

Jasmine Tea 基礎-2D ゲームを作ってプログラミングの基礎を学ぼう（生徒用）

1. プログラムの基本構文の学習

変数の基礎知識と使い方

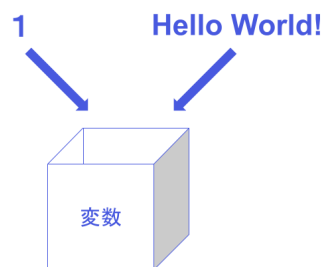
プログラミングを学ぶ上で必ず出てくるのが「**変数**」という言葉です。

変数と聞くと、数学に出てきた「x」や「y」などを思い浮かべる人も多いと思いますが、数学の変数とプログラミングの変数では少し考え方が違うので注意してください。

変数とは

変数を簡単に説明すると、「値を格納しておくための箱」です。

プログラミングの変数は、「1」や「0.5」といった数値はもちろんですが、「こんにちは」や「Hello World!」といった文字列（単語や文章など）など、**数字以外の値**も格納することができます。



プログラミングでは、「値を格納するための箱（変数）」を利用することで、コードが読みやすくなったり、コードの修正が簡単になったりするのですが、実際に触ってみないと分からない部分も多いと思うので実際にコードを書きながら学習していきましょう。

Jasmine Tea を開く

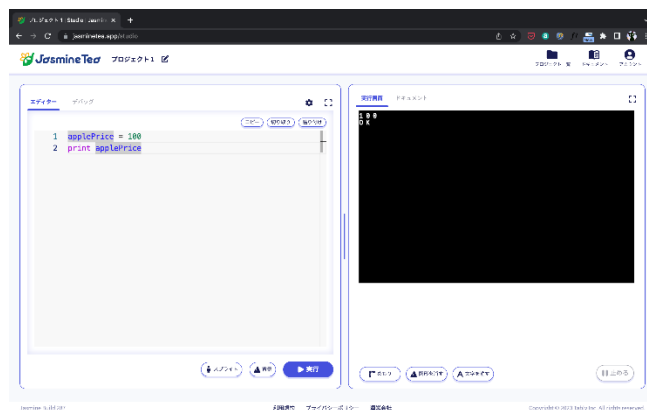
まずはブラウザで Jasmine Tea を開いてください。

Jasmine Tea のトップページが表示された場合は、右上の「ログイン」からプロジェクトページに移動してください。

Jasmine Tea を開いたら、下記のコードを記述し実行してみましょう。

```
applePrice = 100  
print applePrice
```

下の画像のように実行画面に 100 と表示されていれば成功です。



このように、変数に格納した値は取り出して利用することもできます。

プログラミングでは変数に格納した値を取り出すことを**出力**と呼びます。

変数名とは

`applePrice` という変数を使って画面に `100` という数値を出力することができました。

この `applePrice` という変数の名前を「**変数名**」と呼びます。

変数名はその変数にどんな値が入っているのかを表す単語にしてください。

変数の命名規則

変数名は好きなようにつけることができます。

ただし、好きなように変数名を付けることができるからといって、適当な名前をつけるのは良くありません。

例えば、下のコードをみてください。

```
a=12  
b$="桜"
```

変数 `a` に「12」を代入し、変数 `b$`(**文字を入れる変数は\$をつける必要がある**)に「桜」を代入しました。もちろん上記の書き方でもエラーなどは起こらず、問題なくプログラムは動きます。

ただ、上記の変数名だと何の値が代入されているのか見ただけでは分かりませんか？

書いた直後なら「この変数には〇〇の値が代入されている」とすぐに分かるかもしれませんが、1 週間後に自分が書いたコードを見たときにもすぐに思い出せるでしょうか？

また、プログラムは自分以外の人も見る 경우가よくあります。

その時に `a` に代入されている値はなんなのか、`b$`に代入されているテキストはどういう意味なのかを考える必要があります。

そうならないように、変数名は誰が見ても中身がなんなのかが分かるように設定しましょう。

例えば上記のコードを以下に修正します。

```
age=12  
name$="桜"
```

これで「12 は年齢で、桜は名前なんだな」と想像がつくと思います。

コードが少ないときは、まだ変数に代入されている値がなんなのかを理解することは簡単なのですが、コードが 100 行や 1000 行になった時に、いちいちこの変数名に何が入っているのかを確認するのは大変です。

できる限り簡潔に、具体的な情報が入った名前をつけるようにしてください。

また、Jasmine Tea では以下の命名規則があります。

💡 命名規則

- 名前の最初の文字は、アルファベット (a~z または A~Z) または「_」(アンダースコア記号) のいずれかでなければなりません。
- 名前の 2 文字目以降は、アルファベット (a~z または A~Z)、数字 (0~9) または「_」のいずれかでなければなりません。
- 名前の最後に「\$」(ドル記号) をつけると、文字列変数となります。一方、名前の最後に「\$」をつけなかったときは、数値変数となります。
- 名前の長さに制限はありません。
- 予約語 (for や if など) をそのまま変数の名前に使うことはできません。

予約語とは

予約語とは、あらかじめ決まったスペル (綴り) と意味を持っている単語のことです。

既に決まった意味を持っているので同じ名前で変数を作ることができないので注意しましょう。

利用不可 : roll, print, if, else など

利用可能 : rollBall, printScreen など

詳しくはこちらを確認してください。

<https://jasminetea.app/docs/guides/reserves>

代入とは

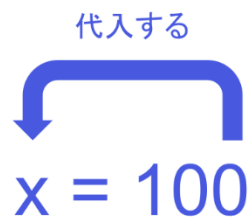
変数を作成しただけでは中身が空っぽのままです。

その変数に格納する値を設定してあげる必要があります。

変数に値を格納することを**代入**と呼びます。applePrice = 100 の「100」の部分ですね。

数学では「 $x = 100$ 」とすると、「右辺（変数 x ）と左辺（100）は等しい」という意味になりますが、プログラミングでは「左辺の値（100）を右辺（変数 x ）に代入する」という意味になります。

さっきの例では、「100 という値を変数 `applePrice` に代入（格納）する」という意味になります。



出力とは

変数は値を出し入れすることができます。

`print applePrice` で画面に 100 という値を出力することができました。

これは変数 `applePrice` に格納されていた値のみを取り出して出力しています。

変数同士の計算

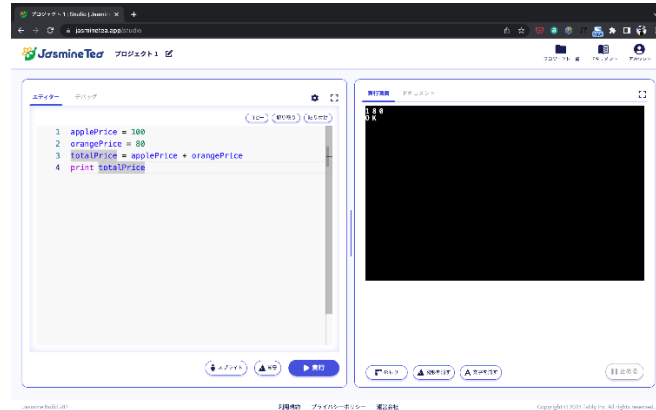
変数は上記の通り、値を出し入れすることができます。

そして変数同士の計算を行うこともできます。

以下のコードをエディタに記述し、実行してみましょう。

```
applePrice = 100
orangePrice = 80
totalPrice = applePrice + orangePrice
print totalPrice
```

そうすると、下の画像のようにりんごの値段とオレンジの値段を足した数が出力されたかと思えます。



変数に値を代入し、その後の計算などを全て変数で定義すれば、例えばりんごの値段が変わったとしても最初の `applePrice` に代入した数値を変更するだけで済むので、コードの量が多くなればなるほど便利になってきます。

再代入とは

再代入とは、あとから変数に格納されている値を変更することをいいます。

たとえば一度合計金額を出した後に、リンゴをもう 1 個追加で購入した場合をコードで表してみたいと思います。下記のコードを記述し、実行してみましょう。

```
applePrice = 100
orangePrice = 80
totalPrice = applePrice + orangePrice
print totalPrice
totalPrice = totalPrice + applePrice
print totalPrice
```

そうすると、画面には `180` と `280` という数字が表示されているかと思えます。

一度 `totalPrice` にリンゴ 1 個とオレンジ 1 個の値段を足した数を代入しましたが、下の行で `totalPrice` にリンゴ 1 個分を足した数を再度代入しました。

これにより `totalPrice` の値が置き換わった形になります。

このように、一度代入した変数はあとから変更することもできるので、その変数に今どんな値が格納されているのかをよく考える必要があります。

また、通常数学では「=」は等式なので「 $x = x + 1$ 」という式は利用しませんが、プログラミングでは「変数 x に 1 を足した値を再度変数 x に代入する」という考え方になるので注意してください。

$$x = x + 1$$

x に 10 が代入されているとすると、 $10 + 1 = 11$ の値が再度変数 x に代入されたことになります。

配列の基礎知識と使い方

配列とは

配列とは、多くの数値や文字列を 1 つの変数で扱うことができる仕組みのことです。

上記で値を格納しておく箱のことを「変数」といいました。変数に値を格納し、必要な時に値を取り出して利用していましたね。

ただ、変数は**値を 1 つしか**代入することができません。

配列は 1 つで複数の値を格納することができるので、同じような値をまとめて管理することができます。

例えば、フルーツの名前を格納する配列を準備して画面に出力してみたいと思います。

以下のコードを記述し、実行してみましょう。

```
fruits@ = ["リンゴ", "オレンジ", "バナナ"]  
print fruits@[0]
```

画面にリンゴと表示されていれば成功です。

Jasmine Tea では、配列を扱う変数の後ろに「@」をつけます。

そして配列に値を格納する時は「[]」の中に1つずつ「,」を利用して値を入れてください。

文字列を扱う場合は「" "」でその文字列を囲いましょう。(数値は囲わなくて OK です)

```
fruits@ = ["リンゴ", "オレンジ", "バナナ"]
```

この記述で「fruits」という配列の中に「リンゴ、オレンジ、バナナ」という文字列を代入することができました。

```
print fruits@[0]
```

この記述で配列 fruits@ に格納されている 0 番目の値(リンゴ)を画面に出力できました。

配列から値を取り出す時は

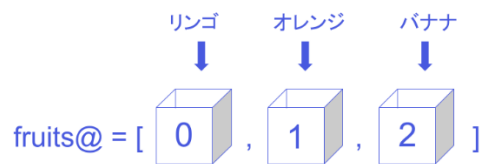
配列名【添字】

と記述します。

配列に格納されている値には全て番号が付けられます。この番号のことを添字(インデックス)と呼びます。

Jasmine Tea では、添字は 0 から始まるので、fruits の 1 つ目の値(リンゴ)を取り出したい時は fruits@[0] と記述しなければならないので注意してください。

2 つ目のオレンジを取り出したい時は fruits@[1] になります。



配列についてもっと詳しく知りたい方はこちらを確認してください。

<https://jasminetea.app/docs/guides/array>

💡 チャレンジ問題

- 配列から「バナナ」という値を取り出して出力してみよう
- 配列の中身（テキスト）を好きなものに変更してみよう

条件分岐の書き方

条件分岐とは、ある条件が満たされたかどうかで次に実行されるプログラムを変化させることをいいます。

例えば、

「もし雨が降っていたら、傘をさす。降っていなかったら傘をささない」

「もし 120 円以上持っていたらジュースを買う、持っていなかったら買わない」

など、日常でもその条件が満たされているかどうかで次の行動を変化させると思います。

プログラムも同様に、例えば

「もしプレイヤーと障害物が衝突したら、HP を減らす」

「もしゲームオーバーになったら、ゲームを止めてスコアを表示する」

といった具合に、その条件に当てはまるかどうかで次の処理を変化させる場面が多くあります。

実際に少し記述してみましょう。

以下のコードを書いて、実行してください。

```
age = 21
if age >= 18 then
  print "あなたは成人です"
else
  print "あなたは未成年です"
end if
```

まず、`age` という変数に `21` という数値を代入しました。

`if age >= 18 then` で「もし `age` が `18` 以上だった場合」という条件分岐を作成し、その条件に当てはまった場合、下のコード (`print "あなたは成人です"`) を実行します。

`else` は「条件に当てはまらなかった場合」下のコードを実行するという意味になります。

つまり上の例では「`age` が `18` 未満だった場合」となりますね。

```
if 条件文 1 then
  条件文 1 に当てはまった場合に実行される
else
  条件文 1 に当てはまらなかった場合に実行される
end if
```

`age` の値に `17` を代入し、再度実行してみましょう。

`if` 文の条件に当てはまらず、`else` 文のコードが実行されていれば成功です。

Jasmine Tea では、条件文の最後に `end if` をつける決まりがあるので注意してください。

また、条件分岐は複数の条件を設定することもできます。

```
if 条件文 1 then
  条件文 1 に当てはまった場合に実行される
else if 条件文 2 then
  条件文 1 に当てはまらないが、条件文 2 に当てはまった場合に実行される
else
  条件文 1 にも条件文 2 にも当てはまらなかった場合に実行される
```

```
end if
```

例えば、上の例で問題なく成人と未成年を分けられたように思いますが、もし `age` に `-1` を代入したらどうなると思いますか？

「もし `age` が `18` 以上だった場合は成人、それ以外は未成年」という条件分岐になっているので

「`-1`」でも未成年と出力されてしまいます。

年齢でマイナスはおかしいので、条件を修正しましょう。

1. 条件 1 を「もし `age` が `18` 以上だった場合」
2. 条件 2 を「条件 1 に当てはまらず、もし `age` が `0` 以上だった場合」
3. 最後に「条件 1 にも条件 2 にも当てはまらなかった場合」

という風にします。以下のコードを書いて実行してください。

```
age = -1
if age >= 18 then
  print "あなたは成人です"
else if age >= 0 then
  print "あなたは未成年です"
else
  print "正確な年齢を入れてください"
end if
```

`age` にマイナスの数値を入れると `else` 文が実行されているかと思います。

このように条件は複数に分けて設定することもできるので覚えておきましょう。

💡 チャレンジ問題

- もし `age` が `60` 以上だった場合、「あなたは還暦です」と表示するように修正してください。
- 変数 `x` に任意の値を代入し、もし変数 `x` の値が `3` で割り切れる時は「これは `3` の倍数です」と出力し、それ以外の場合は「これは `3` の倍数ではありません」と出力するプログラムを作成してください。(下は書き方の例です)

```
x=9
if 3 の倍数だった場合 then
  print “これは 3 の倍数です”
else
  print “これは 3 の倍数ではありません”
end if
```

2. 円を描画してみよう

今回作るゲームについて

これから複数回に分けて、実際に 2D ゲームを制作しながらプログラミングを学習していきます。

今回作るゲームは、プレイヤーとなるキャラクターを操作し、横から飛んでくるボールを避けるというカジュアルなゲームとなっています。

ゲーム自体は単純なのですが、制作する上でプログラムの重要な要素がいくつも入っていますので、しっかり作りながら学んでいきましょう。



円の描画

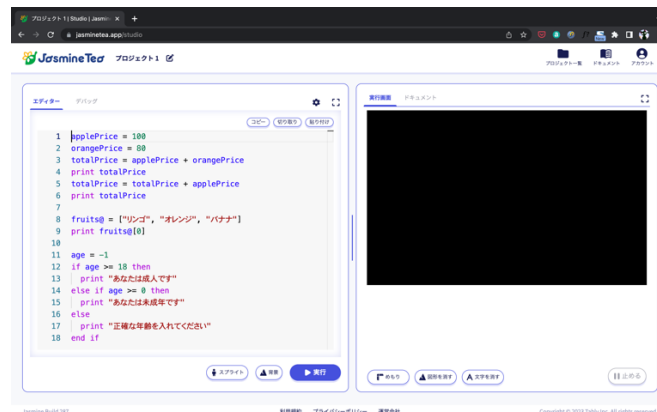
Jasmine Tea を起動しよう

まずは Jasmine Tea を開いてください。

<https://jasminetea.app/studio>

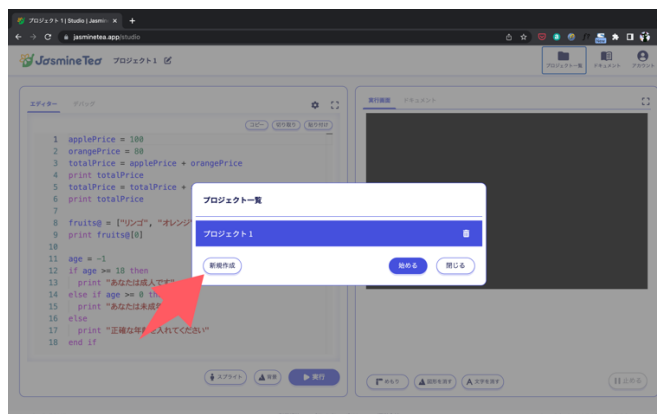
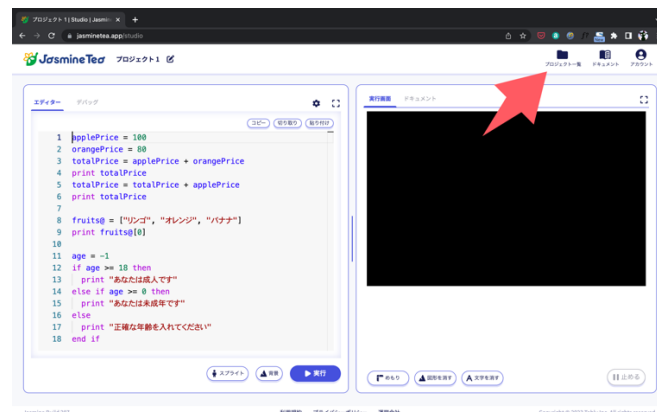
すると、下の図のように前回作成したページが表示されていると思います。

ログイン画面が出た場合は、ログインをしてプログラム作成画面を表示してください。



このように、Jasmine Tea は最後に開いた状態を自動で保存してくれます。

今回は新しくプロジェクトを作成したいと思いますので、画面右上の「プロジェクト一覧」をクリックし、「新規作成」を選択してください。



そうすると、新規画面が開き、左上のプロジェクト名が「プロジェクト 2」になっているかと思います。

円を描画しよう

まずは実行画面に円を描画したいと思います。

プログラムで円を書く方法なんて分からないと思うかもしれませんが、Jasmine Tea では円を描画するための命令が最初から準備されているので安心してください。

💡 命令とは

命令とは、Jasmine Tea に対しての具体的な指示のことです。Jasmine Tea は、命令をされると、忠実にその命令で表される「やらなければならないこと」をすぐに行います。Jasmine Tea が命令を受けて何かを行うことを、「命令を実行する」と言います。

もちろん、何でもかんでも人間が自由にコンピューターに指示をしたとしても、そのコンピューターができることは限られています。予め Jasmine Tea ができることが決まっていて、それぞれに命令があります。例えば、以下のような命令があります。

- circle - 円を画面に描く。
- print - 画面に文字や計算結果を表示する。
- move - スプライト機能を使って、キャラクターを画面上で動かす。
- speak - 音声合成機能を使って、セリフを話す。
- bgmplay - BGM (Background Music) を再生する。

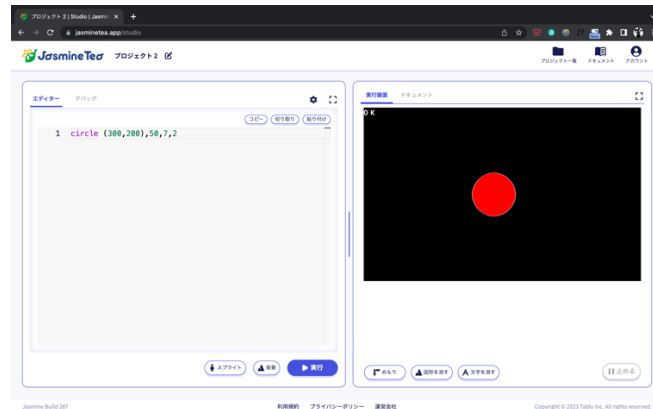
上記は、Jasmine Tea ができることのほんの一部です。数多くの命令があって、Jasmine Tea は個々の命令で指示されたことを忠実に実行し、その結果を画面に表示したり、音を鳴らしたりして伝えます。

では、上記の `circle` 命令を使って画面に円を描画してみましょう。

以下のコードを記述して実行してください。

```
circle(300,200),50,7,2
```

そうすると、下の画像のように実行画面に赤い円が描画されました。



先ほどのコードを日本語にすると下のようになります。

円を描画する命令 (x 座標, y 座標), 半径, 線の色, 塗りつぶしの色

Jasmine Tea に単に `circle` とだけ指示をしたとしても、Jasmine Tea は「円を描けと言われても...」となってしまいます。どこにどんな大きさでどの色の円を描けば良いか、細かく指示をしてあげなければ、Jasmine Tea はどんな円を描いていいかわかりません。

命令の後に指定する具体的な指示のことを、「**パラメーター**」と呼びます。先ほど実行した `circle` 命令の例では、「(300,200),50,7,2」がパラメーターです。`circle` 命令のパラメーターは、(300,200) が円の中心の座標、50 が半径、7 が線の色（白）、そして 2 が塗りつぶしの色（赤）という意味になります。

命令の後にパラメーターを指定する際には、命令の直後に空白を一つ以上入力してから、パラメーターを書くことが必要です。

命令ごとに、パラメーターとして指定するものは異なります。また、パラメーターがない命令もあります。どの命令にどんなパラメーターが必要なのかについては、「命令・関数辞書 (<https://jasminetea.app/docs/references/index>)」にて説明されています。

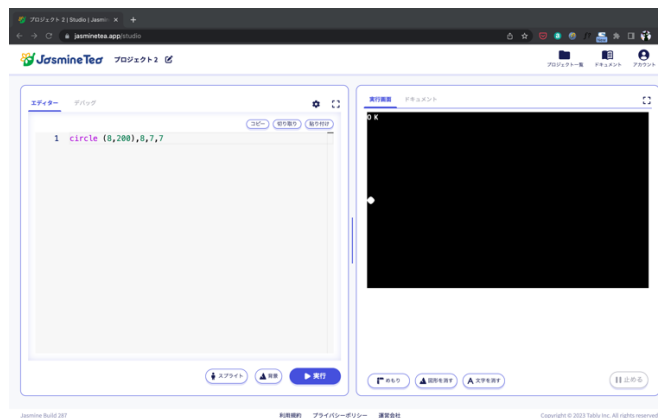
カラーについては、Jasmine Tea では「38 色 (0~37)」指定することができます。

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37		

先ほど書いたコードを以下のように変更して実行してください。

```
circle (8,200),8,7,7
```

そうすると、下の画像のように白い円が実行画面に描画されていると思います。



コメントを書こう

Jasmine Tea では、コメントという機能を使ってプログラム上にメモを残すことができます。

```
// これがコメント
```

文の最初に「//」を記入することにより、その行に書かれたものはプログラムとして実行されなくなります。このコードがどういう意味なのか忘れないようにこまめにプログラム上にコメントを残しておきましょう。

先ほどの円を表示するプログラムの上にコメントを追加します。

```
//円を表示する
circle (8,200),8,7,7
```


円の移動

次に、今描画した円を右に移動させてみたいです。

図形を右に移動させる方法はいくつかあるのですが、今回は「画面をずらしていく」方法を取ります。

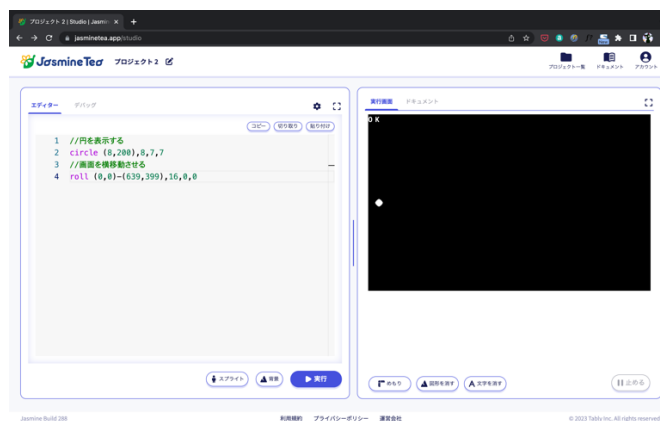
Jasmine Tea で画面をずらす（スクロールさせる）命令は「roll」です。

roll 命令を使うことにより、プログラム実行画面に描画されている全ての図形を、縦や横に移動させることができます。実際にやってみましょう。

下のコードをエディタに追加してください。

```
//画面を横移動させる
roll (0,0)-(639,399),16,0,0
```

追加できたら、実行するボタンを押してみましょう。そうすると、下の画像のように右に少し移動しました。



先ほどのコードを日本語に直すと下のようになります。

画面を移動させる命令 (始点 x, 始点 y)-(終点 x, 終点 y), 横方向の移動量, 縦方向の移動量, 背景の色

少し分かりにくいですが、始点 x,y から終点 x,y までの範囲内を横の移動量、縦の移動量分ずらします。という命令文です。

実行画面の左上の座標が「x=0、y=0」

右下の座標が「x=639、y=399」

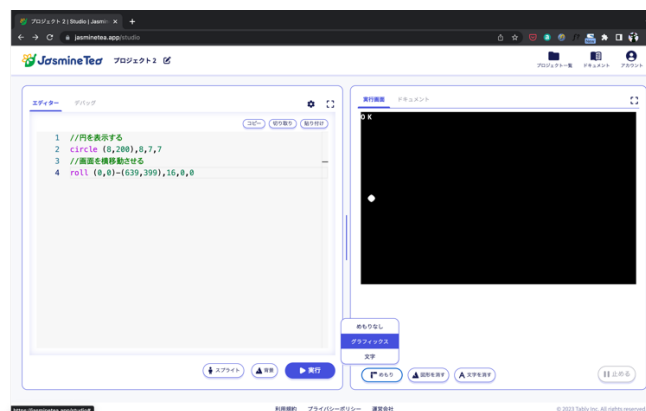
なので、実行画面全てを右に 16 移動させた後に背景色 0（黒色）で塗りつぶした形になります。

座標の確認方法

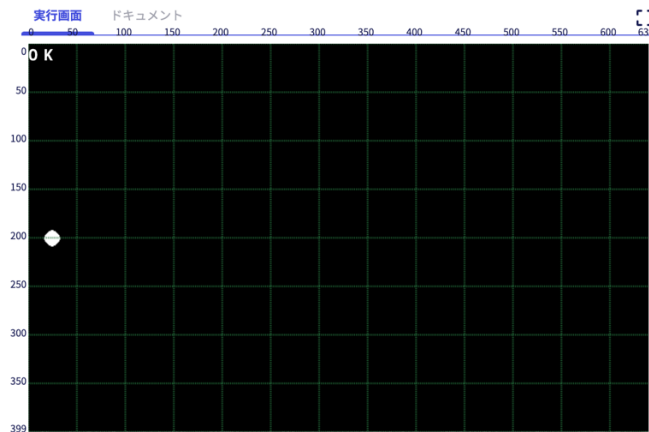
画面右下の座標を「x=639、y=399」としましたが、現在の画面では右下の座標の数分かりません。この座標を確認するために、Jasmine Tea では「メモリ」という機能があります。

実際に試してみましょう。

実行画面下の「メモリ」をクリックしてください。そうすると、3 つ項目が出てきますが、そこから「グラフィックス」を選択しましょう。



そうすると、下の画像のように実行画面に目盛りが表示され、右端が 639、下が 399 となっています。これがこの実行画面の座標です。



グラフィックなどはこの座標に合わせて配置してくので確認する時はこの目盛りを活用してください。

円を複数回移動させる

先ほどのコードで円 1 つ分右に移動できました。では、これを円 5 個分移動させるにはどうすればいいでしょうか？

`roll` 命令を 5 回使えば良さそうです。

実際に記述してみましょう。

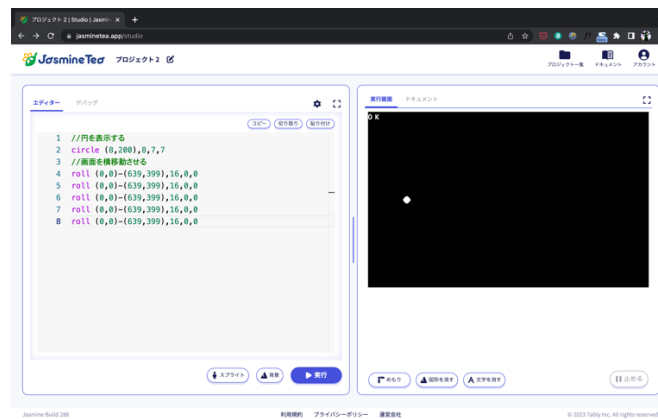
エディタを以下のように修正してください。

既に上 2 行は書いてあるので、残りの部分を追加します。

2 行目以降は全く同じコードなので 2 行目をコピー & ペーストして増やしましょう。

コードが書けたら実行ボタンを押してください。

そうすると下の画像のように円が 5 つ分移動していると思います。



💡 コピペ（コピー&ペースト）のやり方

コピペはプログラミング以外にもよく利用されるものなので、ぜひショートカットを覚えましょう。

まずコピーしたい文字を選択します。

```
1 circle (8,200),8,7,7
2 roll (0,0)-(639,399),16,0,0
```

次に、Windows と Chromebook の人は「**Ctrl キー**」を押しながらキーボードの「**C キー**」を押してください。これでコピーができます。

Mac の人は「**command キー**」 + 「**C キー**」でコピーできます。

```
1 circle (8,200),8,7,7
2 roll (0,0)-(639,399),16,0,0
3 |
```

そして貼り付けたい場所へ移動し、Windows と Chromebook の人は「**Ctrl キー**」を押しながらキーボードの「**V キー**」を押してください。これでペーストができます。

Mac の人は「**command キー**」 + 「**V キー**」でペーストできます。

```

1  circle (8,200),8,7,7
2  roll (0,0)-(639,399),16,0,0
3  roll (0,0)-(639,399),16,0,0

```

複数移動しているのを分かりやすくしてみよう

今の画面だと本当に 5 回移動しているのかよく分かりません。背景色を変更してきちんと等間隔で移動しているのか、確認してみましょう。

コードを以下のように変更してください。

```

//円を表示する
circle (8,200),8,7,7
//画面を横移動させる
roll (0,0)-(639,399),16,0,1
roll (0,0)-(639,399),16,0,2
roll (0,0)-(639,399),16,0,3
roll (0,0)-(639,399),16,0,4
roll (0,0)-(639,399),16,0,5

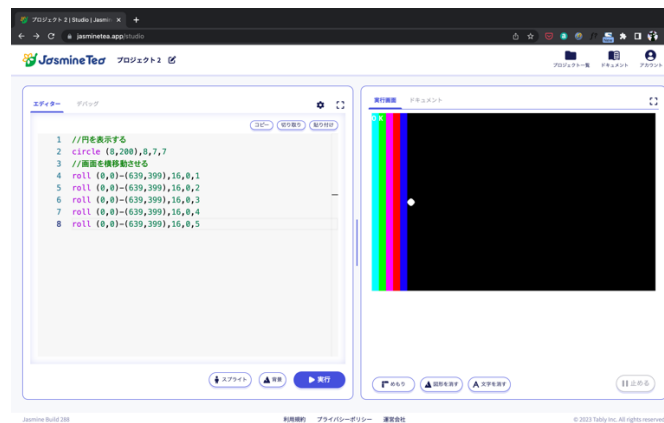
```

`roll` 命令の最後のパラメータの数を変更しました。

こちらを実行してみましょう。

下の画像のように色が変わりながら移動していれば成功です。

`roll` 命令が画面をどのように動かしたのか、色の違いで分かりやすくなったと思います。



繰り返し処理

さて、`roll` 命令で画面を動かすことができました。

しかし、画面の左端から右端までボールを移動させるためには何度も `roll` 命令を書かなければなりません。これはかなり大変です。

何度も同じ命令を実行させるために、プログラミングでは便利な命令があります。

それが「**繰り返し処理**」です。

Jasmine Tea では、`do` 命令と `loop` 命令を使うことで繰り返し行いたい命令をプログラムすることができます。

`do` 命令と `loop` 命令を使うと、プログラムの実行を強制的に止めるまで永遠に実行する（無限ループ）ことができます。

💡 特定の回数だけループ

プログラムで行いたいことの中には、命令を繰り返したい回数が事前に決まっていることもあります。例えば「10 回だけ繰り返し移動させる」などです。

そのような繰り返す回数が決まっている場合は `for` 命令と `next` 命令を利用します。

下の例は 10 回 `for~next` までの命令を実行するものです。

```

for i=1 to 10
  命令 1

```

```

  命令 2
  ...
next

```

繰り返し処理をもっと詳しく知りたいときはこちらのプログラマーズガイドを確認してください。

<https://jasminetea.app/docs/guides/loop>

まずはコードを以下のように変更してください。

```

//円を表示する
circle (8,200),8,7,7
//ずっと繰り返す
do
  //画面を横移動させる
  roll (0,0)-(639,399),16,0,0
loop

```

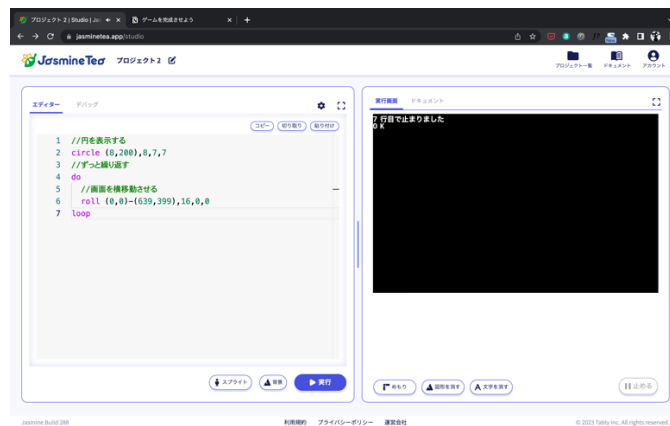
`roll` 命令の前にキーボードのスペースキーを 2 回、または Tab キーを押して少し右にずらしてください。このように `do` 命令と、`loop` 命令の間にある行の先頭に間を開けておくことで、繰り返し実行される命令がどれなのかを見やすくします。

コードを変更した後に、実行してみましょう。

すごいスピードでボールが左から右へ移動したと思います。

実行画面の「止める」でプログラムを停止させましょう。ピーという音が鳴った後にプログラムの実行が止まります。

プログラム実行画面には、プログラムの何行目で止まったのかが表示されます。



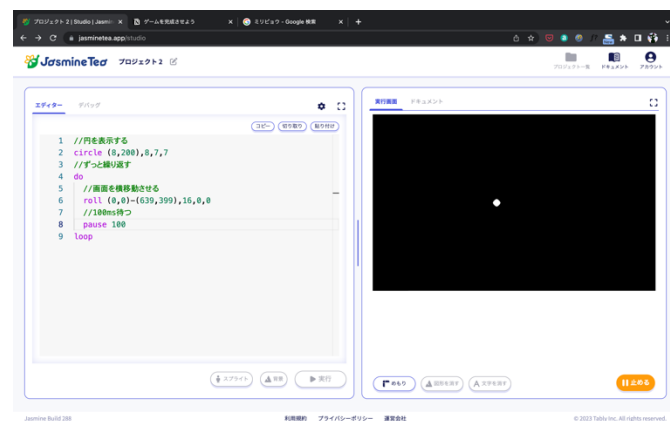
do 命令は、do と loop の間にあるプログラムを延々と実行し続けます。ボールが消えた後も見えないだけで画面は動き続けているのです。

あまりにもスピードが速いので、pause 命令を使って、もっとゆっくり円が移動するようにしてみましょう。do と loop の中に以下のコードを追記してください。

```

//100ms 待つ
pause 100
  
```

実行を押してプログラムを実行すると、先ほどよりもゆっくりとボールが移動するようになりました。



pause 命令は、右の数値分プログラムを一時停止させる命令です。指定はミリ秒なので、100 と指定すると、大体 0.1 秒待つということになります。

💡 ミリ秒とは

ミリ秒とは、時間の単位の一つで、1 秒の 1000 分の 1（0.001 秒）を表すもの。1000 ミリ秒が 1 秒に相当し、コンピュータで時間を扱う際の最小単位としてよく用いられます。

ここで、`roll` 命令で動かす大きさを 16 から増やしたり減らしたりしてみてください。また、`pause` 命令でプログラムを止める時間を 100 から大きくしたり小さくしたりしてみてください。

これらにより、ボールの飛び方がどのように変わるのか、色々と試してみましょう。

ランダムな位置に配置しよう

今は、1 つのボールしか表示されていません。

今回はランダムな場所からどんどんボールが飛んでくるようにしたいと思います。

Jasmine Tea では、`random` 関数を使うことで、一定の範囲内の乱数を作ることができます。

💡 関数とは

Jasmine Tea では、指定された 0 個以上の値に対して、ある決まった計算を行い、その計算の結果を得ることができるものを「関数」と呼びます。

そして、計算に使われる元の値のことを「引数」といいます。行われる計算の種類に従って、各関数は名前を持っています。

Jasmine Tea のプログラムの中で関数は以下のように書きます。

関数名（引数 1, 引数 2, …）

関数名の後ろに、関数の計算で使用する引数を括弧で囲みます。もし 2 つ以上の引数が必要になるときは、「,」（カンマ記号）を区切り記号として使います。

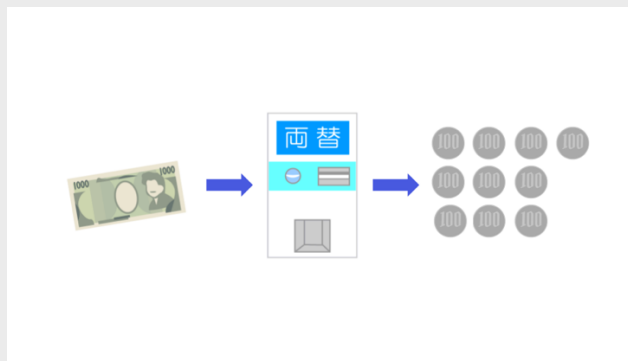
💡 例

`random (1, 6)` → 1 から 6 までの数字の中からランダムに 1 つの値を返します。

関数による計算が行われた結果の数値や文字列のことを「**戻り値**」と呼びます。関数を使った結果戻ってきた値、という意味です。Jasmine Tea では、関数は必ず一つの戻り値が得られます。

💡 例えば、

「両替機に 1000 円札を投入すると、100 円玉が 10 枚出てくる」といった場合、両替機が関数になり、両替機に投入した 1000 円札が引数になります。そして、両替機から戻ってきた 100 円玉 10 枚が戻り値になります。



関数について、より詳しく確認したい場合は下記の URL より確認してください。

<https://jasminetea.app/docs/guides/function>

まずはプログラム全体を以下のコードのように変更してください。

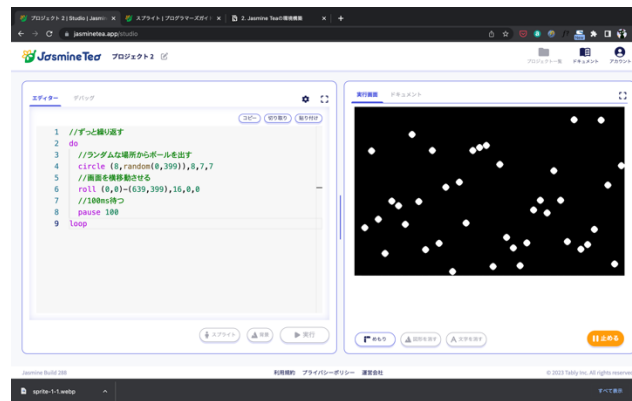
```
//ずっと繰り返す
do
  //ランダムな場所からボールを出す
  circle (8,random(0,399)),8,7,7
  //画面を横移動させる
  roll (0,0)-(639,399),16,0,0
  //100ms 待つ
  pause 100
loop
```

`circle` 命令を `do-loop` の中に入れた理由は、ボールを何個も表示させるためです。

(`do-loop` の中はずっと繰り返し実行されるので、`circle` 命令もずっと実行（ボールを描画する）されるようになります)

コードの修正が終わったら、実行するボタンを押してみましょう。

実行画面左端のランダムな位置からボールが左に流れていくようになったかと思います。



`random` 関数は、`random(最小値, 最大値)`と記述することで、最小値から最大値の間のランダムな数値を取ることができます。今回は画面の縦の幅いっぱい（0~399）のランダムな数値を取りました。

`circle` 命令で円を描く位置について

- 横方向の位置(x) → 8
- 縦方向の位置(y) → `random(0, 399)`

とすることで、ボールが画面のどこから飛んでくるかが、`circle` 命令を実行するたびに変わるようになります。

これでボールをランダムに飛ばし続けるという機能が作り終わりました。

これを使って実際にゲームを作っていくので、今回作成したプログラムは消さないように注意しましょう。

💡 チャレンジ問題

- 画面の縦座標をランダム関数を利用し、ボールが揺れて動くようにしてみよう。

3. キャラクターを動かしてみよう

キャラクターの表示

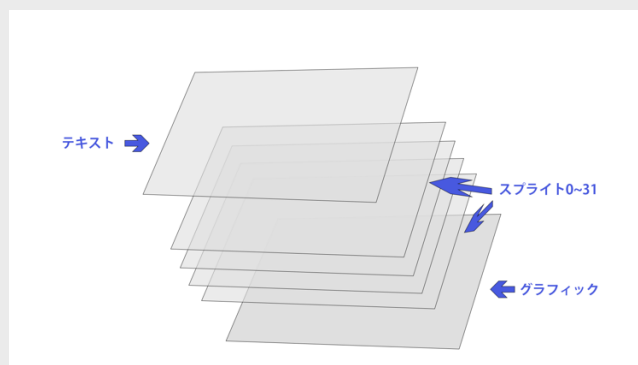
Jasmine Tea では、あらかじめいくつかのキャラクター用のスプライトを準備しています。

💡 スプライトとは

スプライトとは、キャラクターをアニメーションさせながら自由に実行画面を動かすための仕組みです。



プログラム実行画面は、真っ黒な 1 枚の黒板のような画面に見えますが、実際はいくつかの画面が重なって表示されています。



一番下に `circle` 命令によって描かれた円などの図形が表示される「グラフィック画面」があります。 `roll` 命令は、このグラフィック画面を動かします。

グラフィック画面の上にキャラクターが表示される「スプライト画面」があります。キャラクター1つごとに、スプライト画面が1枚使われます。Jasmine Tea では、スプライト画面が32枚あります。つまり、32個のキャラクターを画面に表示することができます。

グラフィック画面の手前にスプライト画面があるので、キャラクターの後ろに何か図形があったときは、図形はキャラクターの後ろに隠れます。スプライト画面は全部で32枚あって、それぞれ番号が振られています。

「OK」などの文字は、テキスト画面に表示されます。テキスト画面は、一番手前にあります。

つまり、文字が一番手前に表示されるので、文字の後ろに図形やキャラクターがあった場合は、図形やキャラクターは文字の後ろに隠れます。

キャラクターを表示してみよう

Jasmine Tea では、`sprite` 命令と `show` 命令を使ってスプライト画面にキャラクターを表示します。

`sprite s[スプライト番号の式], a[アニメーション番号の式]`

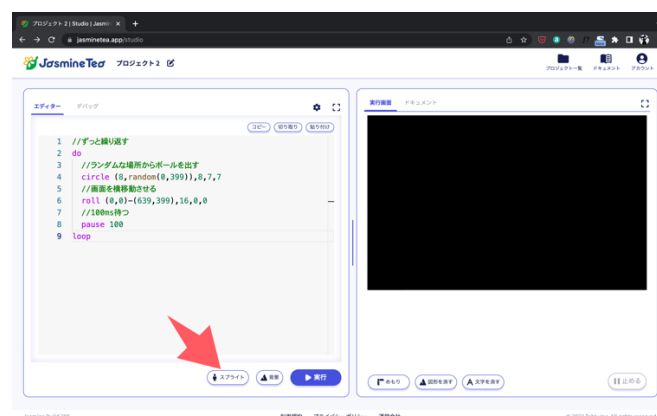
`show n[表示したいスプライト番号の式], (x[スプライトを表示する x 座標], y[スプライトを表示する y 座標])`

`sprite` 命令の最初に、登録したいスプライト番号を書きます。スプライトは、0 から 31 までの 32 個まで番号を使うことができ、それぞれ表示したいアニメーション番号を割り当てることができます。何も登録していないときは0番を利用するので「0」を記入しましょう。その次にアニメーション番号を記述します。



今回は左向きのアニメーションを利用したいので「2」を記入してください。

なお、スプライトやアニメーションは下図の「スプライト」から確認することができます。



```
//キャラクターの登録
sprite 0,2
```

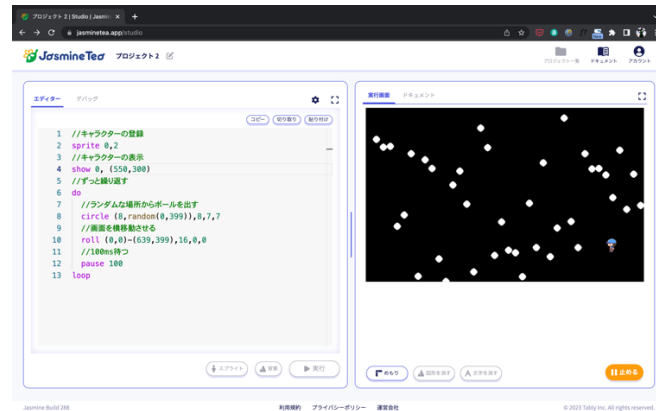
show 命令の最初に、表示したいスプライト番号を記述します。先ほどスプライトの0番目にアニメーションを登録したので0を記入しましょう。次のx,y座標は今回は(550, 300)としておきます。

```
//キャラクターの表示
show 0, (550,300)
```

上の `sprite` と `show` 命令は現在の `do-loop` 命令の前（上の行）に記入してください。

入力が終わったら実行ボタンを押してプログラムを動かしましょう。

下の画像のように、キャラクターが表示されていれば成功です。



キャラクターとボールの当たり判定を取得しよう

キャラクターが飛んでくるボールに当たったらゲームオーバーになるようなゲームを作る予定なので、キャラクターとボールがぶつかっているかどうか判断する必要があります。

当たっているかどうかは人間は見れば判断できますが、コンピュータは、判断方法を命令しないとキャラクターとボールが当たっているかどうかを判断できません。

「〇〇が△△となったときに当たったとする」というように、プログラムを使ってコンピュータに「どういう時に当たったとするか」決めるための方法を教える必要があります。

現在、左からボールが次々と飛んできますが、キャラクターに当たったとしても、ボールはキャラクターを通り抜けていきます。

では、ボールがキャラクターに当たってない時と、当たった時を見比べてみましょう。



ボールは、グラフィック画面に描かれています。キャラクターは、スプライト画面に描かれています。スプライト画面はグラフィック画面よりも手前ですので、ボールはキャラクターの後ろに描画されます。

まず、キャラクターがボールに当たっていない時はキャラクターの背景部分は真っ黒です。

次に、キャラクターがボールに当たっているときはキャラクターの背景には、ボールの白い円があります。

つまり、キャラクターが表示されている部分の背景のグラフィック画面に白色があれば「当たった」ということにできそうですね。

Jasmine Tea では、「touch 関数」を使うことで、キャラクターが表示されている位置のグラフィック画面に、ある色が描かれているかどうかをチェックすることができます。

`touch(n[スプライトの id], c[色番号の式])`

スプライトが表示されているグラフィック画面の位置に、指定された色が描画されているかどうかを確認します。もし描画されていたときは、`-1` がこの関数の結果の値になります。もし描画されていなかった場合は `0` がこの関数の結果の値となります。

実際に利用してみましょう。

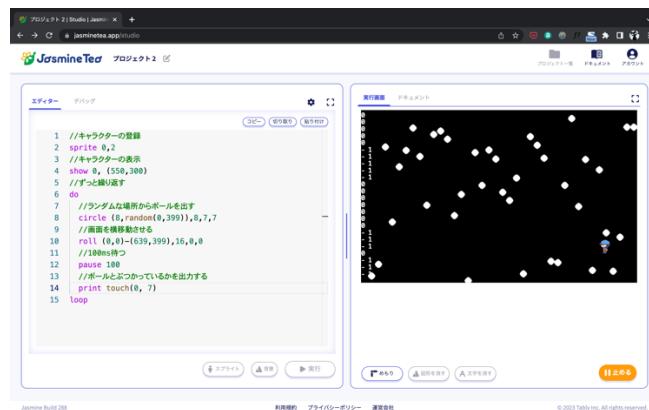
「`pause 100`」で改行し、以下のコードを追記して実行してください。

```
//ボールとぶつかっているかを出力する
print touch(0, 7)
```

重なっているかを判断するスプライトの番号は 0 番、重なっているかを判断する色は白なので 7 になりますね。ボールを他の色にしている人はその色の番号を指定してあげてください。

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37		

下の画像のように、キャラクターとボールが重なったタイミングで `-1` が出力されていれば成功です。



これでキャラクターとボールの接触判定を取ることができました。

キャラクターを動かしてみよう

表示と接触判定はできたので、次はキャラクターを動かしてみましょう。

このゲームでは、プログラム実行画面をマウスやトラックボール、指を使ってクリックやタップした時に、キャラクターを動かしたいと思います。そしてキャラクターを上下に動かすためのルールとして以下の通りにします。

- キャラクターよりも画面の上をクリックした時は、キャラクターを上の方に動かす
- キャラクターよりも画面の下をクリックした時は、キャラクターを下の方に動かす

もしプログラム実行画面をクリックやタップした時は、クリックやタップした位置とキャラクターの位置を比較し、キャラクターを上に移すか下に移すかを判断します。

Jasmine Tea では、`tap` 命令を使って、プログラム実行画面のどこをクリックやタップされたかを知ることができます。

`tap x[x 座標], y[y 座標]`

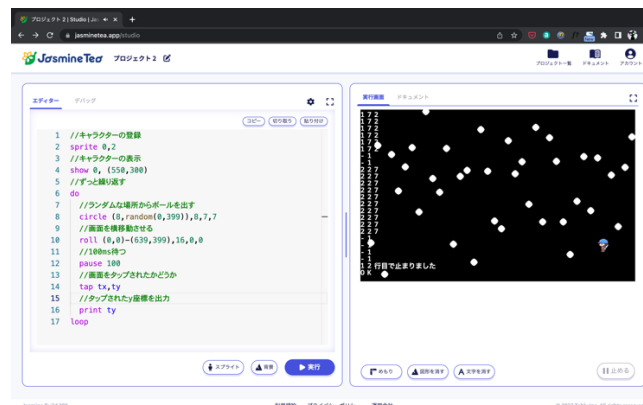
`tap` 命令では、プログラム実行画面がクリックまたはタップされた時の横方向 (x) の位置と縦方向 (y) の位置を、それぞれ別の変数に代入します。

では実際に `tap` 命令を使ってプログラムを作ってみましょう。エディタに以下のプログラムを入力してください。

```
//キャラクターの登録
sprite 0,2
//キャラクターの表示
show 0, (550,300)
//ずっと繰り返す
do
  //ランダムな場所からボールを出す
  circle (8,random(0,399)),8,7,7
  //画面を横移動させる
  roll (0,0)-(639,399),16,0,0
  //100ms 待つ
  pause 100
  //画面をタップされたかどうか
  tap tx,ty
  //タップされた y 座標を出力
  print ty
loop
```

書き終わったら実行ボタンをクリックしてください。

1 秒ごとに「-1」が画面に表示されます。プログラム実行画面のどこかで、クリックやタップをしてみましょう。「-1」の代わりにクリックまたはタップした場所の縦方向（y）の位置の数字が出力されます。



「ty」の変数は、tap 命令で「tap tx, ty」というように、2 つ目の変数として書いたので縦方向（y）の位置が print ty で表示されました。

こちらを「print tx」とすると、横方向の位置が表示されるようになるので試してみましょう。

`tap` 命令はプログラム実行画面がクリックまたはタップされたかどうかで、動きが変わります。

- クリックまたはタップされていた時→クリックまたはタップされた位置の数に変数に入る
- クリックまたはタップされていなかった時→「-1」が変数に入る

つまり、`tap` 命令を実行した後に、変数に入っている数が「-1」だったときは、クリックやタップではなかった、と判断することができます。反対に、変数に入っている数値が「0」以上だったときは、その位置でクリックまたはタップされたと判断することができます。

今までのプログラムでは、プログラムエディタに書いた全ての命令が実行されていました。

このゲームでは、プログラム実行画面をクリックやタップした時に、キャラクターを上下に動かします。つまり、命令を必ず実行するのではなく、画面をクリックやタップされた時だけキャラクターを移動するための命令を実行することになります。

画面をクリックやタップされたかどうかは、`tap` 命令を使う事で分かります。まずは日本語で `tap` 命令を使って画面をクリックやタップされたかどうかを表現してみると、下記のようになります。

もし変数 `ty` の数が 0 以上だったら、キャラクターを動かす

もし画面をクリックやタップされたときは、変数 `ty` の数は 0 以上になるので、キャラクターを動かします。一方、もし画面をクリックやタップされていなかったときは、変数 `ty` の数は -1 になるので「もし変数 `ty` の数が 0 以上だったら」という条件には当てはまらないため「キャラクターを動かす」ことは行いません。

このように、条件に当てはまるかどうかで命令を実行するかしないかを切り替えることを「条件分岐」と呼びましたね。

```
if 条件式 then
  条件に当てはまったときに実行したい命令
end if
```

if と then の間に、条件の式を書きます。

if~then の行と、end if の行の間に、条件に当てはまったときの命令を書きます。

命令は 2 行以上使っていくつも書くことができます。

「条件」という言葉を聞くと、何を連想するでしょうか？「東京駅に着いたら」や「金額が千円以上になったら」、「月が出ていなかったら」と言った表現が条件ですね。

Jasmine Tea では、数を使った条件の式として、以下のように書くことができます。

意味	式	例
A と B が同じ数かどうか	○=□	A=5
A と B が違う数かどうか	○<>□	A<>5
A の数が B の数より大きいかどうか	○<□	A<5
A の数が B の数以上かどうか	○<=□	A<=5
A の数が B の数より小さいかどうか	○>□	A>5
A の数が B の数以下かどうか	○>=□	A>=5

○と□には、数だけでなく、数が入っている変数や、関数も書くことができます。例えば変数 A に 3 が入っているときに、条件の式が「A < 5」だった時は、変数 A の数が 5 よりも小さいので条件に当てはまります。

では実際に画面をクリックやタップされた時だけ、キャラクターを動かせるようにしたいと思います。少し長いですが、以下のコードを記述してください。

```
//キャラクターの y 座標初期値
y=100
//キャラクターの登録
sprite 0,2
//ずっと繰り返す
do
  //ランダムな場所からボールを出す
  circle (8,random(0,399)),8,7,7
  //背景画面を移動させる
  roll (0,0)-(639,399),16,0,0
  //100ms 待つ
  pause 100
```

```
//画面をタップされたかどうか
tap tx,ty
//タップされたら
if ty<>-1 then
  //キャラクターより上をタップされたら
  if ty<y then
    y=y-16
  //キャラクターより下をタップされたら
  else
    y=y+16
  end if
end if
//キャラクターを表示
show 0,(550, y)
loop
```

変数 `y` には、キャラクターの縦方向の位置を入れています。

このように変数の中に値を入れることを代入と呼びます。

まず最初に変数 `y` に `100` という数値を代入したことにより、キャラクターの初期位置が縦座標 `100` の場所となります。

`if` 命令の条件の式として「`ty<>-1`」と書きました。これは「変数 `ty` の数が `-1` ではない場合」という条件になります。画面がクリックやタップされたときは、変数 `ty` の数が `-1` ではなくなりますので、その中の命令が実行されます。

キャラクターの縦位置は変数 `y` の数値です。画面をクリックまたはタップした位置は変数 `ty` の数です。キャラクターを動かすためには、「もし変数 `ty` の数が変数 `y` の数より小さかったら、変数 `y` の数を `16` 減らし、そうでなかったら、変数 `y` の値を `16` 増やす」という挙動にする必要があります。

「そうでなかったら」つまり、最初の条件に当てはまらなかったらという新しい条件が出てきました。そのような場合は `else` 命令を使って記述します。

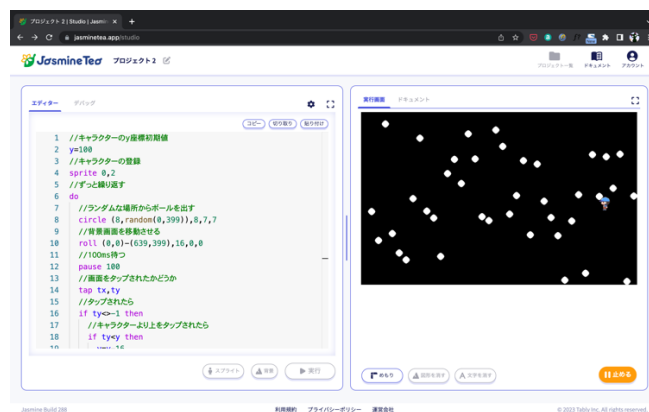
```
if 条件式 then
  条件に当てはまったときに実行したい命令
else
  条件に当てはまらなかった時に実行したい命令
end if
```

最後に `show 0, (550, y)` とすることで、キャラクターの縦座標の位置を変数 `y` に代入された数（クリックやタップにより変化した数値）に変更することにより、キャラクターが移動できています。

早く終わった場合はいろいろカスタマイズを行ってみましょう。

だいぶゲームが完成に近づいてきました。

まだボールに当たってもゲームオーバーとならないので、次回はその機能を追加していきたいと思います。



📌 チャレンジ問題

- タップされた時に移動する距離を増やしてみよう

4. ゲームを完成させよう

ボールの数を調整しよう

前回までで、ボールをランダムに左から右に流すのと、キャラクターを上下に動かすことができました。

今回は、ボールに当たったらゲームオーバーとなるようにしていきたいと思います。

また、前回までに作成したものはボールの数が多すぎて難易度が高すぎるので、ボールの数を減らして難易度を調整していきましょう。

ボールを減らそう

プログラムの 5 行目を確認してみましょう。

この行では、`circle` 命令を使って白い円をグラフィック画面に描くことでボールを作っています。ボールを出す位置は `random` 関数を使って乱数を作ることで毎回違う位置にしています。

```
circle(8, random(0, 399)), 8, 7, 7
```

そして `roll` 命令を使って画面を右に動かした後に、`pause` 命令を使って、`100ms` プログラムを止めています。

```
roll (0, 0)-(639, 399), 16, 0, 0
pause 100
```

「ボールを画面の左端に出して、画面を右に動かす」ということを、`do` 命令と `loop` 命令を使って繰り返します。

画面が右に動くたびにボールが 1 つ作成されるのでボールが大量に出てしまっています。

このボールを出すタイミングを調整しましょう。

例えば「サイコロを振ってもし 3 が出た時にボールを作成する」というようにすればボールの数を減らすことができそうです。

どんな命令や関数を使うといいのでしょうか？

- 「サイコロを振って」 - `random` 関数を使って 1 から 6 までの乱数を決める
- 「もし 3 が出たときは」 - `if` 命令を使って 3 が出たかどうかをチェックする

というように、`if` 命令と `random` を組み合わせて使うとプログラムを作ることができそうです。

ではプログラムエディタの 5 行目を以下のプログラムに変更してください。

```
//3 が出た時だけボールを出す
```

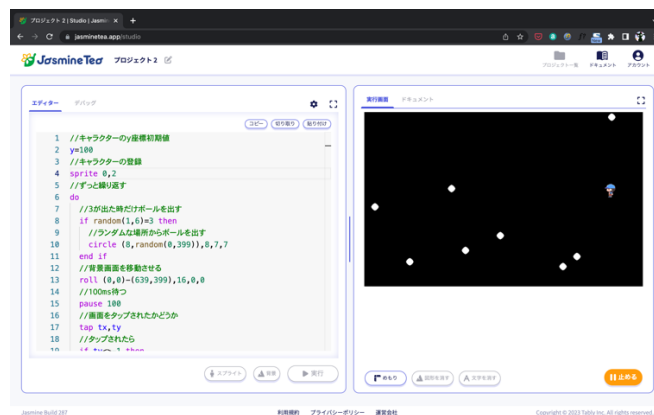
```

if random(1,6)=3 then
  //ランダムな場所からボールを出す
  circle(8,random(0,399)),8,7,7
end if

```

変更が終わったら実行してみましょう。

しっかりとボールの数が減っていれば成功です。



ボールに当たったらゲームを止めるようにしましょう

この当たったかどうかは判断する方法は前回に行いました。

`touch` 関数でしたね。

チェックしたいキャラクターのSprite番号とグラフィック画面でそのキャラクターがいる位置にチェックしたい色があるかどうかを、`touch` 関数で確認します。

前回は `touch` 関数の結果を `print` 命令で表示するプログラムを作りました。

キャラクターがボールに当たったときはゲームを終了させたいので、「もしキャラクターがボールに当たっていたら」という条件でプログラムを作りましょう。

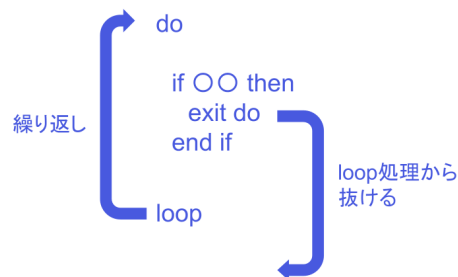
「もし～だったら」というプログラムは `if` 文でしたね。

では、プログラムエディタの `show 0, (550,y)` の下の行に以下のプログラムを追記してください。


```
//もしキャラクターとボールがぶつかったら
if touch(0,7) then
  //do ループを抜ける
  exit do
end if
```

ここで新しい `exit do` 命令が出てきました。

`exit do` 命令によって `do` 命令と `loop` 命令による繰り返しを終わって `loop` 命令の次の行に移動します。



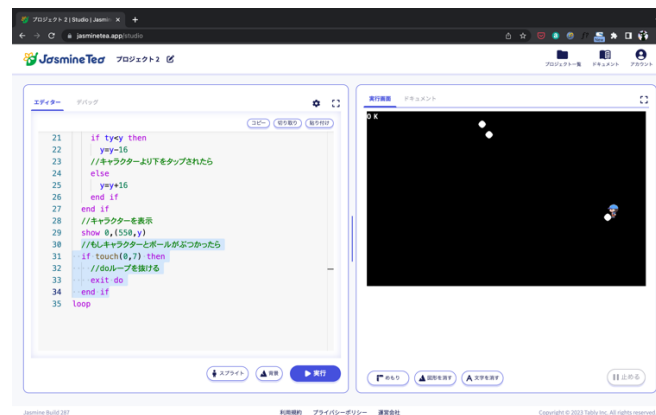
実行ボタンを押してみましょう。

そしてわざとボールに当たってみてください。プログラムの実行画面の左上に「ok」と表示されてプログラムの実行が終わりました。

`if` 命令で `touch` 関数を使ってキャラクターにボールが当たったかどうかをチェックしています。

そして当たったときは `if` 命令の中の `exit do` 命令が実行されます。

これにより最後の `loop` 命令の次の行に移動しますが、`loop` の下には命令がありませんのでプログラムが終了しました。



GAME OVER と表示しよう

ゲームの最後の仕上げとしてボールに当たったときに「GAME OVER」と表示してからプログラムが終了するようにしましょう。

文字を画面に表示するには `print` 命令を使います。

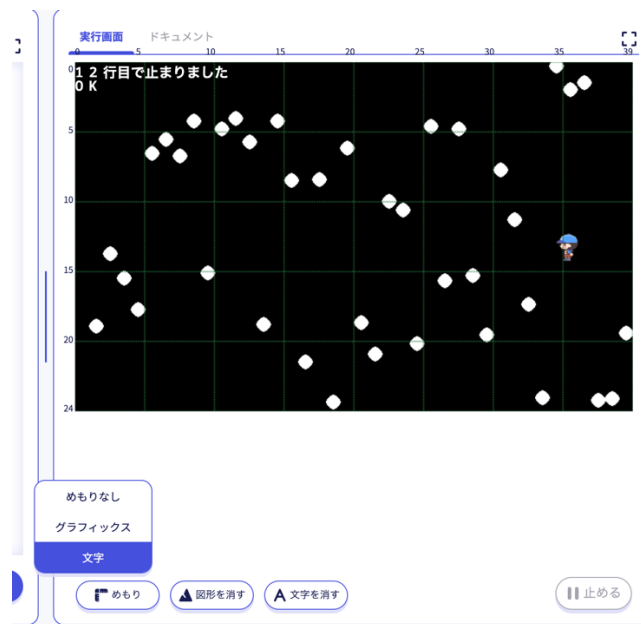
ただ、`print` 命令を記述すると、画面の左側の OK の下に表示されてしまいます。

今回はゲームっぽくするために画面の中央に「GAME OVER」と表示できるようにしましょう。

文字が表示されるテキスト画面は、横に 40 文字、縦に 25 文字の合計 1000 文字を表示することができます。

グラフィック画面と同じように、テキスト画面も左上の文字は `x:0, y:0` です。一番右は横方向に `x:39`、一番下は縦方向に `y:24` です。

こちらの数字の確認は、「メモリ」の「文字」で表示することができます。



Jasmine Tea では `locate` 命令を使って、`print` 命令で表示したい位置をコンピュータに教えることができます。

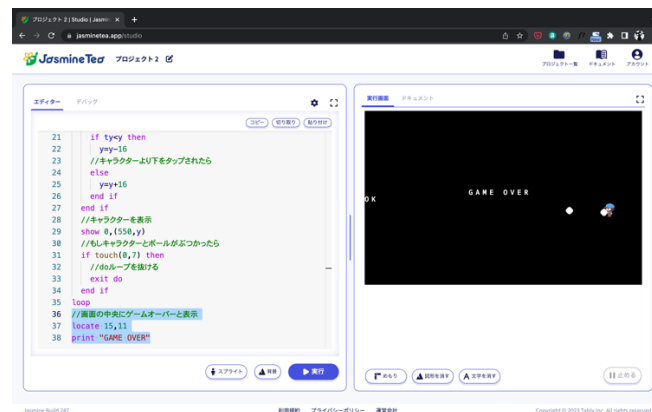
```
locate x 座標, y 座標
```

では、`exit do` 命令で繰り返しが終わった後に、プログラム実行画面の真ん中に「GAME OVER」と表示してみましょう。プログラムエディタの一番下の行に、以下のプログラムを入力してください。

```
//画面の中央にゲームオーバーと表示
locate 15,11
print "GAME OVER"
```

記入が終わったら実行してみましょう。

下の画像のようにプログラム実行画面の中央に「GAME OVER」と表示されていれば成功です。



これで簡単な 2D ゲームが完成しました。おめでとうございます！

ぜひ、どれくらい長くボールを避けることができるか試してみてください。

ここで今完成したゲームのプログラム全コードを下記に記載します。もしエラーなどで動かない場合は、下のコードとプログラムエディタに入力されたコードを見比べてください。

```
//キャラクターの y 座標初期値
y=100
//キャラクターの登録
sprite 0,2
//ずっと繰り返す
do
  //3 が出た時だけボールを出す
  if random(1,6)=3 then
    //ランダムな場所からボールを出す
    circle (8,random(0,399)),8,7,7
  end if
  //背景画面を移動させる
  roll (0,0)-(639,399),16,0,0
  //100ms 待つ
  pause 100
  //画面をタップされたかどうか
  tap tx,ty
  //タップされたら
  if ty<>-1 then
    //キャラクターより上をタップされたら
    if ty<y then
      y=y-16
    //キャラクターより下をタップされたら
    else
```

```

        y=y+16
    end if
end if
//キャラクターを表示
show 0,(550,y)
//もしキャラクターとボールがぶつかったら
if touch(0,7) then
    //do ループを抜ける
    exit do
end if
loop
//画面の中央にゲームオーバーと表示
locate 15,11
print "GAME OVER"

```

自由にカスタマイズしてみよう

完成したゲームをオリジナルにカスタマイズしてみましょう。

基本的に購入したゲーム等は自分で中身を書き換えるということできません（拡張など利用できるものもありますが）

ただし、自分で作成したゲームは好きにプログラムを書き換えることで、どんどん中身を変更することができます。

このゲームでは、プログラムの中の数字を変えることで、ゲームのスピードや難易度を変更することができます。具体的には以下の表のような値を変更することが可能なので、数値の変化でゲームがどのように変化するのか確認してみましょう。

プログラム	数字
random(1,6)	6
roll (0,0)-(639,399),16,0,0	16
pause 100	100
y=y-16	16
y=y+16	16

チャレンジ問題

- circle 命令でボールを描画したら、変数 s を増やしていく。GAME OVER になったら変数 s の数を得点としてプログラム実行画面に表示してみよう。

- random 関数を使って、circle 命令で描画する白い円の大きさを変化させてみよう。
- tap 命令に書いた変数 tx を使って、キャラクターを横方向にも動かせるようにしてみよう。

プログラムを自由に変えてみて、ぜひこのゲームをもっと面白くしてください！