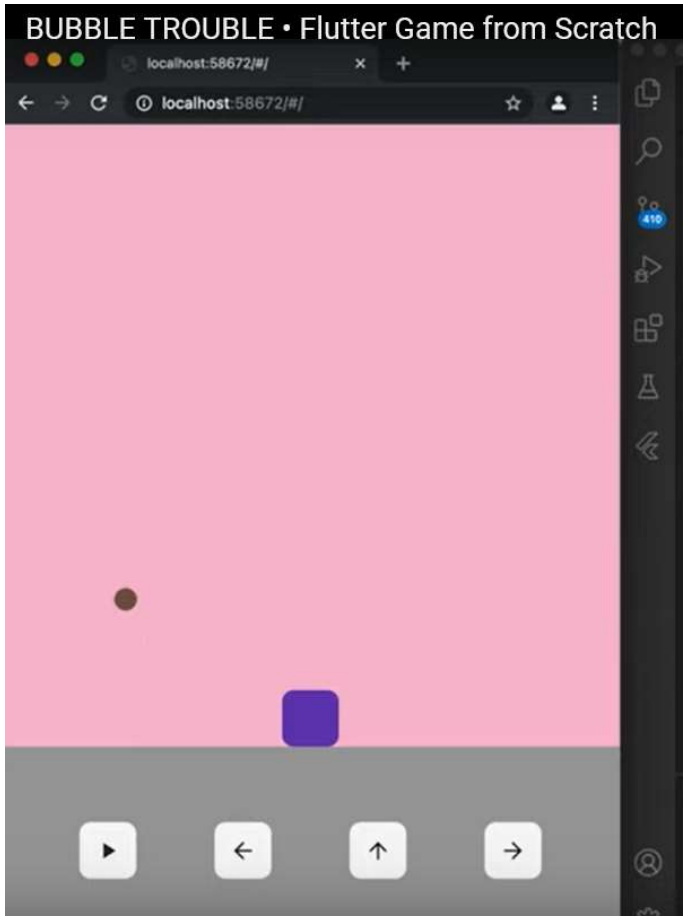
Bubble trouble: a simple game developed in Flutter



Fully featured online version:
https://poki.com/en/g/bubble-trouble

See how a version with minor features was developed from scratch in the following video:
https://www.youtube.com/watch?v=ZBLOxhiym7k

This is part of a complete series of flutter games developed by Mitch Koko, see
https://www.youtube.com/playlist?list=PLlvRDpXh1Se6kipeBLiF1xByAEmxYie6J

Unfortunately I did not find the sources which Mitch Koko used in his video on Bubble Trouble, but I found related sources on GitHub:

Searching in GitHub for “bubble trouble flutter”

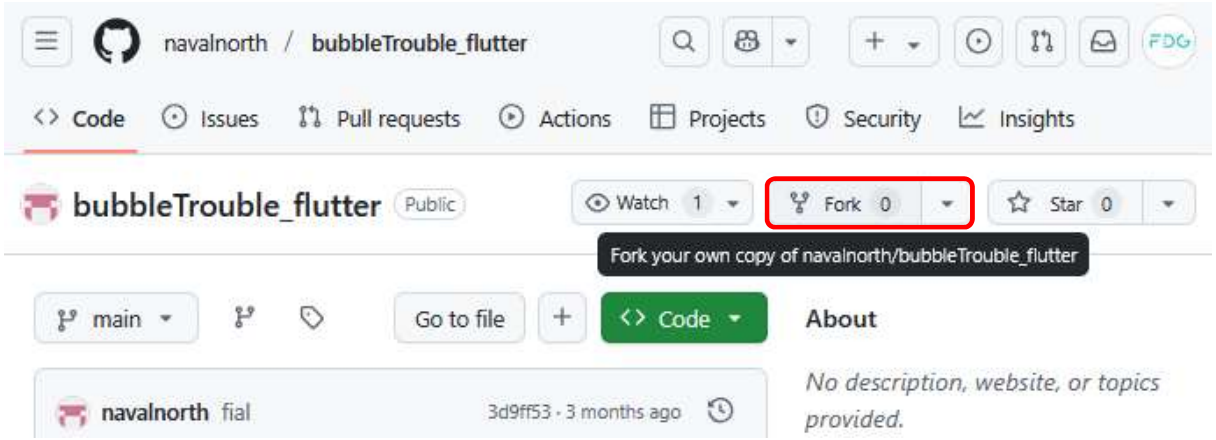
GitHub link: https://github.com/irinavasilescu/bubble_trouble

GitHub link: https://github.com/navalnorth/bubbleTrouble_flutter

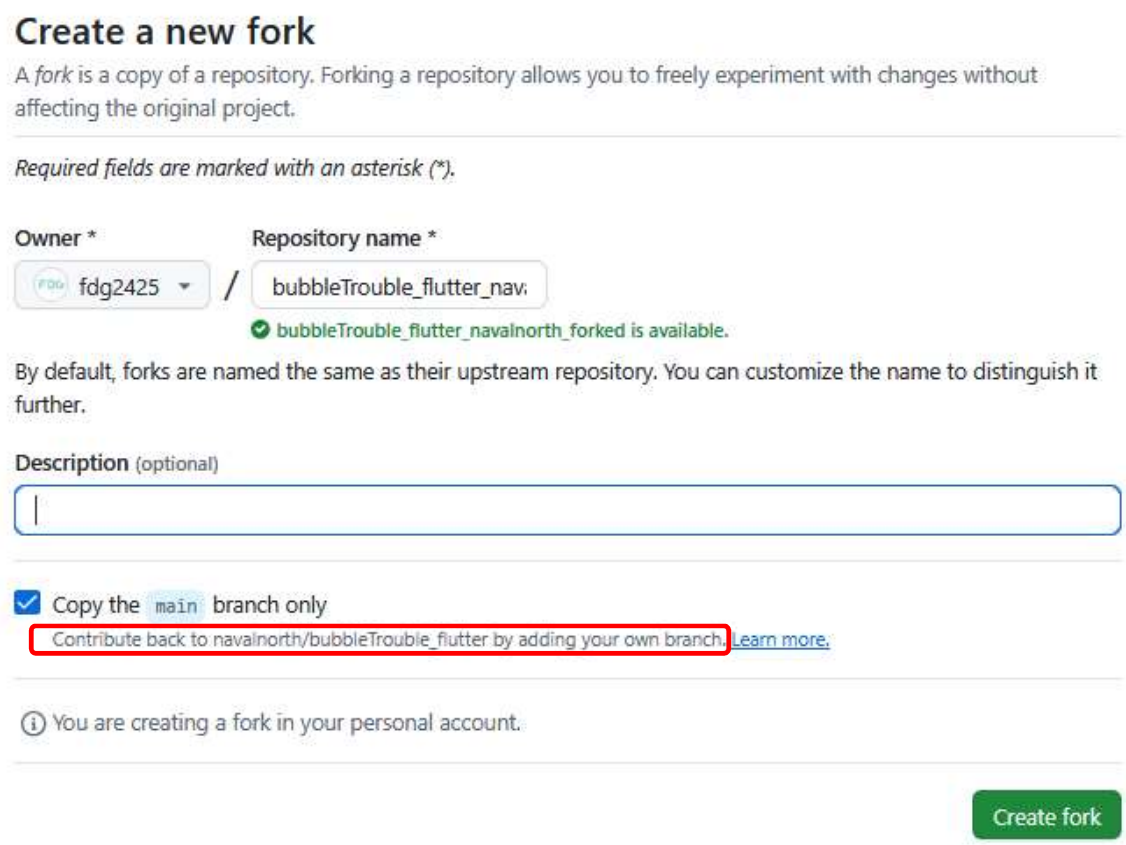
Forking a repository

When you clone a repository, where you are not invited as a collaborator, you are not allowed to push back your commits to the original GitHub repository.

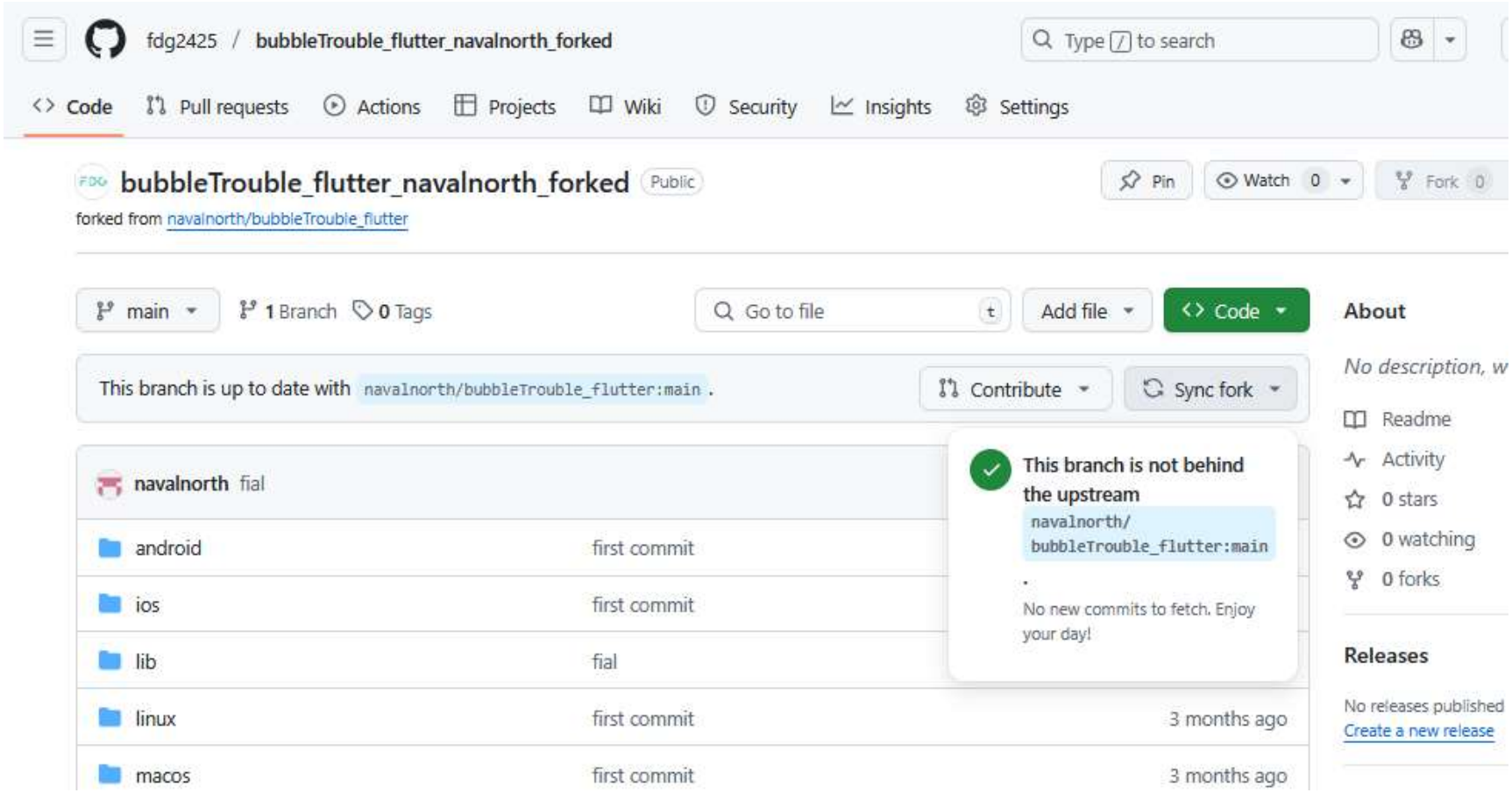
GitHub offers to fork a repository in such cases:



Forking a repository (continued)



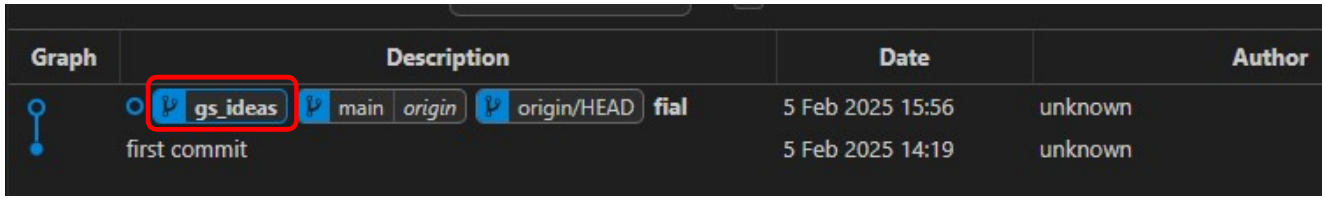
Forking a repository (continued)



Working on the forked repository

After cloning the forked repository to your PC, create a new branch.

Do not work on main branch in order to allow future updates from the original “upstream” repository.



Adapt flutter SDK version

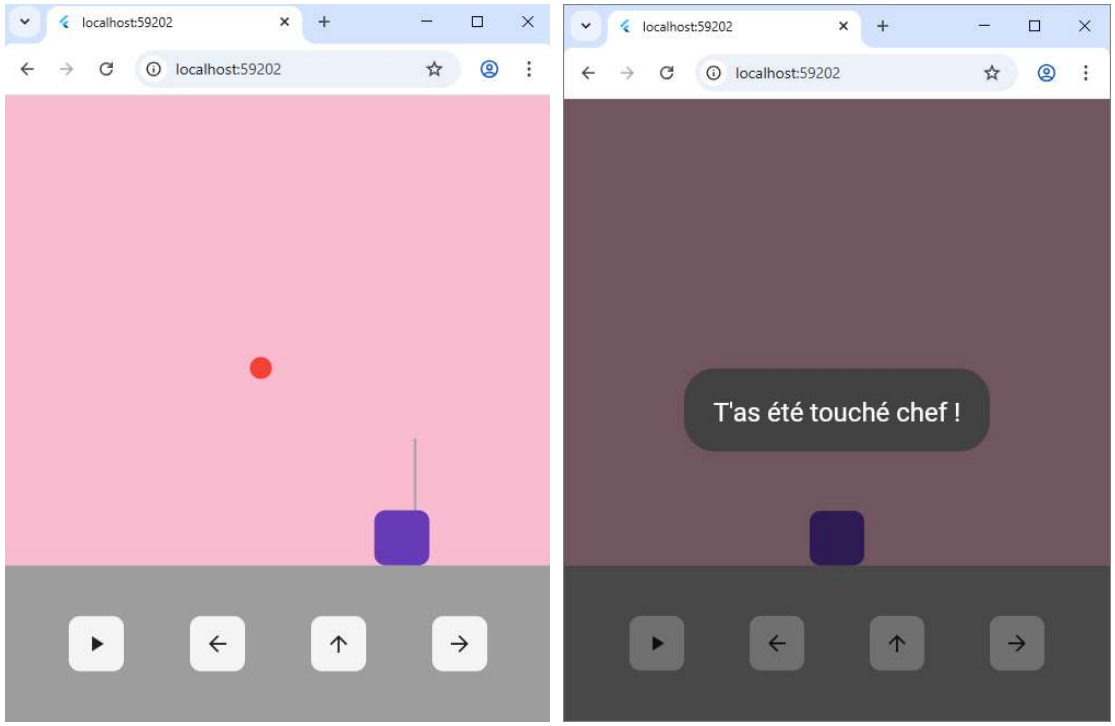
When I ran flutter pub get on the cloned repository, I got:

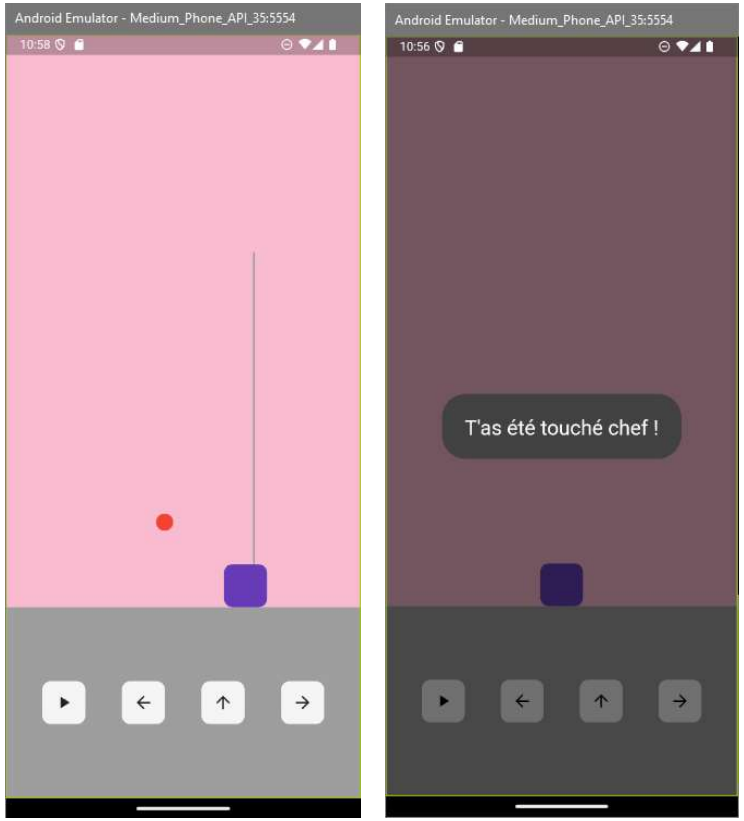
```
[bubbleTrouble_flutter_navalnorth_forked] flutter pub get --no-example
Resolving dependencies...
The current Dart SDK version is 3.5.3.

Because saute_mouton requires SDK version ^3.5.4, version solving failed.


You can try the following suggestion to make the pubspec resolve:
* Try using the Flutter SDK version: 3.29.3.
exit code 1
```

Because at that time I did not want to upgrade my Flutter version (3.24.3) to 3.29.3, I changed the required Dart SDK version in pubspec.yaml:

First experiences

After this change in pubspec.yaml, “flutter put get” executed without errors and the project could be built both for Chrome and for Android emulator:





What could be improved ?

When the game ends because the ball hit the player and when you restart the game, the ball is still behind the player, so the game directly stops again.
So before successfully restarting the game you have to move the player.
Better idea: when the game is restarted, the balls comes in from the right.

While a game is running, you can start a new game resulting in 2 or more flying balls with the same x, but different y positions.

After a ball is shot, it takes some time until the next ball appears.

Different concepts for positioning in the Stack

In flying_balls we used Positioned widget:

```
child: Stack(children: [
  for (var ball in balls)
    Positioned(
      top: ball.top,
      left: ball.left,
      child: Container(
        width: ball.diameter,
        height: ball.diameter,
        decoration: BoxDecoration(
          color: ball.color,
          shape: BoxShape.circle,
        ), // BoxDecoration
      ), // Container
    ), // Positioned
]), // Stack
```

Mitch Koko puts the “ball” inside a Container and uses this Container’s alignment property

```
class MyBall extends StatelessWidget {
  final double ballX;
  final double ballY;

  const MyBall({super.key, required this.ballX, required this.ballY});

  @override
  Widget build(BuildContext context) {
    return Container(
      alignment: Alignment(ballX, ballY),
      child: Container(
        width: 20,
        height: 20,
        decoration:
            const BoxDecoration(
              shape: BoxShape.circle,
              color: Colors.red), // BoxDecoration
      ), // Container
    ); // Container
  }
}
```

Alignment in a Container

```dart
class Alignment extends AlignmentGeometry {
  /// Creates an alignment.
  const Alignment(this.x, this.y);

  /// The distance fraction in the horizontal direction.
  ///
  /// A value of -1.0 corresponds to the leftmost edge. A value of 1.0
  /// corresponds to the rightmost edge. Values are not limited to that range;
  /// values less than -1.0 represent positions to the left of the left edge,
  /// and values greater than 1.0 represent positions to the right of the right
  /// edge.
  final double x;
```

This makes move function simpler:

Mitch Koko at 15:10

```dart
void moveRight() {
  setState(() {
    setState(() {
      if (playerX + 0.1 > 1) {
        // do nothing
      } else {
        playerX += 0.1;
      }
    });
  });
}
```

Irina

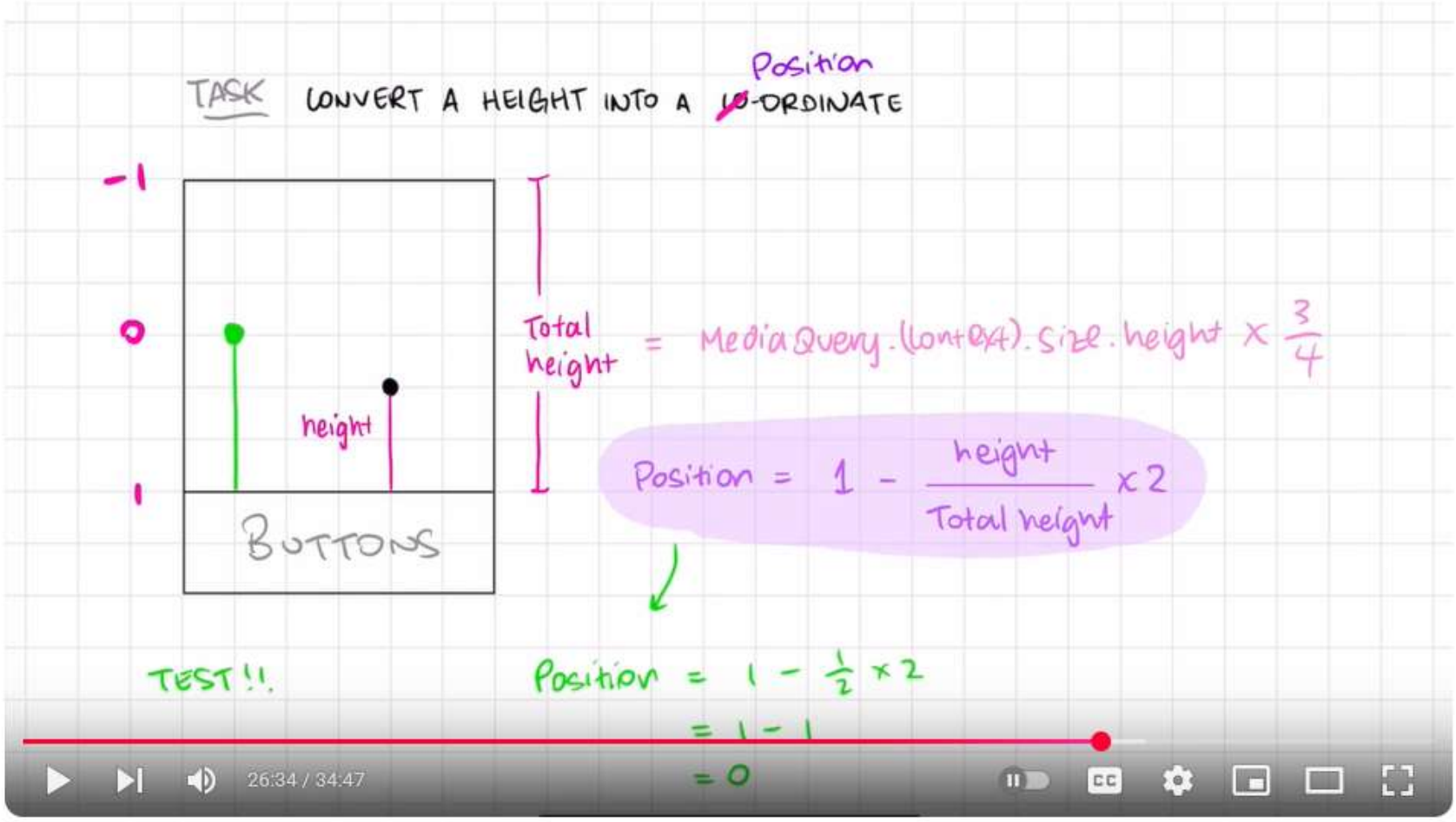
```dart
void moveRight() {
    setState(() {
        if (playerX + 0.1 > 1) {
            // do nothing
        } else {
            playerX += 0.1;
        }
```

navalnorth

```dart
void moveRight() {
  setState(() {
    playerX = (playerX + 0.1).clamp(-1.0, 1.0);
```

```
num clamp(num lowerLimit, num upperLimit)
Type: num Function(num, num)
dart:core
Returns this [num] clamped to be in the range [lowerLimit]-[upperLimit].
```

Conversion from height in pixel to Alignment-Position



Movement of the ball

In flying_balls we used:

Movement with constant speed:

$$s(t_1) = s(t_0) + v*(t_1 - t_0)$$

Velocity with constant acceleration:

$$v(t_1) = v(t_0) + a*(t_1 - t_0)$$

“Physics” says this result in:

Weg-Zeit-Gesetz

Das erste Gesetz ist das **Weg-Zeit-Gesetz**. Mit diesem berechnest du wie viel Strecke bei einer gleichmäßig beschleunigten Bewegung in einer bestimmten Zeit zurückgelegt wird.

$$s = \frac{1}{2} \cdot a \cdot t^2 + v_0 \cdot t + s_0$$

This is used by Mitch (video at 31:00)

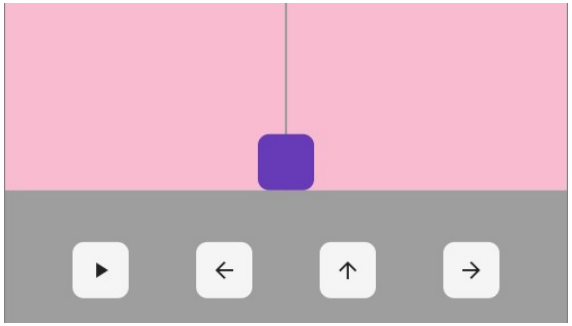
```
void startGame() {
  double time = 0;
  double height = 0;
  double velocity = 50; // how strong the jump is

  Timer.periodic(Duration(milliseconds: 10), (timer) {
    // quadratic equation that models a bounce (upside down parabola)
    height = -5 * time * time + velocity * time;

    // if the ball reaches the ground, reset the jump
    if (height < 0) {
      time = 0;
    }

    setState(() {
      ballY = heightToCoordinate(height);
    });
```

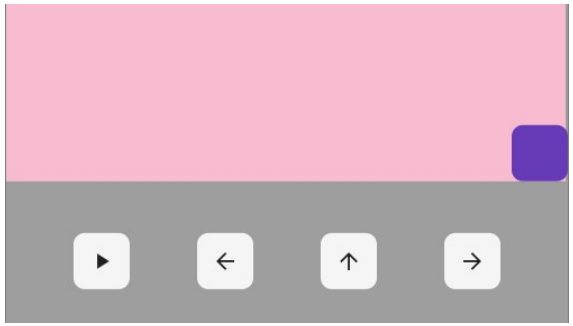
```
//Convertis la hauteur en coodonnées
double heighToCoordinate(double height) {
  double totalHeight = MediaQuery.of(context).size.height * 3 / 4;
  double position = 1 - 2 * (height / totalHeight);
  return position;
}
```

Disadvantage of working with alignment

When the player is at the left side, player and missileX are -1,
when he is on the right side, they are both 1.

This results that both the player and the missile are shown close to the side,
but that means that the missile is no longer shot from the middle of the player.





Another disadvantage or at least effect:

The speed of the ball increases when you increase the width of the browser window.

Mitch separates MyApp and MyHomePage in 2 files

In flying_balls we have in main.dart:

main.dart > ...

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flying balls title',
      theme: ThemeData( // ThemeData ···
      home: const MyHomePage(title: 'Flying balls'),
    ); // MaterialApp
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  double marginRight = 0;
  double marginBottom = 0;
```

Mitch splits this in 2 files (main.dart and homepage.dart):

main.dart > ...

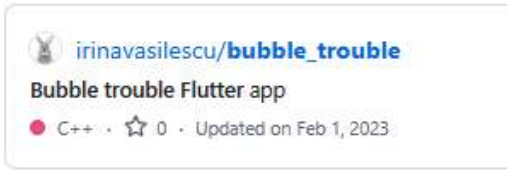
```dart
Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

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

homepage.dart > ...

```dart
class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

enum Direction { left, right }

class _HomePageState extends State<HomePage> {
  //variables joueur
  static double playerX = 0;
  final FocusNode _focusNode = FocusNode();

  //variables missiles
  double missileX = playerX;
  double missileHeight = 10;
  bool midshoot = false;
```

This follows the principle: each class has its own dart file.

“older” projects do not compile for Android



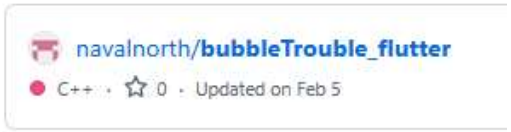
Compile error for Android:

Your project's Gradle version is incompatible with the Java version that Flutter is using

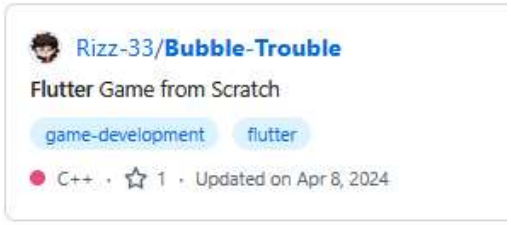
Compile error for Android:

Your project's Gradle version is incompatible with the Java version that Flutter is using

Does not support web



After resetting Dart SDK version from 3.5.4 to 3.5.3
it compiled both for Chrome and Android.



Compile error for Android:

Your project's Gradle version is incompatible with the Java version that Flutter is using

Adapt project’s gradle version

After cloning and building most of these projects for Android, I got:

```
BUILD FAILED in 829ms

┌─ Flutter Fix ──────────────────────────────────────────────────────────────┐
│ [!] Your project's Gradle version is incompatible with the Java version that Flutter is using │
│ for Gradle.                                                                │
```

This error does not occur when building for Chrome !

how to fix Your project's Gradle version is incompatible with the Java version that Flutter is using

Ah, that "incompatible Gradle version" error with Flutter and Java can be a bit of a headache! It usually pops up when the version of Gradle your Android project is using doesn't play nicely with the Java Development Kit (JDK) that Flutter is configured to use. Here's a breakdown of how to tackle this, covering the most common scenarios and solutions:

Solution 1: Update Gradle Version (Recommended in most cases)

This is often the easiest and most recommended approach, as it allows you to use a more recent Gradle version that is likely compatible with a wider range of Java versions.

- Edit `gradle-wrapper.properties`:
  - Open the `android/gradle/wrapper/gradle-wrapper.properties` file.
  - Modify the `distributionUrl` line to point to a newer Gradle version. You can find a list of Gradle releases and their required Java compatibility on the Gradle Releases page.

Adapt project’s gradle version

For our “Irina-fork” we have:

android > gradle > wrapper > gradle-wrapper.properties

```
1  distributionBase=GRADLE_USER_HOME
2  distributionPath=wrapper/dists
3  zipStoreBase=GRADLE_USER_HOME
4  zipStorePath=wrapper/dists
5  distributionUrl=https\://services.gradle.org/distributions/gradle-7.5-all.zip
6
```

In our flying_balls app we have:

android > gradle > wrapper > gradle-wrapper.properties

```
1  distributionBase=GRADLE_USER_HOME
2  distributionPath=wrapper/dists
3  zipStoreBase=GRADLE_USER_HOME
4  zipStorePath=wrapper/dists
5  distributionUrl=https\://services.gradle.org/distributions/gradle-8.3-all.zip
6
```

I changed this in the “Irina-fork”, but then I got other build errors, e.g.

```
A problem occurred evaluating root project 'android'.
> A problem occurred configuring project ':app'.
   > Could not create task ':app:copyFlutterAssetsDebug'.
      > Could not create task ':app:mergeDebugAssets'.
```

Make “older” projects available for Android

Open another VS Code and create a new project in some temporary directory with the name “bubble_trouble”.

Copy the folder android from this “new” project into the “old” project.

In Terminal of the “old” project, run “flutter clean” and “flutter pub get”

Now you should be able to build and run the “old” project on Android:

Finally you can delete the “new” project.