Game: Pick the color!

This exercise will build on the knowledge and skills gained from the previous tutorials. You will be creating a simple game of Pick the color. There is only one rule for the game, the player needs to select the color of the word on the screen from a selection of 6 colors. Get it right - earn a point, get it wrong - loose a point.

At the end your app should look similar to the picture below -

The exercise should be attempted without the use of the following instructions
Why? - As you have already done a few tutorials before hand, you should have the necessary knowledge and skills to attempt this on your own. Simply put “Try and fail, but don’t fail to try” - Stephen Kaggwa. Learning by doing is the best method in my opinion.

Step 1 -

First Let’s think about the game and interface.

The game is simple, On the screen the text and It’s color change every second. When the player selects a color and it matches the color of the text (NOT the word), then they get a point, if not they loose a point. Simple!

Now we need to know what needs to be on screen, and which elements are interactive.

a) We need a Label - To display some text with color
b) We need 6 buttons, each with a different background color - This allows the player to click the appropriate box
c) We need another Label for the score - Allows the player to keep track of their score
d) We need a clock
That’s it!

So let’s begin by dragging the required UI(User interface) elements to the screen.

It’s up to you how large the text should be, however size 40 seems like a good fit.

Reminder:

The arrangement of the buttons can be achieved with the arrangement helpers provided by app inventor inside the “Screen Arrangement” menu on the left.
Now Let's continue by naming all the elements you dragged on to the screen.

The Top Label can be named - “ColorNameLabel”

The Color buttons can be named - [Color of the button]Button - e.g. “BlueButton”

The Score Label can be named - “ScoreLabel”

The Color can be named - “ChangeLabelTime” - *(We name it this way, because it will be specifically used to change the Top label every second, So if we need another timer which has 2 or more seconds interval, we can keep track of all our timers)*

---

**Step 2 -**

Now let's begin by setting background colors for our six buttons, make sure each button has a unique color.

That's it for the interface!... Now let's put the puzzle together.
Step 3 -

Once you have completed changing the background color of the buttons, click on the “Open Blocks Editor” on the top right hand corner of the screen.

You should now see something similar to the above, and empty editor.
Step 4 -

Now let's think about the logic behind the game.

We know that the Text and its background color needs to change every second.

We know that the Text needs to be a name of one of the colors we are using on the buttons.

We know that the Color of the text needs to be one of the colors we are using on the buttons.

So let's begin by selecting our timer, Inside the My Blocks menu, select your timer, and in choose the following block with the "Timer" on the end.

![Timer Block](image)

This block runs any procedure we give it, every second (This can be modified in the interface editor where we can change the timer's interval)

Therefore we can use this to run a procedure which change the text and the color of the text. Which is our next step.

First of all we need to have a variable for our color and our text. so that we may keep track of them and use them.

So let's drag out 2 empty Variable blocks and name them as follows -

![Variable Blocks](image)
We now need to give these variables some default values, so let’s pick a random color and a random text and assign them as the values as shown below -

![Image of variable assignment]

Next we need to start a procedure which changing the values of our “Color” and “Text” variables and another procedure which uses the new values in our variables on the screen.

So let’s begin our first procedure called “ChangeColorAndTextVariableValues” -

![Image of procedure block]

This procedure has to set the new values, so inside “My Blocks” menu, select My Definitions and choose the block which has “Set Global” for the variables “Color” and “Text”

![Image of set global blocks]
Next we need a list of all the colors and text we can use. To do that we need to create a list and add in the appropriate values to the list. Select the “Lists” option in the Built-In menu, and choose “Make a List” block -

Now let's add the colors we can use to this list -

Do the same for a second list for the text we can use
Now that we have some lists, we need to be able to select a random item and assign it as the new variable. To do that we need to use built-in procedure inside the list's menu called “Pick random item” -
Now let's use this inside our procedure called CangeColorAndTextVariableValues -

Great!, now we have a procedure which can change the values of our “Color” and “Text” variables. Now we need a procedure which allows the interface to use the values of “Color” and “Text”, so let’s create a new procedure called “ChangeLabel” and set it up as follows -
Now it's time to use our timer to start the “ChangeColorAndTextVariableValues” and “ChangeLabel” procedures -

That’s It, now let's start a new emulator and connect to it, If you see a “Run Time Error” it means you have a block that is incomplete or incorrect. In which case, go back through the steps to make sure you haven't missed anything.

Step 5

We now need to see if the player clicked on the right color. To Check this, we need to first of all see what buttons the player clicked, So we need to use something that runs when a player clicks on it -

E.G. -

So for example, this block runs when a user clicks on our button called - “BlueButton”. At this point we need to have what’s called a Conditional statement...

For example, let's think of this in plain english.

We need to know WHEN the user CLICKS on the blue button and when clicked, we need to know IF the color of the text matches the color blue.

so keyword - “When”, “Clicks”, and most importantly “IF”, the “When” and “Click” part is already done for you, now you need to check “IF” the user clicked the right button.
And so we can use the built-in control called “IF” -

Now we need to perform logic inside the IF statement... we need to see if the text color of what ever is inside ColorNameLabel is EQUAL to the color of our button -
Now anytime the test passes, we can tell the IF statement to do something, in this case - add to the score. However before we move on to that, replicate the same login as above to the rest of the buttons. So that you will end up with something similar to the following -

Step 6 -

Now we need a variable for our score, and a procedure to add to the current score and display it on the screen.

So first we need a score variable with a default value of 0 -

Now we need a procedure to add 1 point to the current score

Therefore we need to know what the current score is, and add 1 point to it, so you should end up with the following -
Now that we have a procedure to update and display the score, let’s use it inside our “.Click” blocks of all our buttons as follows -

That’s it!! you’re done. You now have a simple and an addictive game.

EXTRA:

- If the user does not get the color right, remove 1 point from their score and display it on the screen. Clue: you will have to use the “IF ELSE” block instead.

- Make the word and color change faster with every correct answer. Clue: you will need to work with the timer.

- Randomize the color of the buttons with every correct answer