

# プログラミング概論

第11回 2024年12月4日

App Inventorによる

Androidアプリ開発の実践

(5) 物理シミュレーション2

# 今回の授業内容

今回は**放物運動**のシミュレーションを行うことを目的とする。  
放物運動とは**空中に投げられた物体の運動**のことである。  
重力により、物体には鉛直下向きに行こうとする力が常にかかっている。

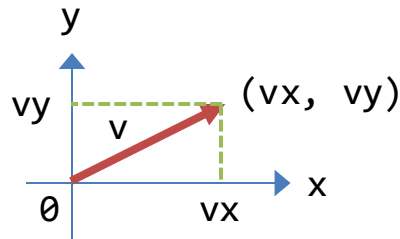
空気抵抗などの重力以外の力を無視し、投げられたものの本質的な動きだけをシミュレートする。

- 等速直線運動
- 等加速度運動（水平方向）
- 落体の運動（鉛直方向の等加速度運動）
- 放物運動
- 的当てゲーム（放物運動版）を作る

# 放物運動とは

# 復習

## コンピュータの画面上で「動き」を表現するには



1秒間に動く量

t秒後の位置は？

$$x(t) = x(0) + v_x \times t$$

$$y(t) = y(0) + v_y \times t$$

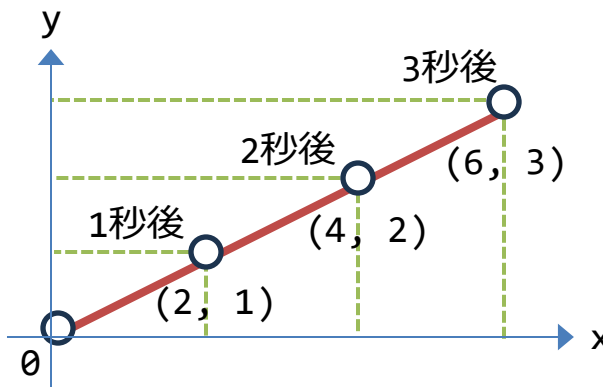
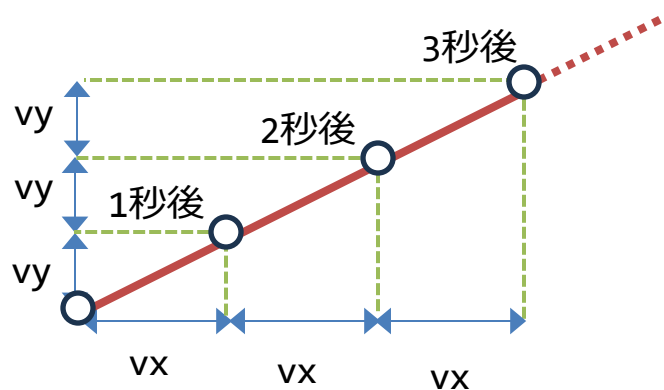
↓

速度が一定でないとダメ

↓

$$x_{(t+1)} = x(t) + v_{x(t)}$$

$$y_{(t+1)} = y(t) + v_{y(t)}$$



$$x_{(4)} = 0 + 2 \times 4 = 8$$

$$y_{(4)} = 0 + 1 \times 4 = 4$$

いまの時刻tから**1秒後**の位置を求める.

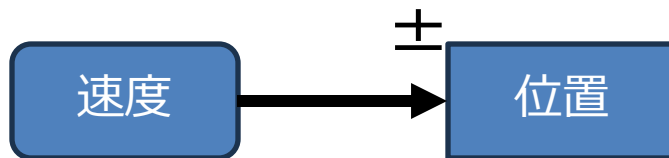
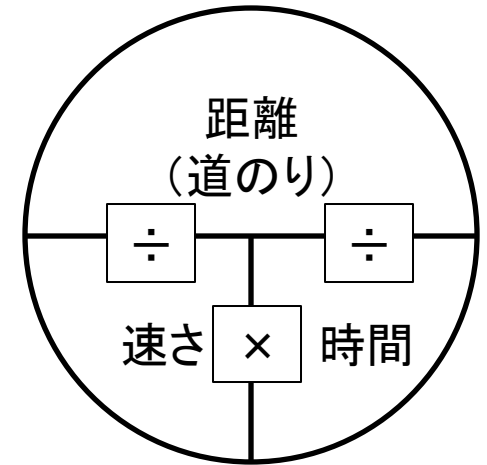
$x(t)$ と $y(t)$ はいまの位置,

$v_x(t)$ と $v_y(t)$ は**1秒間**に動く量を表している.

## 復習

# “動き”のシミュレーション

- 一回あたりの移動量 = 速度
- 毎回速度 × 時間を位置に加えればよい



$$x' = x + v_x \times \Delta t$$

(元の $x$ に速度 $\times$ 時間を足したものが次の $x$ )

$$y' = y + v_y \times \Delta t$$

(元の $y$ に速度 $\times$ 時間を足したものが次の $y$ )

# 等速直線運動

本来は 速度×時間＝距離（位置の変化）

- 1step "1秒" という仮定で時間は省略していた  
（位置＋速度という計算はできない）
- より正確な表現には"微小時間 $\Delta t$ "を考える

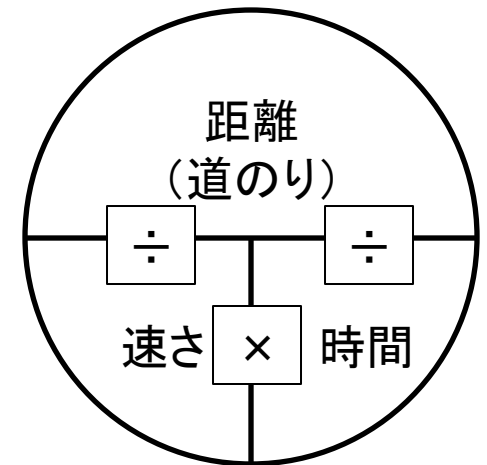
$$x_{(t+1)} = x_{(t)} + v_{x(t)}$$

$$y_{(t+1)} = y_{(t)} + v_{y(t)}$$

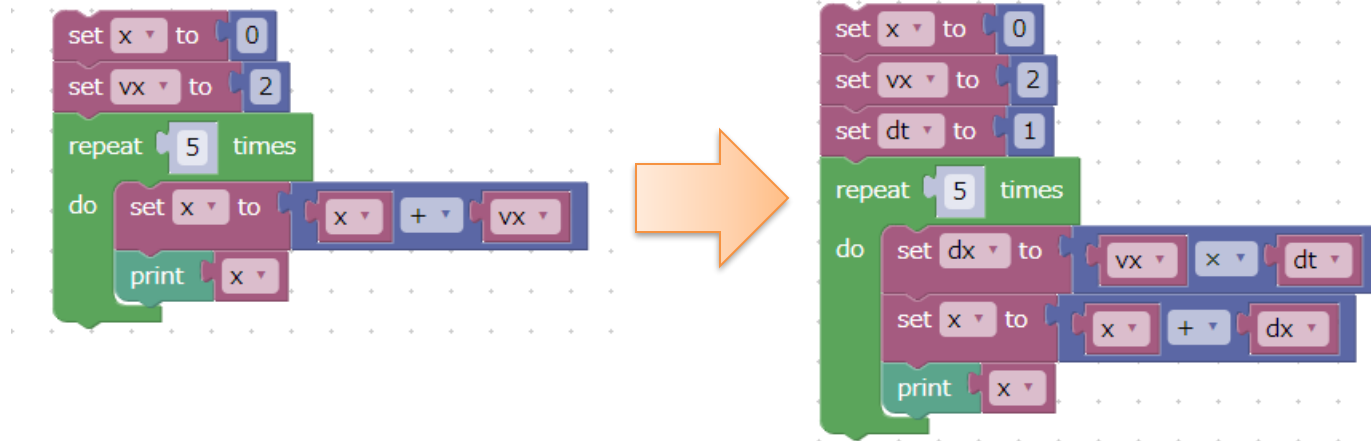


$$x_{(t+1)} = x_{(t)} + v_{x(t)} \times \Delta t$$

$$y_{(t+1)} = y_{(t)} + v_{y(t)} \times \Delta t$$



# 等速直線運動（改）



- 変更点

1. 微小時間 $dt$ の間の位置の変化量 $dx$ を速さ $v_x$ を用いて

```
set dx to vx * dt
```

としていること

2. この位置の変化量 $dx$ を用いて

```
set x to x + dx
```

によって位置情報を更新していること

[Blockly Code](#) で確かめよう

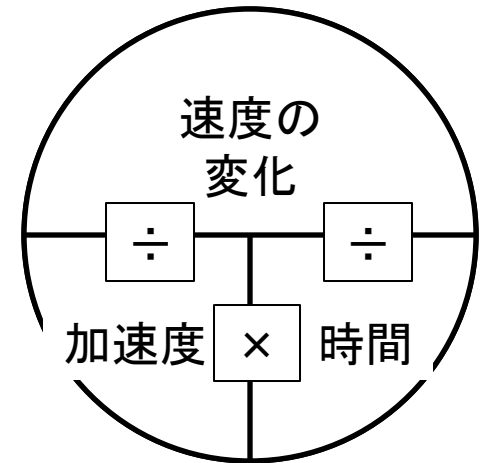
# 等加速度運動

速度が変化する割合を加速度という

速度×時間 = 距離 (位置の変化)

↓ 同様に考える

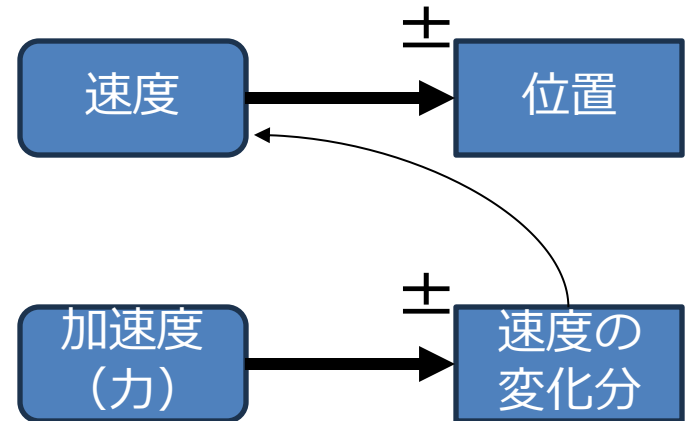
加速度×時間 = 速度の変化



$$x_{(t+1)} = x_{(t)} + v_{x(t)}$$



$$v_{x(t+1)} = v_{x(t)} + a_{x(t)}$$



(加速度は力 ÷ 質量)



# 等加速度運動（水平方向）

加速度とは「単位時間あたりの速度の変化量」を指す。自動車でいえば、アクセルの踏み込みに相当する。秒速10mの速さ（※1秒あたり10m進む速さ。1時間=3600秒に換算すると36000m=36Km,つまり時速36Kmのこと）で走っていたところで、アクセルをぐっと踏無ことで秒速20m（時速72Km）になったとしよう。もし1秒間の間に10m/sの速さから20m/sの速さに変化したのであれば、加速度は $(20-10)/1=10$ （単位は $(\text{m/s})/\text{s}=\text{m/s}^2$ ），もし10秒間で20m/sの速さに達したのであれば、加速度は $(20-10)/10=1\text{m/s}^2$ ということになる。これを数式で表すと次のようになる。

$$\text{加速度} a = (v_2 - v_1) / t \quad (4)$$

この式を変形すると $v_2 - v_1 = at$ となる。先と同様に、微小時間 $\Delta t$ の間に生じた微小な速さの変化 $\Delta v$ として式を書き直すと

$$v_2 - v_1 = \Delta v = a\Delta t \quad (5)$$

となる。

等加速度運動をプログラムとして表現するために次の段を踏むこととする。

- ① 加速度から速さの変化量を求める
- ② 速さの変化量に基づいて、速さの値を更新する
- ③ 更新された速さに基づいて移動距離の変化量を求める
- ④ 移動距離の変化量に基づいて、位置の値を更新する

# 等加速度運動（水平方向）

①表示される値を考えてみよう

②aが-1のとき表示される値を考えてみよう

```
set x to 50
set vx to 0
set dt to 1
set a to 1
repeat 5 times
do
  set dv to a x dt
  set vx to vx + dv
  set dx to vx x dt
  set x to x + dx
  print x
```

①

/	/	/	/
---	---	---	---

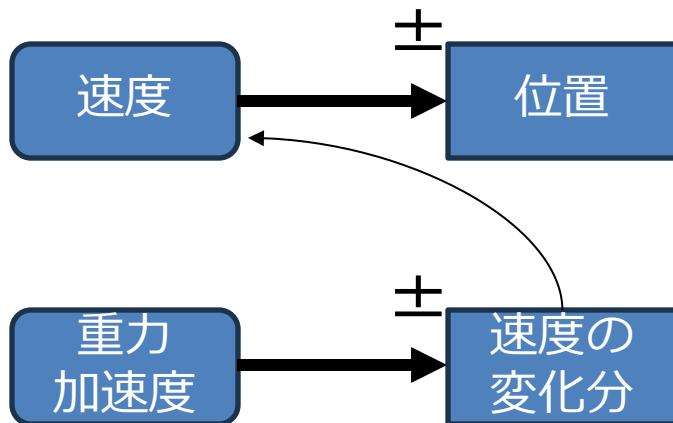
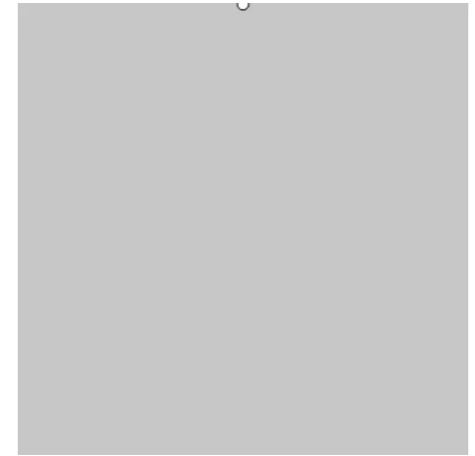
②

/	/	/	/
---	---	---	---

加速度は負の値になっても良い。一定の速さで動いていけば、負の加速度は減速の効果をもたらす。速さが0になっても負の加速度が与えられたままであれば、そこから逆向きの加速度運動になる。

# 自由落下 (初速度0)

- 重力により速度が増える
- “重力加速度”を考える



$$y' = y + v_y \times \Delta t$$

(元の $y$ に速度 $\times$ 時間を足したものが次の $y$ )

$$v_y' = v_y + a \times \Delta t$$

(元の $v$ に**加速度** $\times$ 時間を足したものが次の $v$ )

# 落体の運動（鉛直方向の等加速度運動）

重力加速度：約 $9.8\text{m/s}^2$

- 1秒ごとに移動速度が $9.8\text{m/s}$ ずつ増えていく，という量
- 車になぞらえれば，止まっている状態から2秒間で時速約 $70\text{km}$ に達するような量

```
set x to 50
set y to 50
set vx to 0
set vy to 0
set dt to 1
set a to 1
repeat 5 times
do
  set dv to a x dt
  set vy to vy + dv
  set dy to vy x dt
  set y to y + dy
print y
```

なお，このシミュレーションでは，プログラムの分かりやすさと可視化を重視し，現象のリアリティについては厳密性を求めないこととする．つまり，どれくらいの長さを $1\text{m}$ と見立てるか？といった換算は行わない．よって，重力加速度も $9.8$ という値を用いず，見易さを重視し，小さな値を用いることとする．

y軸方向の初期速度として負の値（例えば $vy=-10$ ）にするとどのような動きを示すか考えてみよう

# 水平投射

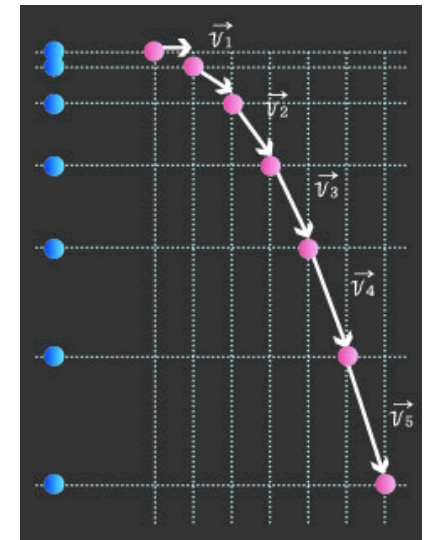
- yは自由落下
  - xは等速直線運動 (例: 40)
- xとyを別々に計算して良い

$$x' = x + v_x \times \Delta t \quad y' = y + v_y \times \Delta t$$

$$v_x' = v_x + a_x \times \Delta t \quad v_y' = v_y + a_y \times \Delta t$$

||  
0

(xも式は同じだが加速度は0 = 等速と考えれば良い)



<https://www.brh.co.jp/>

# 放物運動

放物運動 = 空中に投げられた物体の運動

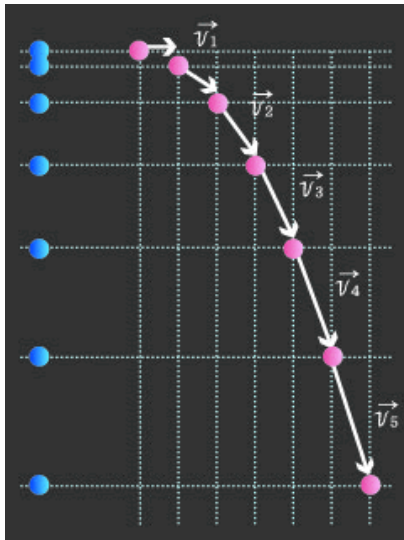
図中（左）：単なる落体の運動

図中（右）：水平方向に球を投げ出した時の運動

について、一定時間ごとの位置を示したもの

水平方向には等速運動、  
鉛直方向には等加速度運動になっている

モノを投げる状態をシミュレートするためには  
水平方向と鉛直方向のそれぞれで計算を独立に行って、それを単純に足し合わせればよい



<https://www.brh.co.jp/>

```
set x to 50
set y to 50
set vx to 2
set vy to 0
set dt to 1
set a to 1

repeat 5 times
do
  set dx to vx x dt
  set x to x + dx
  set dv to a x dt
  set vy to vy + dv
  set dy to vy x dt
  set y to y + dy
  print create text with x , y
```



的当てゲームを作る



# 前回のプロジェクトを別名で保存する

The screenshot shows the MIT App Inventor web interface. At the top, a navigation bar includes 'Projects', 'Connect', 'Build', 'Settings', 'Help', 'My Projects', 'View Trash', 'Guide', 'Report an Issue', 'English', and 'akiyolab5@gmail.com'. A red arrow points to the 'Projects' dropdown menu, which is open and displays the following options:

- My projects
- Start new project
- Import project (.aia) from my computer ...
- Import project (.aia) from a repository ...
- Move To Trash
- Save project
- Save project as ...** (highlighted with a red arrow)
- Checkpoint
- Project Properties...
- Export selected project (.aia) to my computer
- Export all projects
- Import keystore
- Export keystore
- Delete keystore

The main workspace shows a mobile app preview with a red circle at the bottom. The 'All Components' panel on the right lists: Screen1, Canvas1, Ball1, Ball2, HorizontalArrangement1, Button1, Label1, Label2, Label3, and Clock1. The 'Properties' panel on the far right shows settings for 'Screen1 (Form)', including 'Appearance' and 'AboutScreen'.



# 前回のプロジェクトを別名で保存する

The screenshot shows the MIT App Inventor web interface. At the top, a blue banner contains the title "前回のプロジェクトを別名で保存する". Below it, the browser address bar shows "ai2.appinventor.mit.edu/#4783042956492800". The interface includes a top navigation bar with "MIT APP INVENTOR" logo, "Projects", "Connect", "Build", "Guide", "Report an Issue", "English", and "akiyolab5@gmail.com". A notification bubble says "Saved project at Nov 28, 2023, 11:18:15 PM". The main workspace is divided into several panels: "Palette" (left), "Viewer" (center), "All Components" (right), and "Properties" (far right). The "Viewer" panel shows a mobile app preview with a red circle on the screen. A "Save As - Matoate" dialog box is overlaid on the viewer, with "New name:" set to "Matoate2". A blue arrow points to the text "今回は「Matoate2」" (This time it's "Matoate2"). A red arrow points to the "OK" button in the dialog. The "Properties" panel on the right shows settings for "Screen1 (Form)", including "Appearance" and "AboutScreen".

# アプリのタイトルを変える

The screenshot shows the MIT App Inventor web interface. At the top, there is a navigation bar with the MIT App Inventor logo and various menu items like Projects, Connect, Build, Settings, Help, My Projects, View Trash, Guide, Report an Issue, English, and a user profile. Below the navigation bar, there is a left sidebar with a list of components (Spinner, Switch, TextBox, TimePicker, WebViewer) and a list of categories (Layout, Media, Drawing and Animation, Maps, Charts, Data Science, Sensors, Social, Storage, Connectivity, LEGO® MINDSTORMS®, Experimental). The main workspace is divided into three sections: a mobile device preview on the left, a central canvas with a 'Media' component and an 'Upload File ...' button, and a right-hand 'Properties' panel. The 'Properties' panel lists various settings for the selected component, including CloseScreenAnimation, HighContrast, OpenScreenAnimation, PrimaryColor, PrimaryColorDark, ScreenOrientation, Scrollable, ShowListsAsJson, ShowStatusBar, Title, TitleVisible, and Application. The 'Title' property is highlighted with a red box and contains the text '的当てゲーム'. A blue arrow points from the text '前回とは異なる方が混乱しない' to the 'Title' property.

ai2.appinventor.mit.edu/#4783042956492800

MIT APP INVENTOR

Projects Connect Build Settings Help My Projects View Trash Guide Report an Issue English akiyolab5@gmail.com

Spinner Switch TextBox TimePicker WebViewer

Layout Media Drawing and Animation Maps Charts Data Science Sensors Social Storage Connectivity LEGO® MINDSTORMS® Experimental

Media

Upload File ...

CloseScreenAnimation  
Default

HighContrast

OpenScreenAnimation  
Default

PrimaryColor  
Default

PrimaryColorDark  
Default

ScreenOrientation  
Unspecified

Scrollable

ShowListsAsJson

ShowStatusBar

**Title**  
的当てゲーム

TitleVisible

Application

Privacy Policy and Terms of Use

前回とは異なる方が混乱しない ---->

# 変数の追加と初期化

initialize global a to 0

加速度

initialize global dt to 1

微小時間

initialize global dv to 0

initialize global dx to 0

initialize global dy to 0

initialize global f to 0

```
when Button1 .Click
do
  set global vx to 0
  set global vy to 0
  set Ball1 . X to 50
  set Ball1 . Y to 50
  set Ball1 . Visible to true
  set global total to [get global total] + 1
  set Label1 . Text to [join [get global score] ["/"] [get global total]]
  set global a to 0
```

追加

# 球を飛ばす処理

The image shows a Scratch script for a ball game. The script is enclosed in a 'when Clock1.Timer' block. The main logic is in a 'do' block. It starts by calculating the horizontal displacement (dx) as the current horizontal velocity (vx) multiplied by the time step (dt). Then, it updates the ball's X position by adding dx. Next, it calculates the vertical displacement (dv) as the current acceleration (a) multiplied by dt, and updates the vertical velocity (vy) by adding dv. Then, it calculates the vertical displacement (dy) as the current vertical velocity (vy) multiplied by dt, and updates the ball's Y position by adding dy. A collision check is performed using an 'if' block. The condition is the square root of the sum of the squared differences in X and Y coordinates between Ball1 and Ball2, which is less than 30. If this condition is true, the script then sets the vertical velocity (vy) and horizontal velocity (vx) to their negative values, effectively reversing the ball's direction. Finally, it increments the global score by 1. After the 'do' block, the script updates the text of Label2 to the current X position of Ball1 and the text of Label3 to the current Y position of Ball1.

```
when Clock1.Timer
do
  set global dx to (get global vx) × (get global dt)
  set Ball1.X to (Ball1.X) + (get global dx)
  set global dv to (get global a) × (get global dt)
  set global vy to (get global vy) + (get global dv)
  set global dy to (get global vy) × (get global dt)
  set Ball1.Y to (Ball1.Y) + (get global dy)
  if (square root of ((Ball1.X - Ball2.X) ^ 2 + (Ball1.Y - Ball2.Y) ^ 2) < 30)
  then
    set global vy to (get global vy) × (-1)
    set global vx to (get global vx) × (-1)
    set global score to (get global score) + 1
  set Label2.Text to Ball1.X
  set Label3.Text to Ball1.Y
```

# タップしたときと離れたときの処理

```
when Canvas1 .TouchUp
  x y
  do
    call Canvas1 .Clear
    set global f to 0.2
    set global vx to (get global f) × (Ball1 . X - get x)
    set global a to 0.5
```

} 変更

これ以外のブロックは変更なし

```
when Canvas1 .Dragged
  startX startY prevX prevY currentX currentY draggedAnySprite
  do
    call Canvas1 .Clear
    call Canvas1 .DrawLine
      x1 50
      y1 50
      x2 get currentX
      y2 get currentY
```

横方向の速度しかないので  
線は好みで変えても良い  
(横方向のみにするなど)

# 工夫してみよう

- Resetするたびに的のx座標が変わるようにしてみよう
- Resetするたびに的の大きさが変わるようにしてみよう
- 的が常に移動するようにしてみよう
- オリジナルの工夫も考えてみよう