プログラミングで
図形を動かしてみよう！

目的・ねらい

シンプルなプログラムを理解し、自分で変更したパラメータによって図形の動きが変化することを体験することで、重度視覚障害があってもグラフィカルなプログラミングを自作する可能性があることを感じて欲しいと考えています。DXruby を使い、短いコードで動くものを体験します。

実施内容の概要

ruby およびライブラリの DXruby を使って、キーボード入力に反応して図形が動いたり音が再生されたりするプログラムを理解し、自分でパラメータを変更して点図ディスプレイの DV-2 で動きを確認します。

講師用の実施手順の詳細

準備することがら、物品など

- コンピュータに Ruby をインストールし、DXruby を展開しておきます。
- .rb のファイルの規定のプログラムをマイエディッ

トやメモ帳にしておきます。

- DV-2 のドライバと GView をインストールしておきます。DV-2 の Ver.1 を Windows10 で利用する場合は署名なしドライバのインストールを許可しなくてはならないので、セキュアブートを OFF にした後にコマンドラインで
  bcdedit /set TESTSIGNING ON
  と打ち込み、設定の更新とセキュリティ、回復メニューから PC の起動をカスタマイズし署名なしドライバをインストールできるようにする…など、多少面倒な作業が必要になります。
- DV-2 に表示するプログラム GView の動作確認をしておきます。COM ポートは一回指定すれば次から正しいものを最初に選んでくれます。設定としてはフォーカスの当たったウィンドウだけを表示するモードにし、リフレッシュレートを 0.1 秒にしておきます。
- ユーザーディレクトリの下にサブディレクトリを作り、その下にサンプルプログラムをすべてコピーしておきます。もしくは USB メモリなどに入れておきます。

実施手順の詳細

図形を描くコードを学ぶ

Windows キーを押した後に「cmd」と入力してコマンドラインを開きます。
「cd サブディレクトリ名」もしくは「ドライブレター:」などでサンプルのある場所に移動します。サンプルは数字で始まるプログラム名などにしておき、直接入力したり Tab キーのファイル名補完などを使ったりしてコマンドラインから開くようにします。

例えば「1.rb」というファイル名をつけていた場合「数字の 1 を押して Tab キーを押してから Enter キーを押してください」と伝えることで、メモ帳でソースコードを開くことができます。（あらかじめ規定のプログラムをメモ帳にしておく必要があります）

最初のサンプルは以下の四角形を描くコードです。開いたら順に説明します。

```
#まずdxrubyというライブラリを使えるようにする。
require 'dxruby'
#Imageオブジェクトを生成。
img1 = Image.new(640, 480, [0, 0, 0])
#白い四角形をimg1に描く。
img1.box_fill(100,100,250,200,[255,255,255])
#ここから無限ループスタート。
Window.loop do
  #img1領域を描く。
  Window.draw(0, 0, img1)
end
```

- シャープ（＃）以降はプログラムとは関係のないコメントすなわち注釈になる。
- require は「要求する」という意味。dxruby というライブラリを使えるように宣言している。
- ライブラリとは、外部から呼び出せるプログラムをひとまとめにしたものである。
- まず「新しい Image オブジェクト」Image.new という命令でを作る。オブジェクトというのはプロ

グラムの中の「意味のある塊」のこと。とりあえず Image オブジェクトは絵を描くキャンバスのようなものだと説明しておく。

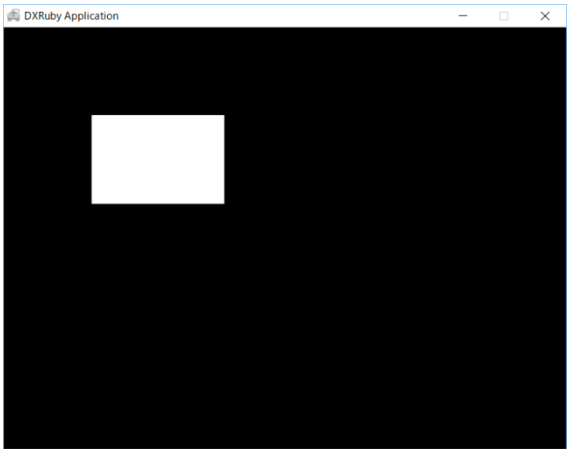
- 最初の数字は横幅、次の数字は高さ、角カッコで囲われた数字は背景色を表す。色は順に赤、緑、青の色の強さの意味であり、255 が最大値である。すべて 0 だと黒を意味する。
- 生成した Image オブジェクトに img1 という名前を付けている。イコールは名前をつけるという意味である。
- 次に img1 オブジェクトに box_fill という命令で四角形を描く。命令のことを「メソッド」と呼ぶ。左上の座標と右下の座標、色を指定する。
- オブジェクトに対する命令はドットでつなげる。
- Window.loop do から end までは無限に繰り返される。Window オブジェクトに対して何も指定しなければ幅 640 ドット高さ 480 ドットのウィンドウが生成される。
- 無限ループの中で draw という命令を使って img1 を描いている。左から 0 ドット、上から 0 ドットの位置を指定したことになる。その位置が img1 の

左上の点になる。位置をずらすこともできる。

これらの解説を終えたら、実行してみます。

`ruby` **プログラム名**

と打ち込ませます。適宜ファイル名補完を使わせた方が確実です。「rubii」などと打ってしまう場合があるので、「アール、ユー、ビー、ワイ、数字、タブキー、エンター」と最初は細かく指示する方が良いでしょう。プログラムを動かすと、下のようなウィンドウが現れます。終了は Alt+F4 です。



四角形を表示したウィンドウ

DV-2 を動かす

ここで Windows キーを押し、「GView」と打ち込んで GView を起動します。移動する図形の表示のために 0.1 秒のリフレッシュレート設定を確認してください。GView が問題なく起動したら、本体の拡大・縮小キーと側面にある上下左右のカーソルキーの使い方を説明します。手前のジョイスティック部分は古い機種はバー状になっていて誤操作してしまうかもしれないので、周辺を押すように指導します。また、拡大縮小のボタンと間違えて横のステータス表示のボタンを押してしまう場合があります。慌てずにもう一度同じボタンを押すことをここで教えておきます。ステータス表示で誤操作をしてしまうと、DV-2 の表示閾値を変えてしまったりすることもあるので注意します。あらかじめ硬い紙などで覆っておくのも有効だと思います。

図形を加えてみる

線や円など、他の図形を描いてみます。img1.box_fill の行に続いて下記コードを box_fill の行の下に入力させ、表示させます。キーボードの打ち込みに不安

がある場合は、テキストファイルを用意しておいて、全選択、コピー、ペーストなどを指示します。

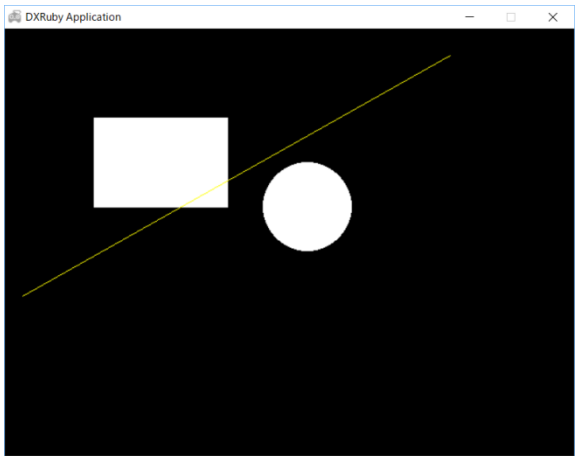
説明のポイントは数値の意味になります。

```
#円を描く。
img1.circle_fill(340, 200, 50, C_WHITE)
#線を描く。
img1.line(20,300,500,30, C_YELLOW)
```

- circle は円、fill は塗りつぶしの意味。
- circle_fill(x 座標, y 座標，半径，色)
- line(端点の x 座標，端点の y 座標，異なる端点の x 座標，異なる端点の y 座標，色)
- C_WHITE は白、C_YELLOW は黄色。

これらの意味を理解させた後で、自分で数値を変えて実行させます。

課題としては、見える生徒さんに対しては色を用いて国旗などを描かせたり、全盲の生徒さんに対しては雪だるまなど、四角形と円で描ける触察可能な図形を作らせたりすると良いと思います。



いくつか図形を追加したウィンドウ

画像を読み込んで表示してみる

ここまではImageオブジェクトをキャンバス代わりに使ってみましたが、実際には画像ファイルを読み込んで表示したりする「画像の入れ物」として使います。次のサンプルを確認し、動かしてみます。

```
require 'dxruby'
#test.pngという画像ファイルを読み込んでimg1という名前をつける。
img1 = Image.load('test.png')
#ここから無限ループスタート。
Window.loop do
  #img1を指定した座標を左上として描画する。
  Window.draw(100, 100, img1)
end
```

新しい部分はImageオブジェクトのloadメソッドだけです。ここを説明した後で、表示する座標値などを変えさせてみてください。画像ファイルには、生徒の年代が皆知っているキャラクターのなかでDV-2に表示しやすい明るい色のものなどを選ぶと良いでしょう。

キーボード入力

キーボードの矢印キーの入力についてはInput.xとInput.yで取得できます。ジョイスティックがつながっている場合はジョイスティックの動きが反映されます。次のサンプルを開いて中を確認します。

```
require 'dxruby'
#test.pngという画像ファイルを読み込んでimg1という名前をつける。
img1 = Image.load('test.png')
#画像の触察が難しいようならDV-2用に以下の行を使う。
#img1 = Image.new(50, 50, C_WHITE)
#xとyの値をゼロで初期化する
x=0;y=0;
```

```
#ここから無限ループスタート。
Window.loop do
  #x と y を更新してその座標で img1 を描く。
  x=x+Input.x;  y=y+Input.y;
  Window.draw(x, y, img1)
end
```

- img1 の生成まではこれまで学んでいる。
- x と y は中の数値が変わる入れ物で「変数」と呼ぶ。最初にゼロを代入して初期化している。
- Window.loop do～end の無限ループの中で x と y は変化する。
- もとの値に Input.x や Input.y で得られる値を加えている。これらは矢印キーが押されていると +1 や -1 になる。

説明を終えたら「もっと速く動かすには」「遅く動かすには」という課題を解かせます。適宜 Input.x が変化分であることをヒントに与えたりします。

跳ね返るボール

座標値を一定割合で変化させることで動くボールを表現してみます。以下のサンプルコードを流れに沿って確認します。

```
require 'dxruby'
img1 = Image.new(30, 30, C_BLACK)
img1.circle_fill(15,15,15,C_WHITE)
x=0;
#ここまでは既に学んだ内容。次の dx は x 方向の変化分。
dx=1
#円を縦方向の真ん中に表示するために以下の式で y を決める。
y=(Window.height-img1.height)/2
#無限ループスタート
Window.loop do
  #x を dx だけ変化させるために x=x+dx とする。x+=dx という書き方もある。
  x=x+dx
```

```
  #x がウィンドウの横幅をはみ出そうになったら方向を変える
  if(x>Window.width-30)then
    #方向を変える。ここは dx*=-1 と書いても良い。
    dx=dx*(-1)
  end
  Window.draw(x, y, img1)
end
```

- dx という変数が新たに加わった。これは変化分を意味する変数。初期値は+1。
- y は固定。ウィンドウの高さの半分から img1 の半分の高さを引いた座標。これにより図は真ん中に来る。
- 無限ループの中で x=x+dx とすることで右に移動。
- 右端で跳ね返るために「条件分岐」を行う。条件分岐とは、「もし～なら○○する」という命令を実現するものである。
- 条件分岐は「x 座標がウィンドウの右端の座標-img1 の横幅より大きくなった場合」に変化分を左方向にすること。
- 右方向への変化（dx=1）から左方向への変化は-1

を掛け算すると実現できる。

しかしこのままだと、右で跳ね返った後、左側に消えていってしまいます。そこで、「x 座標がゼロより小さくなったら」という条件を加えさせます。

```
if(x>Window.width-30 || x<0)then
```

「この縦棒 2 つは、「もしくは」という意味である。ここで dx=dx*(-1)の意味を再確認する。左方向への移動から右方向への移動に変わるのも、実現できている。」ということを理解させます。

ここまで来たら生徒の進度や理解度を確認しつつ「y 方向にも同様に変化するプログラムにせよ。」という課題などを出してみるのも良いでしょう。

```
require 'dxruby'
img1 = Image.new(80, 80, C_BLACK)
img1.circle_fill(40,40,40,C_WHITE)
x=0;dx=3
y=(Window.height-img1.height)/2
dy=3
```

```
Window.loop do
  x+=dx;y+=dy

if(x>Window.width-img1.width||x<0)then
    dx*=-1
  end

if(y>Window.height-img1.height||y<0)the
n
    dy*=-1
  end
  Window.draw(x, y, img1)
end
```

音を鳴らす

スペースキーを押したら音が再生されるサンプルコードを開いて中身を確認します。

```
require 'dxruby'
#Soundオブジェクトを生成してsnd1という名前をつける。
snd1 =
Sound.new("C:¥¥Windows¥¥Media¥¥ding.wav")
#無限ループスタート。
Window.loop do
  #もしSPACEキーが押されたらsnd1を再生する。
  if Input.key_push?(K_SPACE) then
     snd1.play
  end
end
```

- Sound.new でサウンドオブジェクトを生成、snd1という名前にしておく。
- Input.key_push?という命令で特定のキーが押されたかどうか判断がつく。
- スペースキーが押されたならば Sound オブジェクトの play メソッドを呼び音を鳴らす。

プログラムが理解できた後で、snd1=…の行をコピーして snd2=…とし、if 文もコピーして異なるキーを押すと異なる音が鳴るプログラムを作成させます。キーコードは K_A や K_B などを使わせます。

組み合わせ課題

跳ね返るボールのプログラムをベースにして、

- 壁で跳ね返る時に音が鳴る
- 上下矢印キーで速度が変わる

という機能を追加したプログラムを作らせます。

注意すべき点

- 進度や理解度の差が出やすいので、進んでいる生徒さんには比較的自由にやらせて構いません。遅い生徒さんのフォロー時には、

到達目標

- サンプルプログラムの流れを理解する。
- プログラムの中の数値の意味を理解し、自分の思った通りの動きをさせられるように編集することができる。

生徒用資料

生徒用には、サンプルプログラムをあらかじめコンピュータに入れておきます。