プログラミング概論

第10回　2024年11月27日
App Inventorによる
Androidアプリ開発の実践
(4) 物理シミュレーション1

今回の授業内容

- コンピュータシミュレーションとは
- コンピュータの画面上で「動き」を表現するには
- 等速直線運動
- 的当てゲームを作る

コンピュータシミュレーションとは

コンピュータシミュレーション

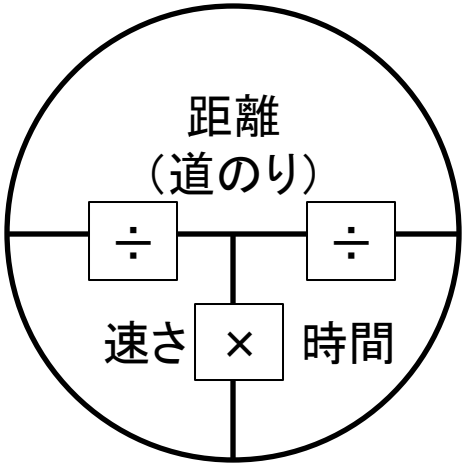
自然や社会に見られる現象をコンピュータ上で模擬的に再現すること　→　分析や予測，ゲームに有用

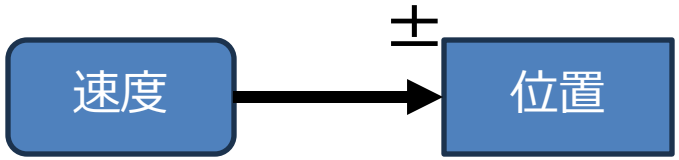
コンピュータ上で模擬するためには必要なこと
・現象の本質をとらえ，そのしくみを単純化
・ルールや数式により記述

第10回～第12回の授業では
物理現象のひとつである力学のうちの
「物体の運動」に焦点を当て，コンピュータ上で
模擬する方法の基礎を学ぶ

“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





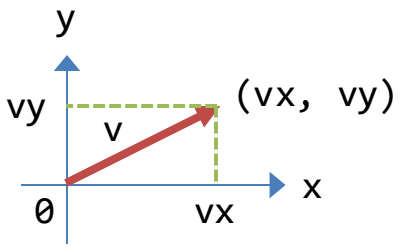
$$x' = x + v_x \times \Delta t$$

（元のxに速度x時間を足したものが次のx）

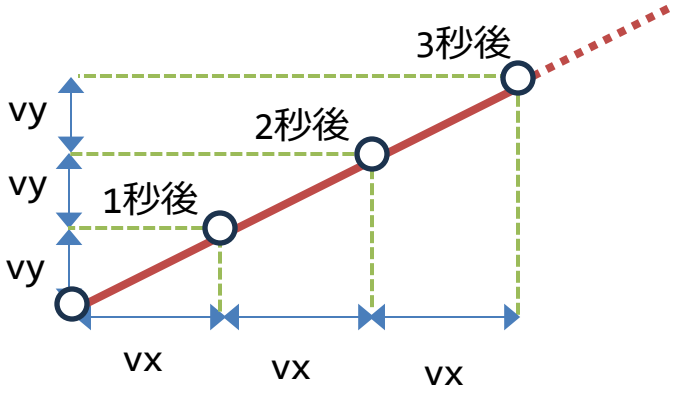
$$y' = y + v_y \times \Delta t$$

（元のyに速度x時間を足したものが次のy）

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？

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

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

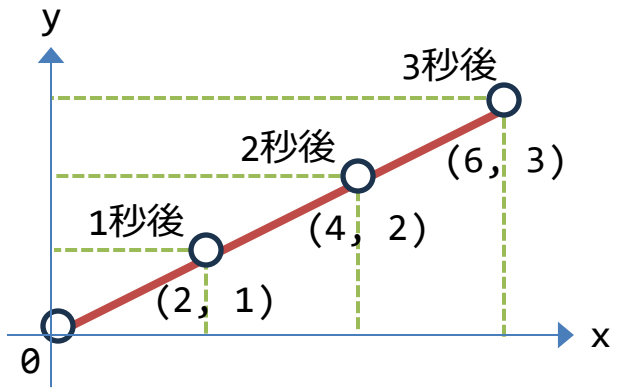
↓

速度が一定でないとダメ

↓

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

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



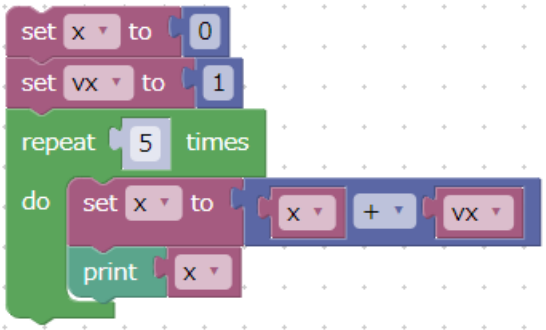
$$x_{(4)} = 0 + 2 \times 4 = 8$$

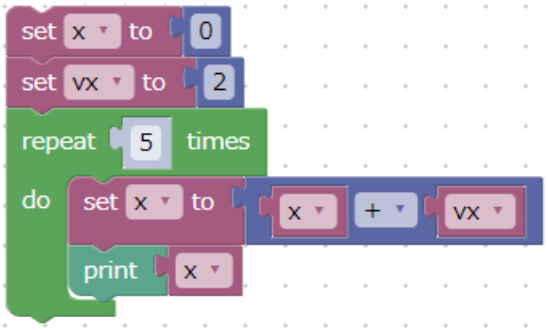
$$y_{(4)} = 0 + 1 \times 4 = 4$$

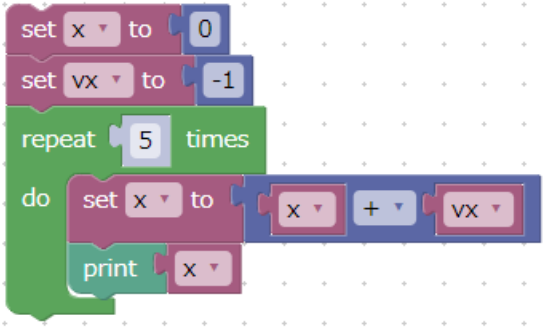
いまの時刻tから1秒後の位置を求める.
x(t)とy(t)はいまの位置,
vx(t)とvy(t)は1秒間に動く量を表している.

等速直線運動

表示される値をそれぞれ考えてみよう







,　　,　　,　　,

,　　,　　,　　,

,　　,　　,　　,

vx（移動量）は繰り返しの間，常に一定

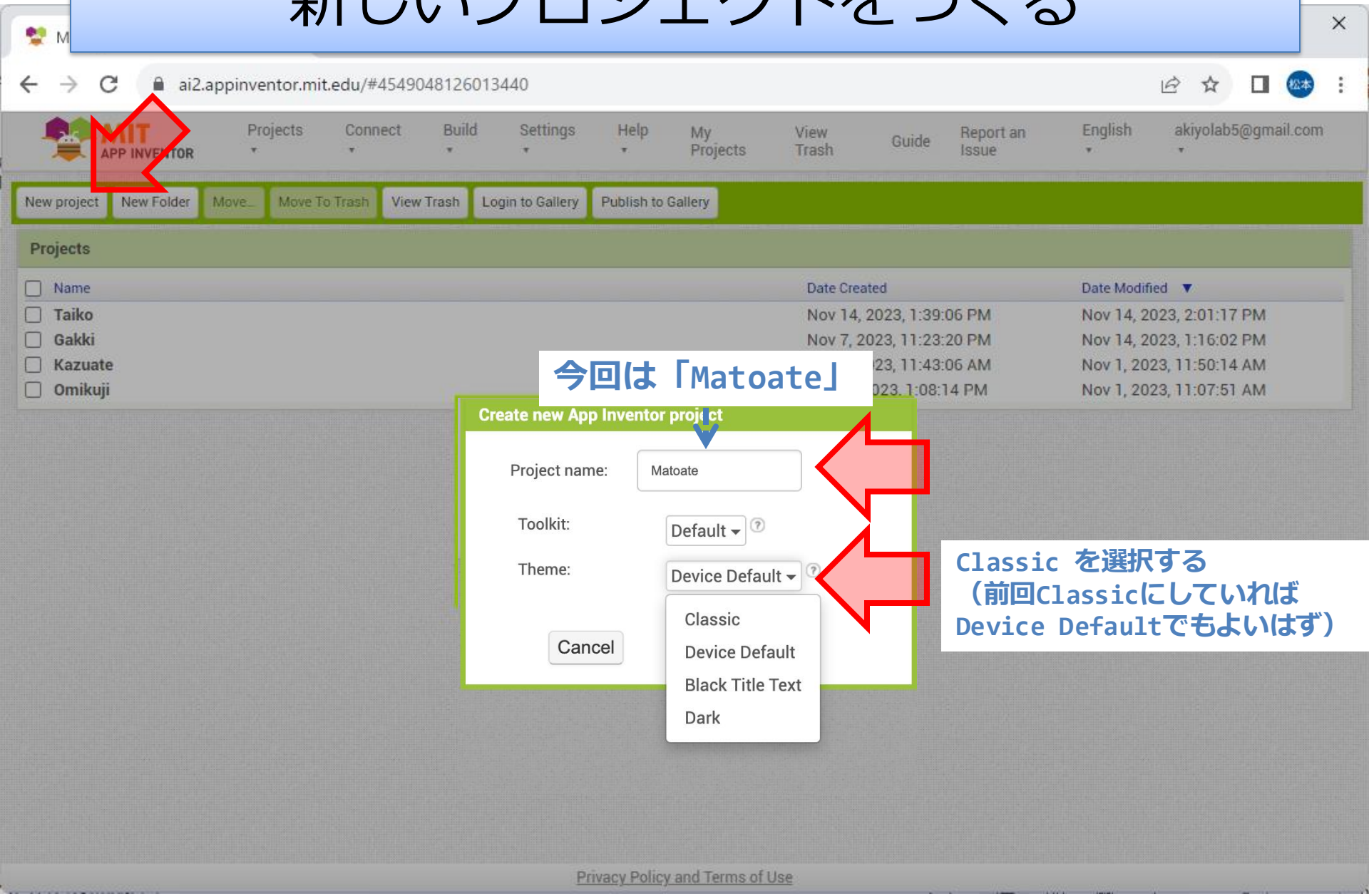
↓

等速直線運動

考えたら Blockly Code で確かめよう

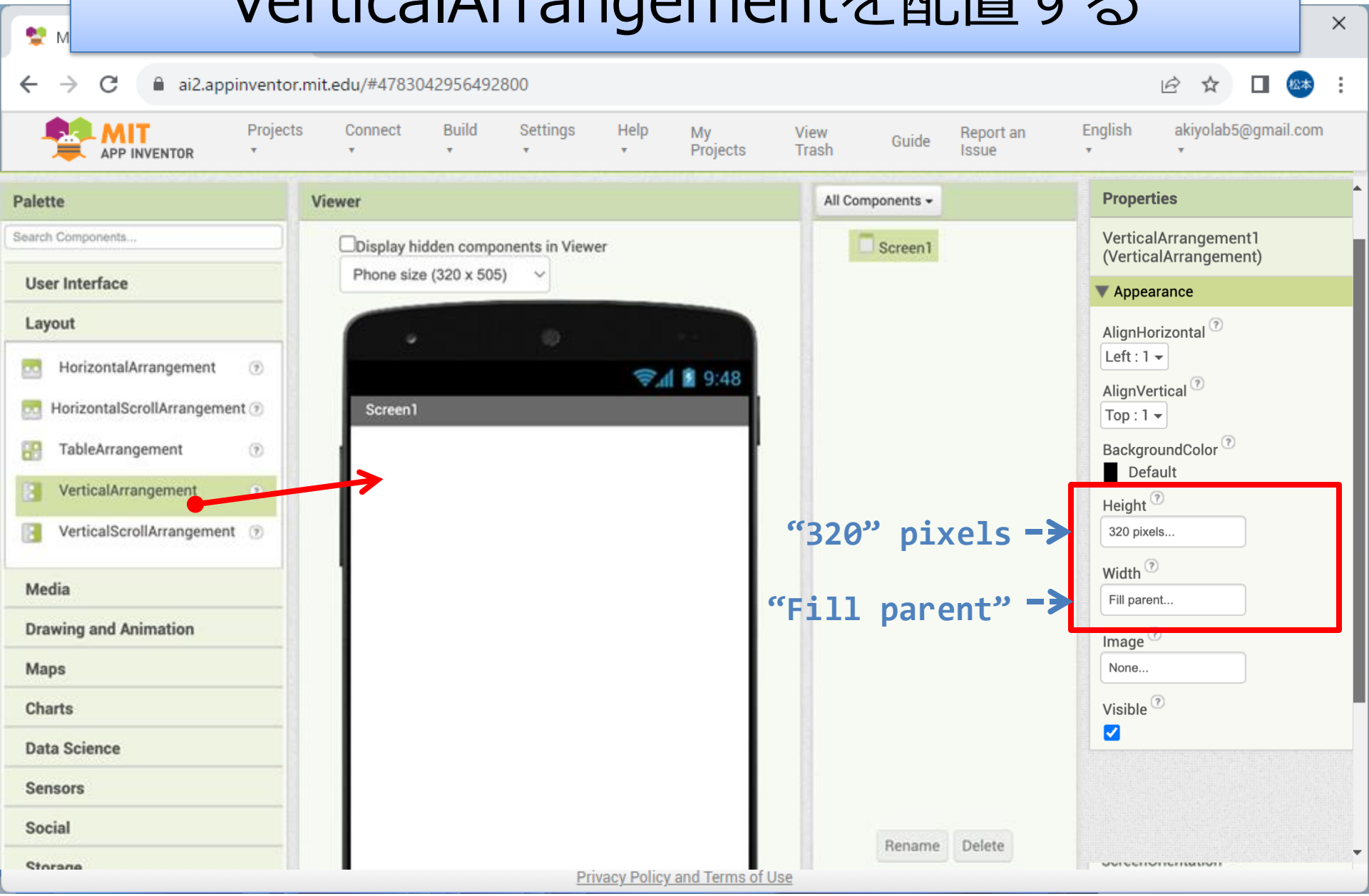
的当てゲームを作る

新しいプロジェクトをつくる

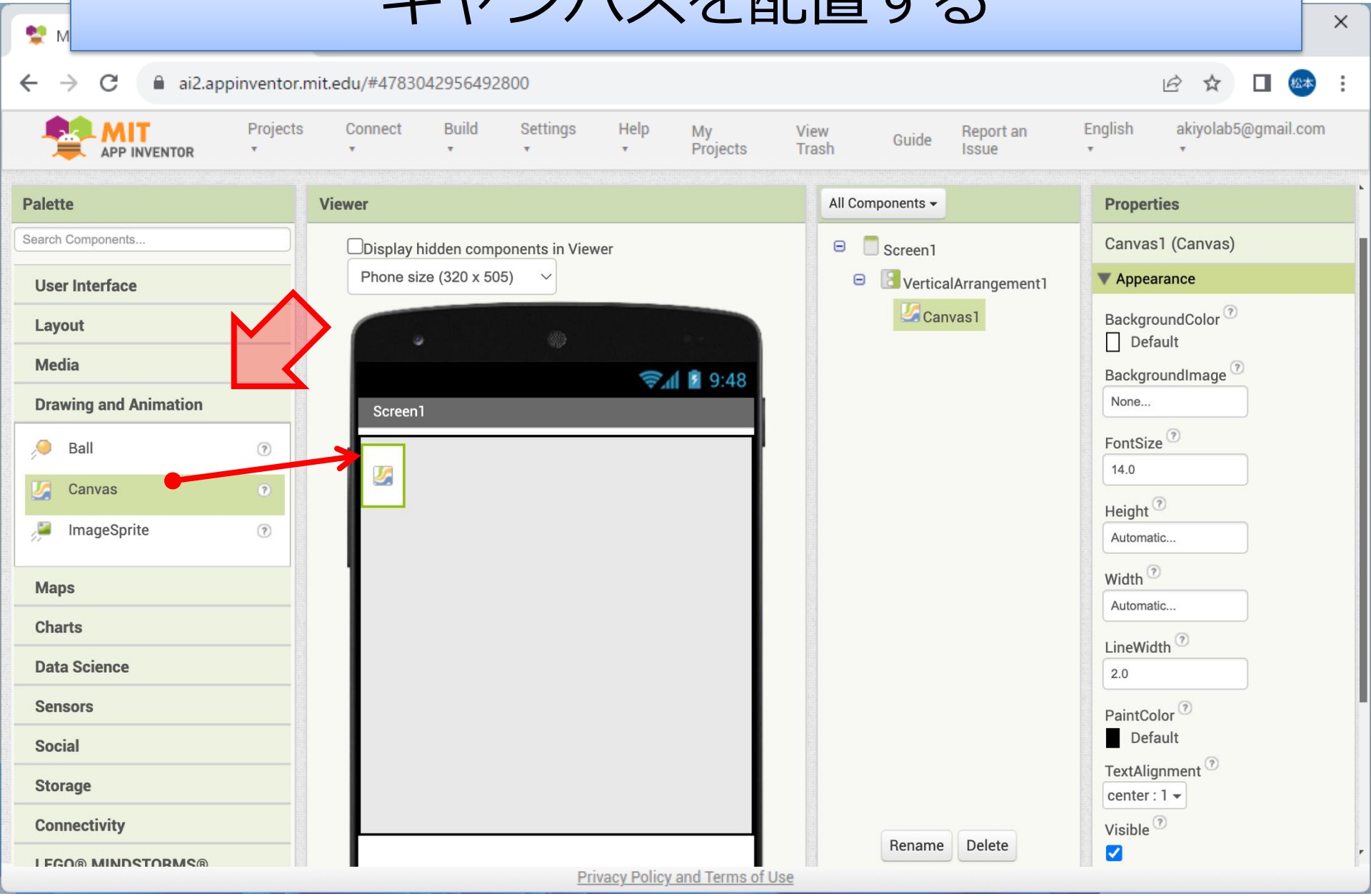


アプリのタイトルを変える

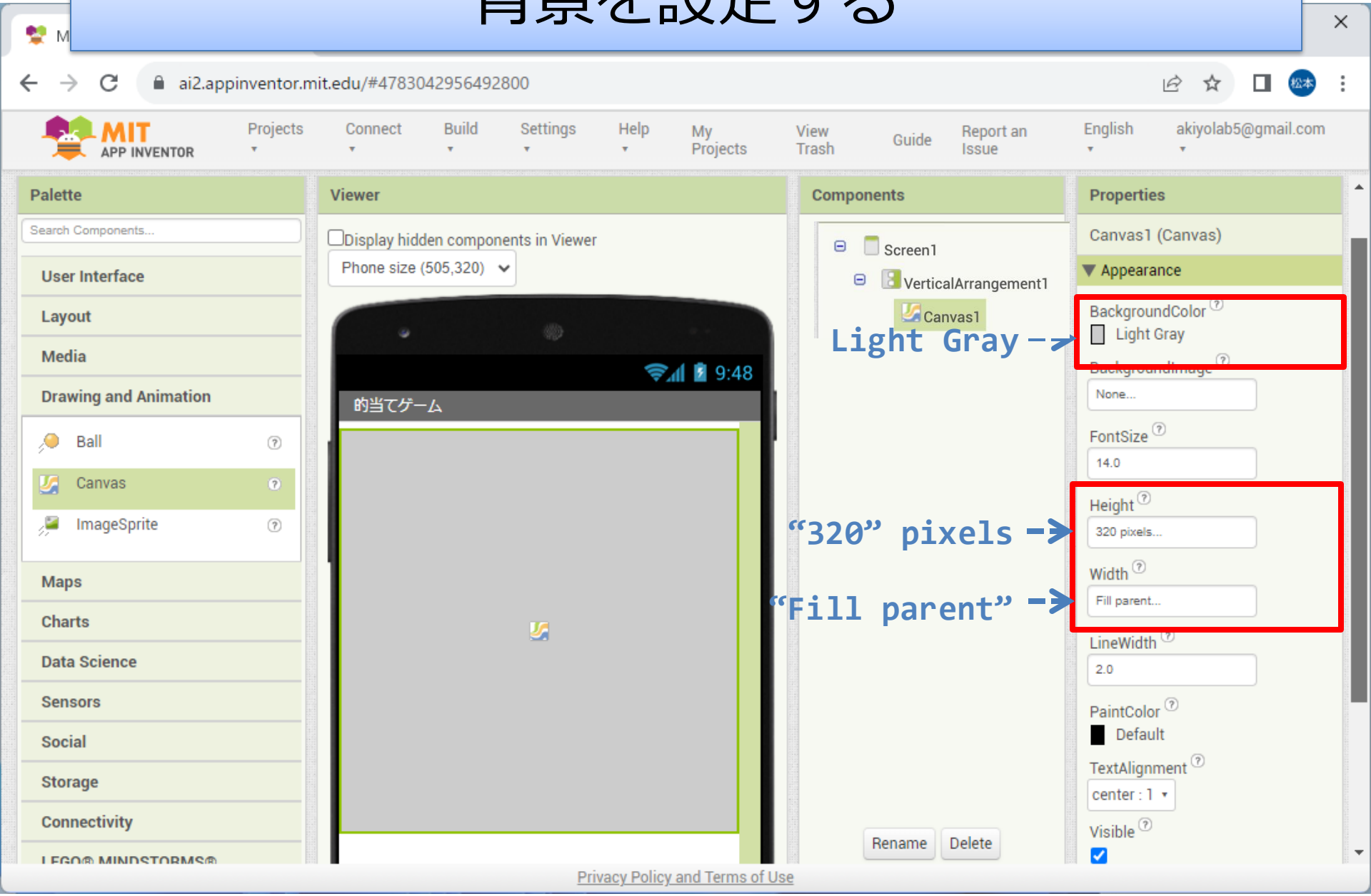
VerticalArrangementを配置する



キャンバスを配置する


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
Palette
Search Components...
User Interface
Layout
Media
Drawing and Animation
Ball
Canvas
ImageSprite
Maps
Charts
Data Science
Sensors
Social
Storage
Connectivity
Viewer
Display hidden components in Viewer
Phone size (320 x 505)
9:48
Screen1
All Components
Screen1
VerticalArrangement1
Canvas1
Rename
Delete
Properties
Canvas1 (Canvas)
Appearance
BackgroundColor
Default
BackgroundImage
None...
FontSize
14.0
Height
Automatic...
Width
Automatic...
LineWidth
2.0
PaintColor
Default
TextAlignment
center : 1
Visible
Privacy Policy and Terms of Use

背景を設定する


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
Palette
Search Components...
User Interface
Layout
Media
Drawing and Animation
Ball
Canvas
ImageSprite
Maps
Charts
Data Science
Sensors
Social
Storage
Connectivity
Viewer
Display hidden components in Viewer
Phone size (505,320)
9:48
的当てゲーム
Components
Screen1
VerticalArrangement1
Canvas1
Rename
Delete
Properties
Canvas1 (Canvas)
Appearance
BackgroundColor
Light Gray
None...
FontSize
14.0
Height
320 pixels...
Width
Fill parent...
LineWidth
2.0
PaintColor
Default
TextAlignment
center : 1
Visible
Light Gray
“320” pixels
“Fill parent”
Privacy Policy and Terms of Use

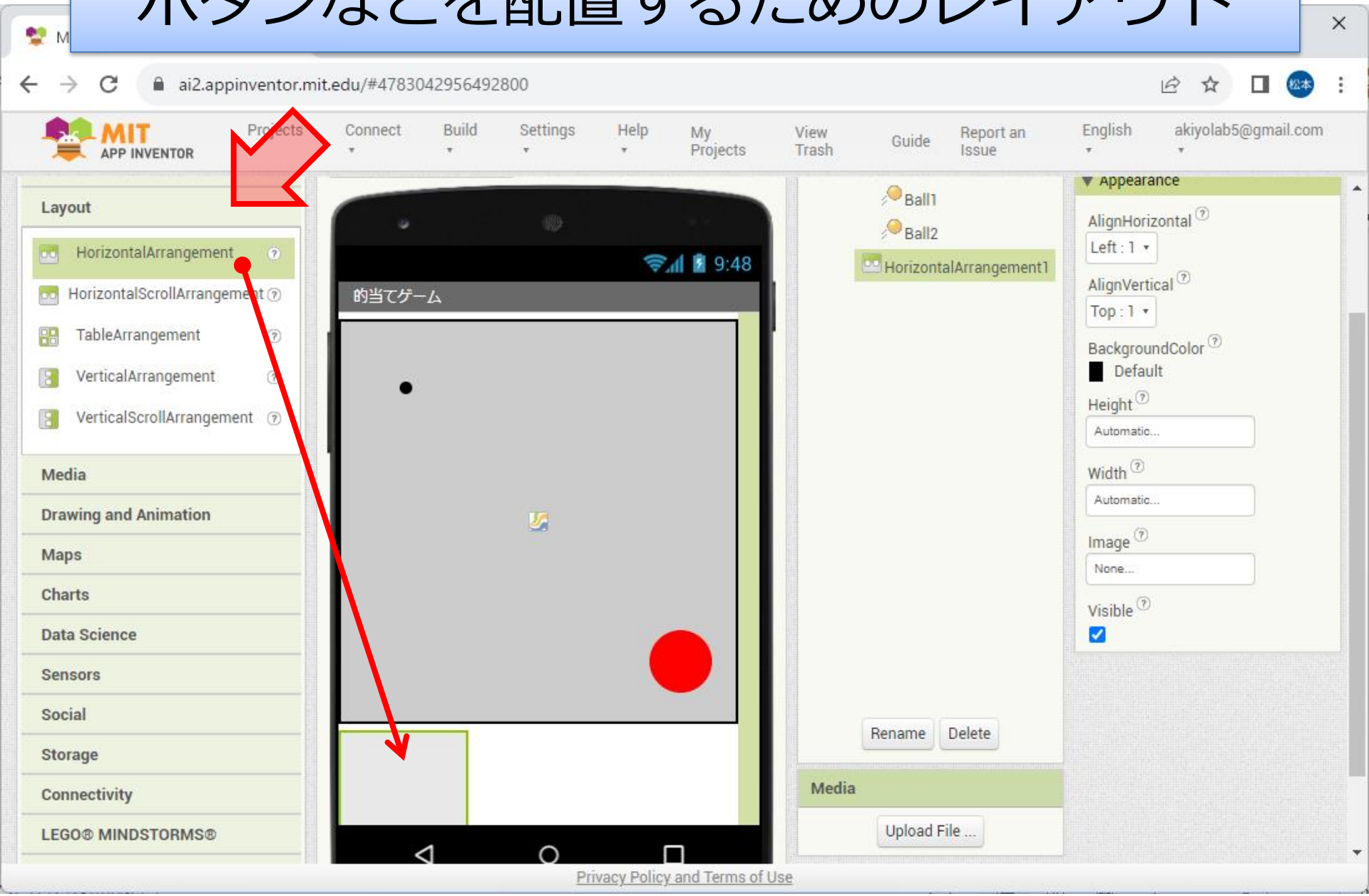
球をつくる


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
Layout
Media
Drawing and Animation
Ball
Canvas
ImageSprite
Maps
Charts
Data Science
Sensors
Social
Storage
Connectivity
LEGO® MINDSTORMS®
Experimental
Extension
9:48
的当てゲーム
Ball1
Rename
Delete
Media
Upload File ...
PaintColor
Default
Radius
5
Visible
X
50
50
Y
50
50
Z
1.0
Behavior
Enabled
Heading
0
Interval
50
50
OriginAtCenter
チェック
Privacy Policy and Terms of Use

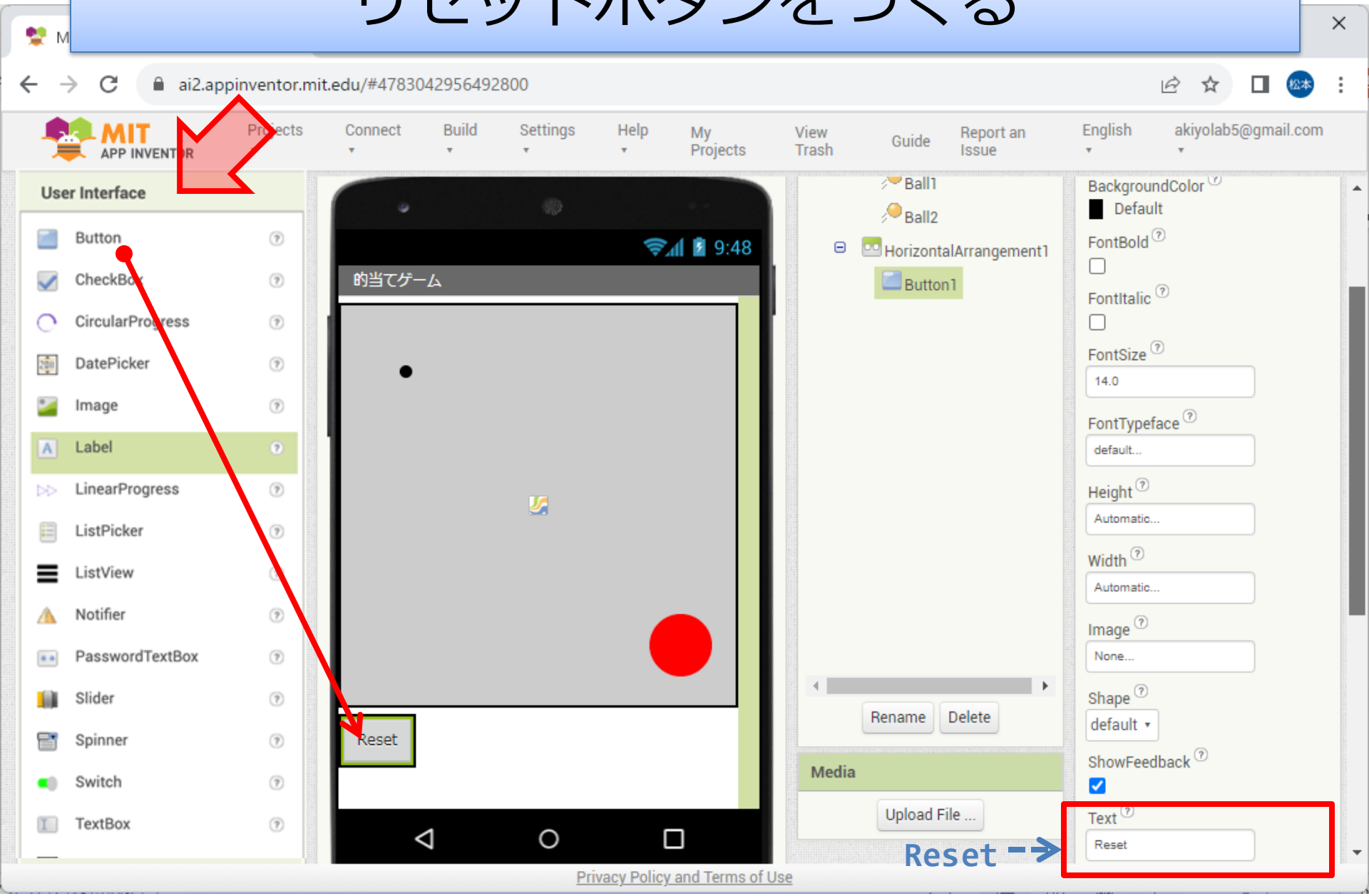
的をつくる


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
Layout
Media
Drawing and Animation
Ball
Canvas
ImageSprite
Maps
Charts
Data Science
Sensors
Social
Storage
Connectivity
LEGO® MINDSTORMS®
Experimental
Extension
9:48
的当てゲーム
Ball1
Ball2
Red
25
270
270
Rename
Delete
Media
Upload File ...
PaintColor
Red
Radius
25
Visible
X
270
Y
270
Z
1.0
Behavior
Enabled
Heading
0
Interval
100
OriginAtCenter
チェック
Privacy Policy and Terms of Use

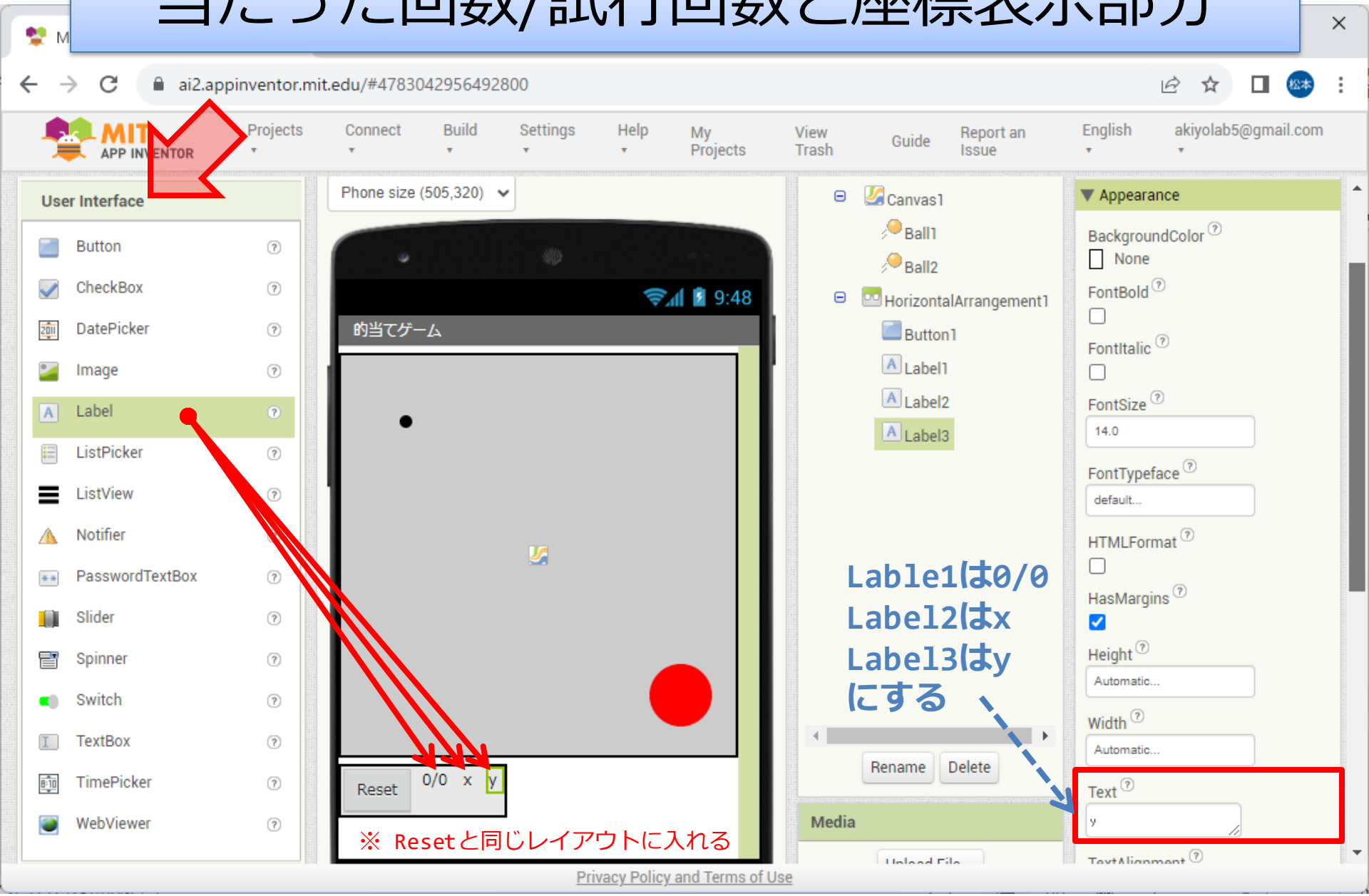
ボタンなどを配置するためのレイアウト



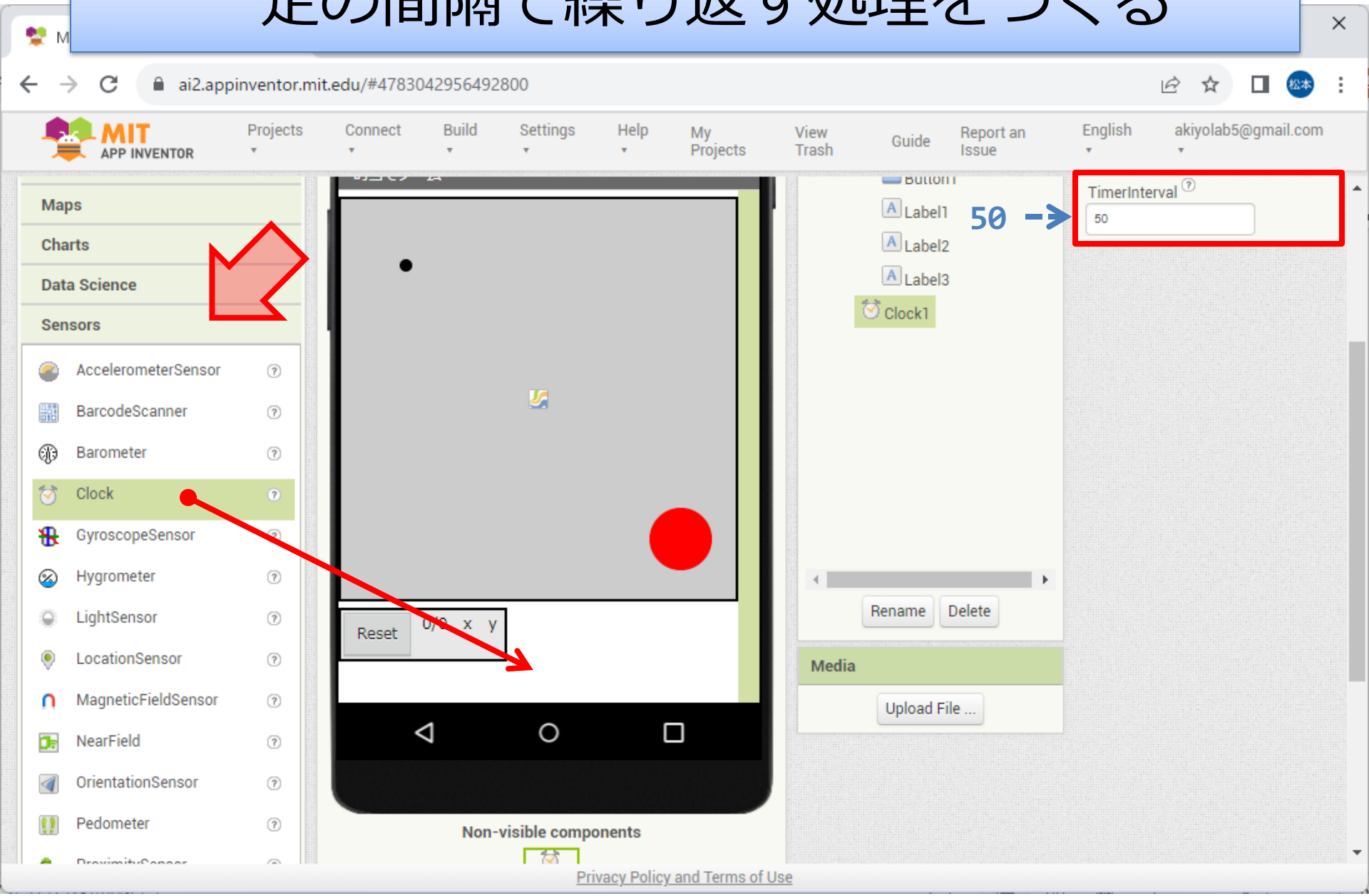
リセットボタンをつくる



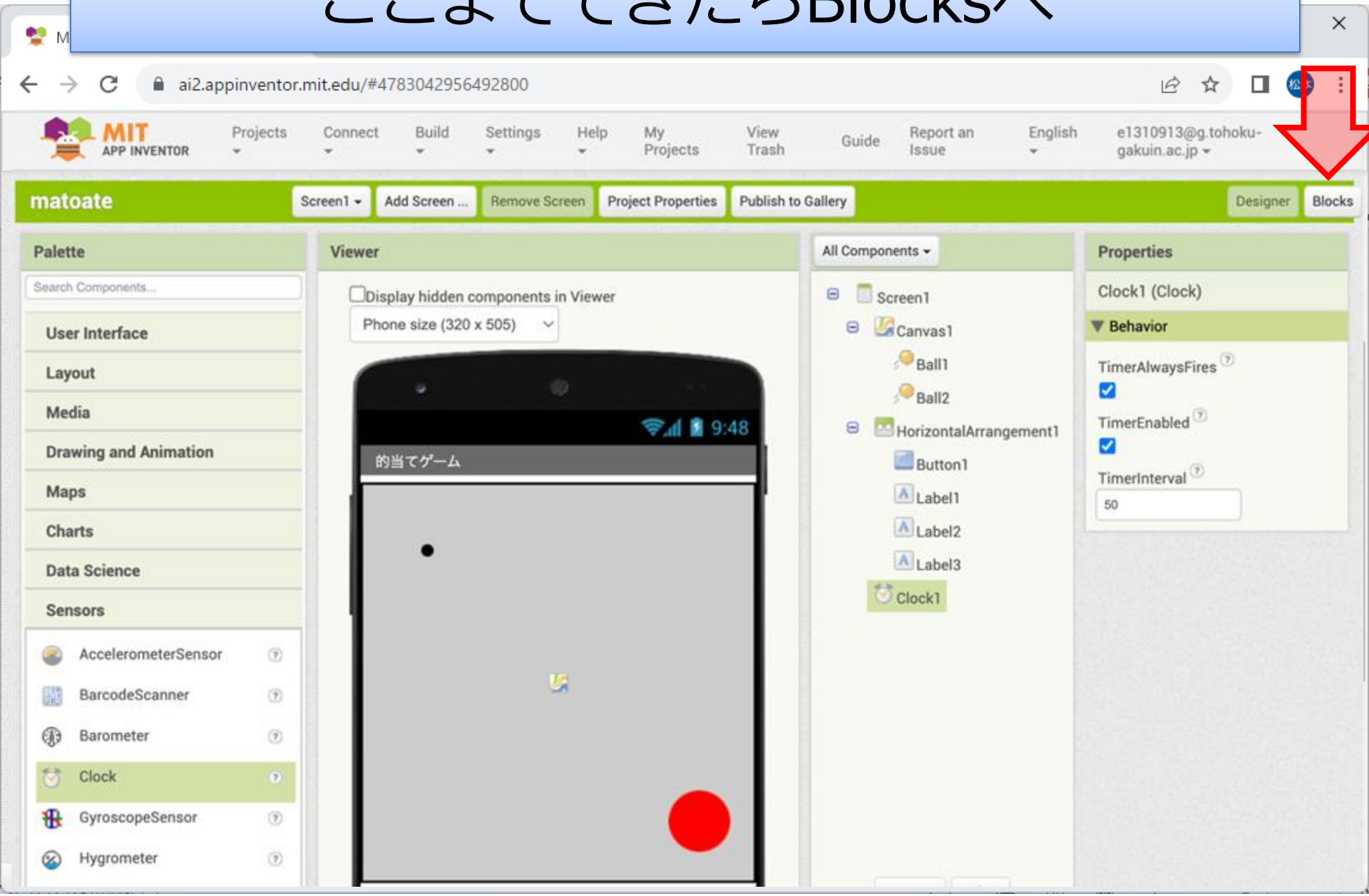
当たった回数/試行回数と座標表示部分



一定の間隔で繰り返す処理をつくる



ここまでできたらBlocksへ



アプリ起動時の初期化とリセット処理

(Built-in) Variables

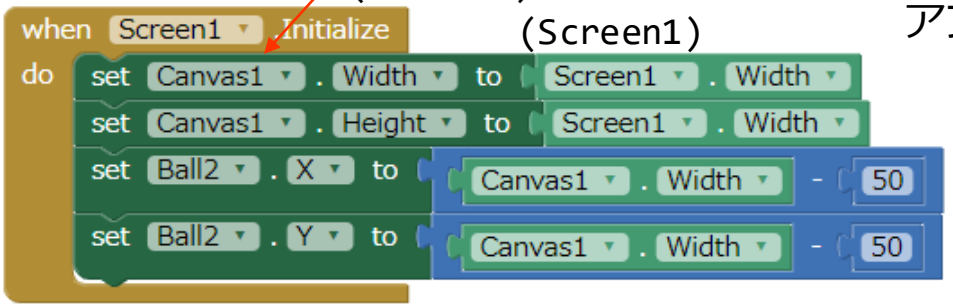
Clock1.Timerが1回実行されるごとのX軸方向の移動量

Clock1.Timerが1回実行されるごとのY軸方向の移動量

(Screen1)

(Canvas1)

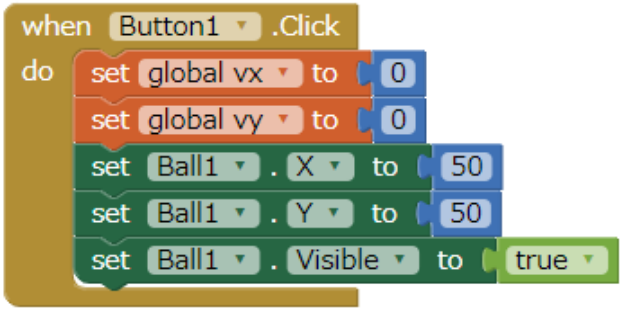
(Screen1)



アプリが起動したら（初期化）
キャンバスの幅を画面の幅に
キャンバスの高さも（正方形）

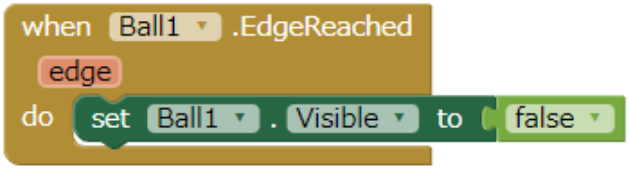
的のx座標は右下から50ずらす
的のy座標も同様に

(Button1)



リセットボタンが押されたら
球は動かないようにする
球のx座標・y座標は初期位置に戻す
球は見えるようにする

(Ball1)



球が画面端に当たったときには
見えなくする

※ Canvas1とかついているBlockはそれぞれのcomponentの場所にある

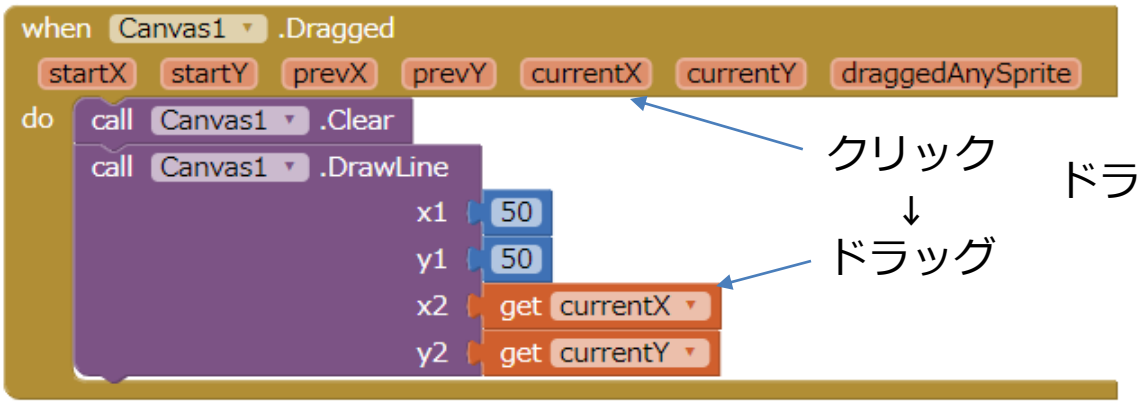
球を飛ばす処理

50ミリ秒に1回行われる処理
球を移動する
球の座標を画面に表示する

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

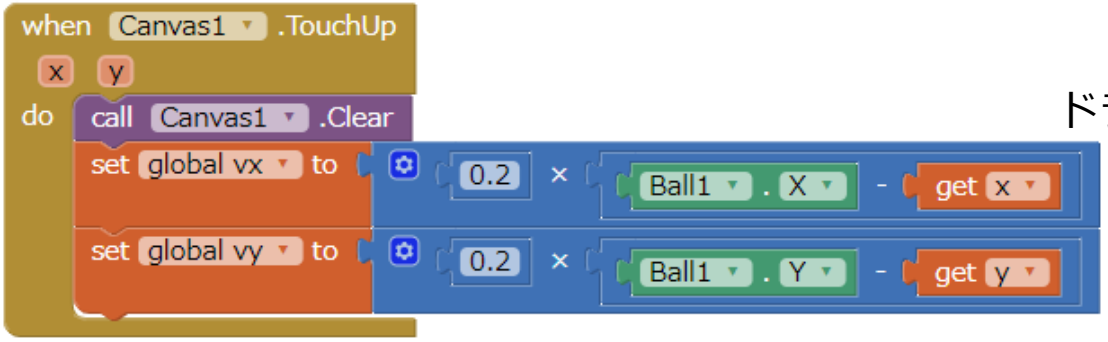
$$y' = y + v_y \times \Delta t$$

(Δt=1 と考える)



クリック
↓
ドラッグ

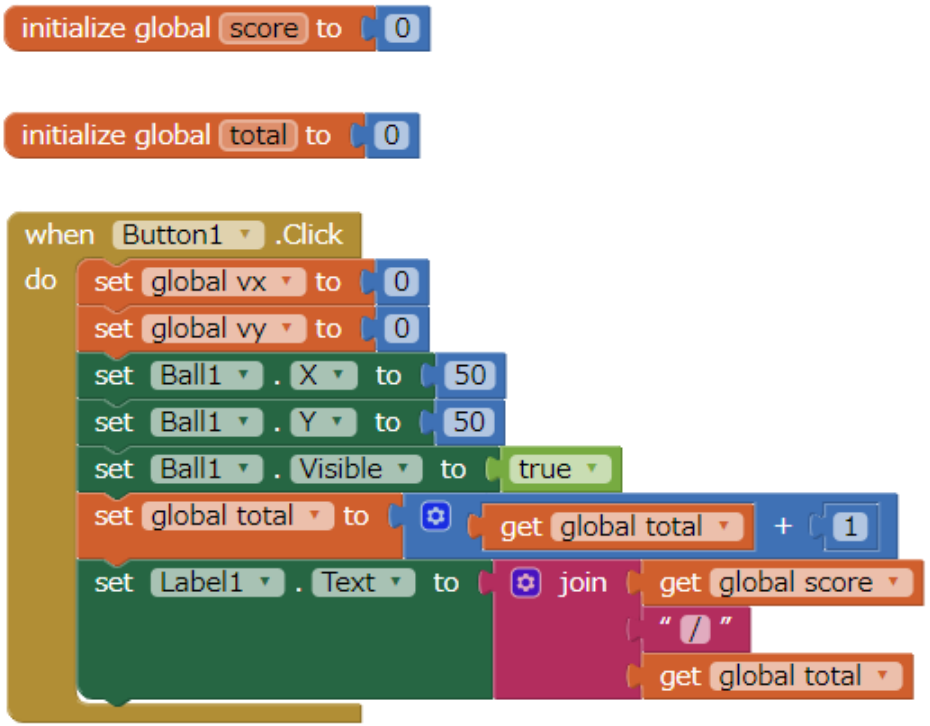
ドラッグしたとき
球の初期位置から現在地まで
直線を描く



ドラッグした手を離したとき
直線を消す
球の移動量を設定する
（手を離した位置が球の位置より
離れている方が大きな値に）

ここまでできたら動作確認すること

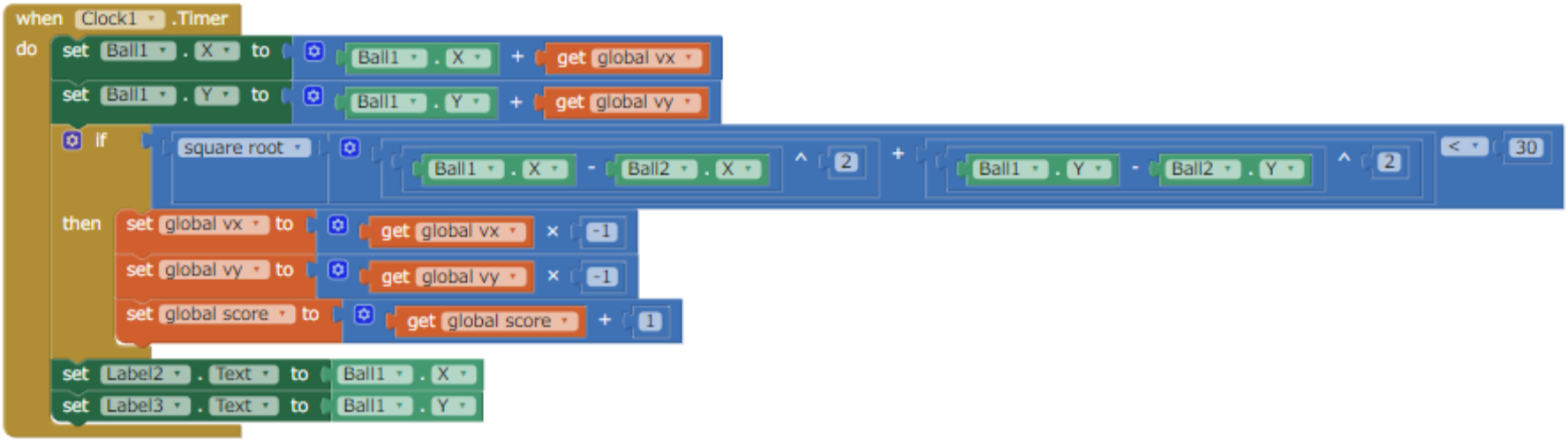
当たった回数と試行回数の表示処理

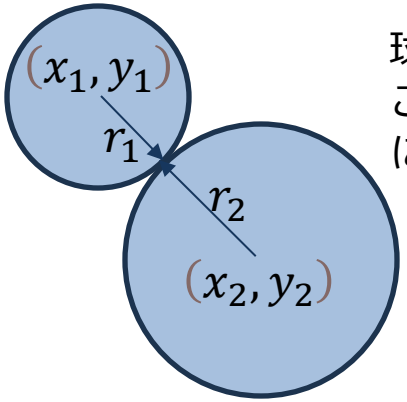


（元からあるブロックに）
追加

ここまでで再び動作確認すること

衝突したら球が跳ね返るようにする



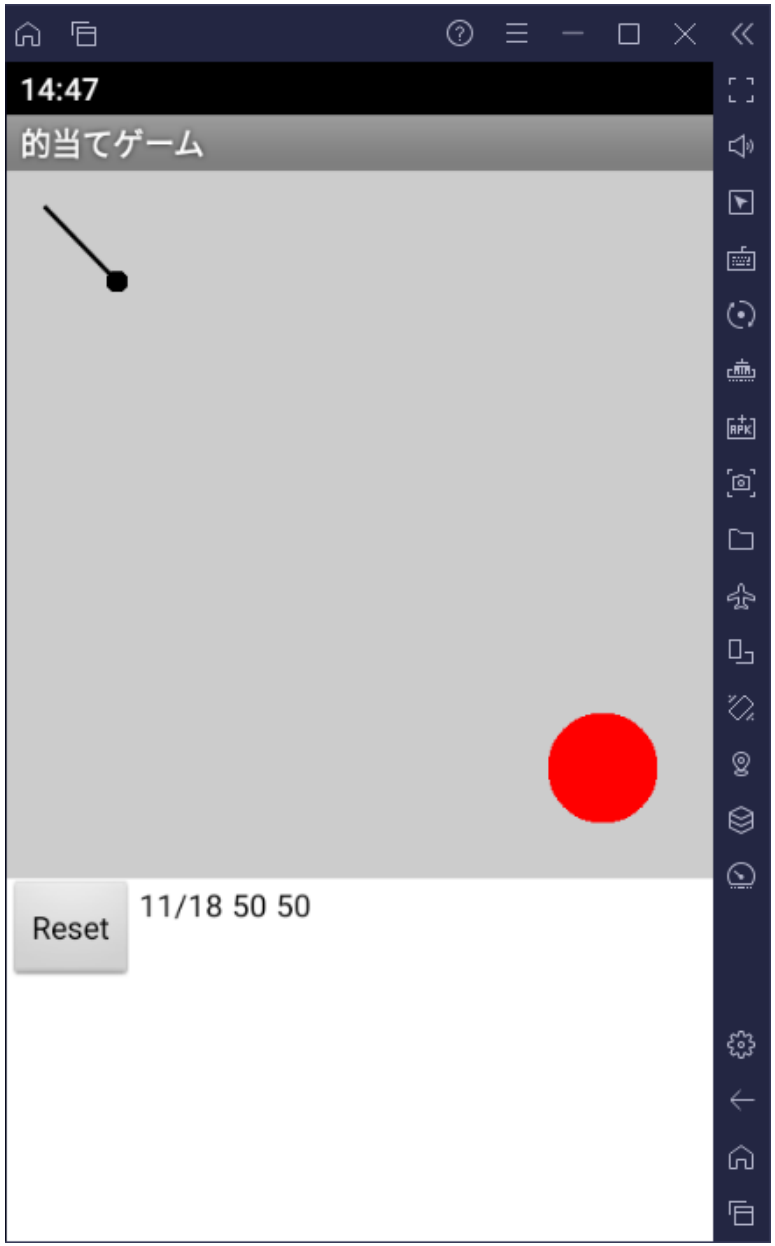


球と的の半径をr_1とr_2とし，中心の座標をそれぞれ(x_1, y_1)と(x_2, y_2)とする。このとき(x_1, y_1)と(x_2, y_2)の距離が$r_1 + r_2$より小さければ重なっていることになり「衝突している状態」と言える。

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad < r_1 + r_2$$

衝突処理を追加したら動作確認

実行画面



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