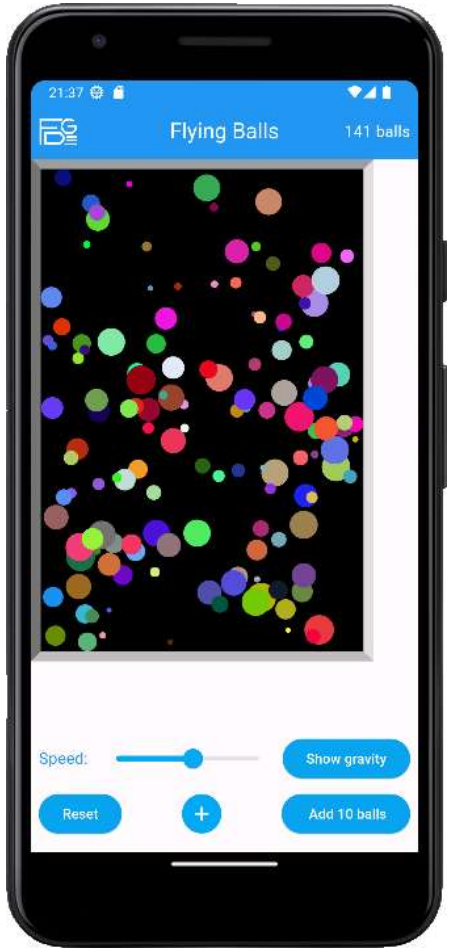
Developing the “Flying balls” application

- Use the Stack and Positioned widgets
- Define a periodic Timer
- Create random numbers
- React on user gestures
- Use object oriented principles when defining the Ball class
- Control the visibility of widget

Our Goal

After Start:

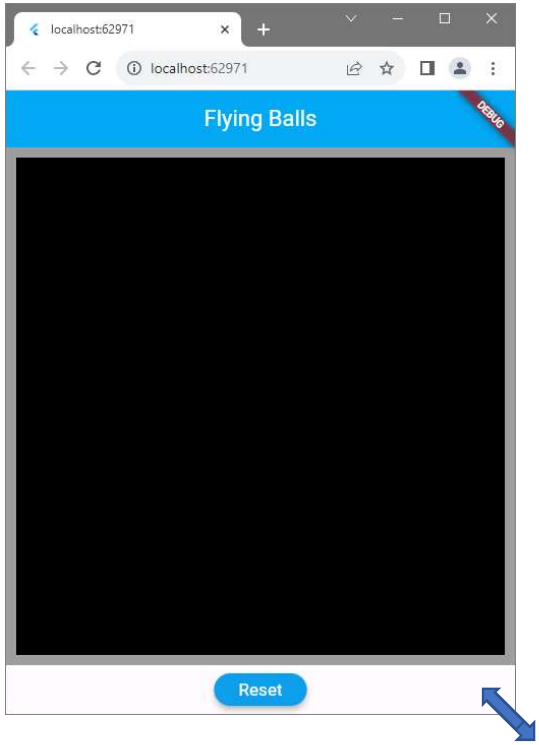
In “Expert Mode” after adding balls & with a shrinked box:



Configuring the box

Expanded widget: box size follows browser window size when running as web app

```
body: Column(children: [
  Expanded(
    child: Container(
      decoration: BoxDecoration(
          color: Colors.black,
          border: Border.all(color: Colors.grey, width: 10)), // Bo
    ), // Container
  ), // Expanded
  Padding(
    padding: const EdgeInsets.all(8.0),
    child: ElevatedButton(
        style: ElevatedButton.styleFrom(
            backgroundColor: Colors.lightBlue, elevation: 5),
        onPressed: () {},
        child: const Text("Reset",
            style: TextStyle(color: Colors.white, fontSize: 17))),
  ), // Padding
```



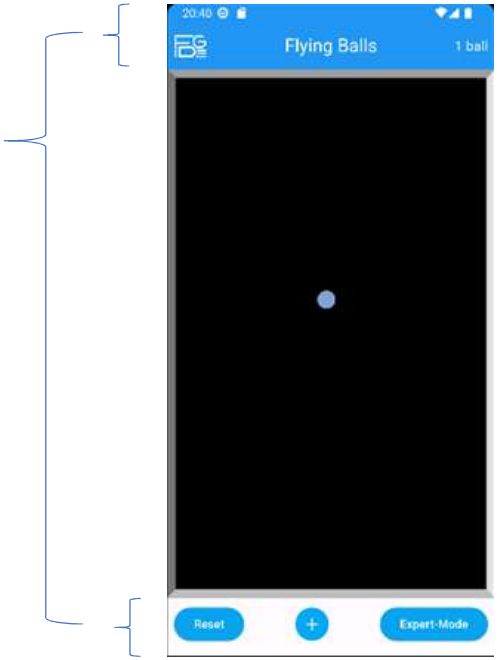
Disadvantage when using Expanded widget

To bounce the balls on the lower border, we need to know the height of the black box. This cannot be directly calculated when using the Expanded widget.

Workaround: estimate the height around the black box (which stays fixed).

```
double getMainBoxHeight() {
  double screenHeight = MediaQuery.of(context).size.height;
  return screenHeight - 150;  // 150 found by try-and-error
}
```

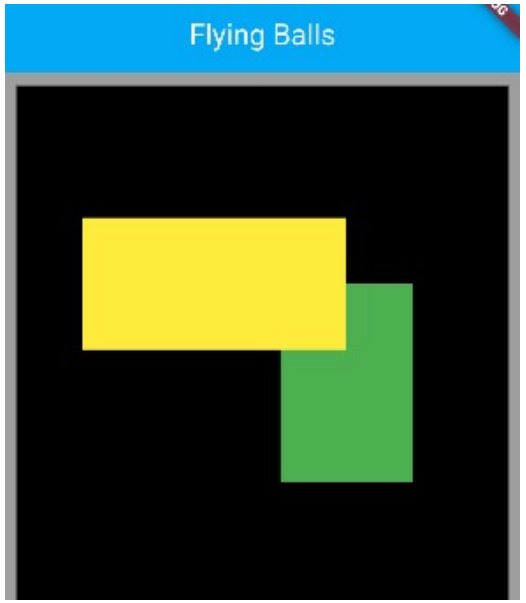
```
body: Column(
    children: [
      //Expanded(  // difficult to calculate the height
      SizedBox(
        height: getMainBoxHeight(),
        child:
            Container(
            decoration: BoxDecoration(
                color: Colors.black,
                border:
                    Border.all(color: Colors.grey, width: 10)),
```



Using Stack widget to position children

With the widgets “Stack” and “Positioned”, you can position and stack children.

```
SizedBox(
    height: getMainBoxHeight(),
    child: Container(
        decoration: BoxDecoration(
            color: Colors.black,
            border: Border.all(color: Colors.grey, width: 10)), // BoxDecoration
        child: Stack(children: [
          Positioned(
              top: 150,
              left: 200,
              child: Container(
                width: 100,
                height: 150,
                color: Colors.green,
              )), // Container // Positioned
          Positioned(
              top: 100,
              left: 50,
              child: Container(
                width: 200,
                height: 100,
                color: Colors.yellow,
              )), // Container // Positioned
        ])), // Stack // Container // SizedBox
```



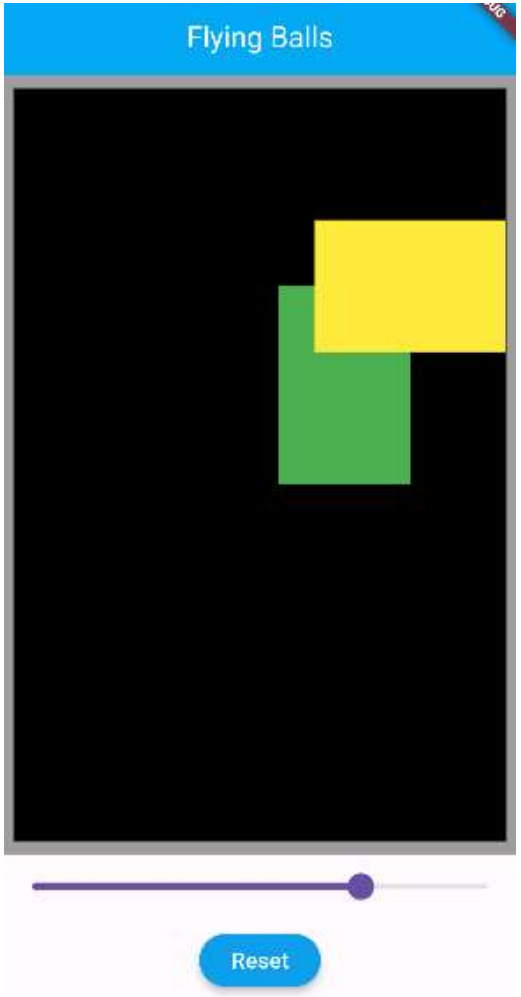
Last child is drawn above previous ones.

First test: Modify the position with a slider

```
Positioned(
    top: 100,
    left: leftYellow,
    child: Container(
      width: 200,
      height: 100,
      color: Colors.yellow,
    )), // Container // Positioned
```

```
Slider(
  min: -199,
  max: 300,
  value: leftYellow,
  onChanged: (value) {
    setState(() {
      leftYellow = value;
    });
  },
) // Slider
```

Negative values for “left” are allowed.
They move the yellow container out of the box.



Modify the position with a periodic timer

Override base class method “initState” to create a periodic timer:

```dart
class _MainPageState extends State<MainPage> {
  double leftYellow = 50;
  double speedX = 2;

  @override
  void initState() {
    Timer.periodic(const Duration(milliseconds: 20), timerCallback);
    super.initState();
  }
```

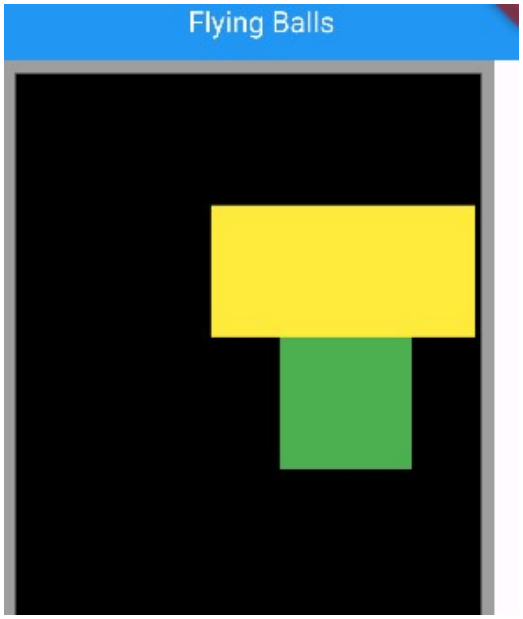
```dart
void timerCallback(Timer timer) {
  var screenWidth = MediaQuery.of(context).size.width;
  setState(() {
    leftYellow += speedX;
    if (leftYellow > screenWidth) {
      leftYellow = screenWidth;
      speedX = -speedX;
    }
    if (leftYellow < 0) {
      leftYellow = 0;
      speedX = -speedX;
    }
  });
}
```

Bouncing inside the box

We have to calculate, when the yellow container reaches the right border:

```
class _MainPageState extends State<MainPage> {
  double leftYellow = 50;
  double widthYellow = 200;
  double speedX = 2;
  double marginRight = 20;
  double borderWidth = 10;
```

```
void timerCallback(Timer timer) {
  var screenWidth = MediaQuery.of(context).size.width;

  setState(() {
    leftYellow += speedX;
    var maxLeft = screenWidth - marginRight - widthYellow - 2 * borderWidth;
    if (leftYellow > maxLeft) {
      leftYellow = maxLeft;
      speedX = -speedX;
    }

    if (leftYellow < 0) {
      leftYellow = 0;
      speedX = -speedX;
    }
  });
}
```



Bouncing in two directions

```dart
class _MainPageState extends State<MainPage> {
  double leftYellow = 50;
  double topYellow = 100;
  double widthYellow = 200;
  double heightYellow = 100;
  double speedX = 2;
  double speedY = 2;
  double marginRight = 0;
  double marginBottom = 0;
  double borderWidth = 10;
```

```dart
Positioned(
    top: topYellow,
    left: leftYellow,
    child: Container(
      width: widthYellow,
      height: heightYellow,
      color: Colors.yellow,
    )), // Container // Positioned
```

```dart
void timerCallback(Timer timer) {
  var screenWidth = MediaQuery.of(context).size.width;

  setState(() {
    leftYellow += speedX * speedFactor;
    var maxLeft = screenWidth - marginRight - widthYellow - 2 * borderWidth;
    if (leftYellow > maxLeft) {
      leftYellow = maxLeft;
      speedX = -speedX;
    }
    if (leftYellow < 0) {
      leftYellow = 0;
      speedX = -speedX;
    }

    topYellow += speedY * speedFactor;
    var maxTop = getMainBoxHeight() - marginBottom - heightYellow - 2 * borderWidth;
    if (topYellow > maxTop) {
      topYellow = maxTop;
      speedY = -speedY;
    }
    if (topYellow < 0) {
      topYellow = 0;
      speedY = -speedY;
    }
  });
}
```

Make a ball out of the yellow container

Old:

```
double yellowWidth = 200;
double yellowHeight = 100;
```

```
Positioned(
  top: topPos,
  left: leftPos,
  child: Container(
      color: Colors.yellow,
      width: yellowWidth,
      height: yellowHeight),
```

New:

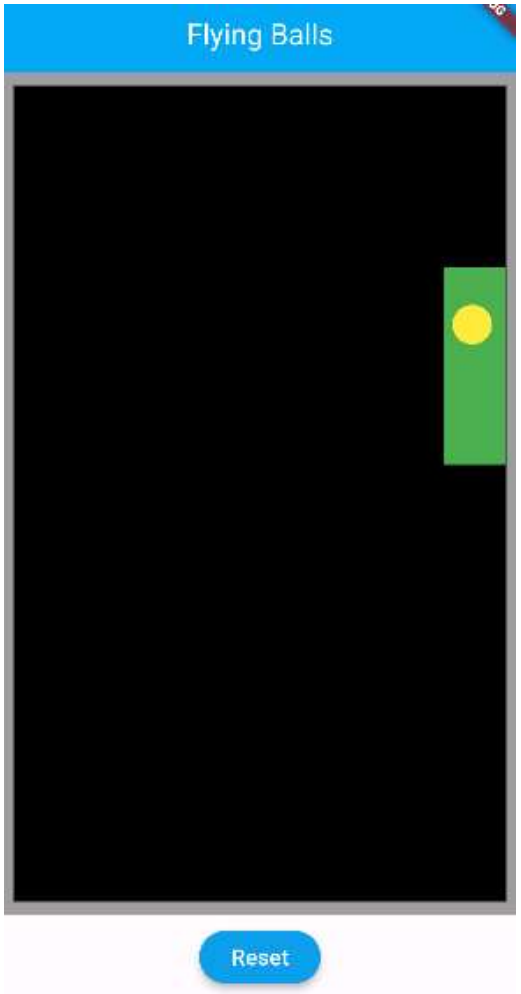
```
double diameter = 30;
```

```
Positioned(
  top: topPos,
  left: leftPos,
  child: Container(
      decoration: const BoxDecoration(
          color: Colors.yellow, shape: BoxShape.circle),
      width: diameter,
      height: diameter), // Container
```

Gesture detection

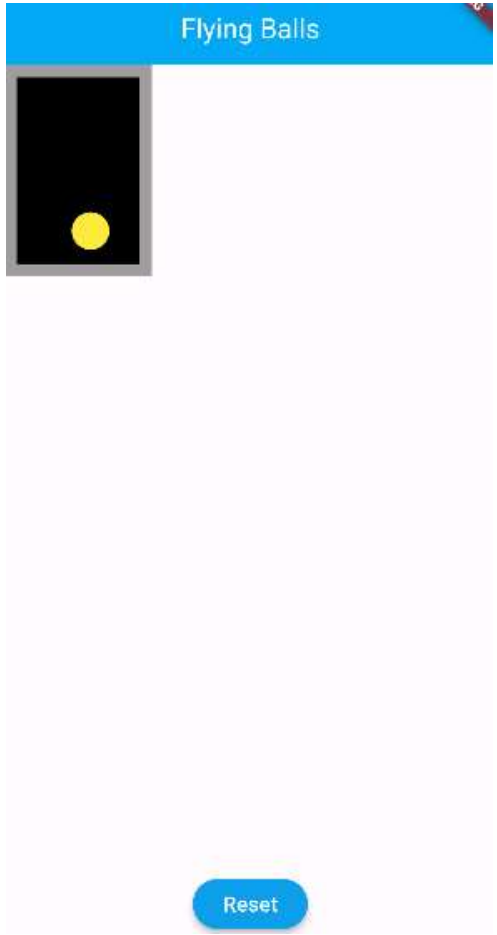
First we try it with the green rectangle

```
Positioned(
    top: topGreen,
    left: leftGreen,
    child: GestureDetector(
      onPanUpdate: (details) {
        //print("${details.delta.dx} ${details.delta.dy}");
        setState(() {
          topGreen += details.delta.dy;
          leftGreen += details.delta.dx;
        });
      },
      child: Container(
          color: Colors.green, width: 100, height: 150),
    )), // GestureDetector // Positioned
```



Change size of black box with panning gesture

```
body: Column(children: [
  Expanded(
    child: GestureDetector(
        onPanUpdate: (details) {
          setState(() {
            rightMargin -= details.delta.dx;
            bottomMargin -= details.delta.dy;
            // do not allow to make box too small:
            var maxRightMargin =
                MediaQuery.of(context).size.width * 0.7;
            if (rightMargin < 0) {
              rightMargin = 0;
            } else if (rightMargin > maxRightMargin) {
              rightMargin = maxRightMargin;
            }
            var maxBottomMargin =
                MediaQuery.of(context).size.height * 0.6;
            if (bottomMargin < 0) {
              bottomMargin = 0;
            } else if (bottomMargin > maxBottomMargin) {
              bottomMargin = maxBottomMargin;
            }
          });
        },
```



Managing many balls (part I)

Create an own class for balls

```
class Ball {
  static double speedFactor = 1;
  double left = 50;
  double top = 100;
  double diameter = 20;
  double speedX = 2;
  double speedY = 2;

  void move(double stackWidth, double stackHeight) {
    left += speedX * speedFactor;
    var maxLeft = stackWidth - diameter;
    if (left > maxLeft) {
      left = maxLeft;
      speedX = -speedX;
    }
    if (left < 0) {
      left = 0;
      speedX = -speedX;
    }
    top += speedY * speedFactor;
    var maxTop = stackHeight - diameter;
    if (top > maxTop) {
      top = maxTop;
      speedY = -speedY;
    }
    if (top < 0) {
      top = 0;
      speedY = -speedY;
    }
  }
}
```

Managing many balls (part II)

Adapted timerCallBack:

```
void timerCallback(Timer timer) {
  var screenWidth = MediaQuery.of(context).size.width;

  setState(() {
    var stackWidth = screenWidth - marginRight - 2 * borderWidth;
    var stackHeight = getMainBoxHeight() - marginBottom - 2 * borderWidth;
    for (var ball in balls) {
      ball.move(stackWidth, stackHeight);
    }
  });
}
```

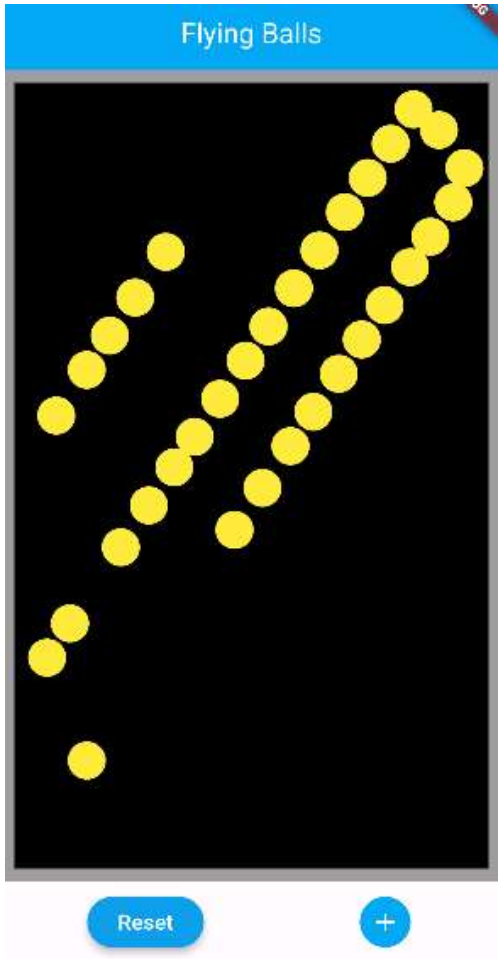
The black Container:

```
child: Container(
    margin:
        EdgeInsets.fromLTRB(0, 0, marginRight, marginBottom),
    height: 600,
    decoration: BoxDecoration(
        color: Colors.black,
        border:
            Border.all(color: Colors.grey, width: borderWidth)),
    child: Stack(
        children: getBallWidgets(),
    )), // Stack // Container
```

```
List<Widget> getBallWidgets() {
  List<Widget> result = [];
  for (var ball in balls) {
    result.add(
      Positioned(
          top: ball.topYellow,
          left: ball.leftYellow,
          child: Container(
            decoration: const BoxDecoration(
              shape: BoxShape.circle,
              color: Colors.yellow,
            ), // BoxDecoration
            width: ball.diameter,
            height: ball.diameter,
          )), // Container // Positioned
    );
  }
  return result;
}
```

Add and Reset Button

```
Padding(
  padding: const EdgeInsets.all(8),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceAround,
    children: [
      ElevatedButton(
          style: ElevatedButton.styleFrom(
              backgroundColor: Colors.lightBlue, elevation: 5),
          onPressed: () {
            setState(() {
              balls.clear();
              balls.add(Ball());
            });
          },
          child: const Text("Reset",
              style: TextStyle(color: Colors.white, fontSize: 17))),
      IconButton(
          style: IconButton.styleFrom(
              backgroundColor: Colors.lightBlue,
              foregroundColor: Colors.white,
              elevation: 5),
          onPressed: () {
            setState(() {
              balls.add(Ball());
            });
          },
          icon: const Icon(Icons.add)) // IconButton
    ],
  ), // Row
), // Padding
```



Random balls

```
Ball() {
  var random = Random();
  speedX = 1 + 3 * random.nextDouble();
  speedY = 1 + 4 * random.nextDouble();
  diameter = 5 + 25 * random.nextDouble();
  top = 30 + 200 * random.nextDouble();

  // to ensure "not so dark" colors
  int minColorValue = 0;

  color = Color.fromARGB(
      255,  // set this e.g. to 200 to make balls a bit transparent
      minColorValue + random.nextInt(255 - minColorValue),
      minColorValue + random.nextInt(255 - minColorValue),
      minColorValue + random.nextInt(255 - minColorValue));
}
```

FDG logo in AppBar

```
appBar: AppBar(
    title: const Text("Flying Balls"),
    centerTitle: true,
    backgroundColor: Colors.lightBlue,
    foregroundColor: Colors.white,
    leading: Padding(
      padding: const EdgeInsets.all(5.0),
      child: Image.asset('assets/images/fdg_logo.png',
          height: 30, color: Colors.white), // Image.asset
    ), // Padding
```

Added in file pubspec.yaml:

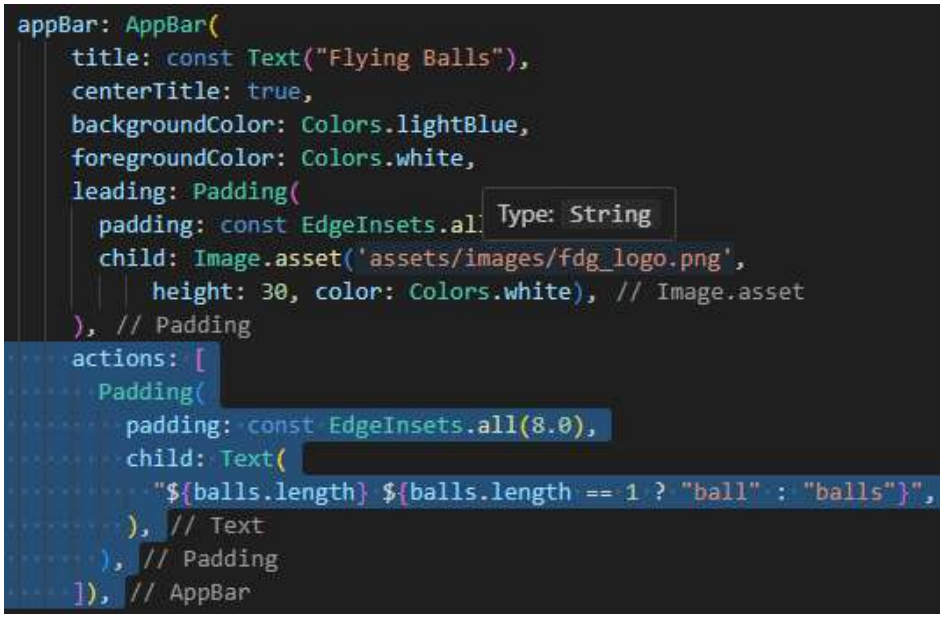
```
flutter:
  uses-material-design: true

  assets:
    - assets/images/
```

FDG logo was downloaded from link
https://fdg-ab.de/wp-content/uploads/2021/03/logo_fdg_neu_freigestellt.png

Then the file was renamed to “fdg_logo.png” and finally it was stored in the new generated project folder “assets/images”.

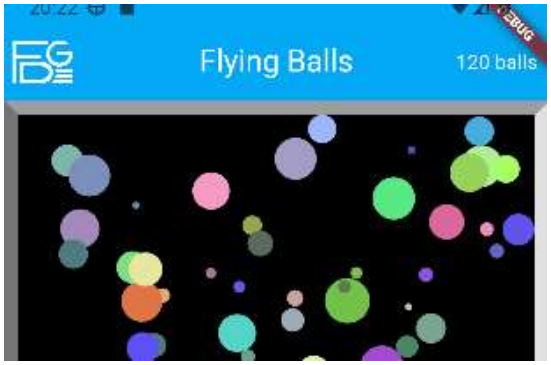
For a reminder how to work with images in Flutter see slides 25 – 27 in our old “04 Create a HelloWorld flutter app and use some basic UI elements.pdf” on https://github.com/GuentherSchmitt/fdg_flutter_2023/tree/main/docs.

Show the number of balls in the AppBar



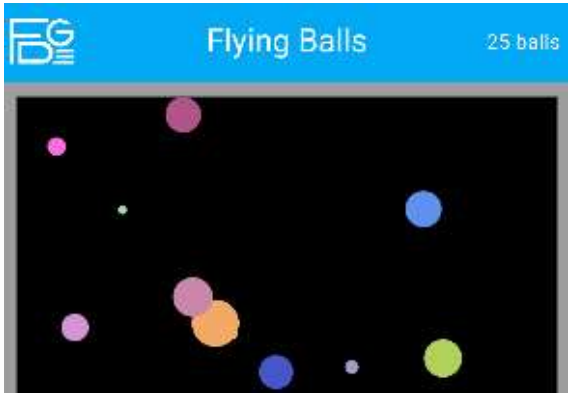
```
appBar: AppBar(
    title: const Text("Flying Balls"),
    centerTitle: true,
    backgroundColor: Colors.lightBlue,
    foregroundColor: Colors.white,
    leading: Padding(
      padding: const EdgeInsets.al
      child: Image.asset('assets/images/fdg_logo.png',
          height: 30, color: Colors.white), // Image.asset
    ), // Padding
    actions: [
      Padding(
        padding: const EdgeInsets.all(8.0),
        child: Text(
          "${balls.length} ${balls.length == 1 ? "ball" : "balls"}",
        ), // Text
      ), // Padding
    ]), // AppBar
```

Type: String



Do not show the debug banner in the app bar

```dart
class MainApp extends StatelessWidget {
  const MainApp({super.key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: MainPage()); // MaterialApp
  }
}
```



Hint found e.g. on https://quickcoder.org/remove-flutter-debug-banner/

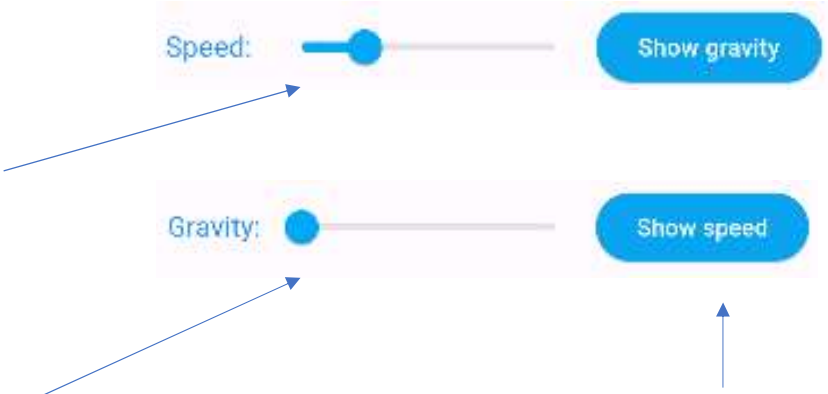
Modify the visibility of widgets

```
IconButton(
    style: elevatedButtonStyle,
    onPressed: () {
      setState(() {
        balls.add(Ball());
      });
    },
    icon: const Icon(Icons.add)), // IconButton
if (expertMode)
  ElevatedButton(
      style: elevatedButtonStyle,
      onPressed: () {
        setState(() {
          for (var i = 0; i < 10; i++) {
            balls.add(Ball());
          }
        });
      },
      child: const Text(" Add 10 balls")), // ElevatedButton
if (!expertMode)
  ElevatedButton(
      style: elevatedButtonStyle,
      onPressed: () {
        setState(() {
          expertMode = true;
        });
      },
      child: const Text("Expert-Mode")), // ElevatedButton
```

Modify speed or gravity in Expert Mode

```
Text(showGravity ? "Gravity:" : "Speed: ",
    style:
        const TextStyle(color: Colors.blue, fontSize: 16)),
if (!showGravity)
  Expanded(
    child: Slider(
      activeColor: Colors.lightBlue,
      min: 0,
      max: 4,
      value: Ball.speedFactor,
      onChanged: (value) {
        setState(() {
          Ball.speedFactor = value;
        });
      },
    ), // Slider
  ), // Expanded
if (showGravity)
  Expanded(
    child: Slider(
      activeColor: Colors.lightBlue,
      min: 0,
      max: 0.7,
      value: Ball.yAcceleration,
      onChanged: (value) {
        setState(() {
          Ball.yAcceleration = value;
        });
      },
    ), // Slider
  ), // Expanded
```

```
speedY = speedY + yAcceleration;
topYellow += speedY * speedFactor;
```




```
ElevatedButton(
    style: elevatedButtonStyle,
    onPressed: () {
      setState(() {
        showGravity = !showGravity;
      });
    },
    child:
        Text(showGravity ? "Show speed" : "Show gravity"))
```

Some “physics” around speed and gravity

Movement with constant speed:

$$s(t) = s_0 + v*t$$

Velocity with constant acceleration:

$$v(t) = v_0 + a*t$$

Beautify the border of the “black box”

```
child: Container(
    margin:
        EdgeInsets.fromLTRB(0, 0, rightMargin, bottomMargin),
    decoration: BoxDecoration(
        color: Colors.black,
        border: Border(
            top: BorderSide(
                color: Colors.grey.shade500,
                width: boxBorderWidth), // BorderSide
            left: BorderSide(
                color: Colors.grey.shade600,
                width: boxBorderWidth), // BorderSide
            right: BorderSide(
                color: Colors.grey.shade300,
                width: boxBorderWidth), // BorderSide
            bottom: BorderSide(
                color: Colors.grey.shade400,
                width: boxBorderWidth))), // BorderSide // Borc
    child: Stack(key: stackKey, children: getBallWidgets()))),
```

Possible next step

We can have a look, how to make these “Flying Balls” available as Web App on GitHub.

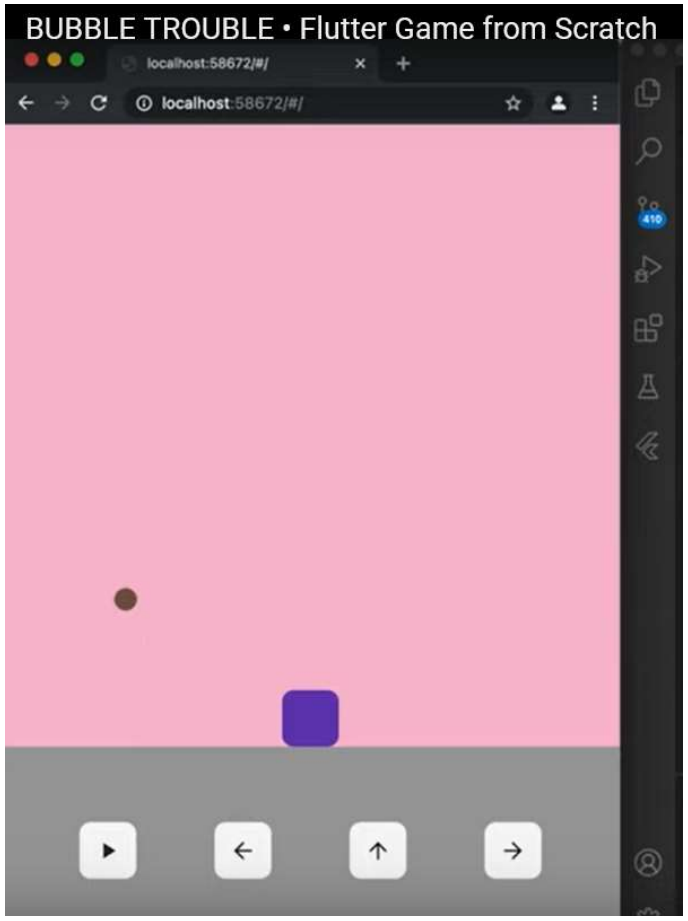
To do so we would create a new user „**fdg2024**“ on GitHub.

Then we would build the web version of „Flying Balls“ and upload it to a public repository of fdg2024 on GitHub.

After that everybody would be able to start “Flying Balls” in his Web-Browser.

Günther tested that with the account “**fdg2023**” and you can find the result under https://fdg2023.github.io/web/flying_balls/

Hint: Another simple game developed in Flutter



See how it was developed from scratch
in the following video:

https://www.youtube.com/watch?v=ZBLOxhiym7k

To play a „more elaborated“ version of the game „Bubble Trouble“ online, you can use the link:

https://poki.com/de/g/bubble-trouble