Text

Description automatically generated

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **Assignment No** | Assignment No 08 |

Assignment Number - 08

**Title :** **Flutter animation by using AnimationController**

**Theory :**

## **Ticker providers**

An [AnimationController](https://api.flutter.dev/flutter/animation/AnimationController-class.html) needs a [TickerProvider](https://api.flutter.dev/flutter/scheduler/TickerProvider-class.html), which is configured using the vsync argument on the constructor. The [TickerProvider](https://api.flutter.dev/flutter/scheduler/TickerProvider-class.html) interface describes a factory for [Ticker](https://api.flutter.dev/flutter/scheduler/Ticker-class.html) objects. A [Ticker](https://api.flutter.dev/flutter/scheduler/Ticker-class.html) is an object that knows how to register itself with the [SchedulerBinding](https://api.flutter.dev/flutter/scheduler/SchedulerBinding-mixin.html) and fires a callback every frame. The [AnimationController](https://api.flutter.dev/flutter/animation/AnimationController-class.html) class uses a [Ticker](https://api.flutter.dev/flutter/scheduler/Ticker-class.html) to step through the animation that it controls.

If an [AnimationController](https://api.flutter.dev/flutter/animation/AnimationController-class.html) is being created from a [State](https://api.flutter.dev/flutter/widgets/State-class.html), then the State can use the [TickerProviderStateMixin](https://api.flutter.dev/flutter/widgets/TickerProviderStateMixin-mixin.html) and [SingleTickerProviderStateMixin](https://api.flutter.dev/flutter/widgets/SingleTickerProviderStateMixin-mixin.html) classes to implement the [TickerProvider](https://api.flutter.dev/flutter/scheduler/TickerProvider-class.html) interface. The [TickerProviderStateMixin](https://api.flutter.dev/flutter/widgets/TickerProviderStateMixin-mixin.html) class always works for this purpose; the [SingleTickerProviderStateMixin](https://api.flutter.dev/flutter/widgets/SingleTickerProviderStateMixin-mixin.html) is slightly more efficient in the case of the class only ever needing one [Ticker](https://api.flutter.dev/flutter/scheduler/Ticker-class.html) (e.g. if the class creates only a single [AnimationController](https://api.flutter.dev/flutter/animation/AnimationController-class.html) during its entire lifetime).

**Late Keyword**

In Dart, we use the **late keyword** to declare variables that will be initialized later. These are called non-nullable variables as they are initialized after the declaration. Hence, we use the late keyword.

**Vsync**

It is the TickerProvider for the current context. It can be changed by calling resync. It is required and must not be null.

**AnimationController**

Every Flutter animation needs at least two elements to be created:

* A Tween to get generate values for the animation
* An AnimationController as parent

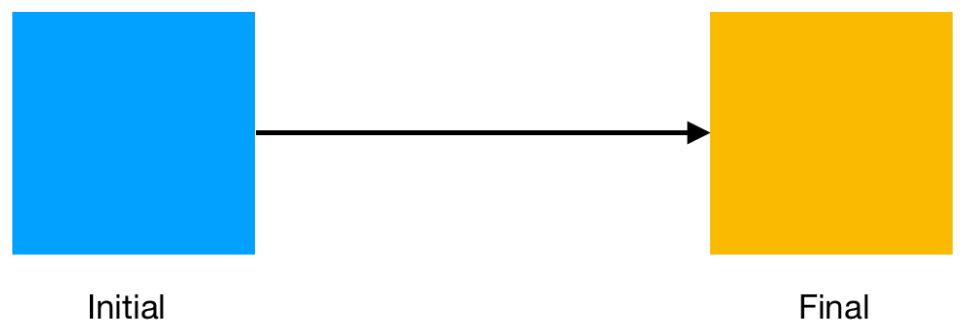
An [AnimationController](https://api.flutter.dev/flutter/animation/AnimationController-class.html) should be [dispose](https://api.flutter.dev/flutter/animation/AnimationController/dispose.html)d when it is no longer needed. This reduces the likelihood of leaks. When used with a [StatefulWidget](https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html), it is common for an [AnimationController](https://api.flutter.dev/flutter/animation/AnimationController-class.html) to be created in the [State.initState](https://api.flutter.dev/flutter/widgets/State/initState.html) method and then disposed in the [State.dispose](https://api.flutter.dev/flutter/widgets/State/dispose.html) method.

[CurvedAnimation](https://api.flutter.dev/flutter/animation/CurvedAnimation-class.html) is useful when you want to apply a non-linear [Curve](https://api.flutter.dev/flutter/animation/Curve-class.html) to an animation object, especially if you want different curves when the animation is going forward vs when it is going backward.

Depending on the given curve, the output of the [CurvedAnimation](https://api.flutter.dev/flutter/animation/CurvedAnimation-class.html) could have a wider range than its input. For example, elastic curves such as [Curves.elasticIn](https://api.flutter.dev/flutter/animation/Curves/elasticIn-constant.html) will significantly overshoot or undershoot the default range of 0.0 to 1.0.

**Twin Animation**

Tweening stands for inbetweening. To understand what that means, look at the image below:



Tween gives us intermediate values between two values like colors, integers, alignments and almost anything you can think of. In Flutter, you can define the tween as Tween<T> or use predefined classes like ColorTween which are specifically designed for this.

**Opacity class**

A widget that makes its child partially transparent.

This class paints its child into an intermediate buffer and then blends the child back into the scene partially transparent.

For values of opacity other than 0.0 and 1.0, this class is relatively expensive because it requires painting the child into an intermediate buffer. For the value 0.0, the child is not painted at all. For the value 1.0, the child is painted immediately without an intermediate buffer.

**Source Code:-**

import 'package:flutter/material.dart';

/// Flutter code sample for [FadeTransition].

void main() => runApp(const FadeTransitionExampleApp());

class FadeTransitionExampleApp extends StatelessWidget {

const FadeTransitionExampleApp({super.key});

@override

Widget build(BuildContext context) {

return const MaterialApp(

home: FadeTransitionExample(),

);

}

}

class FadeTransitionExample extends StatefulWidget {

const FadeTransitionExample({super.key});

@override

State<FadeTransitionExample> createState() => \_FadeTransitionExampleState();

}

/// [AnimationController]s can be created with `vsync: this` because of

/// [TickerProviderStateMixin].

class \_FadeTransitionExampleState extends State<FadeTransitionExample>

with TickerProviderStateMixin {

late final AnimationController \_controller = AnimationController(

duration: const Duration(seconds: 1),

vsync: this,

)..repeat(reverse: true);

late final Animation<double> \_animation = CurvedAnimation(

parent: \_controller,

curve: Curves.easeIn,

);

@override

void dispose() {

\_controller.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return ColoredBox(

color: Colors.white,

child: FadeTransition(

opacity: \_animation,

child: Image.network('https://media.tenor.com/\_bTaLmoLSc4AAAAS/troll-pilled.gif',

height: 100,

width: 100,

fit: BoxFit.fitWidth,

),

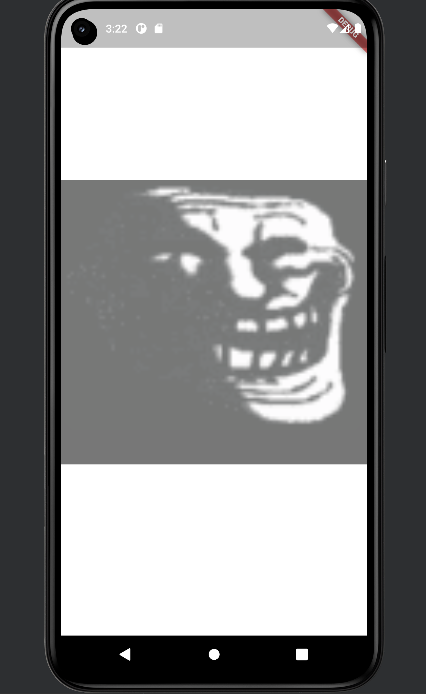
),

);

}

}

**Output:**

****

**Conclusion : in this assignment I have learn about ticker provider twin animation .**