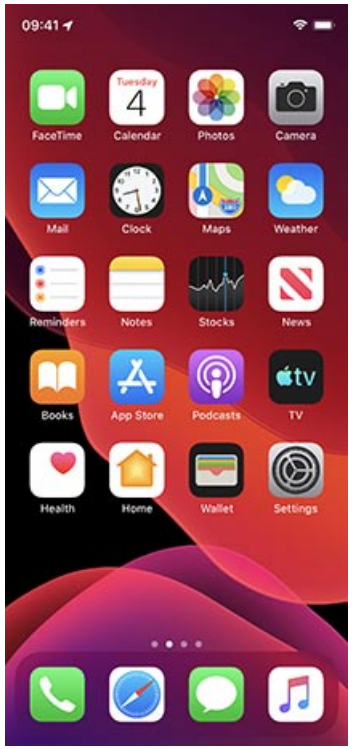
iOS SwiftUI PROGRAMMING EXERCISES

Kari Laitinen
http://www.naturalprogramming.com
2020-09-14 File created.
2020-09-14 Last modification.



EXERCISES WITH SquareBallRectangleSUI

Among the iOS SwiftUI example apps, you'll find an app named SquareBallRectangleSUI. This app displays a square, a ball, or a rectangle, depending on which shape is selected with the so-called Segmented Control.

You can start using this app by downloading the file **SquareBallRectangleSUI.zip** and unzipping it locally. On macOS, you can unzip a **.zip** file by double-clicking it in Finder. Safari may unzip a file automatically after downloading.

Please, ensure that this app works before you start doing the following exercises.

Please, note that there exist older versions of this app. One is made with the UIKit technology and an even older app is based on Objective C. So you must be careful that you start using a correct version of the app. Apps made with SwiftUI have the acronym SUI in their name.

Exercise 1:

Modify the program so that it displays a triangle in place of the rectangle.

By studying the app named ShapesDemoSUI you can find out how a triangle can be shown on the screen. You can copy from **ShapesDemoView.swift** a `struct` named `Triangle`. The struct provides a suitable drawing path to make a triangle. There are also code lines that are needed to show a triangle.

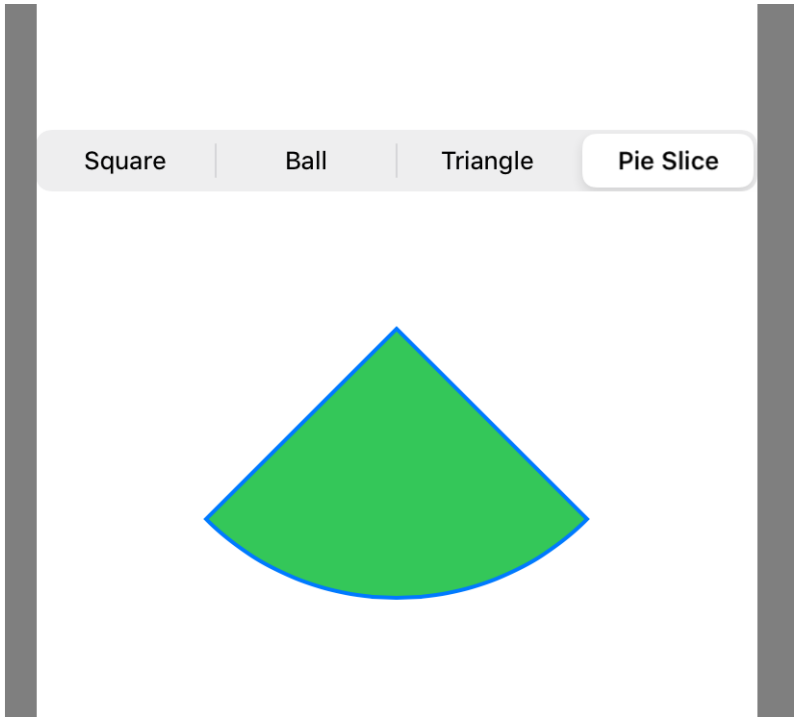
After you have done this exercise, the user interface (UI) must show the word "Triangle" in place of "Rectangle".

Exercise 2:

Improve the program further so that there will be a new selectable shape named "Pie Slice". You must modify the Segmented Control so that it provides four possible selections.

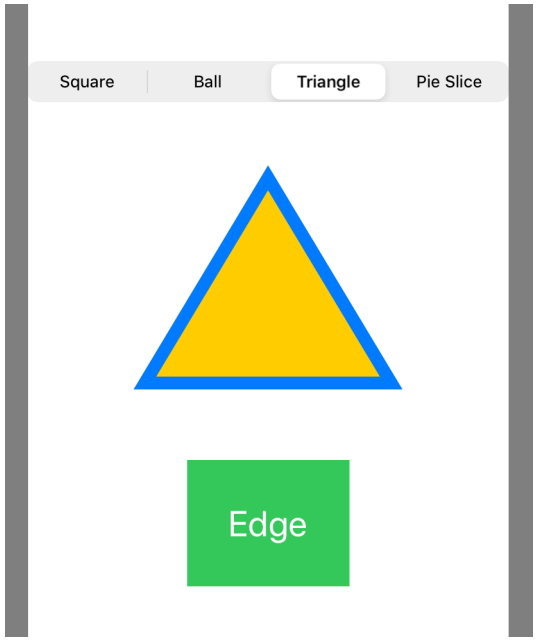
ShapesDemoView.swift provides a `struct` named `Pacman`. By studying that `struct`, you'll discover how to make a `struct` named `PieSlice`.

When the "Pie Slice" is selected in the UI, the screen should look like the following.



Exercise 3:

Improve the program so that the app will have a button with text "Edge" to adjust the thickness of the egdes of the shapes. The thickness, or line width, of the edges can be increased with two (points) always when the button is pressed. If the line width reaches value 10, it should be put back to value 2. After this modification, the UI could look like the following.



To control the line width of the edges, you can use a `@State` variable that can be declared in the following way.


```
@State var edge_line_width : CGFloat = 2.0
```

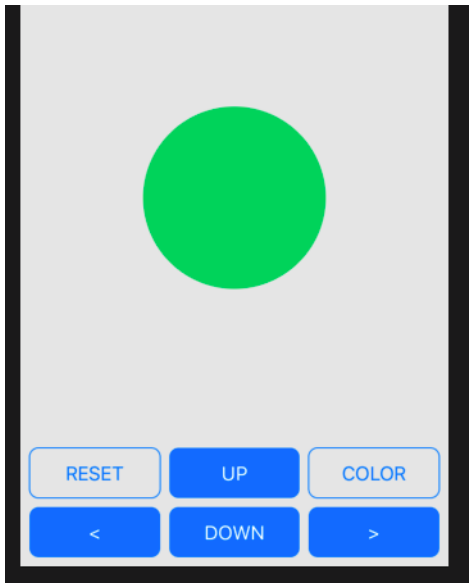
You should study the app named ButtonDemoSUI to find out how a `@State` variable can be used.

A `@State` variable is such that, when the value of a `@State` variable changes, the system automatically executes those blocks of the program in which the variable is used.

EXERCISES WITH MovingBallSUI

Exercise 1:

The original app has four buttons to move the ball, and two buttons for other purposes. If we think about the usability of the app, the buttons for moving the ball should be easily distinguishable from the other buttons. So, your task in this exercise is to modify the program so that the buttons for moving the ball have a darkish background and white texts, in the following way.



In the original program, the style of the buttons is specified with a `struct` named `BallButtonStyle`. You can make a copy of that `struct`, name it `MovementButtonStyle`, modify it as is necessary, and use the style for the buttons that move the ball. You can easily specify a background color for the buttons when you use `.fill()` instead of `.stroke()`.

Exercise 2:

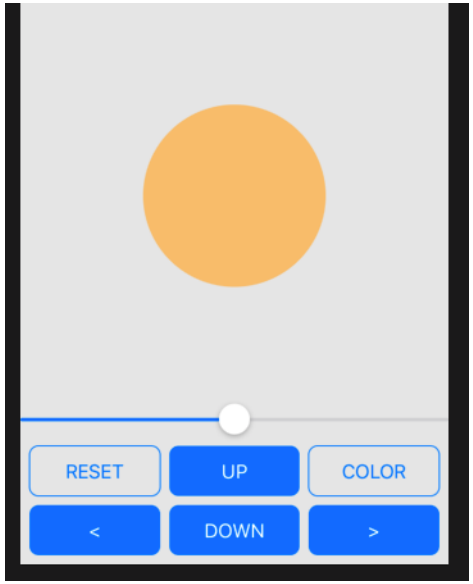
In the current version of the program, a pressing on the RESET button will show an `Alert`. Modify the program so that no `Alert` is shown and a real RESET is made. You should just disable the showing of the `Alert`. We'll later find a better use for the `Alert`.

In a RESET operation, the ball must be placed to the initial position on the screen, and it must also get the same color it had when the program was started.

Exercise 3:

By studying an app named TurningArrowSUI you'll find out how to use a `Slider`. `Slider` is a UI control with which it is possible to adjust some numerical value in a specified range.

Now your task is to add a `Slider` to the MovingBallSUI app so that the UI looks like the following. You can first put the `Slider` to the app, and later make it do something.



The goal in this exercise is that the `Slider` can be used to adjust the transparency of the color of the ball. The transparency of a color can be specified with so-called alpha value. When speaking about transparency, we can also use the term ’opacity’. A fully opaque color has no transparency.

In iOS programming alpha values are decimal values in the range 0 ... 1. Value 0 means a fully transparent color and value 1 means a fully opaque color. The app must change the alpha value of the filling color of the ball always when the `Slider` value changes. The alpha value can be 0.5 in the beginning.

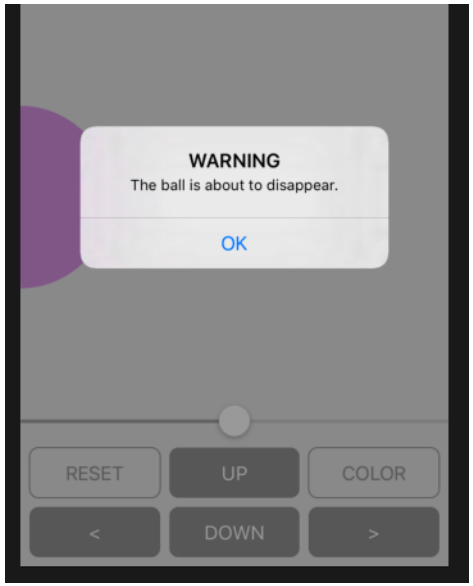
In this exercise you can use a `@State` variable to describe the opacity of the color. The opacity

can be set initially with method `.opacity()` in the following way.

```
Circle().fill( ball_filling_color.opacity( ball_color_opacity ) )
```

Exercise 4:

If you have done the previous exercises according to instructions, there is an `Alert` in the program but it is never used. Now your task is to show an `Alert` when a half of the ball is outside the movement area, as in the following picture



As the ball can move on the entire screen area, you can compare ball offset to the size of the screen. By studying the app ShapesDemoSUI, you can find one way to find out the size of the screen.

In this exercise, you must check the ’walls’ of the screen in all possible directions of movement. For all collisions with the walls, you can use the same `Alert`. When the teacher did this exercise, he put the following to the end of the `VStack` that contains the `HStack`s for the buttons

```
.alert( isPresented: $showingAlert )
{
    Alert( title: Text( "WARNING" ),
           message: Text( "The ball is about to disappear." ),
           dismissButton: .default( Text( "OK" ) ) )
}
```