First App

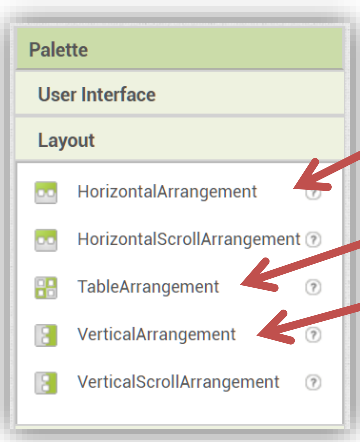
(2) Drawing for Intermediates

For now, you have taken care of canvas and the background image of your app. The next steps will be the layout of the app and the actual painting. You will…

- … learn, how to use **Screen Arrangements** to compose the components of an app.
- … add the functionality to paint with **different colors** on the display.

The appearance of your app

The App Inventor uses so-called ***Screen Arrangements*** to place the objects, i.e. components of an app on the screen. Go to ***Designer*** → ***Palette*** → ***Layout***.

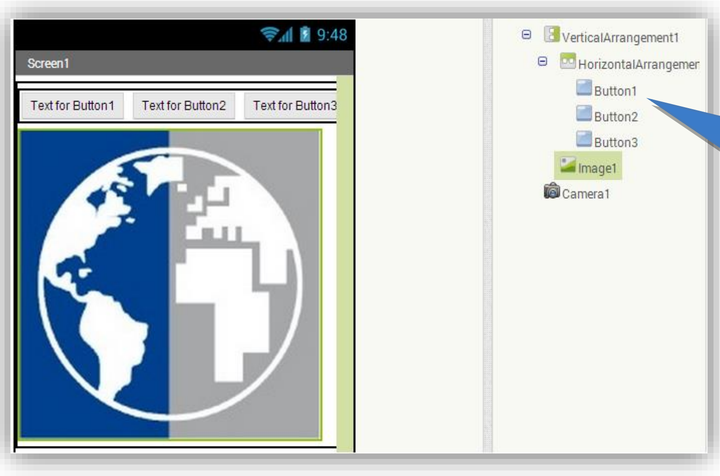


Elements are placed next to each other **horizontally**.

Elements are placed **block-wise** (e.g. 4x4).

Elements are placed above each other **vertically**.

You can choose between scrollable and non-scrollable. For now, use the latter. The arrangements can be combined, as shown below:



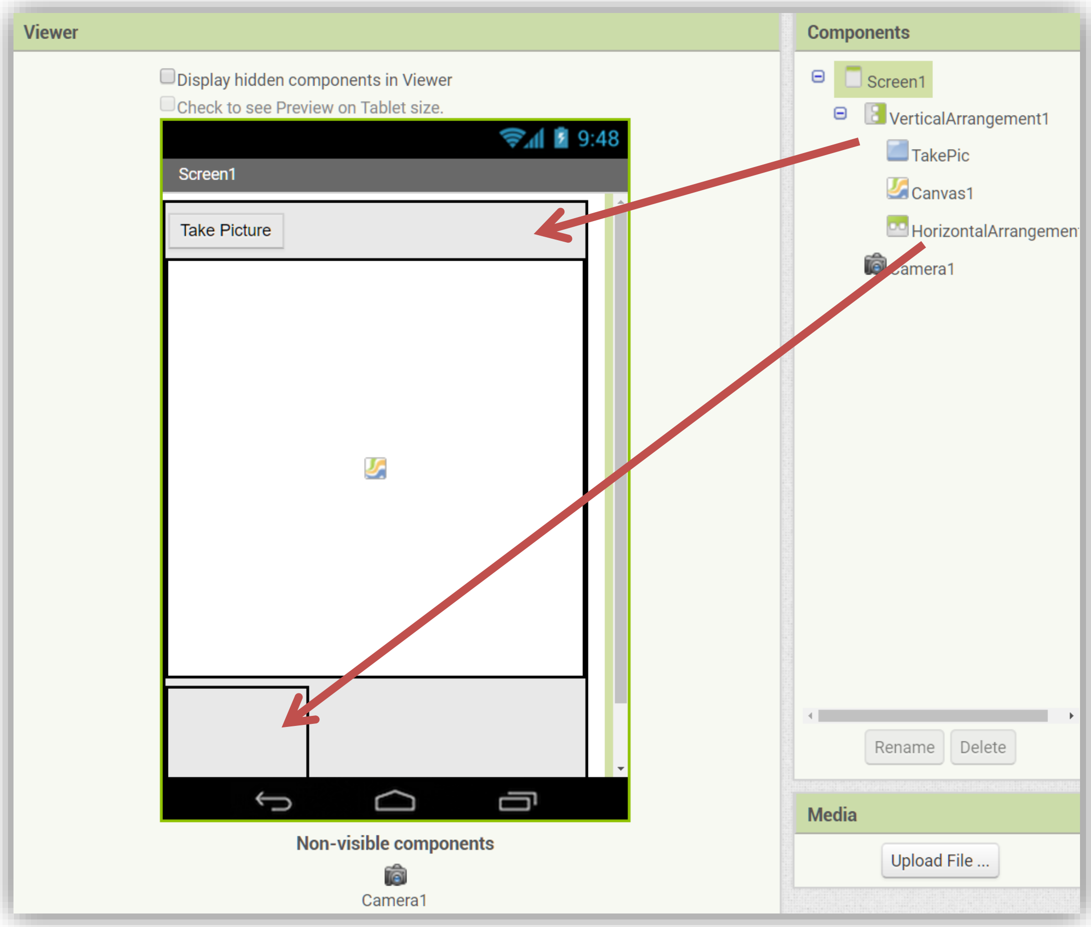
Here, a vertical and a horizontal arrangement are used to place three buttons next to each other (horizontally) above a picture (vertically).

Be careful when deleting a screen arrangement, as you will also delete everything in it!

(2) Drawing for Intermediates

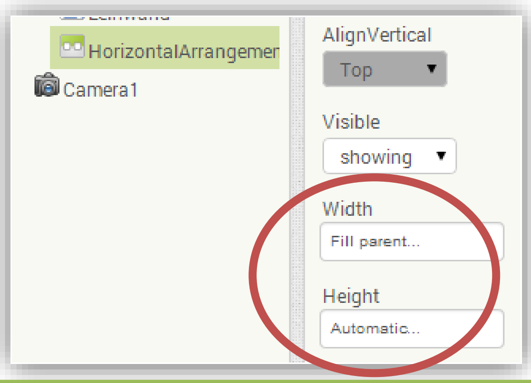
Now it's your turn:

a) Drag a ***vertical arrangement*** into your app on the Viewer.
b) Drag the picture button and the canvas into the arrangement.
c) Drag a ***horizontal arrangement*** into the vertical arrangement.



The necessary elements for the painting will be placed into the empty arrangement in the next step.

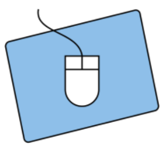
To fully utilize the screen of your tablet, set the **Width** and **Height** of the screen arrangements to **Fill Parent**.



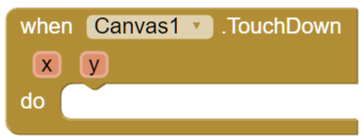
On the next page, we will finally start painting…

(2) Drawing for Intermediates

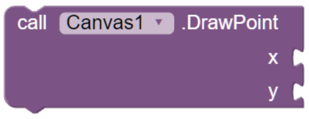
First a dot...



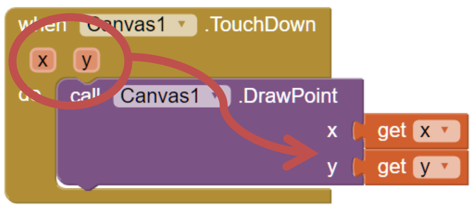
a) Go to the **Blocks Editor** and look for this block in the menu of your canvas:



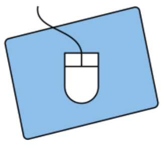
b) Into the space, you put the lilac block on the right. It can also be found in the menu of the canvas.



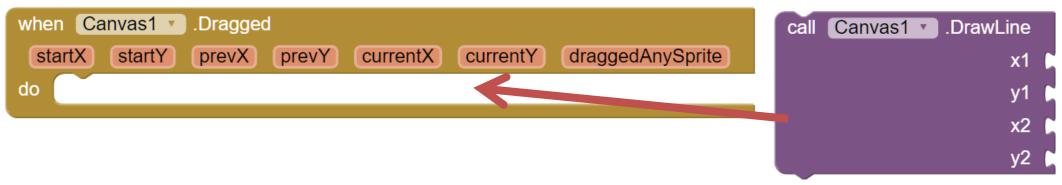
c) Now place the mouse cursor above the X and drag the get-block into the suitable place. Do this for Y also.

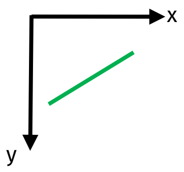


... then a line!



The line works quite similar. Find the blocks below in the menu of your canvas and put them together.

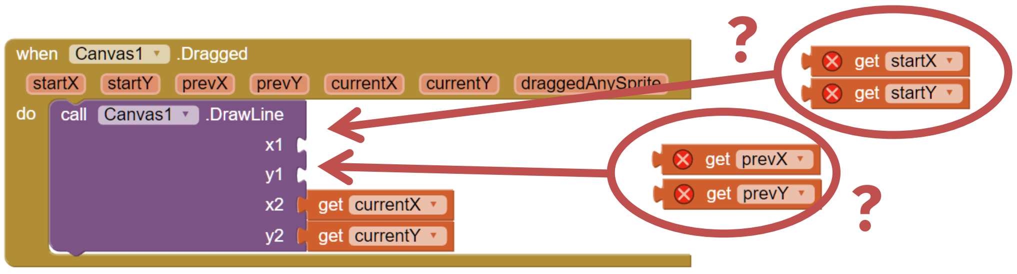




The brown block handles the reaction to dragging the screen with your finger. The lilac block draws a line between the coordinates **(x1, y1)** and **(x2, y2)**. Into the spaces for x2 and y2, you have to put the get-blocks from currentX and currentY.

Which coordinates (start or prev) need to be placed into the spaces for x1 and y1? What happens if you choose wrong?

startX and *startY* refer to the point where the screen is touched first; *prevX* and *prevY* refer to the previous location of your finger on the screen. Just try it out!



Bring in the colors!

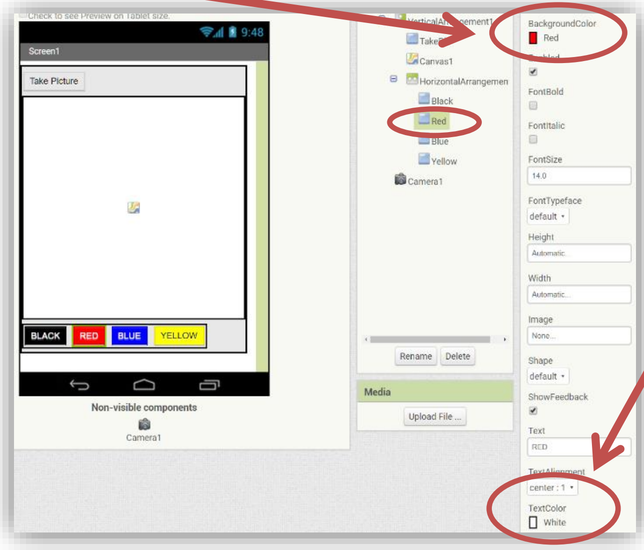
Now you will add different colors and line widths to your app:

a) Switch to the designer and drag **four buttons** into the empty horizontal arrangement.

b) Use “**Rename**”, to give the buttons the names **Black**, **Red**, **Blue** and **Yellow**. Also change the texts of the buttons. Of course, you may use other colors.

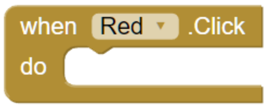
c) Use the Properties to design the buttons. You could use *TextColor* or *BackgroundColor*.

It could look like this:



BackgroundColor: 🟥 Red
Enabled: ✓
FontBold: ☐
FontItalic: ☐
FontSize: 14.0
FontTypeface: default
Height: Automatic...
Width: Automatic...
Image: None...
Shape: default
ShowFeedback: ✓
Text: RED
TextAlignment: center : 1
TextColor: ☐ White

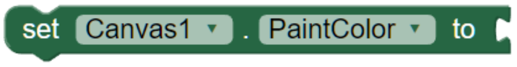
a) Go to the Blocks Editor and select a colour button. Drag the click-block to the workspace:



```
when Red .Click
do
```

.

b) Get this block from the menu of your canvas:

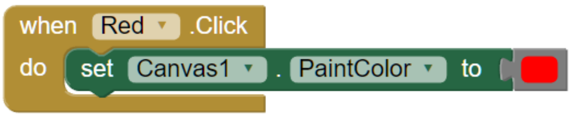


```
set Canvas1 . PaintColor to
```

.

Compose the blocks.

c) In **Colors**, you will find the last block needed:

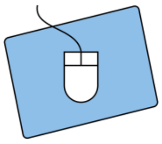


```
when Red .Click
do  set Canvas1 . PaintColor to 🟥
```

.

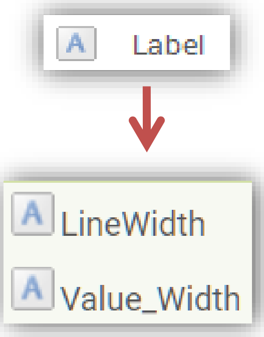
d) Repeat for the other colors.

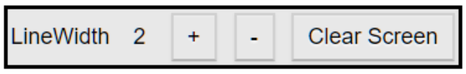
Line width and eraser

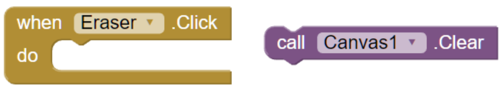


To adjust the line width and add an eraser, you need to switch back to the Designer.

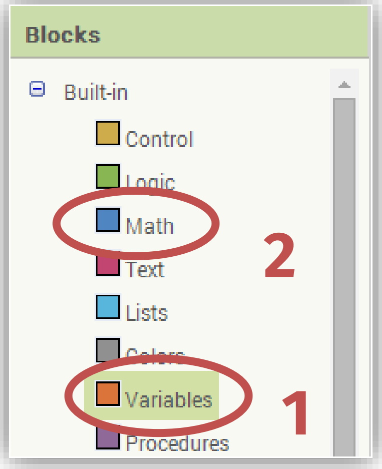
a) You need a **new horizontal arrangement** for the new elements.
b) Drag two **Labels** from *User Interface* into the new arrangement.
c) The first label gets the name and the text *"LineWidth"*. The second label gets the name *"Value_Width"* and a 2 as text (default line width).
d) Next you'll need **two buttons** to adjust the line width. Name them appropriately, e.g. "Width_UP" and "Width_DOWN" and also change their texts (here "+" and "-").
e) Add another button for the eraser that will clean the canvas and change its name and text.
f) **Go to the Blocks Editor.**
g) You need the blocks below for the eraser:



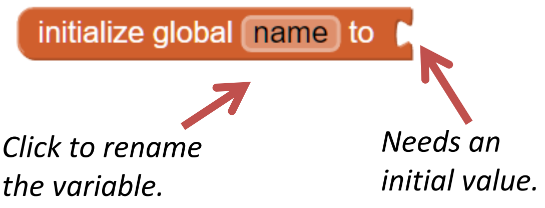




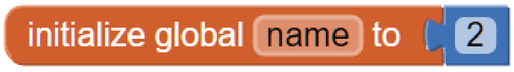
Do you know where to find them?



For the pen (i.e. your finger) to be able to paint, you need to assign it a line width. You create a **variable (see 1)** and name it "lineWidth":


initialize global name to
Click to rename the variable.
Needs an initial value.

The initial value (the default line width) is set with a number block from **Math (see 2)**. Just click on the block and type in the number:



You will learn how to change the line width on the next page.

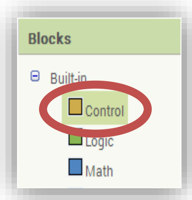
How to adjust the line width…

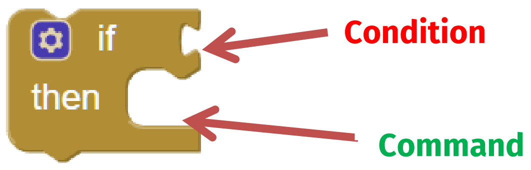
The line width should be between 1 and 5 pixels. So you need to verify that it is within this range before changing it. For that, you need the **if-then-command**. It works like this:

IF the sun is shining tomorrow, THEN I'll go to the swimming pool.

Condition Command

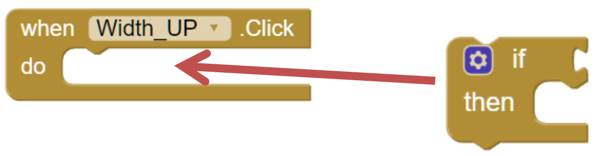
You can find the **if-then-block** in the Blocks-Editor under **Control**.



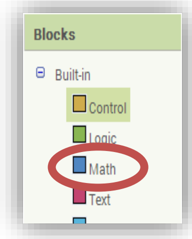


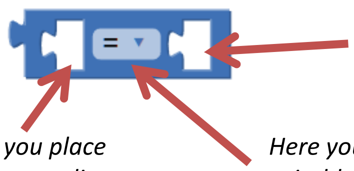
Increasing the line width:

a) Drag the click-block for the button to increase the line width and add the **if-then-block** from **Control**.



b) The condition to be tested is if the current line width is still **lower** than the maximum value. You need the mathematical comparison from **Math**:



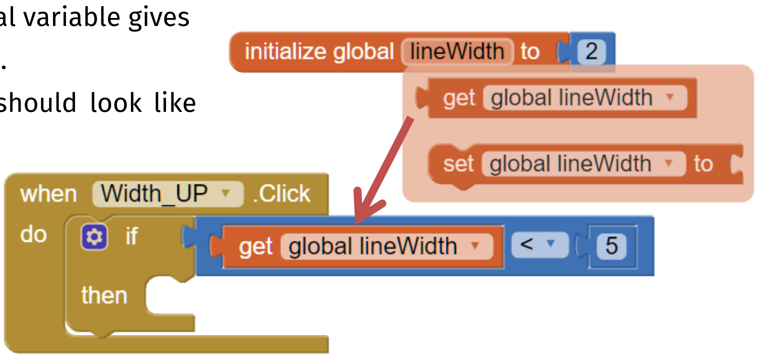


This is the maximum value, so you put in the number 5.

Here you place the current line width.

Here you choose the suitable operater (you need "<").

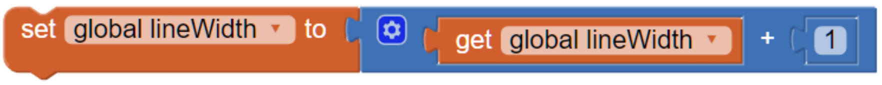
The **get-block** of the global variable gives you the current line width.
The complete condition should look like this:



We will take care of the THEN-part on the next page.

The following steps need to be taken in the then-part:

a) Increase the line width by 1:

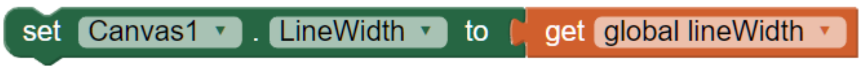


```
set global lineWidth to  get global lineWidth + 1
```

.

You get the *set*- and *get-block* from the global variable and the blocks for the summation and the number from **Math**.

b) Pass on the line width to the canvas:

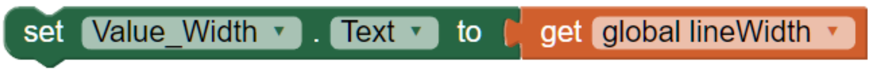


```
set Canvas1 . LineWidth to  get global lineWidth
```

.

You get the green block from your canvas. It sets the line width to the value connected to it.

c) Refresh the label showing the line width:

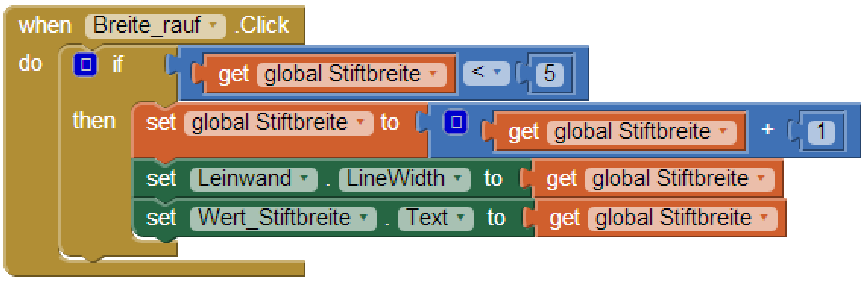


```
set Value_Width . Text to  get global lineWidth
```

.

You get the set-Block from the menu of your label "Value_Width".

When everything is done, your block to increase the line width should look like this.



```
when Breite_rauf .Click
do  if    get global Stiftbreite < 5
    then  set global Stiftbreite to  get global Stiftbreite + 1
          set Leinwand . LineWidth to  get global Stiftbreite
          set Wert_Stiftbreite . Text to  get global Stiftbreite
```

In the same way, you can now compose the block to decrease the line width. Here you need to test if the current line width is **higher than the minimal value** and, if so, **decrease the line width by 1**.

Now your drawing app is done! You can test it and polish up the design. Or you can get the bonus sheet, "(3) Drawing for Experts" and add additional features. If you wish to, you can transfer your app to your own smartphone. Just ask a tutor for help.