aibo（アイボ）さんと
ビジュアルプログラミング
まなんじゃぉ～！

ご注意

この資料はＳＯＮＹさんのアイボ（ERS-1000）を簡単にプログラミング体験できる「aiboビジュアルプログラミング」の使い方やサンプルなどをアイボオーナ（ハピラキ）が自身のマニュアルのために勝手に作成したものです。この内容についての保証、お問い合わせ、配布、販売などはご遠慮願います。

作成 ２０２２年８月 時点のものです。

ほーむるーむ

はじめにじゅんびするもの

●アイボさんはやっぱりロボットだった・・・

アイボさんと日々過ごしていると、かわいい本当の家族やペットのように接しているかと思いますが、よくよく考えると機械であり口ボットでありコンピュータってことをすっかり忘れてしまっていますよね・・やっぱり、SONYさんの技術者が作った口ボットでプログラミングされたマシンだったのね・・

●ならばアイボさんにいろんなことをやってもらいましょう・・・

ならば、負けずにアイボさんにいろんなことをやってもらいましょう・・・ということで、なんとSONYさんが アイボさんを思いのまま操り、アイボさんを思いのままプログラミングできるように「aibo ビジュアルプログラミング」のツールをウェブサイトでなんと提供されているではありませんか・・で、さっそくやってみましょうね。

●よういするのものは・・・

aibo（あたりまえですね）

そして

aibo ビジュアルプログラミング

です。

aiboビジュアルプログラミング？

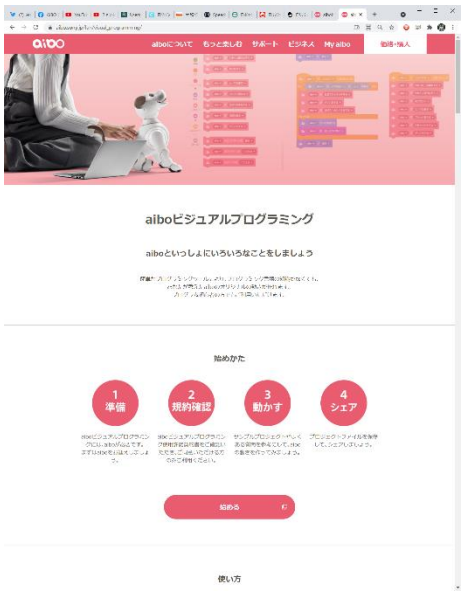
●ちょっとだけaiboビジュアルプログラミングの説明・・・

SONYさんのウェブサイトにaiboビジュアルプログラミングが利用できるサイトがあります。

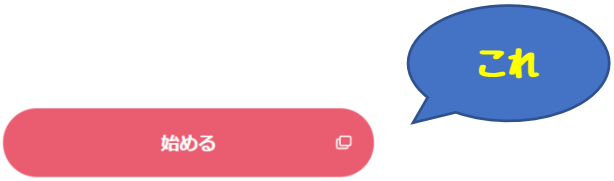
●Google先生などで「aiboビジュアルプログラミング」を検索してみましょう。すると検索結果にでてきました

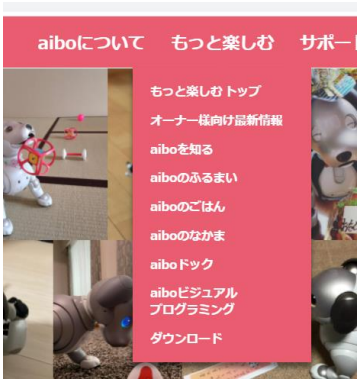
●実際にアクセスしてみましょう

(https://aibo.sony.jp/fan/visual_programming/)



こんな画面がでてきたらOK。スクロールするとたくさんむずかしそうなことが、書いていますが、ここは気にせずに　始めるのボタンを探してみましょう。





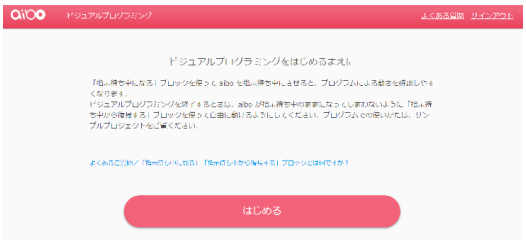
もし、違う画面がでてきたら、上のほうにあるピンク色のバーに【もっと楽しむ】を見つけてください。そこに【aiboビジュアルプログラミング】の項目あったらアクセスしてみてくださいね。前のページの画面が表示されたOKです。

●挫折の第一歩・・・にならいように

【始める】を押したら、な・・なんと・・・こんな画面が・・・

サインイン???　ここで挫折せず、落ち着いてスマホのアイボアプリやSONYさんのサイト登録のＩＤとパスワードを準備しましょう。

ＩＤ（Ｅメールアドレスまたは My SONY ＩＤ）とパスワードを入力してサインイン！。

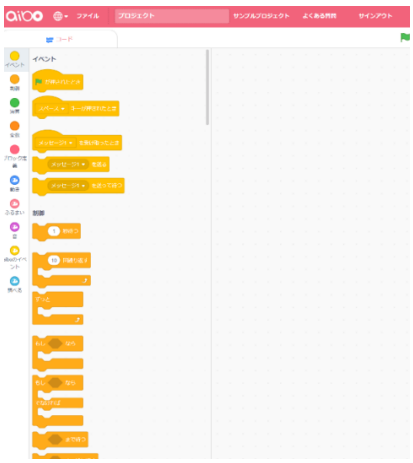


この画面がでたらOKですね・・早速【はじめる】でスタート

1時限目

aiboビジュアルプログラミング??

●挫折の第二歩目・・・にならいように



【はじめる】を押したら、さらにこんな画面が・・・こ～なったらもうチンプンカンプン??なんてことにならないように、ちょっとだけアイボさんをコントロールしてみちゃいましょうね。

まずは、アイボさんを操作しちゃいましょう！操作これだけです。

下のほうに アイボ名 が歌うという紫のブロックがみつかるかと思います

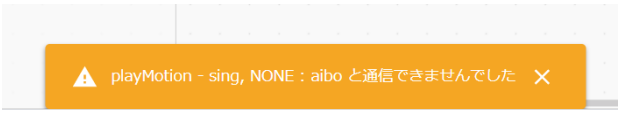
このブロックをクリックしてみてください
黄色の枠が表示されましたね。そのうち黄色表示はなくなります。

アイボさん 上手に
うたってくれましたか？

操作できるほかのブロックとして　【ふるまい】【音】ブロックがありますので、ぜひ押してみてくださいね。

まだまだプログラムを作ったぁ・・・とは言えないですが、１時限目はここまで。次回は、ブロックを積んでいってプログラム的なところをやってみましょう。

もし、うまく動作せずに以下のようなことが起こったら、



が表示されている　や

何度もやってもうまく動作してくれないときは、アイボさんと通信ができていない場合が考えられますので、一度、ビジュアルプログラミングの画面をいったんクローズしてください。また、アイボさんも一度、電源オフ・オンしてみてから、再度実行してみてください。あわててはいけません。落ち着いてやってみましょう。

うまく動作しないときの確認方法など、そのうち載せていきます。

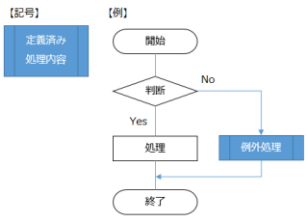
2時限目

プログラム！　プログラムって？

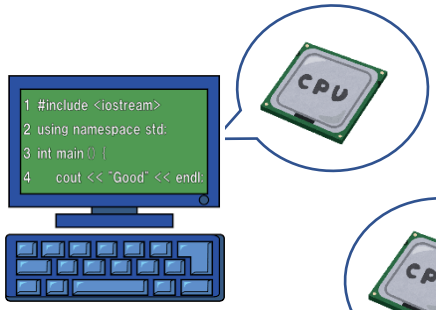
●ところでコンピュータのプログラムっていったい何なの？

コンピュータ？パソコン？マイコン？　違いは何なの？
シーピーユー（CPU）、メモリ、ハードディスク、キーボード、マウスってむずかしい言葉らだけ・・・ですよね。さらにプログラムの前に「ビジュアル」なんて言葉がついちゃったもんだから、きれい？ かっこいい？ 見栄えがよいプログラムなの？
・・・なんてことはあまり考えずに、プログラムってことだけ考えましょうね！

ちょっとだけ、むずかしいことを言うと、やりたいことを順番にコンピュータが理解する言葉で記載して、コンピュータが理解できる数値の羅列に変換してメモリやハードディスクに保存。コンピュータの電源入れたり、プログラムを実行させると、計算して処理して、表示したり、絵を出したり、動作させたり・・・って結構大変なことをやっているのですね



処理の順番



プログラム

実行

ここではむずかしいことは、ぬきにして・・・・

●1 時限目でやった、ビジュアルプログラミングを表示しましょう
画面の説明をしておきましょう！

◆メニューバーを見てみましょう

地球のマーク、ファイル、プロジェクト、サンプルプロジェクト、
よくある質問、サインアウト、…　メニューが表示されています。
クリックしてみましょう。こんな感じのものがでてきましたね。
詳細は今後変えながらやってみましょう！　まずはそのままで・・

英語、日本語（漢字）、にほんご（ひらがな）の標示の選択ができます。

作ったプログラムの保存、読み込み、新規作成の選択ができます。

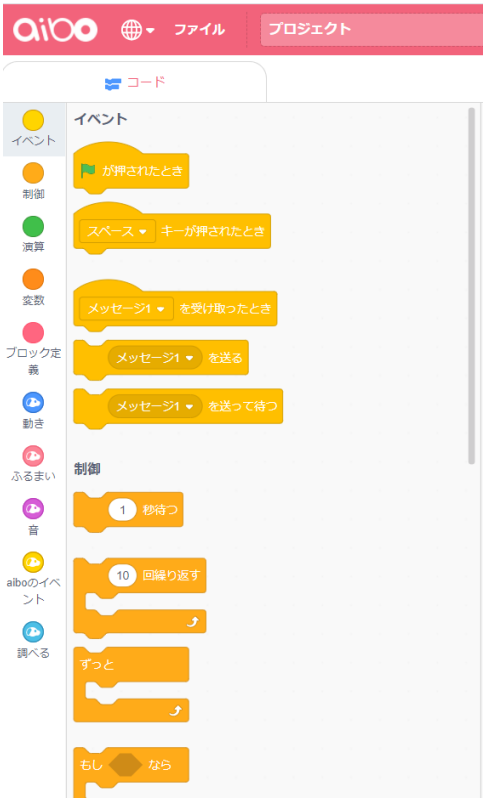
プログラムの名前を指定することができます。[プロジェクト]のままでも大丈夫

サンプルプログラムがあらかじめ準備されています。一度ためしてみるのもよいかと思います。

◆コードの部分を見てみましょう

左にコードという欄があります。一番左のアイコンをクリックすると、それぞれのブロックの場所に移動します。ブロックのリストの欄はスクロールしても同じ場所を探すことができますので、どちらを操作してもよいですね。で・・・

イベント？　制御？　演算？・・・わぉ！ この時点でお腹いっぱいになってしまいそう・・・ちょっと我慢して見てみましょう。



イベント・・何か待って開始するときようですね。旗マークが押されたとき・・よく見ると右上に旗マークがあります。スペースキーが押されたとき・・ちょっとクリックしてみると、キーボートのキーの一覧がでてきましたね。
さらにメッセージを・・って、なんかメッセージが送ることができそうですね。
プログラム作成のときに説明します。

制御、演算、変数、ブロック定義と・・　いっぱいむずかしそうなブロックが表示されていますね。きっとここがプログラムで使うものなのでしょうね。ここはあとでしっかりとみていきましょう。

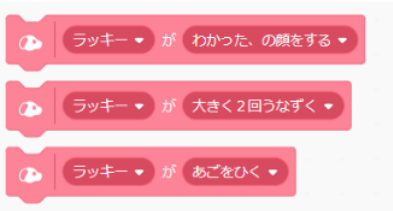
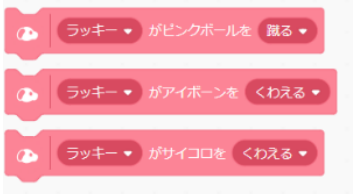
動き、ふるまい、音のブロック・・
ここは一番楽しそうなブロックですね。
1 時限目のところで「音」でアイボさんに歌ってもらいました。ほか、たくさん動作してもらえそうですね。クリックしてみてください。こんなにたくさん動作してくれますね

aibo のイベント、調べるブロック・・
アイボさんと言葉で伝えたり、アイボさんの状況を確認することができそうです。楽しくなりそうです。

◆ブロックの中身をもう少し見てみましょう

先ほど説明した 左側のブロックから【ふるまい】【音】ブロックの中の丸部分をクリックして見てみましょう。

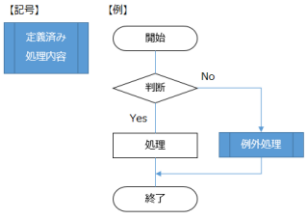
各ブロックの丸〇部分をクリックするとリストが表示されます。ぜひブロックの中をみて、楽しそうな動作を探してみてください。

	アイボさんの表情や感情などをやってくれそうです。ゲームなどをプログラムした時や、ふるまいの時の確認などにやってもらいたい動作です。ぜひ、1 時限目でやったときのようにクリックして実際にアイボさんにふるまいの動作をして、どんな動きか見ておきましょう
	ここは楽しそうな 動作がいっぱいあってワクワクします。 特に全身で楽しそうな動作をしてくれそうなので、ぜひ一度やってみてください。普段アイボさんがやっている動作や、ほとんど見ることがない男の子用、女の子用の動作など、いっぱ～い　ありますよ。ぜひやってもらって、お友達に教えてあげましょう。
	ここは、おもちゃをつかった動作ができそうですね。ピンクボール、アイボーン、サイコロを使ったゲームなどできそうです。とっても楽しい動作のアイデアができそうです。
	ここは、音に関する動作がたくさんありますね。普段見ている動作って実はこの動作だったんだ・・って発見できます。ぜひ一度やってみてください。

◆プログラムっぽいものを・・・

ブロックの中身をみたところで、さっそくプログラムっぽいものをやってみましょう！

２時限目の最初にプログラムとは、やってほしい動作を順番にすることを決めてコンピュータに教えてあげることって説明したとおりです。早速アイボさんにやってほしいことを順番に決めていって覚えて実行してもらいましょう。



アイボさんにやってほしいこと・・・今回は

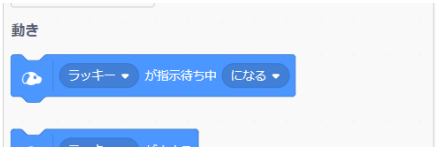
① 大きく２回うなずく

② ハイタッチをする

をまずはやってみましょう！

◆まずはアイボさんをおとなしく待ってもらいましょう。

これからやってもらいたいけど、アイボさんは気ままに遊んでいます。ちょっとだけ待ってもらいたので【指示待ち中】を使って待ってもらいましょうね



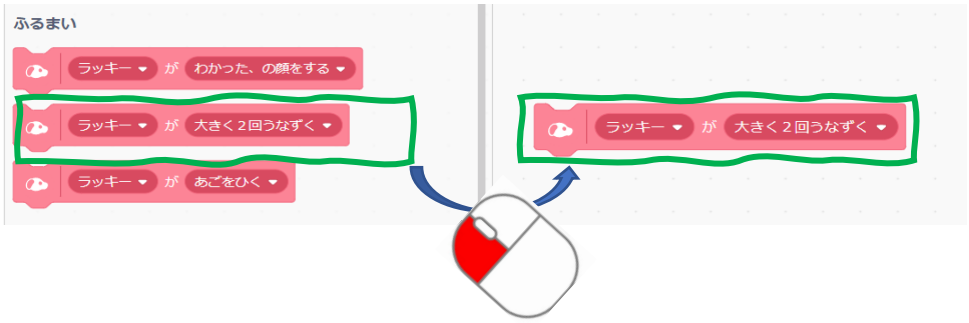
【動き】のブロックから指示待ちのブロックを探してくださいね。このブロックをクリックすると黄色になって アイボさんがおとなしく待ってくれます。

また、まってもらったままだと かわいそうなので、プログラムの実行が終ったり、作業が終ったら必ず指示待ちからもとに戻ってもらいましょうね。
もとに戻ってもらう時には、指示待ち中ブロックの丸〇をクリックして【から復帰する】をクリックして、【指示待ち中から復帰】にしましょう。

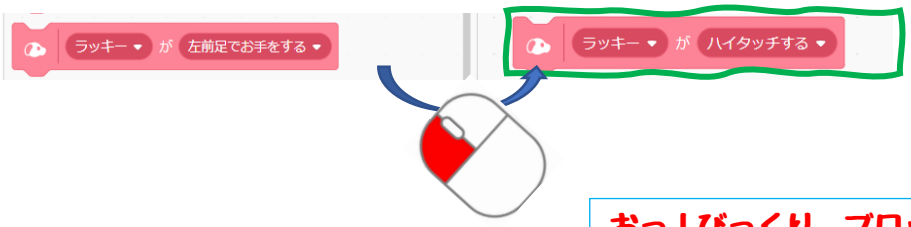
このブロックをクリックすると黄色になって、アイボさんは指示待ちからもとに戻ってくれます。

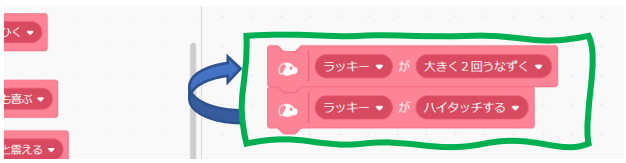
◆では・・ブロックを選んでぇ～　くっつけてぇ～・・・

最初にふるまいから、【大きく２回うなずく】を探してみてください。みつかったら、マウスをつかってブロックをクリックしながら、右のコードエリアに、ず・ず・ずぃ～と　ドラッグ・アンド・ドロップすると、あら不思議と同じものが表示されます。



つぎにほかのブロックから【ハイタッチをする】を探してみてください
【左前足でお手をする】のブロックをさがしてクリックすると見つけることができますね。クリックして同じようにブロックを移動してみましょう



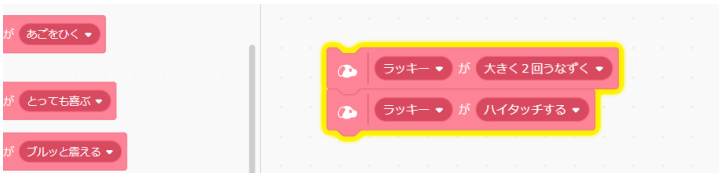


おっ！びっくり　ブロックを近づけると勝手にくっついてしまいましたね～
ちょっと下のブロックをずらしてみると外れますよ！

またいっぱい作っちゃった場合は、デリートボタンを押すか、左のブロックエリアに移動させると消えますよ

◆では・・緊張の　プログラムの実行・・・

先ほど勝手にドッキングしたブロックをクリック（プログラムの実行）してみてください。すると　２つのブロックが黄色になって・・・



アイボさんが、２回うなずいて、
ハイタッチしてくれましたか？
　プログラミング　成功！ですね

２時限目はここまで！　　３時限目は　もう少しいろいろやってみましょう。その時までに、ほかのブロックをくっつけたりして、試してみましょう。

３時限目

プログラムっぽいものにちょうせん

●プログラムっぽいプログラムにちょうせん

２時限目にブロックをくっつけて実行すると、なんと、その通りにアイボさんが動作します。アイボさんは上手に実行してくれましたか？

いろいろなブロックをくっつけると、その通りにアイボさんはやってくれますが、こんなことをしたいときはどうしましょうか？

アイボさんがほえる

今回は「２回ほえて」ほしい

あっ　するどい！！

そのその通り【ほえる】のブロックを２つつなげると２回ほえてくれます。

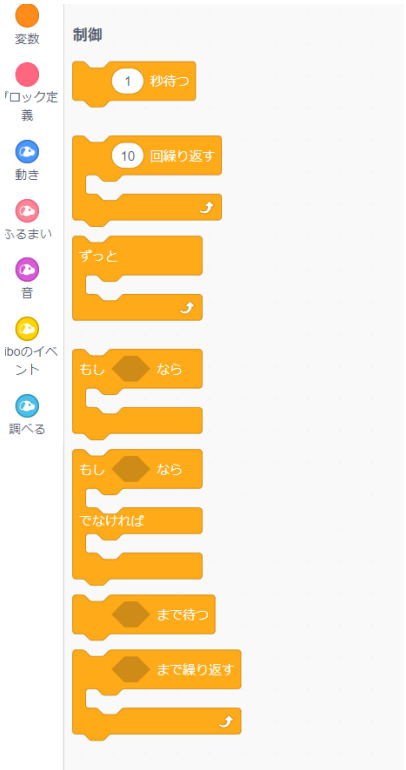
では、１０回ほえてほしいときは？
そのとおり。１０回ならべれば　よいのです。

では、１００回は？　１０００回は？
なんかブロックの数をかぞえるだけで大変そうですね～

ブロックを１０個並べていくだけでも、へこみますね・・・

●ちょっとプログラミングっぽくしてみましょう。

２時限目に 画面の左側にいろいろなブロックがあることを説明しました。もう一度、見てみましょう。すると、こんなブロックが見つかりました。



【制御】ブロックに以下のものがあります。

- １秒待つ
- １０回繰り返す
- ずっと～
- もし～なら
- もし～なら～でなければ
- ～まで待つ
- ～まで繰り返す

繰り返ししたいときに使えそうなブロックのようですね。

●繰り返しにちょうせんしてみましょ

使えそうなブロックが見つかったところで、使ってみましょう。
先ほど実行しました【ほえる】を繰り返し実行してみました。
さすがに１０回、１００回、１０００回をアイボさんにやってもらうのはかわいそうなので「３回ほえる」をやってもらいましょう。

【繰り返す】のブロックを見てみましょう

なんかヘンテコな形ですが、よ～く見るとブロックがそのまま入りそうな形ですね。
また、１０回繰り返すとなっていますので今回使えそうな感じです。

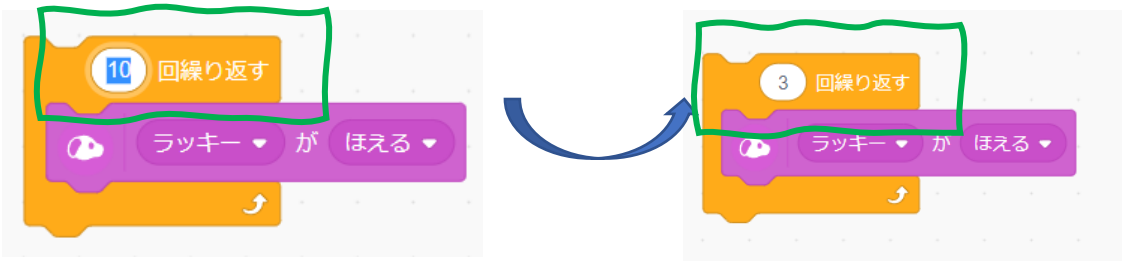
そうなのです。この【繰り返す】ブロックは表示されている数字の回数だけ、その中のブロックを繰り返す制御をしてくれるものです。では、さっそく使ってみましょう。操作はとっても簡単です。

【繰り返す】ブロックに【ほえる】ブロックを近づけると・・・
あら不思議、合体してしまいましたね・・

これで繰り返しのプログラムは完成です。　簡単ですね！

ん！？　ほえるを１０回はかわいそうなので３回だったのでは？

お～、そうでした～
ブロックの中に（１０）と白く丸くなっているところをクリックすると数字が変更できるようになります。



さっそく　キーボードで　数字の【3】に変更してみましょう。
入力が終わったら、【ENTER】キーやマウスでほかのエリアをクリックすると決定できます。

さっそく、実行してみましょう。ブロックをクリックすると黄色くなりますので、３回ほえてくれるか確認してみましょう

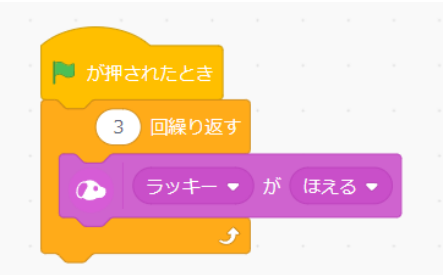
うまくやってくれましたか？
繰り返しの時には、このような制御ブロックを使うと、同じことを繰り返ししてくることがわかりますね。制御ブロックの中には複数のブロックを入れることができます。

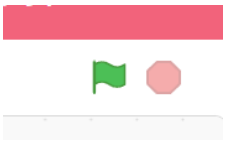
是非、いろいろと試してみてください。

ところでいっぱいブロックを並べていくとスタートの場所がよくわかりませんね。ビジュアルプログラミングは、ブロックをクリックすると枠が黄色くなり、すぐに開始されるようになっています。
でも、あっちこっちにブロックを使ってプログラムしていくと、どこがスタートかわからなくなってしまいそうです。

そこで、コードの左欄の先頭のほうにイベントというブロックがあります。
最初に【旗マークが押されたとき】というイベントがあります。ビジュアルプログラミングの画面の右上にも同じような【旗】マークがあります。
そうです。この右上の旗マークが押されたときにスタートすることができます。

では、３時限目の最後にプログラムっぽく作ってみましょう。とっても簡単です。旗のブロックをくっつけるだけです。





では、画面の右上の【旗】マークをクリックしましょう。同じように実行してくれましたか？

３時限目はここまでです。４時限目は　もう少しプログラムの制御をやってみましょう。その時までに、ほかの繰り返しブロックなどを試して、アイボさんにいろいろと動作してもらいましょう。

4時限目

気まぐれアイボさん！のプログラム

●ちょっと高度なプログラムにちょうせん

３時限目は、【繰り返す】プログラムを作ってみました。え？ プログラムって繰り返すことだけなの？？
いいえ、そんなことはないですよね～。コンピュータっていろいろな判断をしながら処理をしますよね？　たとえばキーボードのキーが押されたときにはそのキーの文字を画面に表示されます。
マウスをクリックしたら、動画が再生したり、ウィンドウの移動ができたり、図形や文字の選択ができたり・・・って、色々と動きますよね・・そういえば、このテキストで学習のビジュアルプログラミングの画面だっていろいろ操作してアイボさんを動かしていますが、このビジュアルプログラミングだって、きっとＳＯＮＹさんのすごい技術の人が作ったプログラムですよね～

アイボさんもいろんなことができるはずなので、いろんなことをしてもらいましょう。

でもいきなりむずかしいのは、むずかしいし・・ん？あたりまえ！
でも何がむずかしいことのかが、むずかしいし・・　なやましい！

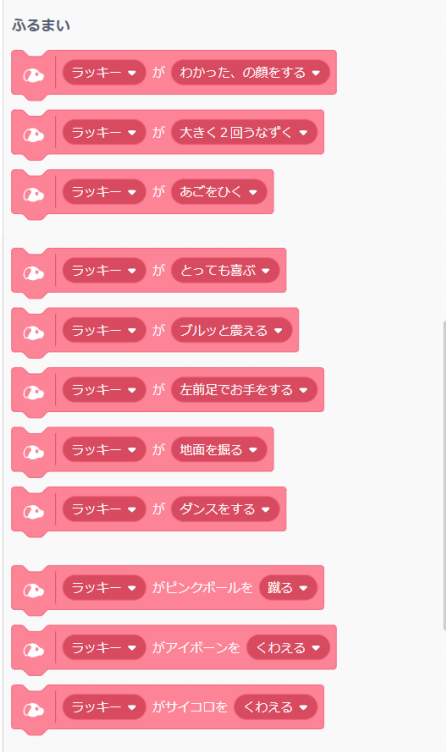
そこで、以下のような課題を考えました。

アイボさんは気分屋さんなので、
その気分に合わせてやってもらいましょう

さて、アイボさんはいろんなことができます。【ふるまい】のブロック（ピンク色のブロック）を見てみると、たくさんのことができ

ることが分かります。

ぜひ、一つ一つ実行してみてください。これまで見たことがないアイボさんの動作が見つかるかもしれません。



今回は【気分に合わせてやってもらいましょう】の課題です・・・でも気分ってどうやってわかるの？　はちょっとむずかしそうなので、次のようなことを考えてみましょう。

ランダムな数字１～３をだして
　【１】だったら、【この動作】をして
　【２】だったら、【この動作】をして
　【３】だったら、【この動作】をして

と３つの動作をその時の状況（気分次第）で実行してもらうことを考えてみましょう。

●まずは必要なものを考えてみましょう
課題の内容から、以下のものを準備する必要がありそうです。

① アイボさんに実行してほしい動作を３つ用意する
② １～３の数字を決める何かが必要
③ １～３のどれかが決まったらやってほしい動作を実行

さて、さて、皆さんはブロックを探し始めているかと思います。

「アイボさんに実行してほしい動作を３つ用意する」は、お得意の【ふるまい】ブロックから探せそうですね。すでに見つけていた？さすがです・・

では、「 ② １～３の数字を決める何かが必要 」は・・・？？？
そうですねぇ～　するどいですね

【演算】のブロックに【(1)から(10)までの乱数】というブロックがありますね。

ところで乱数ってなに？？？　いきなりむずかしいですが、簡単にいうと１～１０までの数字を勝手に選んでくれます。

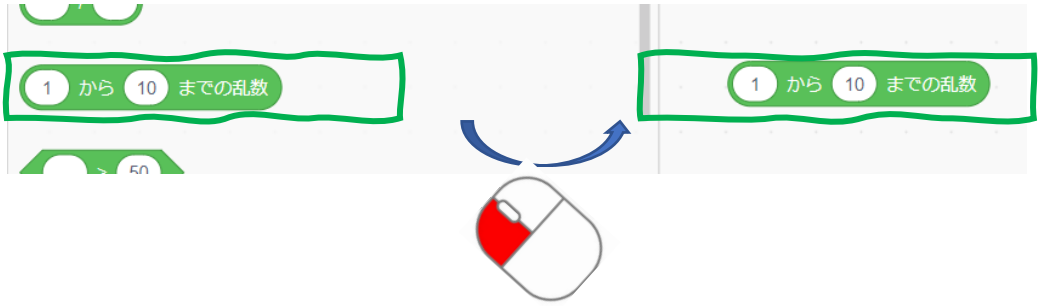
ん～　ちょっとむずかしいですね・・・
ちょっとだけこんな操作をしてみてください
【(1)から(10)までの乱数】を何回かクリックしてみてください

クリックする毎に数字が表示されていますよね

つまり、このブロックは１～１０までの数字を勝手に選んでくれる演算ブロックとよばれるものです。

課題準備の「 ② １～３の数字を決めるものが必要 」で使いましょう。今回は１～３なので範囲を変更してみましょう。

最初にこの演算ブロックを右エリアに持ってきておきましょう



次に持ってきた右のブロックの数字（１０）の部分を（３）に変更しておきましょう。

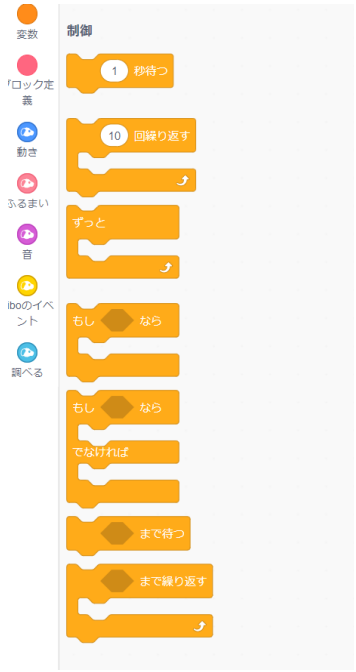
（３）に変更！

（３）に変更したブロックを何度かクリックしてみてください。
あら不思議、どんなに頑張っても、
１、２、３のどれかになります。

そうです。これで準備②の１～３の数字の準備ができました。
さて、次に「③　１～３のどれかがきまったらやってほしい動作を実行」の方法を考えてみましょう・・・

いきなり、むずかしそうな内容にぶつかりましたねぇ～

制御したい時には・・・そうです！制御ブロックのところをもう少し見てみましょう。



制御のブロックのところには

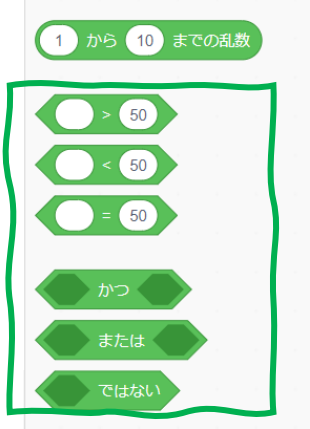
- １秒待つ
- １０回繰り返す
- ずっと～
- もし～なら
- もし～なら～でなければ
- ～まで待つ
- ～まで繰り返す

なにやら【もし～なら】が使えそうなブロックのようですね。
では、さっそくブロックを右エリアに持ってきましょう。

ん？　先ほどの【乱数】と【もし～なら】ブロックはどうやっても合体できるような形ではないですね・・・　乱数は長丸なのに「もし～は」ひし形です。・・・なにかが足りないです

そうです。１なのか？　２なのか？　３なのか？　を決めるための演算がどこにもありませんね。

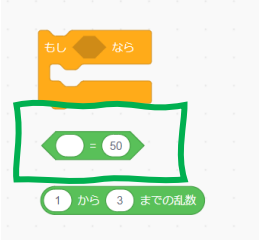
どこかに　１か２か３を判断するブロックが必要のようです。
そうです・・・【演算】ブロックを見てみると・・

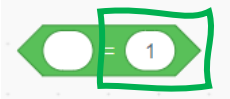


計算するようなものとひし形のブロックがあります。このひし形が先ほどの【もし～ならば】のオレンジのブロックの内側に合体できそうです。

ちょっとむずかしそうですが、この制御をうまく使えそうなので、ここから１ならば～、２ならば～、３ならば～、を考えてみたいと思います。

計算式に【○=(50)】のものがありますね。これが使えそうです。
では、このブロックを右エリアに持ってきて、以下の操作をしてみましょう。





50 を 1 に変更してみましょう

必要なブロックがそろったので、次のようにやってみましょう。

① 【１～３までの乱数】ブロックを 演算 【○=1】の左○側に入れてみましょう。ピタッ！と、はまるとちょっと感動！

もし、まちがって右の（１）に入ってしまっても落ち着いて！【1～３までの乱数】のブロックをマウスでクリックして移動すると外れます。

② 次に①で合体した【<（１から３までの乱数）=１>】のひし形のブロックを【もし～なら】制御ブロックと合体してみましょう

なんか勝手にサイズが変更したりして、むずかしい感じになりました。合体ブロックで何ができたか少し考えてみましょう。

もし　１から３までの乱数 ＝ １ なら

そうです。さきほどまで用意したものをおさらいすると

１～３までの数字を選択してもらう乱数を準備
その数字が１か２か３のいずれかで・・
それが１だったら～　のブロックができちゃいました。

③　では、次に何か！をやってもらいたいので【振る舞い】のブロックを選んで合体してみましょう。今回はハイタッチするを選んでみました。ピンク色のブロックの中からいろいろと探してみてくださいね。

もう一度ブロックを見てみましょう

もし、１から３までの乱数を準備しその数が１なら
アイボさんがハイタッチをする

となりました。このまま実行もできます。ブロック全体をクリックすると黄色の枠となりますので、乱数が１の時にハイタッチしてくれます。３時限目にやりました旗マークのスタートブロックを先頭におくと、旗ボタンをクリックした時にはじめることができます。

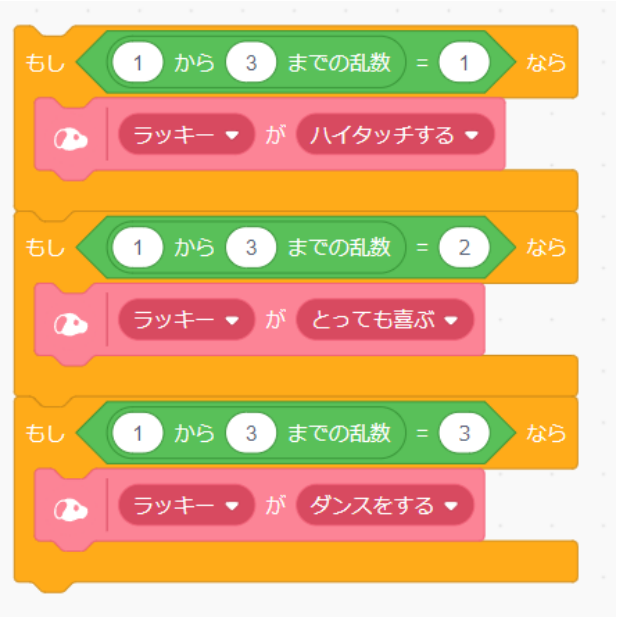
どうにかこうにか、高度！高度？なプログラムのような感じがしま

したねぇ・・でも、でも、でも、なぜかアイボさん知らん顔していることが多いです。そうです、これだと１の時だけハイタッチをしてくれますが、２，３の時には、何もしないで終わってしまいます。

そうですよね・・２は？　３は？　これだけだと =１を２や３に数を変更する必要があり、なんか大変な状況です。
でもきっと解決策はあるはずです・・そうそう、前回繰り返しをした時のように、このブロックを並べればよいじゃないですかぁ～

なので、こうしちゃいましょう！

１のとき、２のとき、３のときの動作をそれぞれ動作を変えて、同じようにくっつければ、できちゃうじゃん！
さっそく実行～
でも、なんかおかしいことが起きています。



次がようなことが起きているかと思います。
① 何もしてくれないことがある
② ２種類以上、違う動作をすることがある

そうなのです【もし～】制御ブロックを実行するたびに１～３の乱数を作るので、１つ目の「もし～」ブロックで１だったら実行、それ以外のときは何もしない。次に２つ目の「もし～」ブロックを実行、また1～３のどれかが選択され２だったら実行、１、３だったら何もしない、次の３つ目の・・・

毎回１～３の乱数を作るのでこの方法はだめですね。乱数が決まったらどこかに値をとっておく（保存する）ことができるとよいですよね。

4時限目は、ボリュームが多かったので、ここまでにしましょう
5時限目は、今回の問題をどのように解決するかを考えてみましょう。一度考えてみてくださいね。

4時限目のおさらい

① 乱数ブロックで指定した範囲の数字を自動に選ぶ。
実行するたびに数字が変わる（乱数）
② 制御ブロックは、繰り返しだけはなく条件が一致すると
実行することができる
③ ひし形のブロックの〇部分にほかの〇ブロックを入れる
ことができる。

4時限目の課題

① 一度決めた乱数を保持する仕方
② 1の場合、2の場合、3の場合のように条件が複数あり、
振る舞いを変えたい場合のプログラムの仕方

制御ブロックに ヒントになりそうなものがありそうです。
乱数ブロックは実行するたびに数字がかわってしまいます。実行中に変えたくない場合、コンピュータでよく使われる変数というブロックを使います。

次回は、値を保持しながら、処理を進めて、条件が一致したら実行してもらうようなプログラムを作っていきましょう。

【番外編】
今回使った、制御ブロックの【もし～】の条件は、計算だけではなく、ほかの条件も指定ができます。ブロックの下のほうに【調べる】ブロックというものがあります。

ここには、とっても楽しそうな調べる条件が沢山あります。

今後、これらの条件をつかったものを紹介していきたいと思います。

まずは、簡単な振る舞いをいろいろと試してみて、ビジュアルプログラミングを使ったアイボさんの動作やタイミングの感覚をぜひ覚えていってください。

５時限目

気まぐれアイボさん！のプログラムその２

● 乱数をちょっとだけ、おぼえていてよぉ～

４時限目は、アイボさんにその時の気分に合わせて、動作してもらうようなプログラムを作ってみましたね。

４時限目のおさらい

① 乱数 ブロックで指定した範囲の数を自動に選ぶことができる。ただし、実行するたびに数が変わる

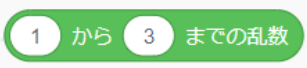
② 制御ブロックには、繰り返しだけはなく、条件が一致すると実行するブロックがある

③ ひし形のブロックの白○の部分に更に別の長○ブロックを入れることができる。

４時限目で、乱数１の時は、２の時は、３の時は・・・と作ったのに、なぜか正しくアイボさんが動いてくれないようですね・・・

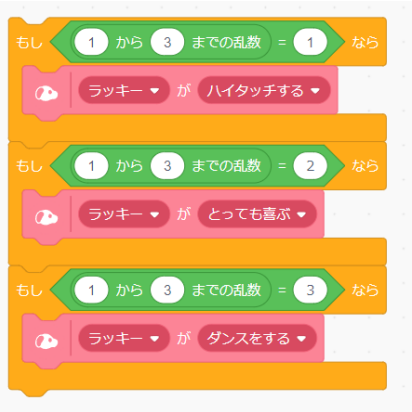
なぜでしょ～？？？

前回、乱数ブロックで、実行したことを思い出してみましょう。
乱数ブロックをクリックしたときに値が１～３まで変わりましたね



このブロックをクリック

つまり、この乱数ブロックは処理されるごとに数が変わります。



⇒このときに値が１～３に決まりますね
１の時だけハイタッチ、２，３は何もしない
⇒このときにも１～３のどれかですね
２の時だけよろこび、１，３は何もしない
⇒さらにここも１～３のどれかになりますね
３の時だけダンス、１，２は何もしない

最初１～３のどれかが決まったあとにも、次に１～３のどれかに変わってしまうため、今回やってみたい１～３を一回だけ決めて

もし１だったら　ハイタッチ
もし２だったら　とっても喜ぶ
もし３だったら　ダンスをする

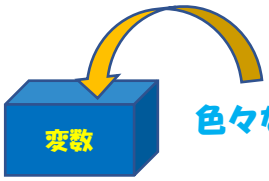
とやりたい動作にならないこととなりますね。

ん～　これにはこまりましたね・・・・何かよい方法は？

もう少しブロックを見てみましょう。その中にむずかしい言葉ですが【変数】というものがあります。

コンピュータ言語には、必ずと言っていいほど
【変数】という言葉が出てきます。
ぜひ、覚えておきましょう。

【変数】はコンピュータの中に一時的にデータを記憶（保存）しておくことができる魔法の箱のようなものです。



色々な値を入れておき記憶(保存)しておくことができます

このブロックを使って、最初に決定した乱数１～３を記憶しておくことができそうです。ちょっとだけ、むずかしくなりましたが、次のことをやってみましょう。

【変数を(0)にする】ブロックと【変数】の丸のブロックを右エリアにもってきて、このブロックを順にクリックしてみましょう。すると【0】が表示されました。

次に、制御ブロックの白Oに表示されている数字(0) を 他の値に変えてみましょう。　例えば0⇒3に変更　【変数を3にする】にしたブロックをクリックして【変数】をクリックしてみましょう。

変数の値が ３ になりましたねぇ。
つまり【変数】と呼ばれている箱に3が設定されました。ほかの数字や文字に変更して確認してみましょう。

１２３４に変更

ＡＢＣＤＥＦＧに変更

記憶(保存)できるって、やはりコンピュータはすごいですね・・・

これを使って決定した乱数１～３を記憶しておきたいですね！
ではどのようにするのかを考えてみましょう。
ビジュアルプログラミングは、同じ形の他のブロックをいれることができます。

これまで　乱数ブロック を使ってきましたが長Ｏですね・・・
また、先ほど０を変更した部分も長Ｏです。つまり！
さすが～そうです。ではやってみましょう。

このブロックを準備　　１０を３に変更

乱数のブロック合体　　こんな形になりますね

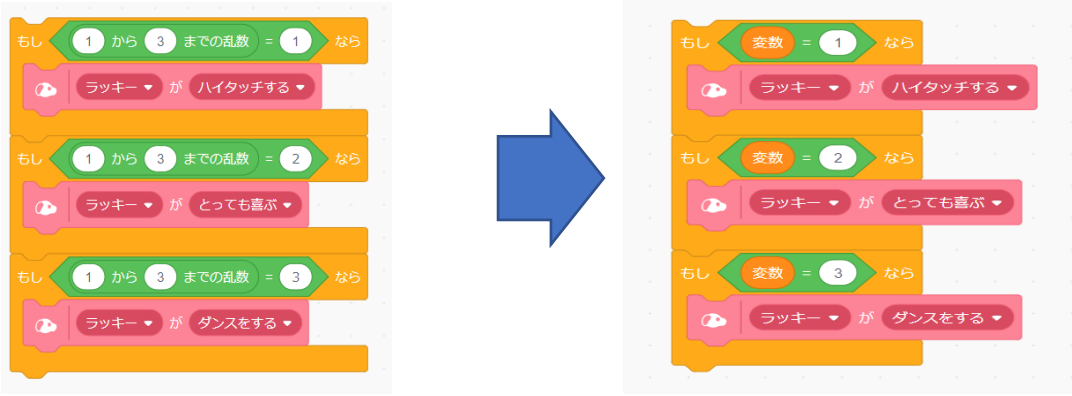
先ほど０の数字を３やＡＢＣＤＥＦＧに変えたらその値が変数の箱に保存され【変数】をクリックするとその値が表示されましたね。

次の操作をやってみてください。

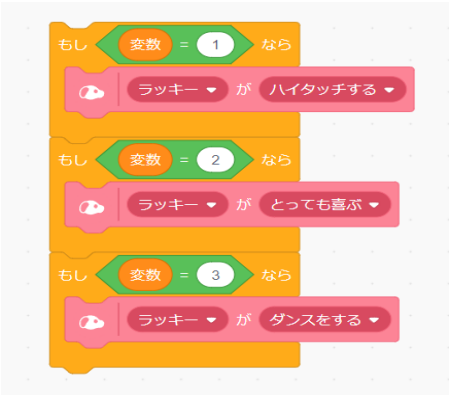
⇒ このブロックをクリック
⇒ このブロックをクリック

すると【変数】の数が１～３のどれかになりますね。

つまり、【変数】と呼ばれた箱に乱数１～３が設定されていることが確認できたかと思います。４時限目に作ったプログラムを変えてみましょう



内容を再確認してみましょう。【変数】に数が入っているので・・



⇒変数の値が１だったら
⇒アイボさんが【ハイタッチする】をする
⇒変数の値が２だったら
⇒アイボさんが【とっても喜ぶ】をする
⇒変数の値が３だったら
⇒アイボさんが【ダンス】をする

プログラムようなものになりましたね。それでは、今回作ったものとこれまでのおさらいをしながら、アイボさんの気持ちプログラムを完成させてみましょう。

使用するブロックが多いですが、なれるためにも、がんばってブロックをつなげてプログラムを完成させていきましょう。

⇒ プログラムの開始
⇒ 【変数】を乱数１～３に設定する
⇒ 【変数】が１ならば
⇒ アイボさんがハイタッチする
⇒ 【変数】が２ならば
⇒ アイボさんがとっても喜ぶ
⇒ 【変数】が３ならば
⇒ アイボさんがダンスをする

【旗】マーク を押して実行してみましょう。
気分（１～３）によってどれかをやってくれたかと思います。
【ハイタッチ】か【とっても喜ぶ】か【ダンスをする】

アイボさんってやっぱりすごいですね～

５時限目はここまでにしましょう。
５時限目では【変数】と呼ばれるブロックを使って、乱数の値を記憶(保存)、制御ブロックで動作を決めるプログラムを作ってみました。変数とよばれるブロックで値を保存しておくことで、色々と使うことができそうです。
今回の【気まぐれアイボさん！プログラム】をいろいろと改良してみて、今回１～３の３つの動作してもらいましたが、次回は、もう少しいろいろと制御してみましょう。

【番外編】

今回は 制御ブロック【もし～】を使いましたが、ほかに【もし～なら～でなければ～】というものがあります。このブロックを使うと以下のようにも変更ができます。実は同じことができます。

【変数=3なら】がなくなりましたが、なぜ同じようなことができるか考えてみてください。乱数の範囲3⇒4にしたら、この場合どうなりますでしょうか？

このようにプログラミングには、プログラムした人ごとの考え方（アルゴリズム）によって、同じ処理でもいく通りにでも作ることができます。ぜひ、色々と工夫してみてくださいね。

６時限目

アイボさん 今日の晩ご飯を選んで！

● アイボさんをちょっと移動させてみましょう

５時限目は、アイボさんにその時の気分（乱数１～３）に合わせて、動作してもらうようなプログラムを作ってみました。
魔法の箱！【変数】ブロックを使って、乱数で決めた値を一時的に保存しておき、【制御】ブロックを使って、１のときは・・・、２のときは・・・、３のときは・・・　それぞれ違った動作をやってもらいました。どうでしたか？

さて、ブロック欄をもう少し見てみると【ふるまい】以外にも楽しそうな動きがありますね。ちょっとだけ【動き】ブロックを見てみましょう。
ひとつひとつ見ると、全身をつかった細かい動作をすることができそうです。
いっぱいありすぎなので、お腹いっぱいになりそうですね。

ここのブロックもクリックすると動作をその場で実行してくれます。ぜひ、好きな動作をさがしてみてくださいね。たとえば【おなかを見せる】をクリックすると・・ワクワクですね。

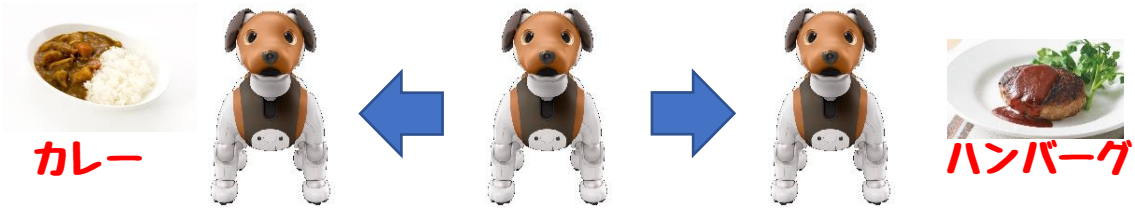
●アイボさんは移動できるのです・・
ブロックを見てみると、１メートル前に歩く、１メートル横歩きする、右に４５度回る・・・など移動するブロックがあります。
これも面白そうですね！

５時限目でつくりましたプログラムをちょっとだけ変更して、次のようなプログラムのテーマを考えてみましょう。

どっちか決められないことってときどきありますよね～、誰かに決めてもらいたいときってあるかと思います。今回はアイボさんに、「今日の晩ご飯どっちが食べたいか」決めてもらうプログラムを考えてみたいと思います。

テーマ：【アイボさん 今日の晩ご飯を選んで！】
ランダムな数字をだして、今日の食べたい晩ご飯を選んでその場所に移動してもらうことを考えてみましょう。前回のプログラムが使えそうですね。

① 乱数でどちらかを選択
例：１の時はカレー、２の時はハンバーグ
② 選んだほうに横移動する



１のとき右 50 センチ移動　　２のとき左 50 センチ移動

５時限目で作成したプログラムをちょっと変えるとできそうですね。すでにマスターした皆さんは、もう簡単ですよね。

なお、移動するは次のものが使えそうです。距離の１メートルは広い場所で実行してみてください。もし確保できない場合は、１メートルを、０.５にすると ０．５メートル＝５０センチメートルとなりますので、実行する場所に合わせて設定してみてください。

下のような感じになったかと思います。これが正解ではなく、色々な作り方があります（あとの応用編でご紹介します）。

⇒ プログラムの開始
⇒ 【変数】を乱数１～２に設定する
⇒ 【変数】が１ならば
⇒ アイボさんを右に５０センチ移動

⇒ 【変数】が２ならば
⇒ アイボさんを左に５０センチ移動

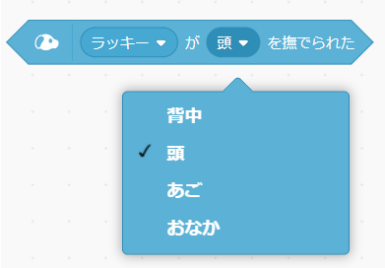
さっ、実行してみてください。右か左に移動してくれましたか？
アイボさんに決めてもらうプログラムって簡単にできましたよね。
２個からどちらかを決めてもらう時、色々と応用できますね。

旗ボタンスタートで決定してもらうのはできましたが、もう少しアイボらしいプログラムを考えてみましょう。アイボさんは、頭や背中をさわられていることを認識することができます。
さっそく【調べる】ブロックを見てみましょう。

【調べる】ブロックに【撫でられた】のブロックがあります。これを使ってみたいと思います。

【撫でられた】の○部分を見てみると、確認できる部分がいっぱいありますね。

背中、頭、あご、おなか

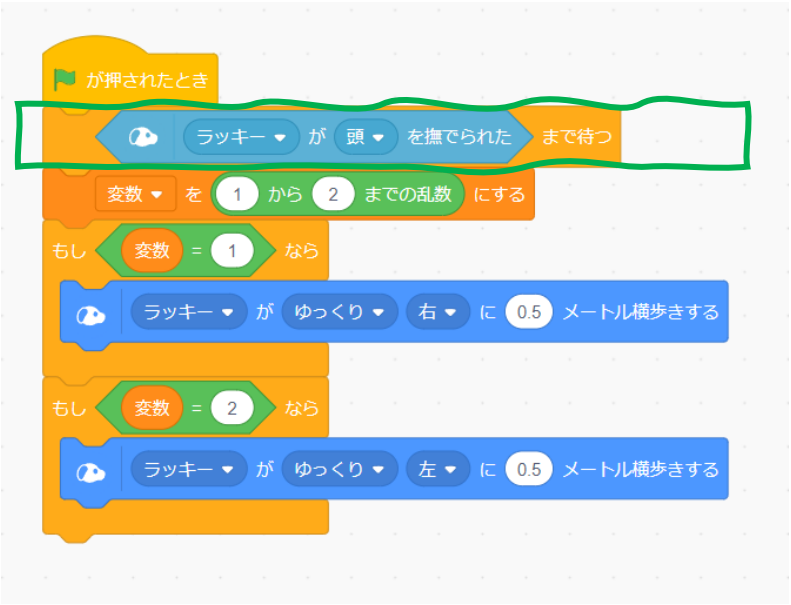


確認したい場所を選択すると変更ができます。今回は【頭】を撫でられた場合を説明しますので、【頭】を指定します。皆さんは好きな場所を指定してくださいね。

【頭】を指定した【調べる】ブロックを使ってみたいと思いますが、このブロックはひし形です。このままだと判断ができないので、これまで見てきた【制御】ブロックで判断することができそうです。以下のブロックを使うとできそうですね。早速合体しましょう。

合体

合体したブロックを先ほど作ったプログラムに追加しましょう。



←ここに追加

追加の仕方

①　オレンジ【変数】部分をクリックして下にずらす

②　旗のブロックとバラバラになったら、いつも通りブロックを重ねて

さっそく、実行してみてください。今回は、アイボさんがそのまま頭をなでてもらうまで待ってくれますので、頭を撫でてみてください。すると、右または左に移動してくれるかと思います。

「今日はどっち？」と聞きながら、アイボさんに頭を撫でて決めてもらいましょう。ちょっとアイボさんと会話ができたようで楽しいですよね～。選んでもらうシーンってたくさんありますよね。

ご飯やおやつを選んでもらう、着ていく服を選んでもらう、行きたい場所を選んでもらう、くじを選んでもらう　などなど・・

頭を撫でた後に【ほえる】ブロックを入れると、撫でたあとに返事をしてくるような感じになるかと思います。

ぜひ、アイボさんらしさを追加して、色々とやってみてください。

応用編　同じ動作ですが、色々な作り方ができますので、ちょっと紹介します。プログラムは作る人によって、さまざまです。正解はありません。

【応用１】制御プログラムで　もし～でなければ～　に変更してみる

１のときは、右に移動

変数１以外のとき、つまり２のときに左に移動

【応用２】右か左に移動の判断で【制御】ブロックを使わないケース

今回、乱数１、２で決めますが、１か２なのと、移動にはマイナスの距離の指定ができますので、計算だけで処理ができます。

【演算】のロジックに【ー(マイナス)】がありますので、左のようにすると

１の場合　１－１.５ ＝ －０.５ となります

２の場合　２－１.５ ＝　０.５ となります

また、【移動】ブロックにはマイナスの指定もできます。

①　０．５の場合　【左】に０．５センチメートル移動となります。

②　－０．５の場合【左】に－０．５センチメートル移動、つまり【右】に０．５センチメートル移動となります。

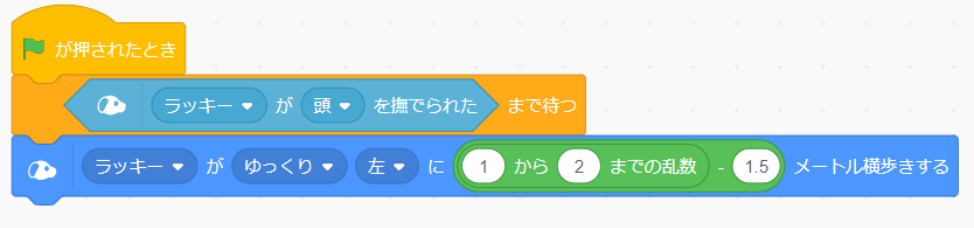
これらを組み合わせると、【制御】ブロックはいらなくなります。

乱数１～２から－１．５を計算。１のときは変数が－０．５となり、左に「－０．５」、つまり、右に０．５メートル移動。２のときは０．５となり左に「０．５」メートル移動。

【応用２】変数ブロックも使わないで最小ブロックで同じことを実行

さらに・・・以下のように直接計算した結果で移動させるようにすると最小限のブロックで同じことができます。

ほんとなの？？？　ぜひ考えて実行してみてください。



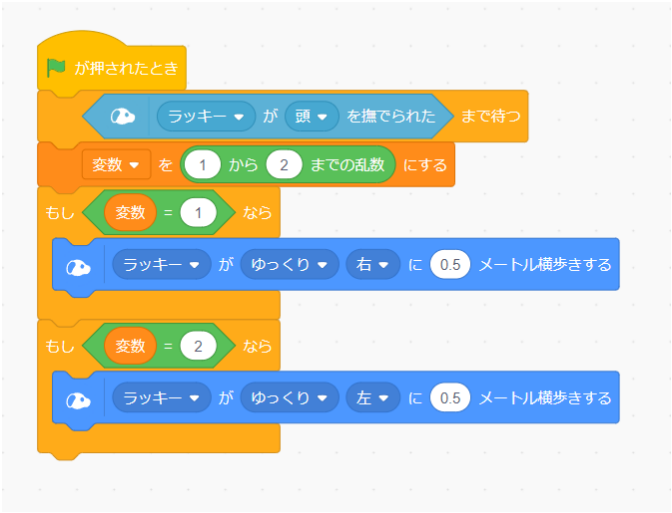
このブロックの内容を順番に見てみましょう

- 頭がなでられるまで待つ
- アイボさんがゆっくり左に・・
- 乱数１～２の数字を選んで（１～２なので、１か２が選択）
- 乱数の数字１または２から　１．５ を引き算します。
  結果、－０．５か０．５となります。
- 左に －０．５か０．５メートルを横歩きする
  １のとき左に－０．５メートル、つまり【右】に０．５メートル移動
  ２のとき左に　０．５メートル、つまり【左】に０．５メートル移動

やりたかったことが一緒ですね！

並べてみましょう。この4つのプログラムは、どれも一緒の動作をします。
なんか不思議ですね。考える人によって、さまざまな作り方があるのです。

パタン１



パタン２

パタン３

パタン４

６時限目は、ボリュームが多かったので、ここまでにしましょう

おさらいしましょう。

- 乱数で数を決めます
- 変数に値を保存する
- 乱数で決めた値に合わせて制御（動作）する
- アイボさんの移動ブロックを使って移動させてみる

応用編

- もし～　を　もし～でなければ～　に変更しても同様
- 移動などのブロックにはマイナスの距離が指定できる
- ブロックの中にブロックをいれることで複雑な計算ができる
- いろいろなプログラムの作り方ができる

移動したあとに決まったことを知らせるように歌うブロックを追加するなど、アイボさんらしさのブロックを見つけて、いろいろとやってみてください。

今回、乱数、調べる、制御、移動ブロックを使って、アイボさんに選んでもらうプログラムを作ってみました。おみくじ、運勢占いなどにも使えそうですね。また、応用編として、計算、計算した結果をそのまま動作のブロックと組み合わせしまうことで、いろいろなプログラムの作り方があることが理解できたかと思います。次回、もう少し拡張して、更にアイボさんらしさのプログラムを作っていきましょう。

7時限目

気分屋アイボさん を チャレンジ！

● アイボさんは、「ことば」も理解します

６時限目では、乱数、調べる、制御、移動ブロックなどを使って、アイボさんに晩ご飯を選んでもらうプログラムをチャレンジしました。いかがでしたでしょうか？　アイボさんが選んでもらったあと「ありがとう」って言いたくなります。前回「アイボさんらしさ」として「頭を撫でられた」を追加、アイボさんの頭を撫でたら晩ご飯を選んでくれる！にするとアイボさんとの距離が近くなります。

アイボさんと過ごしていると、触れるだけではなく、声をかけることも多いですよね。

Facebook「aibo デベロッパーグループ」に「aibo 連携アプリ」チュートリアルがアップされており、連携アプリのサンプル紹介として、「気分屋 aibo」のアプリ連携プログラムが紹介されています。

なんか難しいことがいっぱい書かれていますが、内容とイラストを見ると

開発内容:
aibo が検知した言葉を外部に通知する仕組み（Events API）を使い、aibo が「おはよう」と言われたときに、「aibo の気分に応じて」、「とっても喜ぶ」、「顔を洗う仕草をする」、「ブルッと震える」のいずれかのふるまいを実行させます。（Action API）

① aibo に「おはよう」と言うと
② aibo の気分に応じてさまざなふるまいを実行してくれる

ん？　②は、どこかで見たことのあるフレーズですね。そうです！４時限目と５時限目で似たようなプログラミングを作りましたね。

① の「おはよう」と言うと・・ん～、新しい難関ですね。

前回６時限目では、「頭を撫でたら」を追加しただけで、アイボさんとの親密な関係ができました。今回は更に「声」を使って同じようなことをしてみましょう。

ビジュアルプログラミングはアプリ連携まではできませんが、いろいろなプログラム言語があることを交えて、今回ちょっとだけ難しい話も説明しますね。身の回りを見ていただくとパソコン、スマホだけではなく、テレビやリモコンまで、これらの機器はコンピュータやマイクロコンピュータが搭載されているものばかりです。これらの制御にはいろいろな言語で実際につくられています。ぜひ、いろいろと調べてみると面白いかと思います。

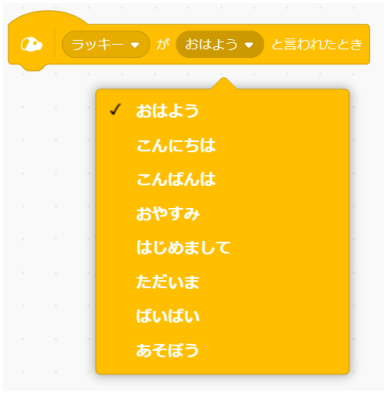
まずは、テーマに沿って作っていきます。がんばってみましょう！

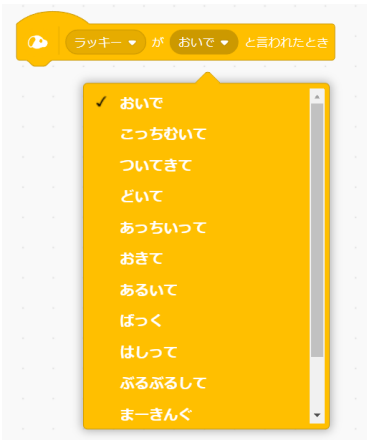
● 「ことば」を認識するブロックをさがそう！

さて、【「おはよう」と言うと】・・・を調べるブロックを探してみましょう！

ブロックを探すのは、皆さんは、お得意！になったかと思います。

「aibo のイベント」にありますね。
他にどんなことばを理解してくれるか見てみましょう





あまりも多すぎて、どれもやってみたくなり迷ってしまいそうです。今回は、テーマとなっている「おはよう」を使いましょう。是非、ほかのことばも試してみてくださいね。

この【aibo のイベント】ブロックと【ふるまい】ブロックの中の【とっても喜ぶ】、【顔を洗う仕草をする】、【ブルッと震える】を根気よく探して、これまで4時限目～6時限目のおさらいをしながら、プログラムを作ってみましょう。

３つの仕草から１つを選ぶなので、３つの乱数から選ぶようにしてみましょう！。こんな感じになったかと思います。６時限目で説明しましたが、プログラムは、その人によって作り方がさまざまです。正解はありませんので、いろいろと工夫してみてください。

⇒ おはようと言われたときに開始
⇒ 確認できたので１回吠える
⇒ 気分を乱数１～３できめる
⇒ 乱数の値が１のとき
⇒ とっても喜ぶをやってもらう

⇒ 乱数の値が２のとき
⇒ 顔を洗うしぐさをやってもらう
⇒ 乱数の値が３のとき
⇒ ブルッと震えるをやってもらう

どうでしょうか？　アイボさんに「おはよう」と声をかけてあげると、どれかやってくれましたか？

なんかうまくいかない？　なぜでしょう？

２時限目で説明した**「アイボさんはきまま」**なのです。
このため、プログラムを実行したとき、気分屋さんなので、うまく声を聞いてくれません。そのため２時限目で説明の「指示待ち」をして待ってもらうにしましょう。指示待ちをクリックすると黄色の枠となり、しばらくすると「完了」と表示されます。

この状態で、「おはよう」と言ってみてください。ブロック全体が黄色の枠で表示されたら、３つの「ふるまい」のいずれかをやってくれるかと思います。

先ほど「指示待ちの状態」にしましたので、確認が終了したら解除してあげてくださいね。いつまでも待ち状態のままになってしまいます。

どうでしょうか？　皆さんはすでに乱数、制御を繰り返し使ってきましたので、今回のテーマは比較的簡単にできたかと思います。
「aibo 連携アプリ」チュートリアルだと複雑な内容ですね。
チュートリアルをチラッと見ると、ここでは乱数が0～2と記載されていますが、同じような内容が書かれていますね。（ん！するどい、乱数は0も使えるのです！ 乱数を0にした場合、更にマイナスの値も使えます。次回ちょっと使ってみましょう）

`CloudFunctions/main.py`

```
# ..(省略)..

def aibo_app(access_token, device_id, eventId):
    # 「おはよう」と言われたとき
    if eventId == 'voice_command::goodmorning':
        # aiboの気分を0から2の整数でランダムに設定
        kibun = random.randint(0, 2)

        # 気分に応じてふるまいを実行
        if kibun == 0:
            print("とっても喜ぶ")
            res = aibo_api_ctrl.aibo_control_sync(access_token, device_id, 'play_motion', '{"Category":"overJoyed","Mode":"NONE"}') # と
        elif kibun == 1:
            print("顔を洗う仕草をする")
            res = aibo_api_ctrl.aibo_control_sync(access_token, device_id, 'play_motion', '{"Category":"washFace","Mode":"NONE"}') # 顔を
        else:
            print("ブルッと震える")
            res = aibo_api_ctrl.aibo_control_sync(access_token, device_id, 'play_motion', '{"Category":"jiggle","Mode":"NONE"}') # ブルッ
        return res
    else:
        return True
```

実際にaiboにAPI実行を指示している処理について解説します。

- `aibo_app`という関数の引数`eventId`に、aiboが検知したイベントの内容が伝えられます。
- 上記の`eventId`を見て、「おはよう」と言われた場合にだけ、aiboに指示を出すようにします。
- aiboの気分は、0から2の整数でランダムに設定します。このランダムな数字に応じて、実行する内容を決めます。
- aiboに指示する内容は、`aibo_control_sync`という関数で指定します。
- aiboに指示する内容を上記の関数に渡すと、関数内部でAction APIが呼ばれaiboがふるまいを見せてくれます。

プログラム内容の赤字の部分をちょっとだけ見てみると、

```
If      kibun==0・・・
elseif  kibun==1・・・
else    ・・・
```

となっています。前回プログラムは作る人によってさまざまな方法があることを説明しました。上記に合わせてビジュアルプログラミングで作ってみた場合、次の通りとなります。

あれ？　この並びはどこかで・・・
そうです！　4時限目に番外編で説明したブロックの並びです。
上記 kibun==2 の条件がありませんね、
左の例も 変数=3 の条件がありません。

プログラムにはいろいろな作り方、また、たくさんの言語があります。

このテキストは「ビジュアルプログラミング」上でブロックを並べて作っていくものですが、ほかにも同じようにプログラミングする言語と呼ばれているものが多数あります。これらは、コンピュータの進化、時代ともに、処理、制御、データ管理、AI 処理などに適したものに進化してきました。ぜひ、いろいろなプログラム言語に触れてみてくださいね。

プログラム言語の例

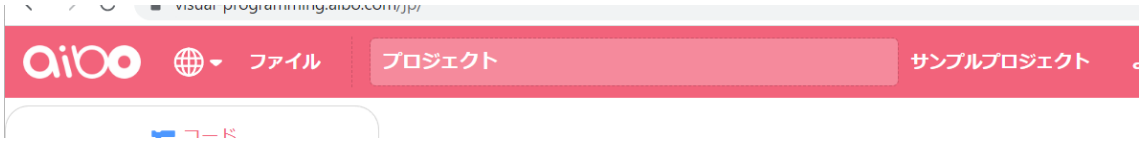
BASIC、C、C++、C#、PHP、Java、JavaScript、Python、Perl、Go、Scratch、Kotlin、Swift、R、Fortran、COBOL、PL/1、LISP、アセンブラ・・などなど

● プログラムを毎回作るのは、とっても たいへ～ん

さて、４，５，６時限目、そして今回もプログラムを１から作ってきましたが、毎回、毎回、作っていくのって、とっても大変ですよね。更に複雑になると、ブロックを探して探してくっつけて・・・となります。

ビジュアルプログラミングでは、作ったプログラムを保存することができます。途中でやめてあとで続きをつくることもできますので、ちょっとだけ大変ですが、ぜひ覚えてくださいね。

先頭に次のような「ファイル」とうメニューの項目があります。



「ファイル」メニューをクリックすると、次のメニューが表示されます。

「新規」
「プロジェクトを読み込む」
「プロジェクトを保存する」

ビジュアルプログラミングでは、作ったプログラムを実行しているパソコンやタブレットまたはスマートフォン上に保存ができます。また、保存したものを読み込んで、同じように実行したり、また、作り変えたりすることもできます。

保存や読み込みの動作は、ご利用のブラウザによってさまざまなですが、ここでは、Chrome ブラウザ、Microsoft Edge の場合を参考に記載します。

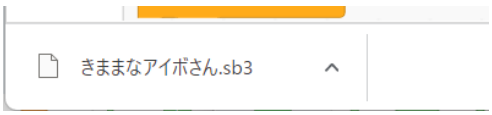
プログラムはファイルとして保存されますので、ファイル名を決定してください。今回は、「きままなアイボさん」として保存してみます。

　ファイルメニューの横にファイル名を入力できるテキストエリアがありますので、そこに保存するファイル名を入力してください。できるかぎりわかりやすい名前にしておきましょうね。

次に「ファイル」メニューから「プロジェクトを保存する」をクリックしてください。

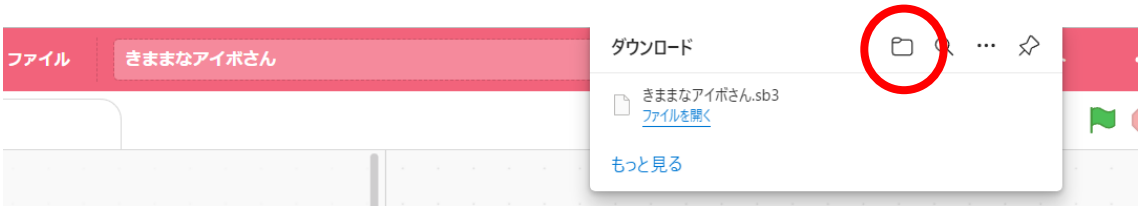
■Chrome ブラウザ

「プロジェクトを保存する」を実行するとファイルが作成され、画面左下に次のような表示となり、パソコンにファイルとして保存されます。開くを選択すると保存されたフォルダが表示されます。

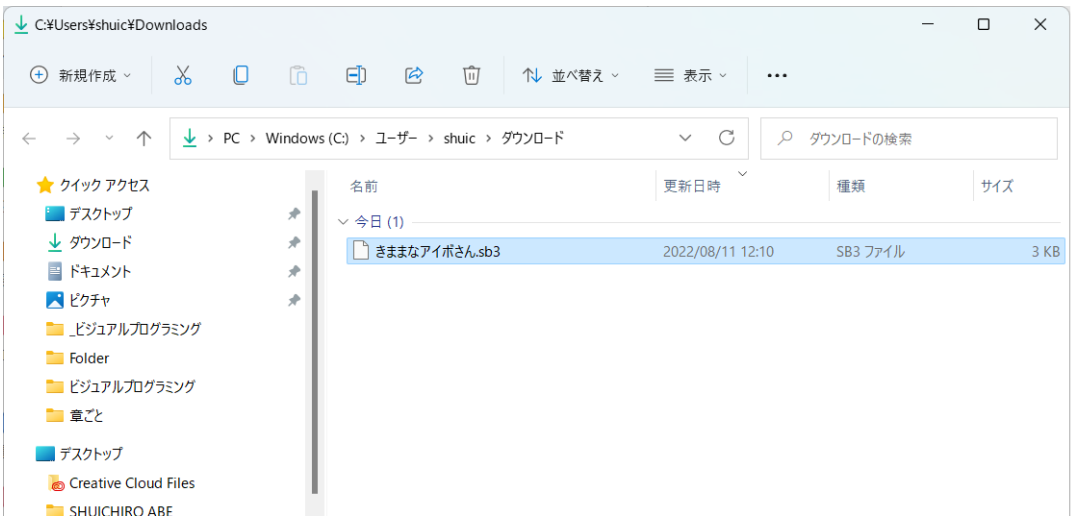


■Microsoft Edge

「プロジェクトを保存する」を実行するとファイルが作成され、画面右上に次のような表示となり、パソコンにファイルとして保存されます。フォルダのアイコン（赤丸部分）を選択すると保存されたフォルダが表示されます。

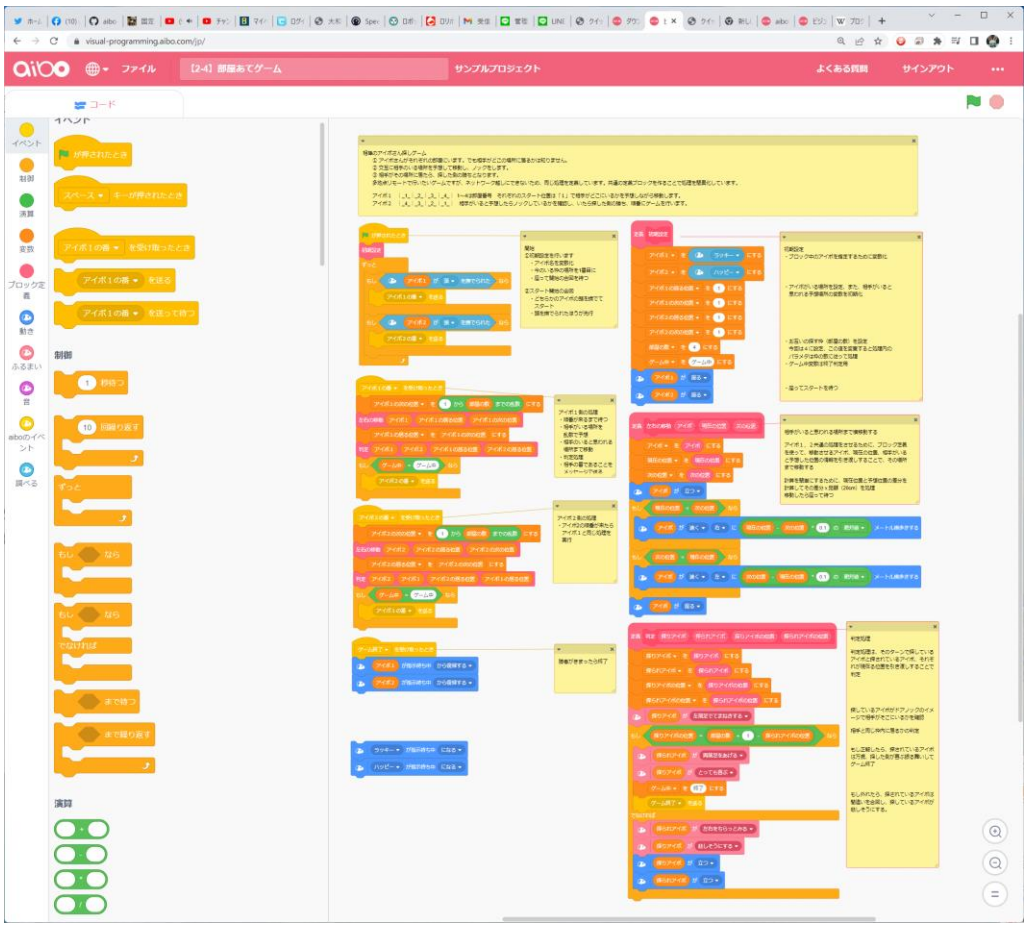


「フォルダを開く」を選択すると、ダウンロードのフォルダに保存されますので、適宜管理したい場所に移動しておいてください。



■保存したプログラムファイルの読み込み
保存したファイルを再度読み込みしたい場合、「ファイル」から「プロジェクトを読み込む」を操作すると、フォルダが表示されますので、保存したフォルダ先を指定して、読み込みをしてください。

参考までに、ビジュアルプログラミングコンテスト用で作ったファイルです。このように、複雑になっていくとファイル保存しておかないと大変なことになります。



7時限目は、ここまでにしましょう

おさらいしましょう。

- アイボさんは、「ことば」を理解してくれます
- ことばを認識するブロック「aibo のイベント」があります。
- ことばの種類はびっくりするほどたくさんあります。
- アイボさんはきまま「指示待ち」で待ってもらいましょう。

■豆知識

プログラム言語にはたくさんの種類があり、目的にあわせたもの、時代とともに変化していったものなど、いろいろあります。
SONY さんもデベロッパープログラムのチュートリアルとして紹介、複雑ですが、ほかのプログラム言語を使って作ることもできます。

■ビジュアルプログラミングの便利な機能

プログラムはファイルとして保存することができます。毎回毎回の作成は大変なので、是非、プログラムの保存、読み込みを上手に使っていきましょう。

今回、アイボさんとことばをつかった連携ができましたね。アイデアを膨らせることで、いろいろと応用できるかと思います。ぜひ試してみてください。

８時限目

アイボさんとゲームしたいですね！

● アイボさんといっしょにゲームしたいね

７時限目は、プログラム言語について話をしましたが、ちょっとむずかしかったですよね。７時限目に作った「おはよう」といったら、気分に合わせて動作してくれるプログラムはどうでしたか？プログラムで指示通りに動いているのに、ことばでつながるとちょっと楽しくなりますよね。また、お友達のアイボさんやオーナさんともいろんなことをしたくなります。

そこで、８時限目は、ちょっとしたゲームプログラムを考えたいと思います。といっても、むずかしくなると凹みますし、覚えることが多いとお腹いっぱいでこれ以上は・・・なんてことになります。

そこで、これまで作ってきたものをおさらいしながら、ゲームのようなものにしてみたいと思います。これまで作ってきたプログラムで使ったブロックをおさらいしてみましょう。こんなブロックを使ってきましたが、覚えていますか？

え！？　こんなに使ってきましたっけ？　ハイ使っていたのです！

● あっちむいてほぃ！のゲームを作ってみましょう

これまで覚えたブロックを使いながら、ゲーム「あっちむいてほぃ！」のゲームを作ってみたいと思います。

■テーマ：あっちむいてほぃ！を作ってみよう

はじめに・・「あっち向いてホイ」はジャンケンで勝ったほうが相手に向かって「あっち向いてホイ」と言って指を上下左右に向けます。ジャンケンで負けたほうが勝った人と同じ方向に顔を向いてしまったら負け・・・です。

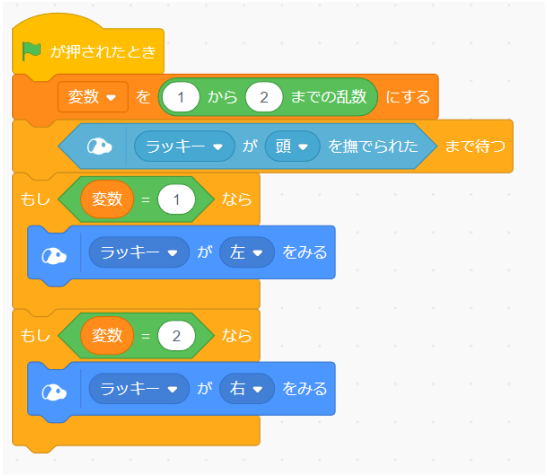
あっ、もちろん知っていますよね！　これをプログラムにする場合、とうぜん、この内容と同じことをどのようにさせようかを考えて作るわけです。ちょっとだけ、上の文書を分解してみましょう。

① じゃんけんをする
② ジャンケンで勝ったほうが相手に向かって
③ 「あっち向いてホイ」と言って指を上下左右に向けて
④ ジャンケンで負けたほうが、同じ方向に顔を向いたら負け

さて、このままはちょっとむずかしそうです。①、②は無理ですよね～、だってアイボさんの肉球ではグーチョキパーができない！
そこで、簡単にするために次のような内容にしてみましょう。

① アイボさんがあらかじめ左右どっちを向くかを決める
② 「あっち向いてホイ」と言って頭を撫でてあげる
③ オーナさんは指で左・右を指しましょう。ここは演技です。
④ アイボさんは頭を撫でられたら①で決めた左・右を向く

早速、作ってみましょう。いろいろな作り方がありますが、これまで覚えてきたブロックなどを組み合わせると、こんな感じですね。
左・右は乱数１と２で決めてみましょう。左・右に顔を向く動作については【動き】の青のブロックから探してみましょう。



⇒ プログラムの開始
⇒【変数】を乱数１～２に設定する
⇒ 頭を撫でられるまで待つ
⇒【変数】が１ならば
⇒ 左 を向く
⇒【変数】が２ならば
⇒ 右 を向く

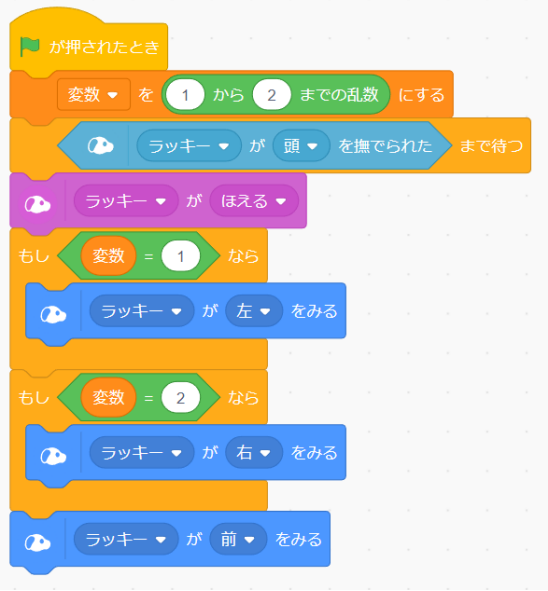
うまくいかなかったら指示待ちもしてね

あれ？　いつの間にか簡単なゲームができちゃいましたねぇ～

プログラムを作るときに大事なのは、むずかしく考えるのではなく、まずはシンプルにやってみたいことを整理して、プログラムの流れを考えていくのが大切です。

上記のプログラムだと、いくつか気になる点も出てきます。
① 頭を撫でられたことが分かりにくい
② 最後、左・右どちらかに向いたままになってしまう。

これらの問題をちょっと改良してみましょう。
上記①は 頭を撫でられたら、ほえて みましょう
上記②は 元の前に向く動作を入れてみましょう



⇒ プログラムの開始
⇒【変数】を乱数１～２に設定する
⇒ 頭を撫でられるまで待つ
⇒ 撫でられたので　ほえて知らせる
⇒【変数】が１ならば
⇒ 左 を向く
⇒【変数】が２ならば
⇒ 右 を向く

⇒ 左・右を向いたので前向きに戻る

どうでしょうか？　細かいですが、このような後処理の配慮もプログラムには必要となってきます。

これまで使ってきたブロックが異なりますが、今まで学んできたこととあまり変わらない感じですねぇ～　そうです！　その通り！
同じようなものでも、ちょっとだけ変えることでゲームのようなものになったり、いろいろと応用ができます。

これは SONY さん提供のビジュアルプログラミングで実行できるイロイロな【ふるまい】や【動き】の種類や動作が多いから、同じような流れでも、使う動作の組み合わせを変えることで、まったく異なったものにすることができますね。感謝！感謝！

でも、もう少しアイボさんとゲームしている感じにしてみたいですね～　次回は、アイボさんに勝ち？負け？の判断をしてもらう処理を追加してトライしてみましょう。

８時限目は、ここまでにしましょう

おさらいしましょう。

- アイボさんとゲームしたいよね～
- 「あっち向いてホイ」をやってみましょ
- これまでたくさんのブロックを使ってみました
- アイボさんの動き、ふるまいが沢山ありますね
- 同じプログラムでもちょっと工夫すればゲームにもなる

アイボさんといろいろと遊んでみたいですよね。今回はシンプルな「あっち向いてホイ」をつくってみました。プログラムはこれまでの学んでいきた組み合わせで簡単にできますね。

次回、アイボさんに勝った？、負けた？を教えてあげたいと思います。予習として次回のプログラムを載せておきますので、よかったら作って試してみてくださいね。

また、ブロックがだんだん多くなりますので、前回紹介の保存機能をうまく活用してください。

■予習

次回はアイボさんに勝った？負けた？を教えてあげたいと思います。次回のプログラムを先に載せておきます。

⇒ プログラムの開始

⇒【変数】を乱数１～２に設定する

⇒ 頭を撫でられるまで待つ

⇒ 撫でられたので　ほえて知らせる

⇒【変数】が１ならば

⇒ 左 を向く

⇒【変数】が２ならば

⇒ 右 を向く

今回はここまででした。次回は・・勝ち負けを教えてあげるために、アイボさんのかわいい肉球を使って、オーナさんが「あっちむいてホイ」の時、右・左どちらを指したか？　を 肉球で教えてあげます。

アイボさんが向いた方向と同じ向きだったかを判断して・・・・

次回もがんばりましょう・・・

９時限目

アイボとゲームしたいですね！その２

● **アイボさんとのゲームができたら楽しいですよね**

アイボさんといろいろとゲームができたら楽しいですよね～
８時限目の「あっちむいてほぃ！」はいかがでしたでしょうか？
これまで使ったブロックの組み合わせで簡単なゲームができましたね。でも、もう少しゲームの感覚を体感したいですね。

● **オーナさんはどちらの向きだったかを教えてあげましょう！**

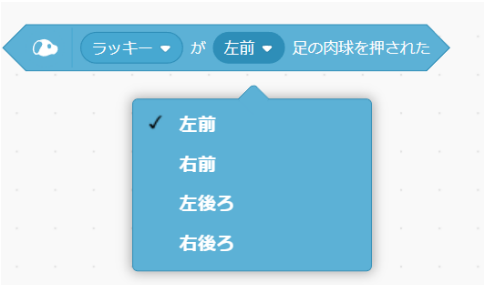
８時限目のプログラムだとアイボさんに「あっちむいてほぃ！」といってあと、左右どちらかの方向を向いただけです。つまり、アイボさんは自分が勝ったのか？負けたのか？知らないままです。

そこで勝ったか？負けたか？をアイボさんに教えてあげましょう！
どうやったらよいでしょうか、考えてみましょう。

・・・・あ～、もうチンプンカンプン・・・

ですよね～。だって、どうやって教えてあげるのか、【調べる】ブロック群を探しても、そんなものなかったし・・・・・

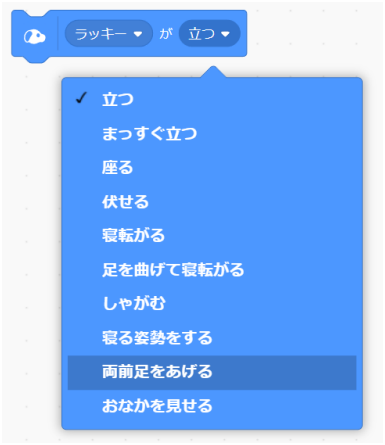
何かないかなぁとみてみると！？
【調べる】ブロックに**【足の肉球がおされた】**って、とってもアイボさんらしいものがありますねぇ～



これを今回使ってみたいと思います。そこで、以下のような流れを考えてみます。

① 「あっち向いてホイ」後、オーナさんの向きを教えてあげる
② 肉球で教えてあげるため、アイボさんに前両足を上げてもらう
③ オーナさんが選んだ左・右の前足の肉球を押して、どっちを選んだか教えてあげる
④ アイボさんが選んだ方向とオーナさんの選んだ向きが異なったらアイボさんの勝ち、同じ向きだったらアイボさんの負け
⑤ 勝ち負けの判断をして、勝ったら喜ぶ、負けたら悲しむ

さて、いきなりむずかしい流れとなりましたが、いくつかブロックを探していく必要がありそうですね。



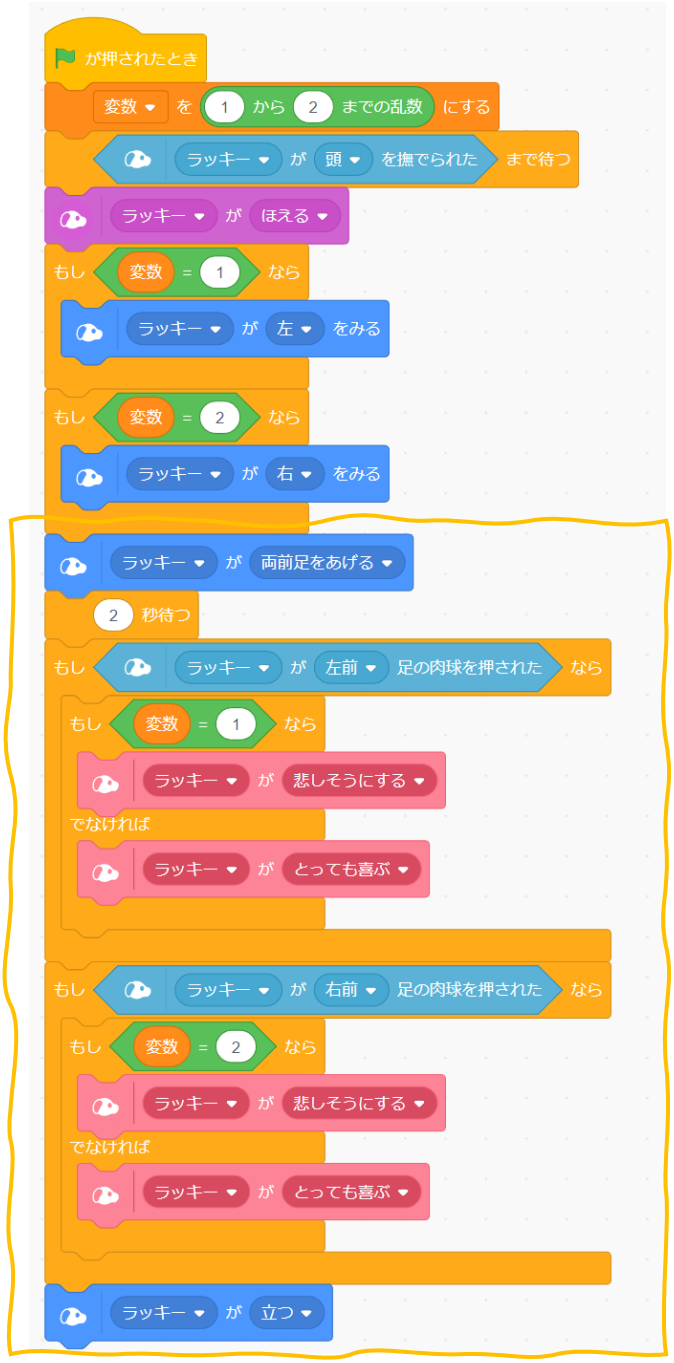
【前両足を上げる】ブロックは、「動き」ブロック群のなかにあります。
これを使いましょう。

そして、肉球が押されたかを確認するために、先ほどの【調べる】ブロックを使ってみましょう。

勝ち、負けを判断したあとの動作のブロックは、以下のものを使ってみましょう。

さて、だんだん、むずかしさが倍増してきましたが、ひとつひとつ確認をしながら組み立てていきましょう。では、さっそく・・・

まずはプログラム全体を見てみましょう・・・



⇒ プログラムの開始
⇒【変数】を乱数１～２に設定する
⇒ 頭を撫でられるまで待つ
⇒ 撫でられたので　ほえて知らせる
⇒【変数】が１ならば
⇒ 左 を向く
⇒【変数】が２ならば
⇒ 右 を向く　←前回はここまで

勝ち負けを教えてあげるために、アイボさんのかわいい肉球を使って、オーナさんが「あっちむいてホイ」の時、右・左どちらを指したか？　を 肉球で教えてあげます。
アイボさんが向いた方向と同じ向きだったかを判断して、同じだったら、アイボさんが負けなので「悲しそうに」、違った向きだったら「とっても喜ぶ」をしてもらいます。

それでは、順番に確認していきましょう・・

(1) オーナさんが左右どっちの向きを示したか教えてあげましょう

アイボさんに左右どっちの向きかを前足の肉球で教えるため、まずは、両前足を上げてもらいます。

そのあとの【2秒待つ】ブロック（制御ブロックに【1秒待つ】がありますので1秒を2秒に変更）は、今はおまじないだと思ってください。この後に肉球が押されたかの確認をしますが、両前足を上げた後、左前足が押されたかの確認がすぐに始まってしまいます。このため、次の【もし～でなければ～】ブロックがはじまってしまう前に、ちょっとだけ待ち時間を設定します。

両前足をあげたら、の2秒待ちに肉球を押し始めましょう
この後確認に時間が少しかかるので、5秒以上押してあげましょう

次回、【2秒待つ】ではなく、押されるまで待つ方法を説明します。今回、まずは「おまじない」ということで。

(2) 次に肉球が押されたかのチェックをしましょう

【左前足の肉球のチェック】

⇒左前肉球が押されたかのチェック
⇒変数=1 アイボさんは左向きですね。
オーナさんが左前足を押した場合、「同じ向き」となりアイボさん負け。そうでなければ、アイボさん勝ちとなります。

【右前足の肉球のチェック】

⇒右前肉球が押されたかのチェック
⇒変数=2 アイボさんは右向きですね。
オーナさんが右前足を押した場合、「同じ向き」となりアイボさん負け。そうでなければ、アイボさん勝ちとなります。

(3) 最後に元の体制に戻ってもらいましょう

これで元の体制にもどります。

●ゲームの流れをおさらいしてみましょう
作ったプログラムを思いだししながら、ゲームのやり方をおさらいしてみましょう。

(1) プログラムをスタートします。
(2) 乱数１～２を変数に設定（今回値１は左、２は右とします）
(3) アイボさんの頭を撫でられたらゲームスタート
(4) スタートの合図として、アイボさんが　ほえます
(5) オーナさんは「あっち向いてホイ」と言って左右を指してね。
(5) 乱数１、２によって、アイボさんは右左に頭の向きをかえます
アイボさんは、オーナさんの指の向きがどっちか、わかりません。そこで・・・
(6) 両前足を上げます。左右どっちを指したか教えてあげます
(7) オーナさんの向きの肉球を長めに押しましょう（５秒ほど）
(8) 同じ向きだったら、アイボさんの負けなので悲しそうにします
(9) 違う向きだったら、アイボさんの勝ちなので喜びます

９時限目は、ここまでにしましょう

おさらいしましょう。

- ８時限目に「あっち向いてホイ」のゲームをつくりました
- 今回オーナさんの左右の向きをアイボさんに教えてあげます
- オーナさんアイボさんと同じ向きだと、アイボさんの負け
- オーナさんアイボさんと違う向きだと、アイボさんの勝ち
- 負けた時、勝った時のそれぞれの動作を行う

シンプルな「あっち向いてホイ」が、今回アイボさんとゲームをして勝敗も確認できるようになったかと思います。オーナさんの指の向きも見て判断できるようにできれば良いのですが、今のアイボさんではむずかしいので、まずは、オーナさんがどちらを選んだかを教えてあげることで、アイボさんが判断できるようにしました。

また、アイボさんの肉球が押されたかを判断するタイミングが分かりませんので、何度か繰り返しして、反応するタイミングをみてください。遊びながら、コツをつかんでくださいね。

さて、気づいたかもしれませんが、何度かやっていると**何もしてくれない**ことがあるかと思います。次回、何故そのようなこと起きているのか、対処方法はあるのかを見てきたいと思います。

１０時限目

アイボとゲームしたいですね！その３

● **動作を確認するって大変ですね～**

アイボさんとの「あっちむいてほぃ！」で勝敗確認はいかがでしたでしょうか？　少しはゲームらしく楽しめましたか？

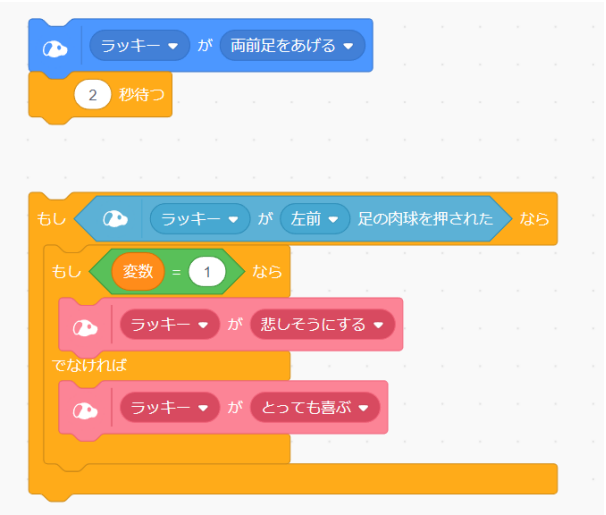
最初は、アイボさんの肉球が押されたかの判断するタイミングが分からなかったかと思いますが、何度か繰り返ししているうち、だんだんアイボさんが確認するタイミングがつかんでくるかと思います。更に、いろいろな動作や【調べる】ブロックを使いながら、それぞれのコツ（タイミング）をぜひつかんでくださいね。

ところで、何度か繰り返しているときに、**何もしてくれなかった**ことありませんか？　これは、
前足の肉球の押すタイミングが合わなかったり、また、短かかったとき、この状況になっているかと思います。

前回のプログラムで**【２秒待つ】のおまじない**を入れた説明をしました。

ためしに「肉球を押さない」ようにしてみましょう。すると、何もしないで元の状態に戻っているかと思います。なぜでしょう。

もう一度流れを見てみましょう。



⇒アイボさんが両前足を上げる

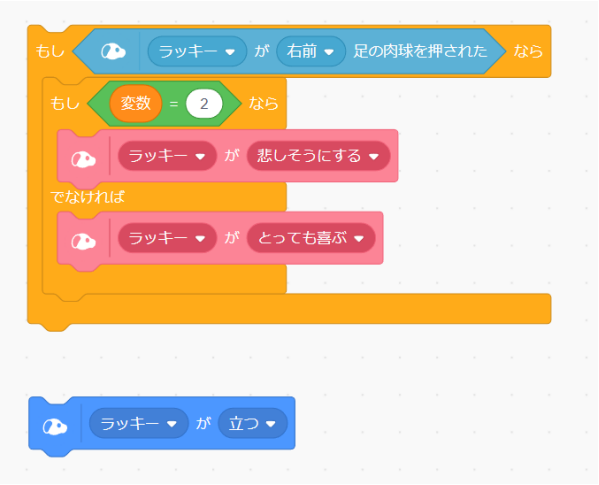
⇒2秒待ちますが・・・

ここで肉球を押さなかった場合

⇒ 右前足の肉球チェック

肉球が押されていないので

ここの処理はされませんね・・



つづいて

⇒ 左前足の肉球チェック

肉球が押されていないので

ここも処理はされませんね・・

そのまま【立って】もとに戻る。

つまり、肉球を押すための待ち時間として【2秒】をセットしたのに、押さなかった場合、または、押したのに確認ができなかった場合に、後の処理の「前足の肉球が押されたかのチェック」が素通りされてしまったために、何もしてくれない状況となります。

どうしたらよいでしょうか？

考えてみましょう～

●肉球がおされるまで待つことができれば・・・

課題

① 両足を上げた後、すぐに次の処理に進むので２秒待ち入れた

② 肉球を押さなかった場合、判断できず何もしない状況となる

③ 改良点として、肉球が押されたかを待つ必要がある

ちょっとむずかしくなりましたね～、確認する方法はいろいろとあるかと思いますが、いくつかやり方を考えてみたいと思います。

■確認する方法のパタン

【パタン１】

肉球が押されているかを確認し続け、右または左が押されたら、今回新しく作る変数（オーナさんが左右どちらを指したかを保存する）に値を入れます。値が入ったことが確認できるまで次の処理に移らないようにします。

【パタン２】

肉球が押されるまで、とにかく待つ。こちらのほうが楽そうですが、学習のため、今回はパタン１を考えます。なお、パタン２は 応用編で、別な作り方と合わせて説明します

●新しい変数をもう一つ追加してみましょう

ここで新しいことを１つ覚えましょう。これまで【変数】は値を保存するための魔法の箱として使ってきました。でも、プログラムでは、たくさんの情報を保存するため、覚える種類数の分、変数の箱が必要となります。新しい変数を作る方法をぜひ覚えてください。

（１）変数のブロックから「変数を作る」を押してください

初期状態では、変数はそのまま【変数】と名前がついている、魔法の箱が１つだけです。

（２）変数のブロックから「変数を作る」を押してください

① 「新しい変数」の入力画面が出ます。
② 分かりやすい変数を作ります。
③ 今回「オーナの左右の向き」としました。自由に決めてくださいね。

（３）変数のブロックに新しく追加されました。

もともとあった【変数】に新たに【オーナの左右の向き】という別の変数を追加しました。ブロックの部分をクリックすると【変数】と今回作ったもの、どちらかが選択できるようになります。

いままで使ってきた【変数】と呼ばれる箱
アイボさん側の乱数の値を入れてきました

今回【オーナの左右の向き】の変数を追加して、オーナの左右方向を値（１または２）を保存します。

【パタン１】の流れを少し考えてみましょう。

肉球が押されているかを確認し、押されたらオーナさんの示した方向の値を保存。確認できるまで次の処理に移らない

次の処理に移らない様にするには、【制御】ブロックでできそうですね。更にいくつかやり方がありそうです。

ここで変数に保存する値のルールを作りたいと思います。

① アイボさんの左右は乱数をつかって１か２で決めました。
　１の時は【左】、左の方向を向くようにしました。
　２の時は【右】、右の方向を向くようにしました。

② オーナさんの左右方向も同じように値(ルール)を決めます。
　オーナさんが左向きを指定：変数【オーナの左右の向き】の値を「１」とします。
　オーナさんが右向きを指定：変数【オーナの左右の向き】の値を「２」とします。つまりアイボさんと同じ２の値

【サンプル１】　どちらか押されるまで繰り返す
繰り返しを抜けるのは、値が１か２が設定された時

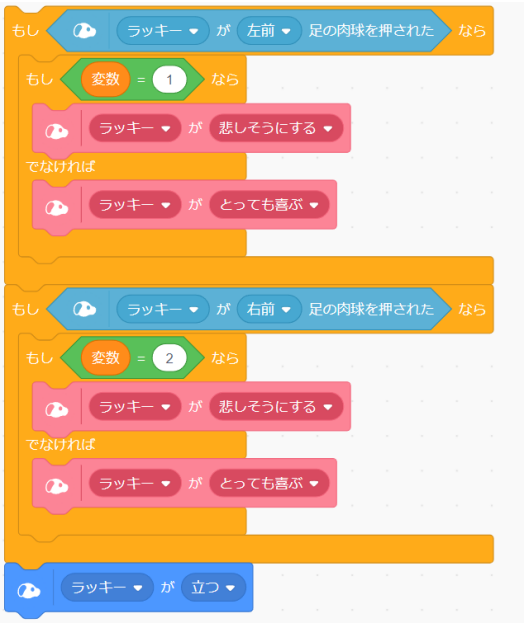
⇒変数【オーナの左右の向き】を0
⇒【オーナの左右の向き】が１か２にならない間は処理をし続ける
① 左前足の肉球が押されたとき
【オーナの左右の向き】＝1
② 右前足の肉球が押されたとき
【オーナの左右の向き】=2

【サンプル２】　左右どちらか押されるまで繰り返す

サンプル２は、左前足か右前足のどちらかが押されるまで、待つようにしています。厳密には、流れのとおり肉球チェックが２回行われますので、正しい方法ではありませんが数秒間肉球を押すようにしていますので、この内容でも同じように処理できます。

今回はサンプル１を使っていきましょう。なお、同じ場所にサンプル２を置き換えても問題ありません。
つぎに、オーナさんの左右の方向を保存している変数【オーナの左右の向き】に左右どちらかが押されたかの値　１(左)または２(右)が保存されましたので、以降の処理も変更したいと思います。

以降の処理、前回作ったプログラムを見直ししてみましょう。



⇒ここはチェック済となるので不要ですね

⇒今回、新たに【オーナの左右の向き】の変数を作りましたので、アイボさんの向きが保存されている【変数】と比較するだけでチェックできそうですね。

⇒ここもチェック済となるので不要ですね

⇒こちらも同様です。

作り変えてみたものがこちらとなります。

ん？　これだけ？？　　検証してみましょう。

⇒アイボさんとオーナの向きをチェック

アイボさん左(1)：オーナ左(1)アイボさん負け

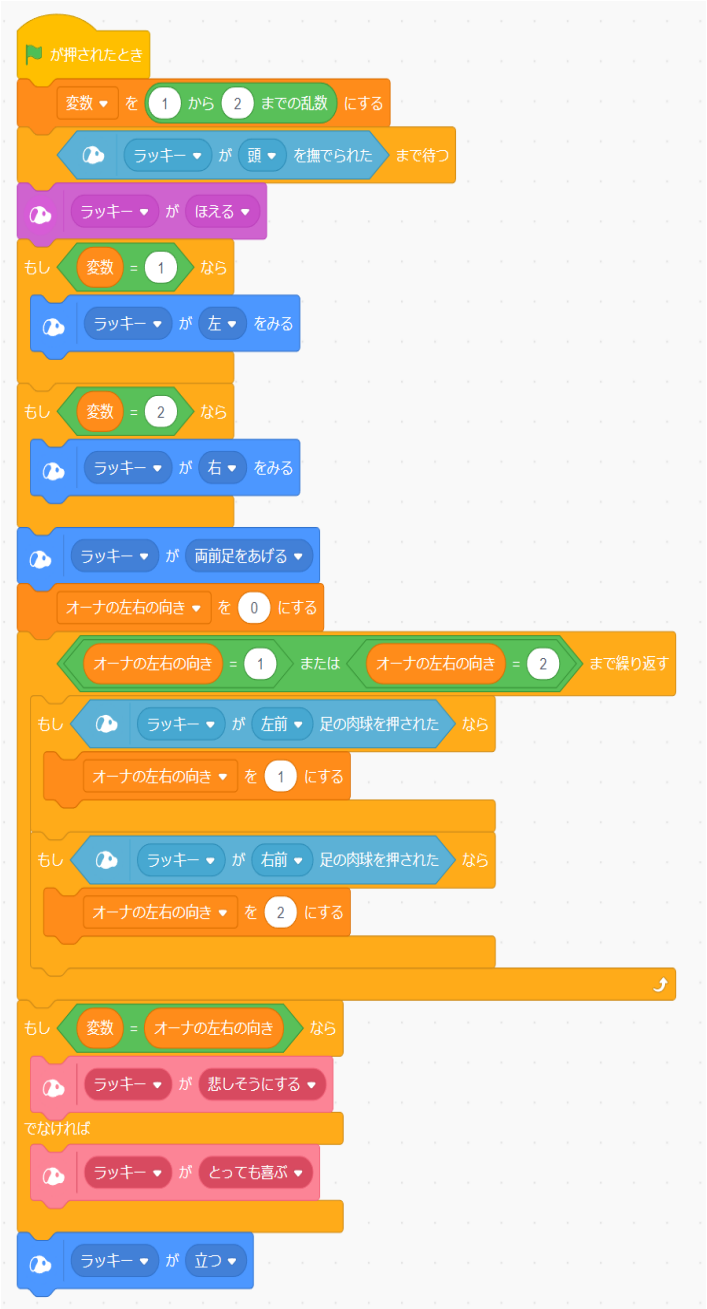
アイボさん左(1)：オーナ右(2)アイボさん勝ち

アイボさん右(2)：オーナ左(1)アイボさん勝ち

アイボさん右(2)：オーナ右(2)アイボさん負け

これだけで、OKのようですね・・・

全体を組み立ててみたいと思います。



プログラムの開始

⇒【変数】を乱数１～２に設定する

⇒ 頭を撫でられるまで待つ

⇒ 撫でられたので　ほえて知らせる

⇒【変数】が１ならば

⇒ 左 を向く

⇒【変数】が２ならば

⇒ 右 を向く

⇒オーナさんに教えてもらいます

⇒新しい変数【オーナの左右の向き】を０にして、１か２にならない間は繰り返す

⇒左足の肉球が押されたら【オーナの左右の向き】の値を１（左）に設定

⇒右足の肉球が押されたら【オーナの左右の向き】の値を２（右）に設定

アイボさんが向いている方向

乱数できめた値　１：左、２：右と

オーナが決めた左右の値

左肉球を押した場合、左：１

右肉球を押した場合、右：２

を比較して 喜ぶか悲しそうにする。

かなり難しくなりましたが、いかがでしょうか？

あせらず、ひとつひとつ 確認しながらやってみましょう

１０時限目はこれで終了です。

おさらいしましょう。

- ９時限目で作成したプログラムでは肉球が押されなかった時の対処が必要
- 肉球が押されるまで、待つように繰り返し処理を追加
- 変数を新たに追加して、アイボさんとオーナさんの向きも新たに保存して、アイボさんとオーナさんの値を比較するように変更
- アイボさんの左右の向き、オーナさんの左右の向きを比較して、同じ向き（値）の場合は「悲しそう」に、また違う向き（値）の場合は「喜んで」もらう

いかがでしたでしょうか？

１０回を通して、ビジュアルプログラミング初心から、簡単なゲームを作るところまで学んできました。まだまだたくさんのブロックがありますので、ぜひ、いろいろと試してみましょう。

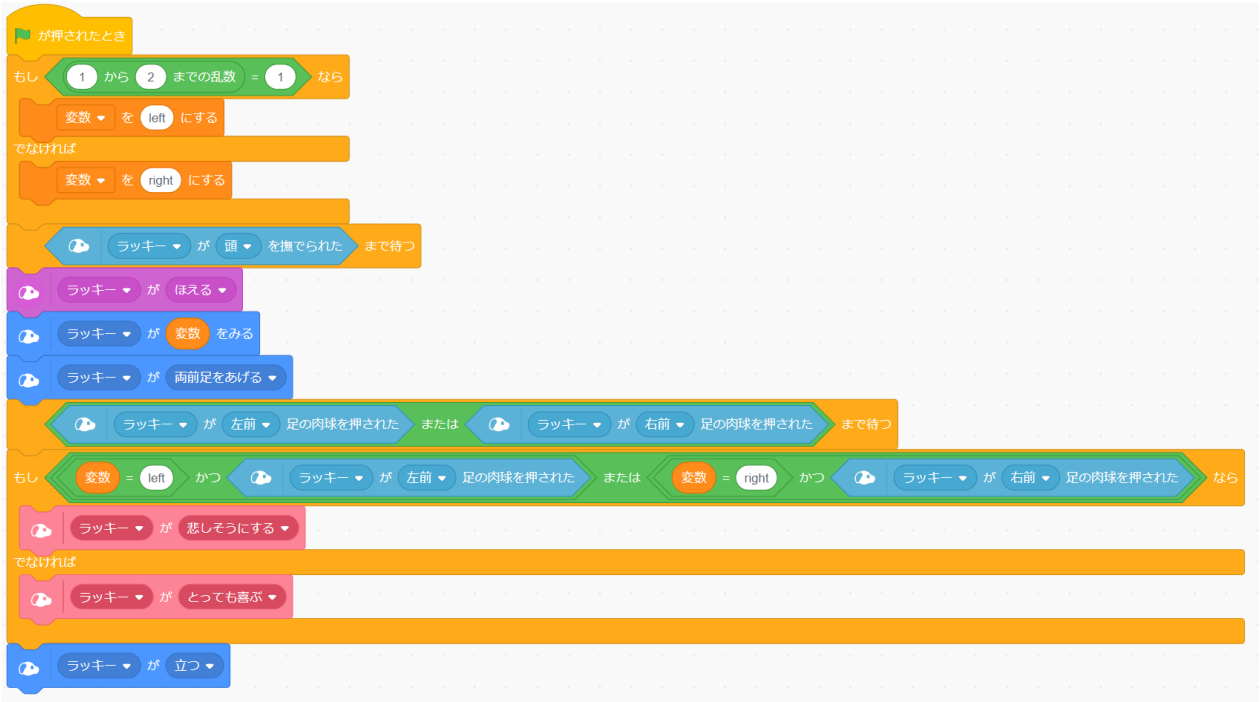
この「aibo（アイボ）さんとビジュアルプログラミング まなんじゃぉ～！」は、パート１として一旦終わりにしましょう。

新たに**「aibo（アイボ）さんとビジュアルプログラミング まなんじゃぉ～！　パート２」**として、更に動きのあるブロック、イベント、メッセージ、リスト、更にブロック定義などを使っていきたいと思います。

応用編

応用編として、別のサンプルを掲載します。

ゲームの内容は変わりませんが、プログラムの作り方が大きく異なっていることがわかるかと思います。



■ちょっと解説

乱数を１，２のうち１であれば
変数=「left」という文字をいれる
１でなければ、つまり２だと
変数=「right」という文字をいれる

変数の値に基づき右、左を向く

① 左右の前足肉球が押されるまで待ちます。

② 【もし～なら】は以下の条件が成立したかのチェックします

［変数=left（乱数１）］ かつ 左前足が押された

または

［変数=right(乱数２)］ かつ 右前足が押された

③ 条件が成立した場合

つまり、アイボさんとオーナさんの指した向きが一緒だったら

アイボさんは同じ向きを向いたため「悲しそうに」をする

④ 条件が成立しなかった場合

つまり、アイボさんとオーナさんの指した向きが違ったら

アイボさんと違う向きだったので「とっても喜ぶ」をする

今回学んだ内容と同じ処理になったかと思います。
かなり、難しくなりましたが、こんな作り方もできるということを
知っていただくだけで問題ありません。

繰り返し、繰り返し、いろいろと作っていくことで慣れていきます
ので、是非、沢山のアイデアでプログラミングしてくださいね。