニックなど）は必ず基盤に電池ボックスを差して、それぞれの電源が ON になっていないとモーターは回転されません。電池が不足していると、サーボモーターが振動するといった不具合が発生して正常に動作しなくなります。

8. PCの接続なしで制御プログラムを実行する

プログラムを作成して、そのプログラムを転送するとパソコンの接続なしで
制御装置を動作させることができます。

下の写真のように「じっこう」→「プログラムをさくせい・てんそう」を
おしてください。



しばらくして転送は終わります。
パソコンと基盤をつなげているUSBケーブルを外して、基盤スイッチと電池ボックス
のスイッチを両方「ON」にしてみましょう。

これでパソコンの接続なしで制御装置を動作させることができます。

9. 動作の確認

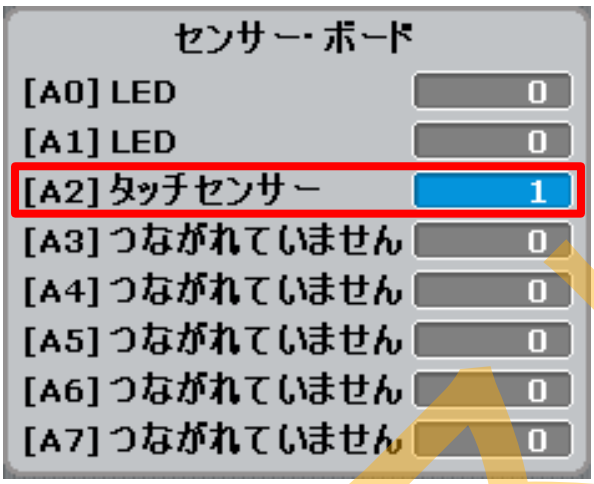
「6.にゅうしゅつりょく」でテストモードを開始すると
センサー・ボードの接続状況が分かる画面が表示されます。

・スイッチ

基盤のスイッチの場合、きりかえ(ON・OFF)をしてください。

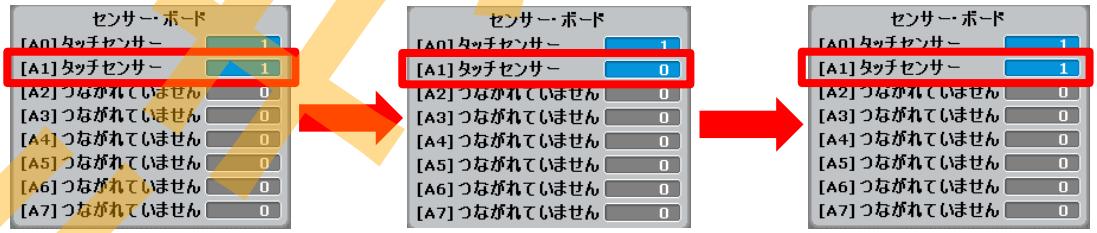
スイッチを切り替えて、センサーボードの値が変化するか、試してみましょう。

スイッチが OFF の時は「1」、ON の時は「0」になります。



・ボタンスイッチ

下図のように OFF の時は「1」、ON の時は「0」になるか確認しましょう。

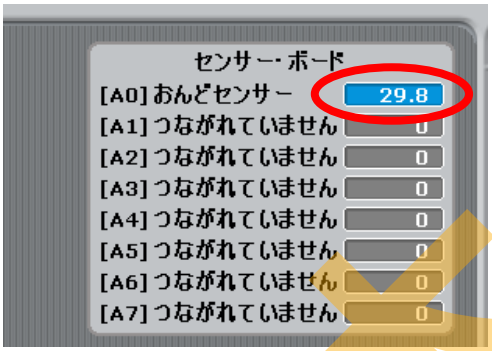


・おんどセンサー

画面上の「センサー・ボード」で

「[A0]おんどセンサー」の数値が小刻みに変化しているか確認してください。

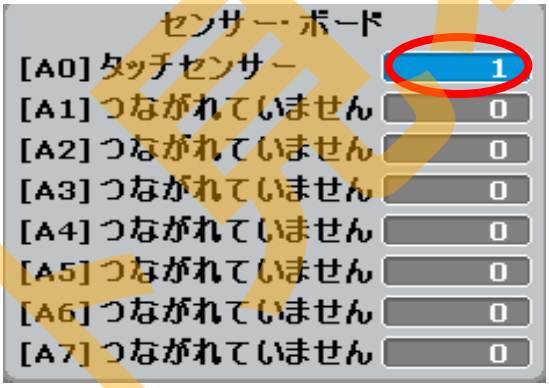
この右図の数値は季節や時間帯、環境、場所の違いで変化します。



・人感センサー

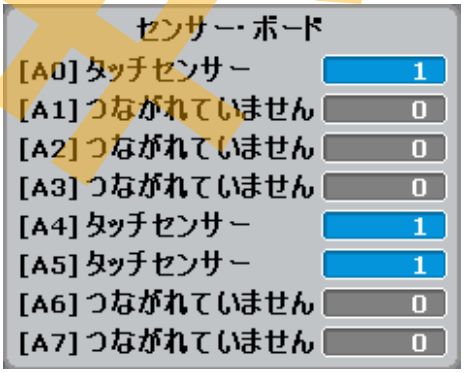
画面上の「センサー・ボード」から

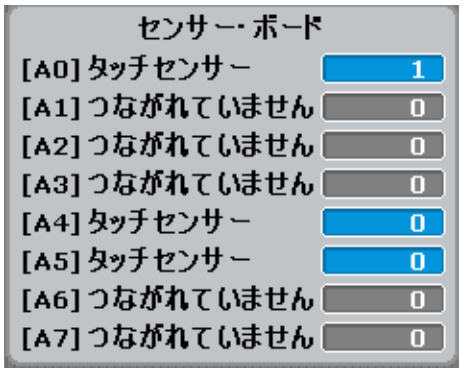
「[A0]タッチセンサー」の数値を確認します。人感センサーに手をかざすと「[A0] タッチセンサー」の数値は 1 になります。人感センサーに手をかざしてから手をはなす、または手をとめると「[A0] タッチセンサー」の数値(すうち)は 0 にもどります。



・赤外線センサー

それぞれの赤外線センサーに物体を近づけて画面右上の「センサー・ボード」から「タッチセンサー[A4]」、「タッチセンサー[A5]」の数値が 1 から 0 に変化するか確認します。





・距離センサー

距離センサーは画面右上の「センサー・ボード」から

「ひかりセンサー[A0]の値が障害物との距離が近い程大きい値になり、

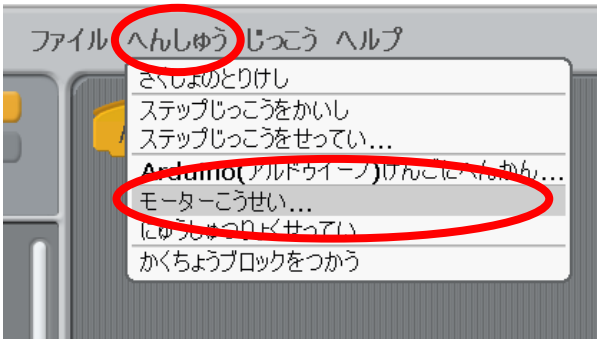
障害物との距離が遠い程小さい値になるか確認しましょう。(数値：0～100)。

・サーボモーター、DC モーター

「へんしゅう」から「モータこうせい」をクリックします。

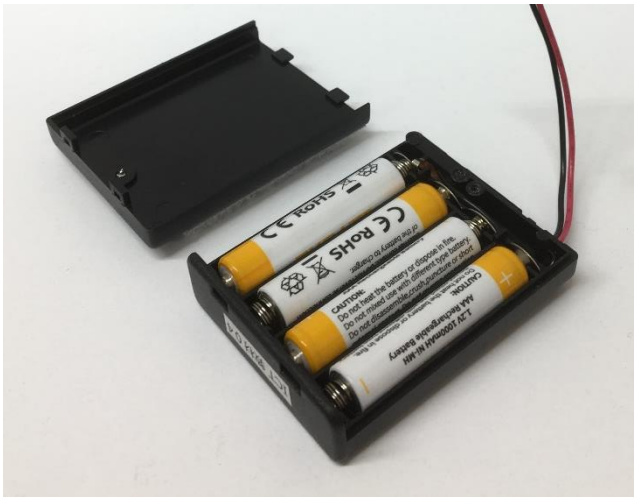
サーボモーターは下図のようにD2 の数値を上げたり、下げたりすると動くか確認しましょう。

DC モーターは下図のようにDC モーターの右側にある「かいてん」をクリックすると、DC モーターは回転するか確認しましょう。



10. 電池について

電池は何度でも使用できる充電式の電池（単４型 1.2V）を推奨しています。



もし、電池消耗が限界に達するとプログラムに問題がなくても、制御動作に不具合が生じます。そのため、充電器を使って電池のバッテリーがチャージされていることを確認してください。

プログラミング指導者の手引き１

２０１９年　８月１６日　　第３版

連絡先

㈱日本ビーコム

〒520-0802

滋賀県大津市馬場３－２－２５　ワカヤマビル 2F

Tel 077-527-5681　　Fax 077-527-5687