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

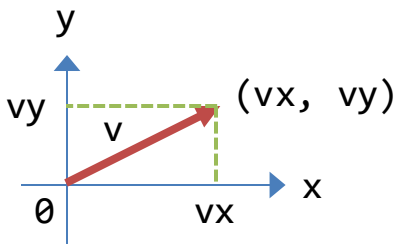
今回は放物運動のシミュレーションを行うことを目的とする.
放物運動とは空中に投げられた物体の運動のことである.
重力により，物体には鉛直下向きに行こうとする力が常にかかっている.
空気抵抗などの重力以外の力を無視し，投げられたものの本質的な動きだけをシミュレートする.

- 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

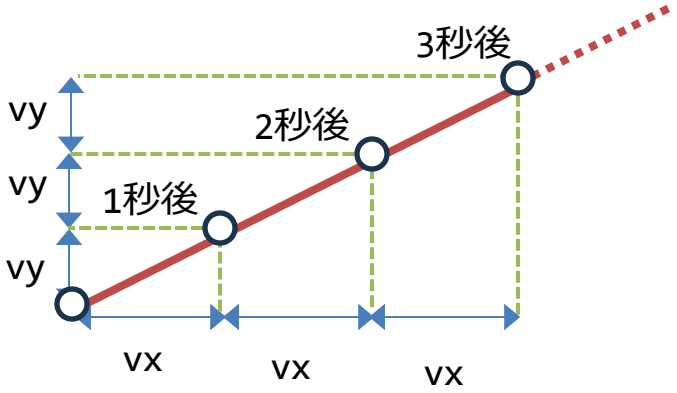
放物運動とは

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？

$$x_{(t)} = x_{(0)} + v_x \times t$$

$$y_{(t)} = y_{(0)} + v_y \times t$$

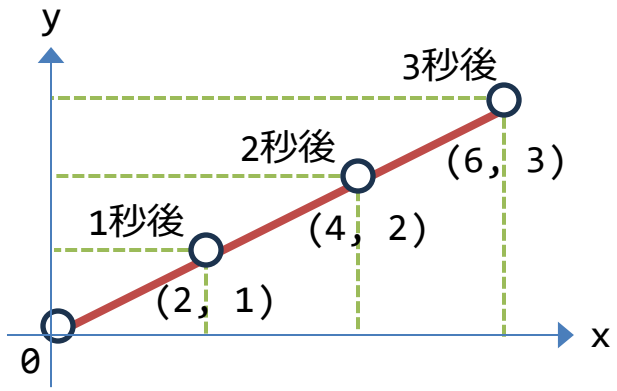
↓

速度が一定でないとダメ

↓

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

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



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

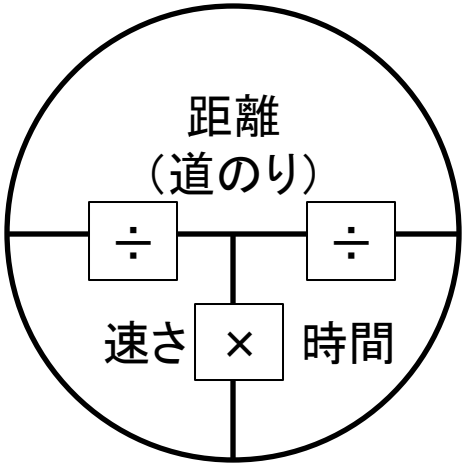
$$y_{(4)} = 0 + 1 \times 4 = 4$$

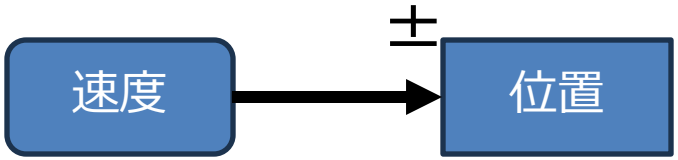
いまの時刻tから1秒後の位置を求める.
x(t)とy(t)はいまの位置,
vx(t)とvy(t)は1秒間に動く量を表している.

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x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x時間を足したものが次のy）

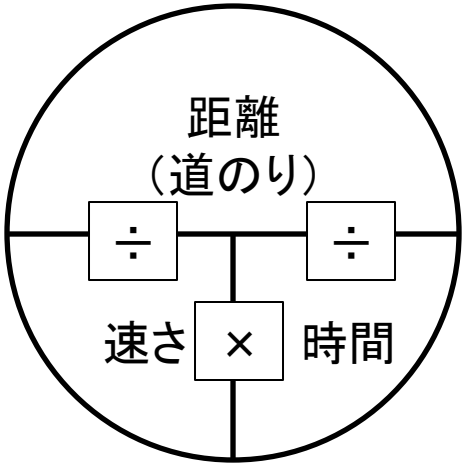
等速直線運動

本来は　速度×時間＝距離（位置の変化）

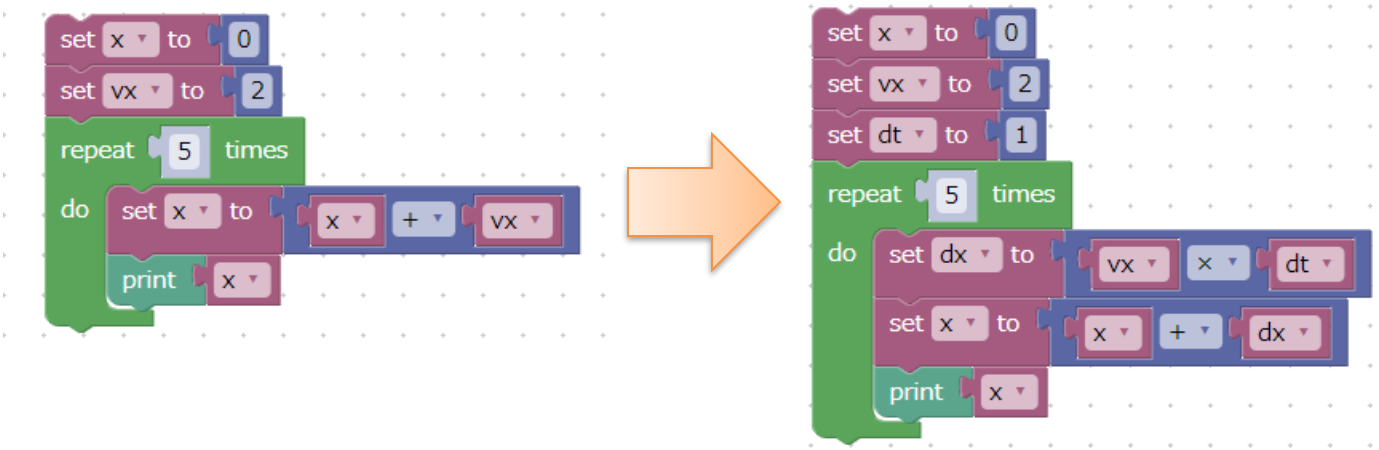
- 1step ”1秒“という仮定で時間は省略していた
  （位置＋速度という計算はできない）
- より正確な表現には“微小時間Δt”を考える

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

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

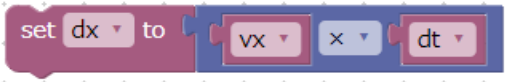


等速直線運動（改）



- 変更点

1. 微小時間dtの間の位置の変化量dxを速さvxを用いて



としていること

2. この位置の変化量dxを用いて

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

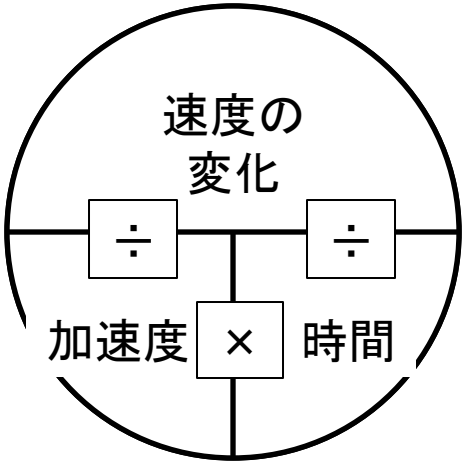
等加速度運動

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

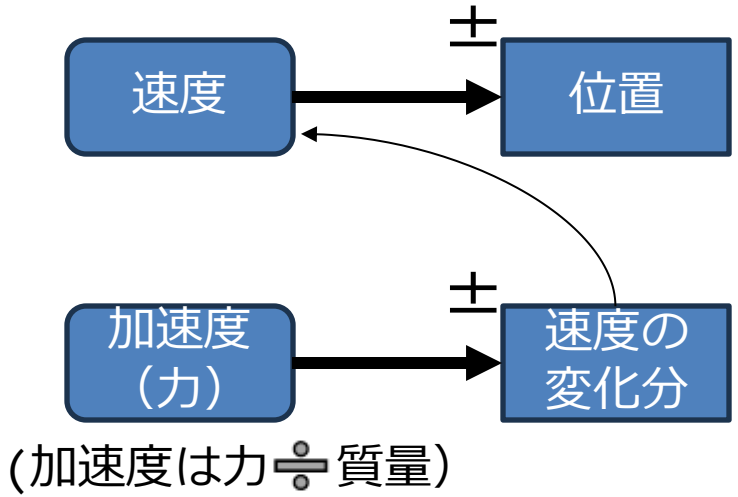
加速度×時間＝速度の変化



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

⇩

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



(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(m/s)/s=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=1m/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Δtの間に生じた微小な速さの変化Δv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

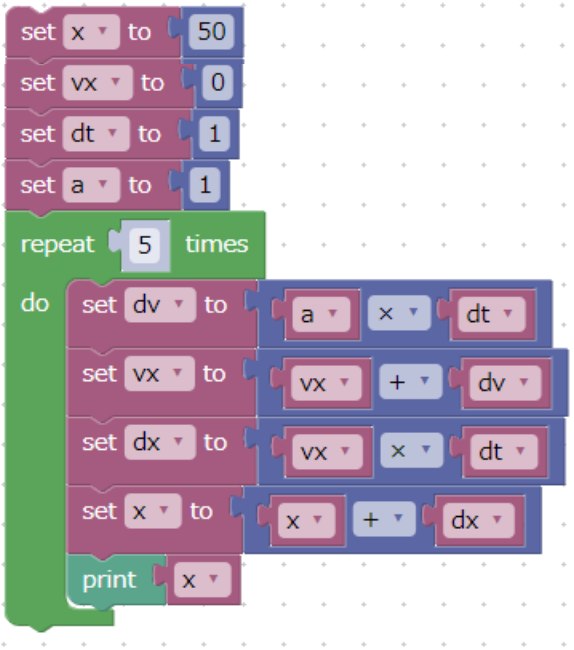
③ 更新された速さに基づいて移動距離の変化量を求める

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



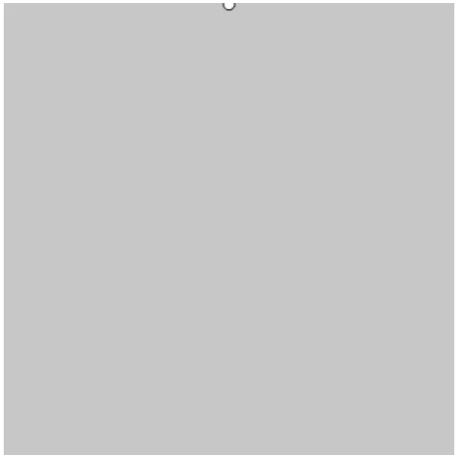
①

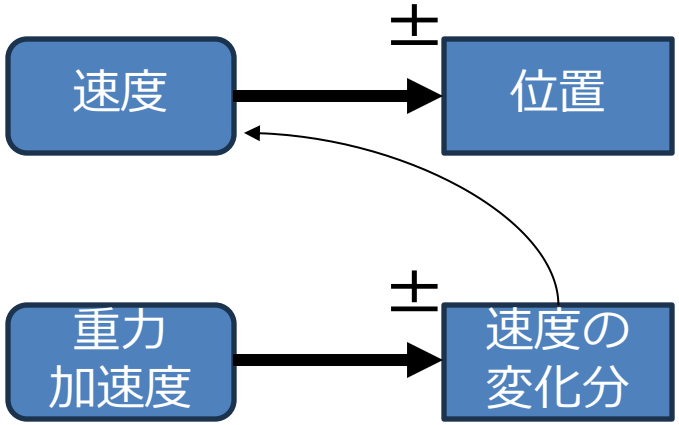
②

加速度は負の値になっても良い．一定の速さで動いていれば，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x時間を足したものが次のy）

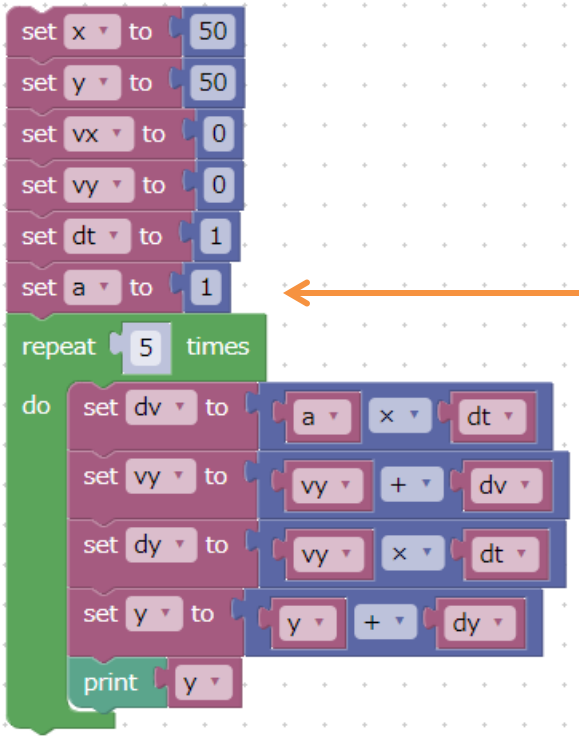
$$v_y' = v_y + \alpha \times \Delta t$$

（元のvに加速度x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9.8m/s^2

- 1秒ごとに移動速度が9.8m/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70kmに達するような量



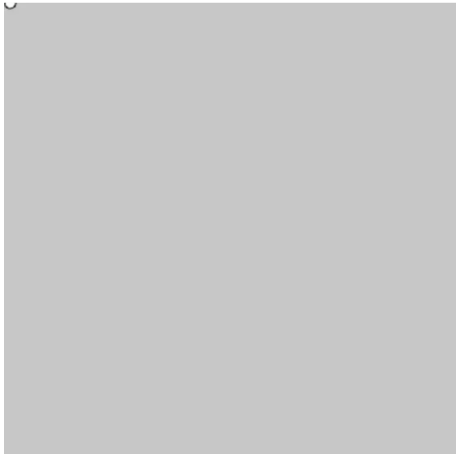
なお，このシミュレーションでは，プログラムの分かりやすさと可視化を重視し，現象のリアリティについては厳密性を求めないこととする．つまり，どれくらいの長さを1mと見立てるか？といった換算は行わない．よって，重力加速度も9.8という値を用いずに，見易さを重視し，小さな値を用いることとする．

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

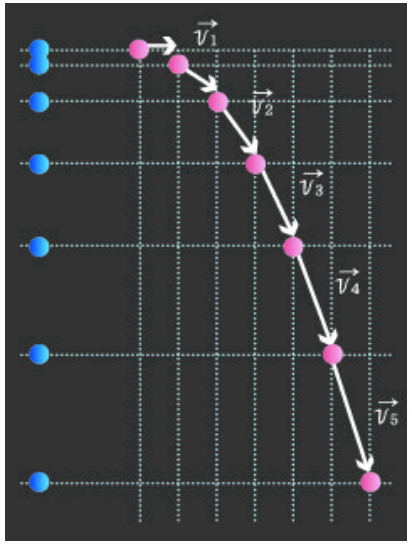


$$x' = x + v_x \times \Delta t \qquad y' = y + v_y \times \Delta t$$

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

$a_x = 0$

（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

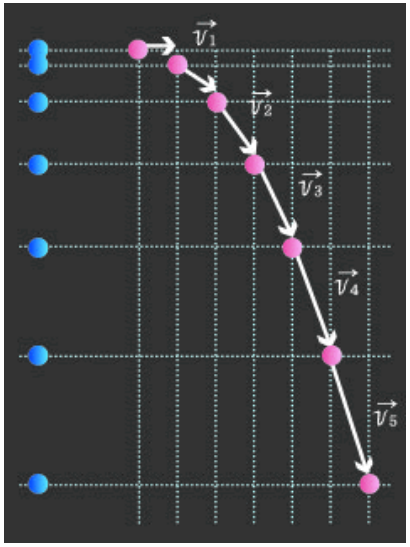
放物運動＝空中に投げられた物体の運動

　図中（左）：単なる落体の運動

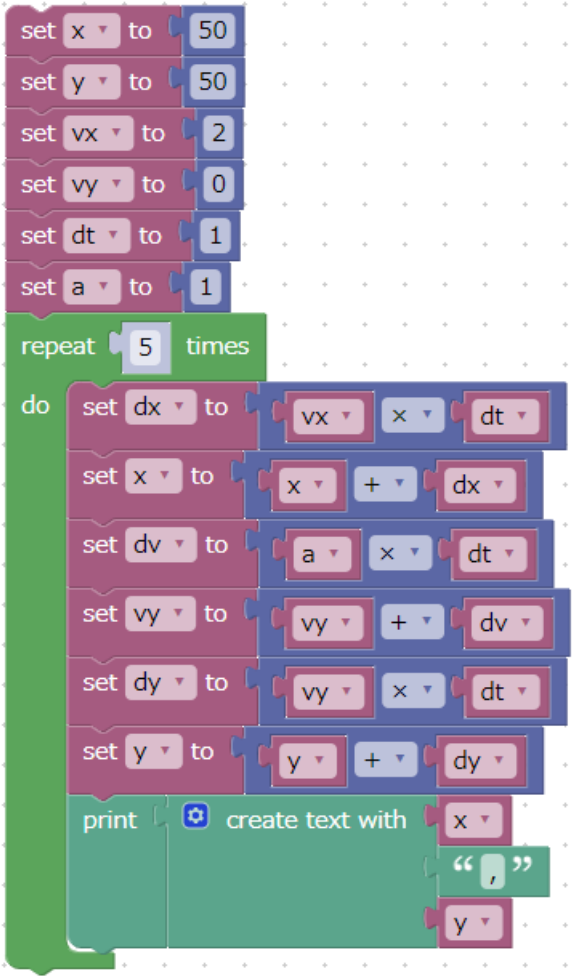
　図中（右）：水平方向に球を投げ出した時の運動

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

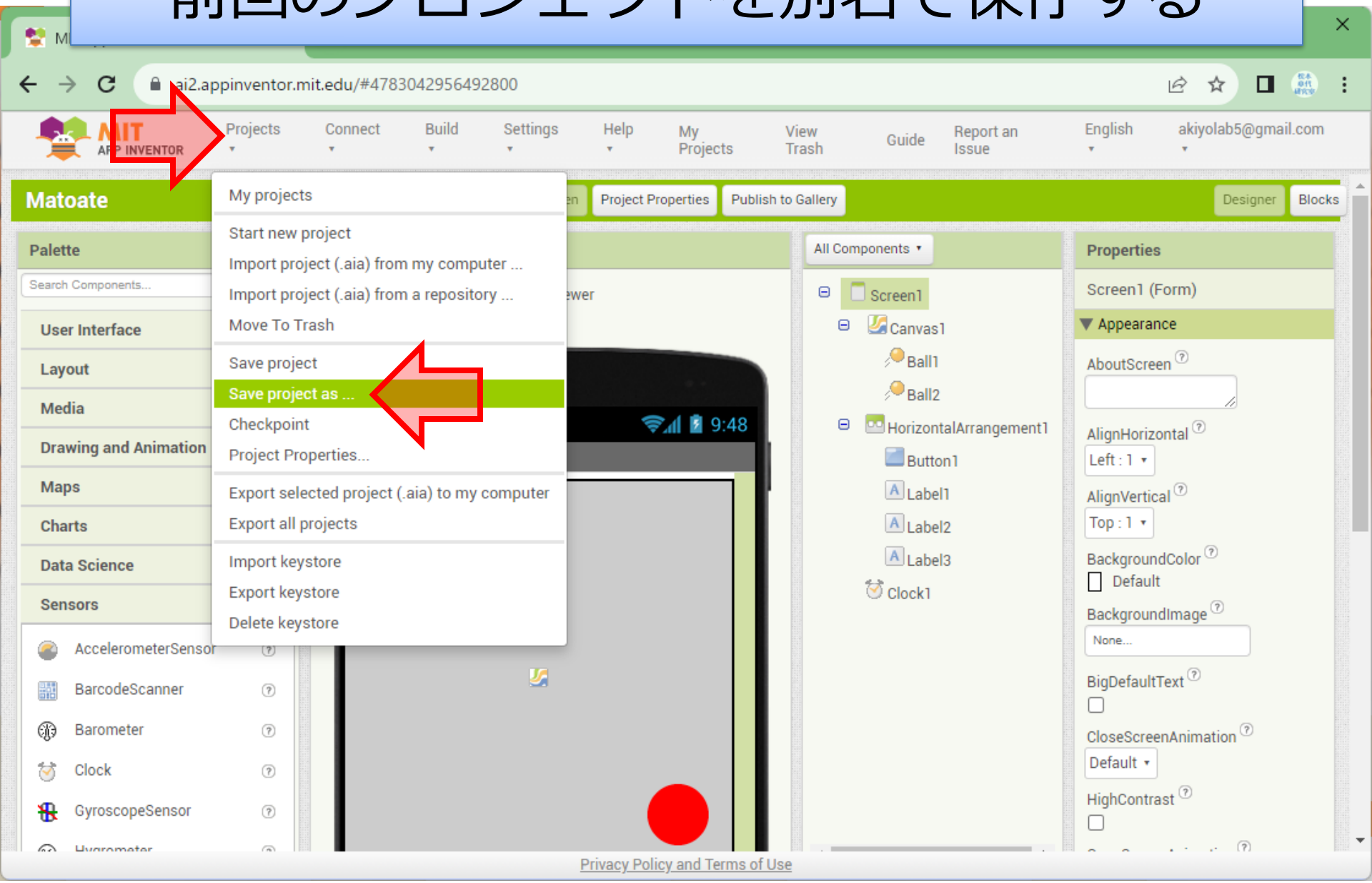


https://www.brh.co.jp/

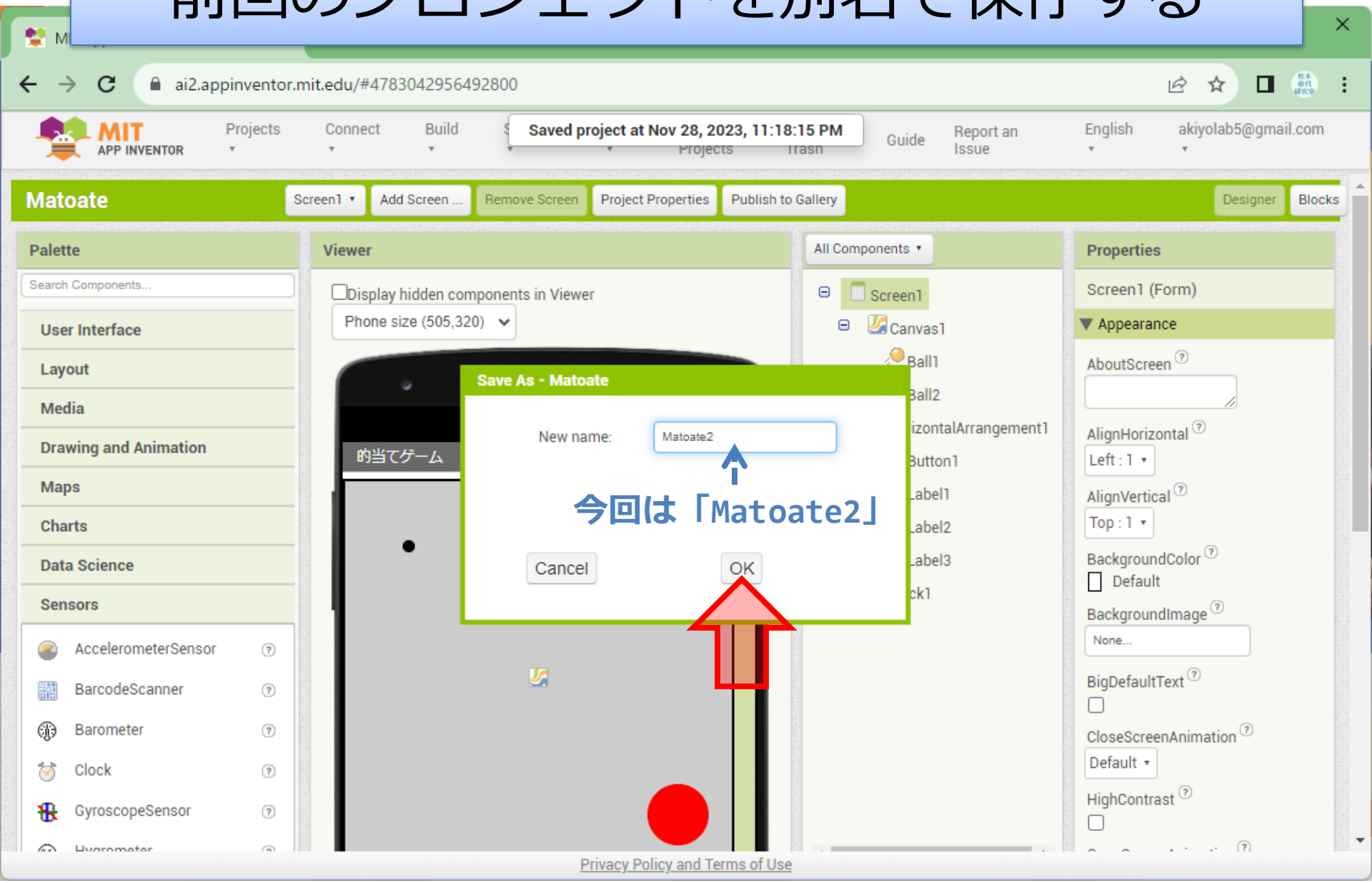


的当てゲームを作る

前回のプロジェクトを別名で保存する



前回のプロジェクトを別名で保存する



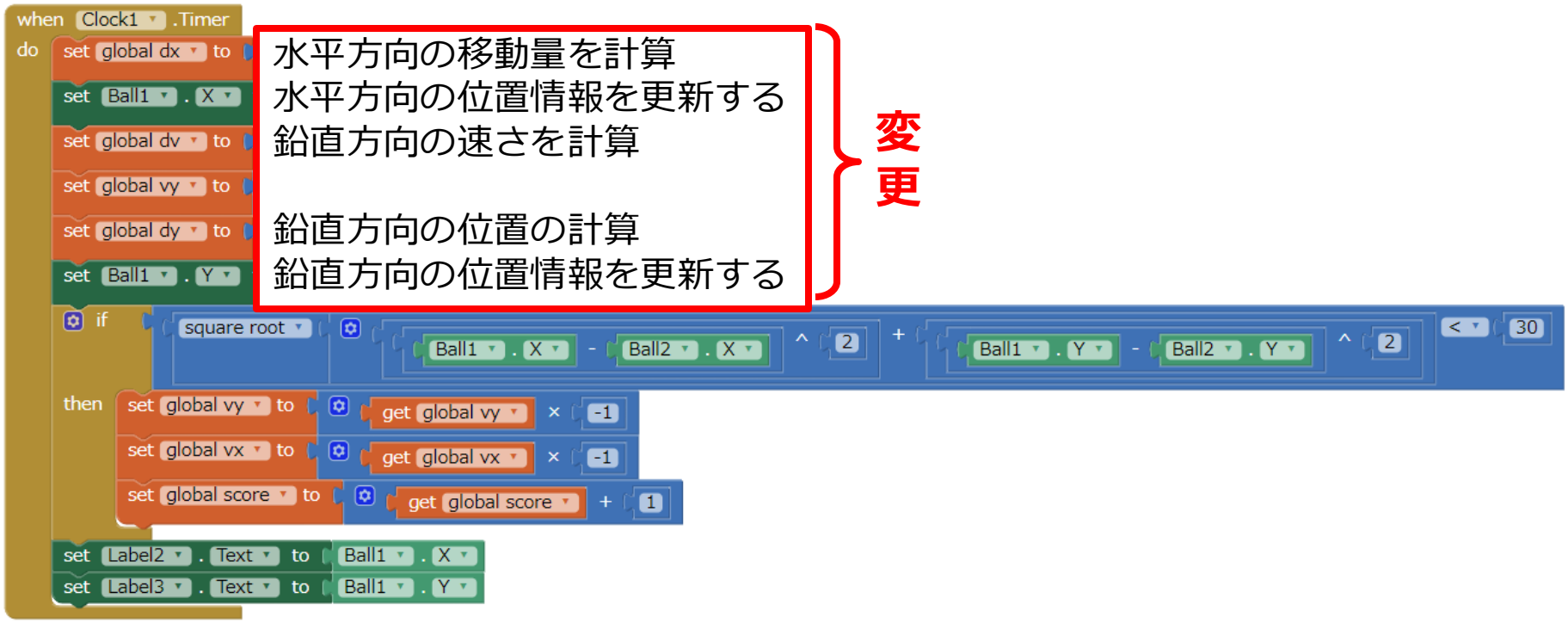
アプリのタイトルを変える

変数の追加と初期化

加速度

微小時間

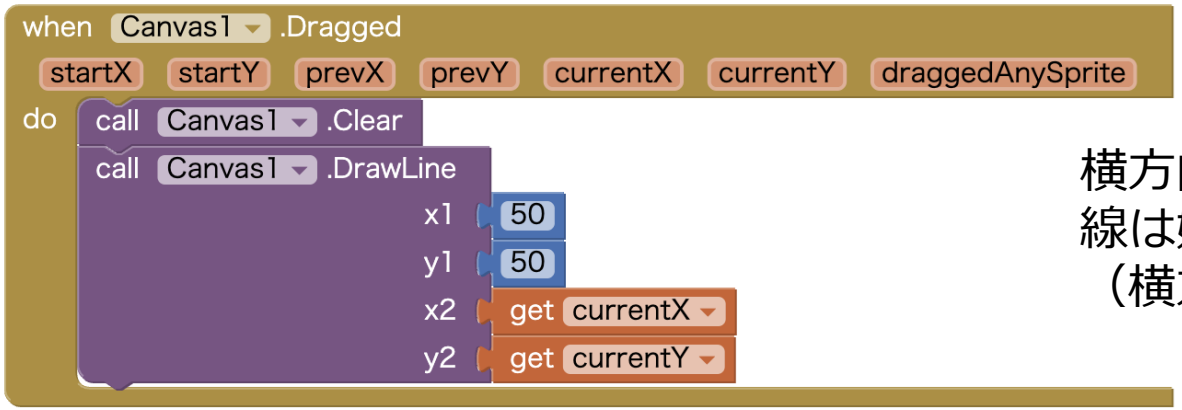
球を飛ばす処理


when Clock1 .Timer
do
set global dx to
set Ball1 . X
set global dv to
set global vy to
set global dy to
set Ball1 . Y
水平方向の移動量を計算
水平方向の位置情報を更新する
鉛直方向の速さを計算
鉛直方向の位置の計算
鉛直方向の位置情報を更新する
変更
if
square root
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したときの処理

変更

これ以外のブロックは変更なし



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