Effective **プログラミン**

4J Iselix

プログラミンって？

プログラミンとは、文部科学省が作成したマウス操作だけで簡単にプログラミングをすることが出来る Flash サイト (http://www.mext.go.jp/programin/) のことです。あらかじめ用意された絵や、自分で描いた絵にチップを並べ、絵の動作を簡単にプログラミングすることが出来ます。

ビジュアルプログラミング言語は、プログラミン以外にも Squeak(スクイーク) というものや、それから派生した教育用の Scratch(スクラッチ) というものもあります。興味がありましたら、そちらのほうも調べてみてください。

プログラミンのプログラミン

プログラミンは、様々な機能を持ったチップを並べてプログラムを作成するのですが、そのチップのことをプログラミンと言います。以下ではプログラミング言語としての “プログラミン” と、チップとしての “プログラミン” が混在していますが、がんばってください。

プログラミンは全部で 28 種類ありますが、名前と基本的な役割は公式サイト[1]で紹介されています。詳しく解説すると、次のようになります。

1　ミギーン

2　ヒダリン

3　ウエーン

4　シターン

それぞれ、オブジェクトを右、左、上、下の方向に移動させます。移動距離と移動時間を指定できます。

プログラミンの基本となるプログラミンと呼んでもいいほど、重要なプログラミンです。演出や、プログラムの構造を作成するなどに使用します。

5　ジャンピン

オブジェクトをジャンプさせることが出来ます。ジャンプする高さ、滞空時間を指定できます。

このプログラミンを実行する前と後で位置が変わらないため、主に演出に使われます。

6　スケールン

オブジェクトを拡大縮小させます。拡大・縮小率と、時間を指定できます。“5%” と指定すると 5%大きく、“-5%” と指定すると 5%小さくなります。

7　ミギクルリン

8　ヒダリクルリン

それぞれ、オブジェクトを時計回り、反時計回りに回転させます。回転時間と回転角を指定できます。

9　カメロン

オブジェクトの色相を変えることが出来ます。明度、彩度を変えることはできません。変化時間、変化量を指定できます。

10　ヨコカエリン

11　タテカエリン

それぞれ、オブジェクトを左右、上下で反転させることが出来ます。この処理は瞬時に行われます。

12　ミエルン

オブジェクトの可視状態、不可視状態を設定することが出来ます。トグル (切り替え) ではなく、設定です。

13　キガエルン

オブジェクトに表示されている絵を他で使用されている絵に変えることが出来ます。

14　リセットン

オブジェクトの位置や傾き、表示されている絵、色相、可視属性などを最初の状態に戻します。

15　フキダシン

オブジェクトに吹き出しを付け、文章を表示することが出来ます。全角、半角関係なく 30 文字[2]まで表示できます。

16　フキケシン

フキダシンによって出した吹き出しを消します。

17　オンプン

あらかじめ用意された効果音や BGM を鳴らすことが出来ます。繰り返し設定をすることが出来ます。

このプログラミンは瞬時に処理が終わります。

[1] http://www.mext.go.jp/programin/data/help/programin_chart.pdf

[2] Twitter でも 140 字書けるのに、それより短いなんて……

18　ミュートン

そのときにオンプンで再生されているすべての音を止めます。他のオブジェクトのオンプンによって再生されている音も停止します。

19　トケイン

指定した時間プログラムの処理を停止することが出来ます。

20　ズットン

この中にあるプログラミンを順番にずっと繰り返します。後述するヨブーンを使用しなければ、この繰り返しから抜けることは出来ません。

21　ナンカイン

この中にあるプログラミンを指定回数繰り返します。

ズットン、ナンカインでは繰り返し処理で 0.1 秒停止します。

22　イッペンニン

この中にあるプログラミンを同時に実行します。すべてのプログラミンの処理が終了すると、イッペンニンの処理が終了します。

イッペンニンの中で随時処理をしたい場合は、イッペンニンの中にナンカインを入れ、そのナンカインに随時処理させたい内容を入れ、繰り返し回数に 1 回を指定するといいでしょう。

23　ヨブーン

すべてのオブジェクトのプログラムの実行場所を、後述するハターンの場所へと変更します。対応するハターンが存在しない場合、実行場所は変化しません。

24　ハターン

ヨブーンによって呼び出される場所を指定します。ハターンは 1 から 12 までの内、1 つを指定します。このプログラミンは、ズットン、ナンカイン、イッペンニンの中に配置することは出来ません。また、プログラムの処理中にヨブーンが呼ばれずにハターンの場所に到達した場合、処理はその番号を呼ぶヨブーンが呼ばれるまで停止します。

25　クリックン

このオブジェクトがクリックされるまで待機します。

26　キーボン

キーボードのキーが入力されるまで待機します。スペース、上下左右の方向キーのどれかを指定することが出来ます。

27　ブツカッタン

このオブジェクトと、指定した絵が衝突するまで待機します。

28　タマーン

指定した絵を、指定した速度で発射することが出来ます。発射する絵の大きさや方向は、タマーンを呼ぶオブジェクトの大きさと角度に依存します。

補足

プログラミンに時間を指定する場合、0.1 秒単位で指定することが出来ます。また、0 秒を指定した場合、そのプログラミンは即時に実行されます。

早速遊んでみよう！

それでは、早速プログラミンでプログラムを作成してみましょう。まずは、図 1 の球が板の間を跳ね返るプログラムを作ってみます。

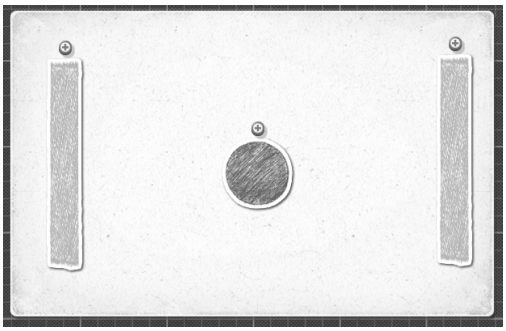

図 1: オブジェクトの配置。真ん中の球が左右の板で跳ね返るプログラムを作る

初めてのプログラミンプログラミング

プログラミンの公式ページ[3]を開いたら “プログラミンであそぶ” をクリックするとプログラミンの Flash が開きます。“使い方をおぼえる” をクリックすると、プログラミンの基本的な操作を動画で確認することが出来ます。また、“おてほんであそぶ” をクリックすると、予め用意されたサンプルを見ることが出来ます。“プログラムをつくる” をクリックして、“新しいプログラムをつくる” となっているところをクリックすれば、

[3] http://www.mext.go.jp/programin/

プログラムの編集画面になります。

まず、デフォルトでいる犬をクリックして選択し、“絵をすてる” をクリックして捨ててしまいましょう。次に、“絵をたす” をクリックして適当な絵を選択し、追加します。このサンプルでは、“かたち” の丸と正方形を適当に拡大・縮小して使用しています。

あとは、図 1 のとおりに絵を配置し、図 2 の通りに画面の下のプログラミンをドラッグして丸のオブジェクトに追加し、移動量などの数値を調節すれば完成です。再生ボタンを押して実行してみましょう！

図 2: プログラム (……？) のソース。プログラミンは下から順番に実行される

こんなプログラムで大丈夫か？

一応、球が板の間を跳ね返っています。ですが、これでは弾は板と衝突して跳ね返っているのではなく、決められた距離を進んでいるだけに過ぎません。板をずらすと正しく跳ね返らなくなります。こんなひどいプログラムはさっさと投げ捨てて、ちゃんとしたプログラムを作り直しましょう。

一般的なプログラミング言語だと、条件分岐を使用して弾と板がぶつかった時に球の移動方向を変えればいいです。しかし、残念ながらプログラミンには条件分岐がありません。しかし、プログラミンにはイッペンニンがあります！イッペンニンに、ずっと右方向に球を進ませる処理と、板とぶつかったら球を左方向に進ませる処理にジャンプする処理を入れればいいでしょう。実際のプログラムは図 3 のようになります。これで球がきちんと板とぶつかったか判断して移動方向を変える、ちゃんとしたプログラムになりました。

一番ヨブーンを使わないプログラムを頼む

ところで、ハターン、ヨブーンでも解説しましたが、ハターンには 12 通りの番号しかありません。またハターンの番号は、このプログラムの全てのオブジェクトで共有されています。そのため、ハターンの番号は保護が必要なほど貴重な資源なのです。

ところが、あろうことか、ただ単に板の間を往復する、というだけの簡単なお仕事で既にその貴重なハターンを 2 個も消費してしまっているのです！そんなプログラムを「大丈夫だ、問題ない」と片付けているとプログラムを作っているうちにハターンが足りなくなるのは火を見るよりも明らかです。なので、ヨブーンを使わないプログラムに書き換えましょう。

図 3: ちゃんとしたプログラムのソース。長い……

ところで、なぜヨブーンを使用しなければならなくなったのでしょうか？それは、プログラミンが、直接状態を保持するための手段を持たないからです。簡単に言えば、プログラミンには変数がないからです。変数があれば、ヨブーンを使う代わりに変数の値を変更し、進行方向を変えることが出来ます。しかし、残念ながらプログラミンには変数がないため、そのようなことをすることが出来ません。

しかし、他の手段を使うことによって、プログラミンでも状態を保持することが出来ます。主な手段は次の通りです。

- プログラム自体で状態を保持する
- オブジェクトが表示する絵を変更することで状態を保持する
- オブジェクトの位置によって状態を保持する

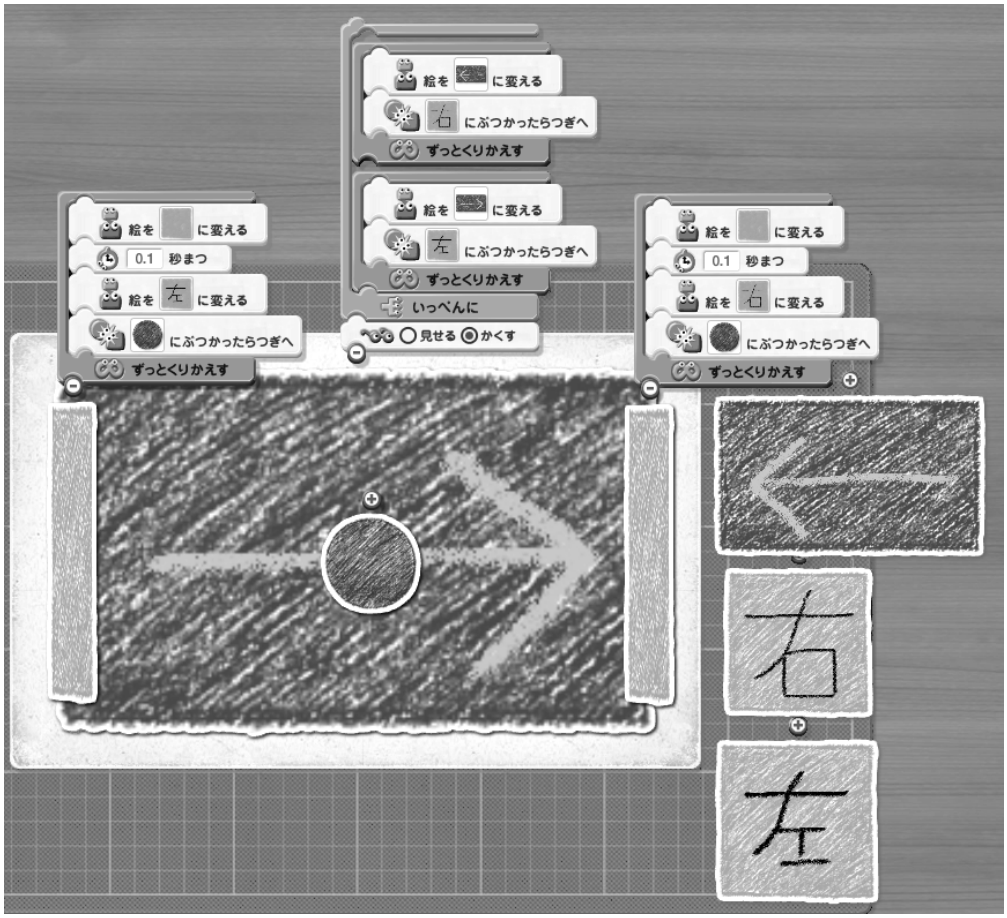

図 4: 状態を保持するオブジェクトと、板の配置とプログラム

1 番目の方法は、図 2 で作成したプログラムで取った手法です。右に進むプログラム、左に進むプログラムをヨブーンを使って切り替えることで、2 つの状態を 1 つのプログラムの中で保持することが出来るようになっています。

しかし、この方法の欠点として、プログラムが長くなる、同じようなプログラムを複数書かなくてはならない、たいていの場合はハターンとヨブーンを使用しなくてはならない、ということが挙げられます。

2 番目の方法は、キガエルンを使用して自身の絵を変化させ、状態の保持や通知を行うというものです。状態を得る必要がある絵に、常に衝突しているような巨大なオブジェクトを用意してそのオブジェクトが表示している絵を状態によって切り替えることで、ヨブーンを使わなくてもイッペンニンとブツカッタンで処理を分ける事ができるようになります。

この方法の欠点としては、現在自分はどの絵を表示しているのかを知ることが出来ないため、現在の状態を元に新しい状態を得る場合はとても面倒になってしまう[4]ことが挙げられます。

今回はこの手法を使用します。

球に状態を伝えるためにブツカッタンを使用するので、球と状態を保持するオブジェクトは常に接触している必要があります。そのため、球が動く範囲全てをカバーするようにオブジェクトを拡大・縮小して調整します[5]。そのオブジェクトが表示する絵に、板と球が接触したときにその情報を伝える必要があるため、板が球と衝突した場合、一瞬だけ絵を変えて衝突したことを通知します[6]。状態を保持するオブジェクトがその通知を受けたときに、表示する絵を変更します。

球は、ブツカッタンを使用して状態を受け取り、その方向に移動するだけとなります。

この方法だと、左右だけでなく上下にも跳ね返らせたい、というときでも単純に状態保持用のオブジェクトとそこに衝突を通知するための板を新しく追加し、球に少し編集するだけで実現できます[7]。もしも 1 番目の方法で上下にも跳ね返らせたい、となった場合、右上、右下、左上、左下に移動するときのプログラムをつくり、それぞれにヨブーンを使って移動することになります[8]。

[4] ブツカッタンは他の絵と衝突しているかどうかしか判別できないため、現在自分が表示している絵を他のオブジェクトで表示して、それをブツカッタンで判別する、という方法があります

[5] 邪魔に見えるかもしれませんが、ミエルンの効果によって実行時には見えなくなります

[6] このせいで、一瞬板の表示が変わってしまいます。気になる人は、この上にただの絵を置いて隠すなり、ミエルンを使って見えなくするなりしてください

[7] 次のページでプログラムを見ることが出来ます。紙面の都合上、短縮 URL を使用しています。http://bit.ly/ciypxt

[8] これだけでハターンが 4 個消費されてしまう……

図 5: 球のプログラム。短い！簡単！

そんな球をずっと見ていても大丈夫か？

いくらサンプルプログラムといえど、ずっと球が跳ね返り続けているのを見ていると飽きてしまうので、何回か跳ね返ったら終了するようにしましょう。

というわけで、球が板に 10 回衝突して跳ね返ったら停止するようにします。ただ停止するだけでは面白くないので、せっかくなのでフキダシンを使って “Hello,World!” と喋らせましょう[9]。

そのためには、球が板に何回衝突したのか覚えておかなければなりません。2 番目の方法では、現在の状態を更新することが難しいので適していません。1 番目の方法でも、ヨブーンを使わずに済む方法[10]はありますが、せっかくなのでここは 3 番目の方法を使用しましょう。

3 番目の方法とは、オブジェクトの位置によって状態を保持する、という方法です。詳しく言うと、状態変化の通知を受けた場合にオブジェクトを移動することで状態を更新する、という方法です。そのため、新しい状態を得るのに現在の状態が必要な場面[11]で適しています。また、この方法を応用することで、簡単な演算[12]をすることも可能となります。

欠点として、オブジェクト数が増えるためプログラムが複雑になることと、オブジェクトの位置関係がプログラムの動作に影響するため、通常以上にオブジェクトの配置に神経を使う必要がある、ということです。

球と壁が何回衝突したかを記憶し、判定するオブジェクトは図 6 のようになっています。球と壁が衝突したことが下のオブジェクトに通知されると、下のオブジェクトは少しずつ上に進んでいきます。そして、球と壁が 10 回衝突したとき、上のオブジェクトの位置を下のオブジェクトと衝突するような位置に調整しておきます。そして、上のオブジェクトは、下のオブジェクトと衝突した場合、ヨブーンを呼んで終了処理を呼びます[13]。これらのオブジェクトは画面外にあるので、特にミエルンなどで不可視設定にする必要はありません。

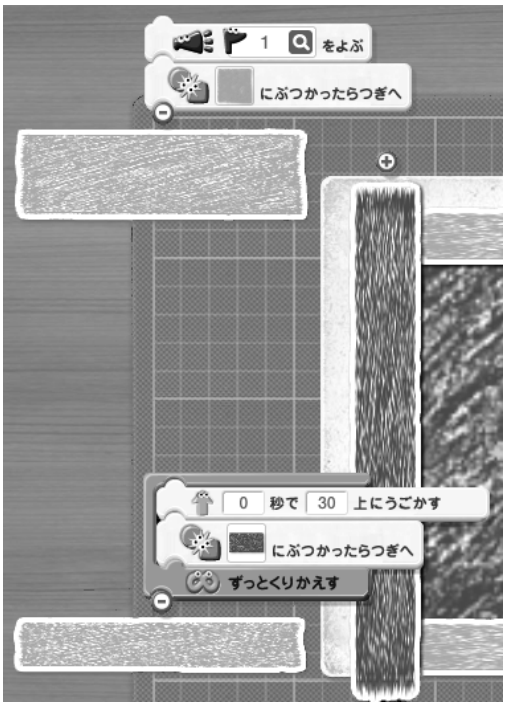

図 6: 下が状態を保持するオブジェクト。下のオブジェクトが 10 回上に移動すると、上のオブジェクトと接触する

球と壁が衝突したことを、回数を記憶するオブジェクトに通知しなくてはいけません。その通知するためのオブジェクトは、図 7 のようになっています。それぞれ、板と接触して板が通知する球との接触を受け取ったとき、回数を記憶するオブジェクトに自身を衝突させて球が板と衝突したことを通知します。これらのオブジェクトは、ミエルンで見えなくなっています。

このプログラムは、次のページ (http://bit.ly/c2cri3) で見ることが出来ます。

[9] “Hello,World!” を先にやるべきな気がするけど、気にしない

[10] ナンカインを使用して 10 回ブツカッタンが呼ばれるまで待つようなプログラムならヨブーンを使わなくても済みます。が、終了条件の設定がナンカインの繰り返し数の設定となっているので、プログラムから操作することが出来ません……

[11] 今回のような物を数える場面など

[12] 加減算や比較など。がんばるとライフゲームも実装できます。たぶん

[13] この処理も、全てのオブジェクトと接触するような不可視の巨大なオブジェクトを作れば、ヨブーンを使わずに処理することが出来ます。しかし、そのような構造にしてしまうとあまりにも複雑すぎるので、たまにはこういった妥協も必要です。

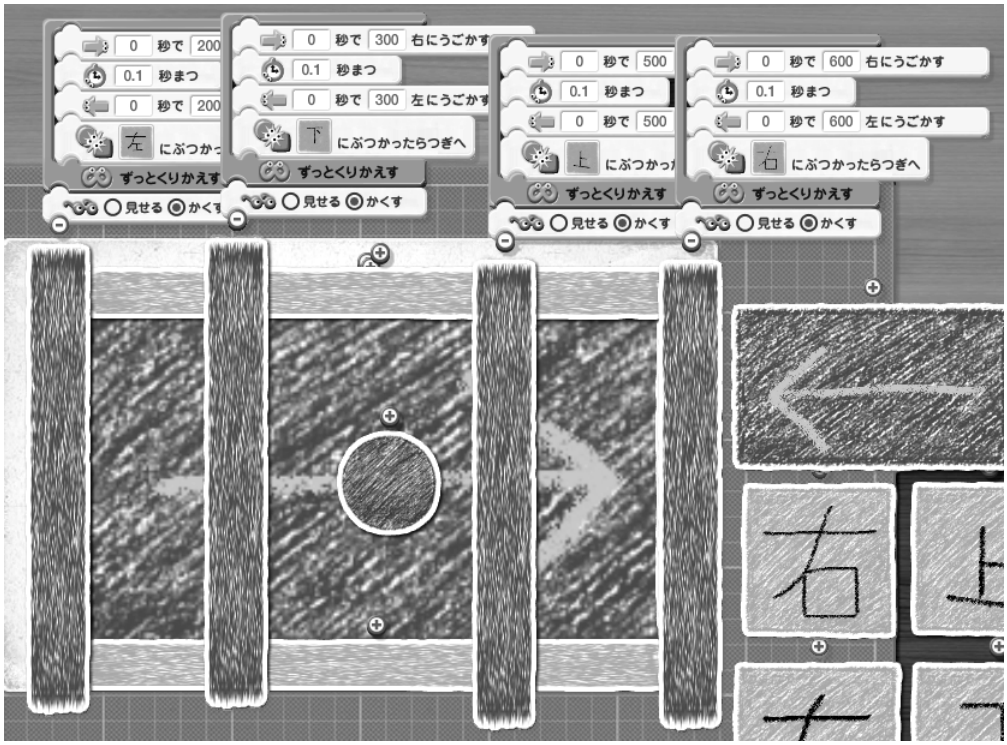

図 7: 球と板が衝突すると、それぞれ状態を保持しているオブジェクトに衝突してそのことを通知する。このように 4 つばらばらにするのではなく、ひとつの大きなオブジェクトに全ての板を接触させ、イッペンニンを使用してひとつのオブジェクトで通知する方法もある

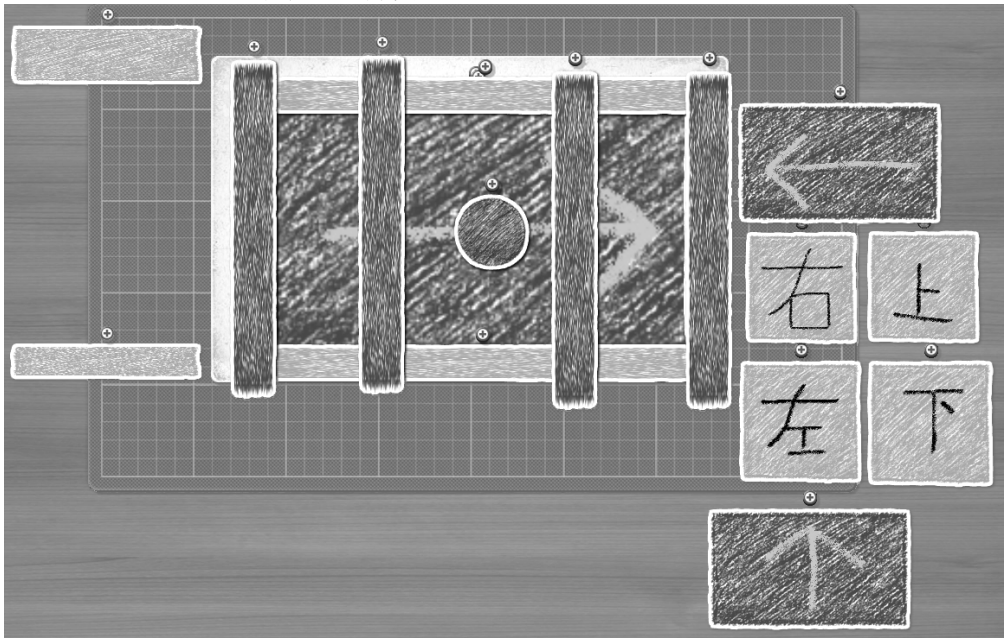

図 8: このプログラムのオブジェクト配置図。右にあるオブジェクトはキガエルン用のプログラムを設定していないただの絵

覚えておくといいかもしれないこと

プログラミンで遊んでるうちに気づいたちょっとしたことを書いておきます。

四角の描き方

プログラミンでは、プログラムの構造をオブジェクトの衝突などを使ってあらわし、絵単位でしか衝突の判定が出来ないことから、多くの種類の四角い絵が必要となる機会があります。しかし、今回のサンプルで行ったような、“絵をたす”、“かたち” にある正方形や長方形に “絵をかきたす” を使って多くの種類を作るのは推奨できません。プログラミンが扱える絵のサイズに制限があるので、そんなに多くの種類を作ることは出来ないからです。それに、最初から用意されている正方形や長方形は若干大きめに作られているので、縮小して使う必要があります。

そのため、最初から小さい四角形を自分で用意する必要があります。しかし、プログラミンの “絵をかく” では、クレヨンのように描いた線の右側と下側はガタガタになってしまいます。そこで、少し大きめの四角形を描いた後消しゴムで右と下の辺を消すと、きれいな直線にすることが出来ます。

指定した時間をかけて行う処理

処理時間を指定することが出来る処理[14]では、0 秒から 0.1 秒刻みで秒数を指定することが出来ます。

その処理中にヨブーンが呼ばれ、ハターンの位置に処理が移ったとしても、その処理は最後まで実行されます。たとえば、2 秒かけて 200 進む、というウエーンの処理を開始してから 1 秒後にヨブーンが呼ばれた場合、100 進んでとまるのではなく、バックグラウンドで 2 秒かけて 200 進む処理が続いています。

ブツカッタンと各プログラミン

ミエルンで不可視状態になっていても、ブツカッタンでの衝突判定は行われます。

カメロンでオブジェクトの色相を変えても、ブツカッタンでの衝突判定は同じ絵として扱われます。

穴の開いた絵とブツカッタン

穴の開いた絵の中に入っているオブジェクトで、ブツカッタンでその絵と衝突判定を行った場合、衝突していないと判定されます。しかし、穴の外側での判定と違い、衝突判定の精度が低いのであまり使用しないほうが無難でしょう。

円運動をさせたい

プログラミンには円運動をさせる命令がありません。どうしてもオブジェクトに円運動をさせたい場合、上下左右移動と回転を使って手作業で円運動を指示する必要がありまあ巣。

ところで、オブジェクトを回転させるミギクルリン、ヒダリクルリンの回転軸は、オブジェクトの中心となっています。そのため、円運動させる絵の回転軸と、絵の中心が一致するようにうまく小さいゴミを置くと、回転を使って擬似的に円運動を再現することが出来ます (http://bit.ly/csyMNF)。スケールンの拡大・縮小の中心も絵の中心なので、同じことが言えます。

絵のコピーでプログラムのコピー

“絵のコピー” を押すと絵がコピーされますが、一緒にプログラムもコピーされています。そのため、ナンカインやズットンなどの中に入っているプログラムは、ナンカインやズットンごと別のオブジェクトに移動、というようにすると簡単にプログラムをコピーすることが出来ます。稀にコピーしたプログラムの挙動がおかしかったり、編集できなかったりする場合がありますが、そういう場合は保存して閉じて、開きなおすと大丈夫です。

不可視オブジェクトもサムネイルに表示されてしまう

ちょっと舞台裏が見えてしまって恥ずかしい。それでネタばれなんかしちゃったときには……。嫌な人は、不可視オブジェクトは画面外においてプログラム開始時に画面内に移動させましょう。

それでは、プログラミンで衝突指向プログラミングをたのしんでください！

[14] ウエーンなどの移動、ミギクルリンなどの回転など