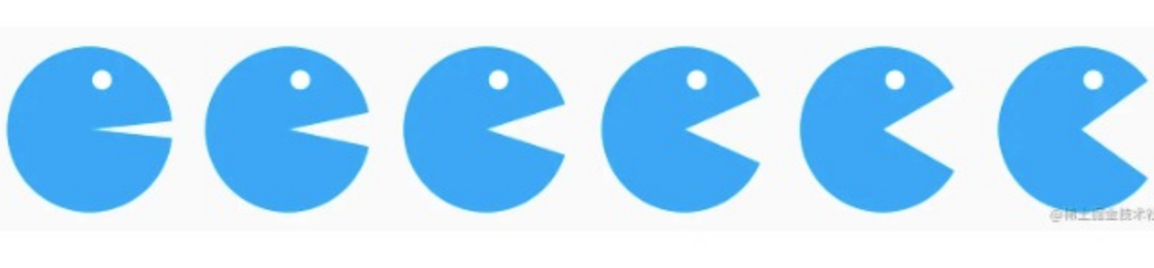
本节目标:

```dart
[1]. 练习绘制静态效果。
[2]. 了解如何在画布中 [使用动画]。
[3]. 了解 [Animation] 和 [Animatable] 对象的作用。
[4]. 学会使用动画器完成多种属性的过渡效果及 [自定义过渡器]。
```

一、绘制静态效果

> 在进入动画之前，我们先准备好进行动画的素材。经过我深思熟虑之后，决定用 吃豆人 的样子进行动画。一者 绘制比较简单 、二者 比较有趣 。我们可以让豆人的嘴巴开合，也可以让豆人有颜色渐变的动画。可以让它有移动动画、缩放动画、眼睛动一动什么的。



1.创建 PicMan 组件

> 这里使用 `StatefulWidget` 是为了之后的动画做准备。传入 `angle` 属性用于控制开口大小。

```dart
---->[p10_anim/s01_static/pic_man.dart]----
class PicMan extends StatefulWidget {
  final Color color;
  final double angle;
  PicMan({Key? key, this.color = Colors.lightBlue, this.angle = 30 }) : super(key: key);

  @override
  _PicManState createState() => _PicManState();
}

class _PicManState extends State<PicMan> with SingleTickerProviderStateMixin {

  @override
  Widget build(BuildContext context) {
    return CustomPaint(
      size: Size(100, 100),
      painter: PicManPainter(color: widget.color, angle : widget.angle), // 背景
    );
  }
}
```

---

2. 创建画布

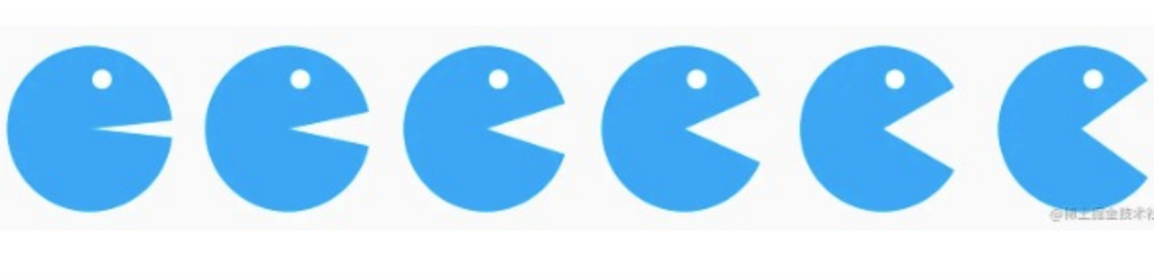
> 通过 `PicManPainter` 进行绘制，在构造函数中传入 角度 和 颜色 。

```dart
class PicManPainter extends CustomPainter {
  final Color color; // 颜色
  final double angle; // 角度(与x轴交角 角度制)

  Paint _paint = Paint();

  PicManPainter({this.color = Colors.yellowAccent, this.angle = 30});

  @override
  void paint(Canvas canvas, Size size) {
    canvas.clipRect(Offset.zero & size); //剪切画布
    final double radius = size.width / 2;
    canvas.translate(radius, size.height / 2);

    _drawHead(canvas, size);
    _drawEye(canvas, radius);
  }

  //绘制头
  void _drawHead(Canvas canvas, Size size) {
    var rect = Rect.fromCenter(
        center: Offset(0, 0), height: size.width, width: size.height);
    var a = angle / 180 * pi;
    canvas.drawArc(rect, a, 2 * pi - a.abs() * 2, true, _paint..color = color);
  }

  //绘制眼睛
  void _drawEye(Canvas canvas, double radius) {
    canvas.drawCircle(Offset(radius * 0.15, -radius * 0.6), radius * 0.12,
        _paint..color = Colors.white);
  }

  @override
  bool shouldRepaint(covariant PicManPainter oldDelegate) =>
      oldDelegate.color != color || oldDelegate.angle != angle;
}
```

---

3. 使用 PicMan 组件

> 使用 `Wrap 组件` 包裹六个 PicMan 组件，开口大小依次增加，效果如下。



```dart
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        title: 'Flutter Demo',
        debugShowCheckedModeBanner: false,
        theme: ThemeData(
          primarySwatch: Colors.blue,
        ),
        home: Scaffold(
            body: Padding(
          padding: const EdgeInsets.only(top: 58.0, left: 20),
          child: Center(
            child: Wrap(spacing: 20, runSpacing: 20, children: buildChildren()),
          ),
        )));
  }

  List<Widget> buildChildren() =>
      List<Widget>.generate(6,
      (index) => PicMan(
            color: Colors.lightBlue,
            angle: (1 + index) * 6.0, // 背景
          ));
}
```

二、为画布添加动画

> 动画的本质就是在 数据不断变化 时，让画布 重绘 ，达到连续的视觉效果。这小节实现如下效果：

---

1. 创建 AnimationController 对象

> AnimationController 的构造器中需要传入 `TickerProvider` 对象
> 可以将 State 对象混入 `SingleTickerProviderStateMixin` 来成为该对象
> `lowerBound` 是运动的下限， `upperBound` 是运动的上限， `duration` 是运动时长。
> 下面的 _controller 会在两秒之内将数值从 10 连续变化到 40。

```dart
---->[p10_anim/s02_animate/pic_man.dart]----
class _PicManState extends State<PicMan> with SingleTickerProviderStateMixin {
  late AnimationController _controller; // 动画控制器

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      lowerBound: 10,
      upperBound: 40,
      duration: const Duration(seconds: 1),
      vsync: this,
    )..repeat(reverse: true);
  }
```

---

2. 在画板中使用动画控制器

> `CustomPainter` 中有一个 `_repaint` 的 `Listenable` 对象。当监听到这个对象的变化时，画板会触发重绘，这是触发重绘的 最高效的方式 。

```dart
class PicManPainter extends CustomPainter {

  final Animation<double> angle; // <--- 定义成员变量
  final Color color;
  Paint _paint = Paint();

  PicManPainter({this.color = Colors.yellowAccent, required this.angle})
      : super(repaint: angle); // <--- 传入 Listenable 可监听对象

  // 其他同上,略...

void _drawHead(Canvas canvas, Size size) {
  var rect = Rect.fromCenter(
      center: Offset(0, 0), height: size.width, width: size.height);
  var a = angle.value / 180 * pi; // <---使用动画器的值
  canvas.drawArc(rect, a, 2 * pi - a.abs() * 2, true, _paint..color = color);
}
```

三、动画使用介绍

1. `Animation<T>` 和 `Animatable<T>`

> 了解 Flutter 中的动画，首先要认清 `Animation` 和 `Animatable` 这两个类
> `Animation` 是继承自 `Listenable` ，它是将两个量在一段时间内变化的 根源 。

```dart
Animation<T>
        |--- AnimationController extends Animation<double>
  |--- CurvedAnimation extends Animation<double>
  |--- ReverseAnimation  extends Animation<double>
  |--- ...
```

> `Animatable` 本身并 不能 进行动画，但它可以将 `Animation` 对象进行加强。
> 通过 `Tween` 的子类可以对很多联系的数据进行动画变换，你也可以自定义自己Tween
> 。
> `CurveTween` 可以是跳转动画的速率变化， `TweenSequence` 可以执行一个动画序列。

```dart
Animatable<T>
        |--- TweenSequence<T> extends Animatable<T>
  |--- CurveTween extends Animatable<double>
        |--- Tween<T extends dynamic> extends Animatable<T>
        |--- IntTween extends Tween<int>
        |--- ColorTween extends Tween<Color?>
        |--- RectTween extends Tween<Rect?>
        |--- StepTween extends Tween<int>
        |--- AlignmentTween extends Tween<Alignment>
        |--- Matrix4Tween extends Tween<Matrix4>
        |--- TextStyleTween extends Tween<TextStyle>
        |--- BoxConstraintsTween extends Tween<BoxConstraints>
        |--- ....
```

---

2. Tween的使用

> 一般 AnimationController 并不设置上下界，否则在使用 Tween 时是在已定的上下界之上变化的。
> AnimationController 只负责 让数字在 `0~1` 间变化，其余的都交给 `Animatable` 来完成。
> 通过 `Animatable#animate(Animation)` 可以返回 `Animation` 对象。 通过 `Animation#drive(Animatable)` 也可以返回 `Animation` 对象，两者的作用是一致的。 下面通过 Tween 和 ColorTween 实现如下角度和颜色一起渐变的动画。

```dart
---->[p10_anim/s03_animate_tween1/pic_man.dart]----
class _PicManState extends State<PicMan> with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<Color?> _colorCtrl; // 声明颜色控制器
  late Animation<double> _angleCtrl; // 声明角度控制器

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(seconds: 1),
      vsync: this,
    );
    // 使用 double 范围 [10,40] 的 Tween 创建动画器
    _angleCtrl = _controller.drive(Tween(begin: 10,end: 40));
    // 使用 color 范围 [Colors.blue,Colors.red] 的 ColorTween 创建动画器
    _colorCtrl = ColorTween(begin: Colors.blue, end: Colors.red)
        .animate(_controller);
        // 重复执行动画
    _controller.repeat(reverse: true);
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return CustomPaint(
      size: Size(100, 100),
      painter: PicManPainter(
          color: _colorCtrl, angle: _angleCtrl, repaint: _controller),
    );
  }
}
```

> 这样使用可以将对应的变换通过构造传入 `PicManPainter` 。
> 关于多个 `Animation` 的 `dispose` ，只需要释放根源的 `AnimationController` 即可。因为它们使用的是同一个 `Ticker` ，释放资源也只是对 `Ticker` 的释放。

```dart
class PicManPainter extends CustomPainter {
  final Animation<double> repaint;
  final Animation<double> angle;
  final Animation<Color> color;
  Paint _paint = Paint();

  PicManPainter({required this.repaint, required this.color, required this.angle})
      : super(repaint: repaint);
```

> 上面的方式是将动画器提供给 `CustomPainter` 使用，你也可以只将主动画器传入。在 `CustomPainter` 中进行创建其他的构造器，这两种方式在源码中都有使用创建，感兴趣的可以自己看看： `flutter/lib/src/material/input_decorator.dart # _InputBorderPainter` `/flutter/lib/src/material/dropdown.dart # _DropdownMenuPainter`

---

3. `Tween#evaluate` 的使用

> 使用 Tween 并不是一定要生成对应的动画器，使用 `Tween#evaluate(Animation)` 可以获取传入动画动画器在当前 Tween 的值。比如传入的动画器是 `0~1` ，运动到了 0.4 。Tween 对象是 `0~10` ，那么该方法就返回 4 ，也就是直接算出 Tween 对应的值。

```dart
---->[p10_anim/s04_animate_tween2/pic_man.dart]----
class _PicManState extends State<PicMan> with SingleTickerProviderStateMixin {
  late AnimationController _controller;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(seconds: 1),
      vsync: this,
    );
    _controller.repeat(reverse: true);
  }

  @override
  Widget build(BuildContext context) {
    return CustomPaint(
      size: Size(100, 100),
      painter: PicManPainter( repaint: _controller), // 背景
    );
  }
}

class PicManPainter extends CustomPainter {
  final Animation<double> repaint;

  final ColorTween colorTween = ColorTween(begin: Colors.blue, end: Colors.red);
  final Tween<double> angleTween = Tween(begin: 10.0, end: 40.0);

  Paint _paint = Paint();

  PicManPainter({this.repaint}): super(repaint: repaint);

      // 略同...
  void _drawHead(Canvas canvas, Size size) {
    var rect = Rect.fromCenter(
        center: Offset(0, 0), height: size.width, width: size.height);
    var a = angleTween.evaluate(controller) / 180 * pi;
    canvas.drawArc(rect, a, 2 * pi - a.abs() * 2, true,
        _paint..color = colorTween.evaluate(repaint)??Colors.black);
  }

  void _drawEye(Canvas canvas, double radius) {
    canvas.drawCircle(Offset(radius * 0.15, -radius * 0.6), radius * 0.12,
        _paint..color = Colors.white);
  }
      // 略同...
}
```

> 另外其余的 XXXTween 都只是为 XXX 类型渐变 准备的，使用方式是一致的。 如果需要渐变的属性太多，当然也可以自己定义需要类型的 Tween 方便使用。

---

4. 自定义 Tween

> 只对于 Tween 主要就是实现在 `t` 变化的过程中应该根据 `t` 变成什么新值。这里写一个 `ColorDoubleTween` 示意一下自定义流程。这个 Tween 负责颜色和 double 类型 同时变化的渐变。 首先要有一个基本结构 `ColorDouble` 来定义需要变化的类型，然后继承自 `Tween<ColorDouble>` 之后重写 `lerp` 方法，根据 `t` 生成新对象即可。如果有十几个渐变的属性，这样可以更方便操作，不然要写十几个不同的 Tween 。这样的话，打包成一个就行了。

```dart
---->[p10_anim/s05_animate_tween3/color_double_tween.dart]----
class ColorDouble {
  final Color? color;
  final double value;

  ColorDouble({this.color = Colors.blue, this.value = 0});
}

class ColorDoubleTween extends Tween<ColorDouble> {
  ColorDoubleTween({required ColorDouble begin,required ColorDouble end})
      : super(begin: begin, end: end);

  @override
  ColorDouble lerp(double t) => ColorDouble(
      color: Color.lerp(begin?.color, end?.color, t),
      value: (begin!.value + (end!.value - begin!.value) * t));
}
```

> 使用方式如下：

```dart
---->[p10_anim/s05_animate_tween3/pic_man.dart]----
class PicManPainter extends CustomPainter {
  final Animation<double> repaint;
  // 创建 ColorDoubleTween
  final ColorDoubleTween tween = ColorDoubleTween(
      begin: ColorDouble(color: Colors.blue, value: 10),
      end: ColorDouble(color:Colors.red, value:40));

  Paint _paint = Paint();

  PicManPainter(this.repaint) : super(repaint: repaint);

  @override
  void paint(Canvas canvas, Size size) {
    canvas.clipRect(Offset.zero & size); //剪切画布
    final double radius = size.width / 2;
    canvas.translate(radius, size.height / 2);
    _drawHead(canvas, size);
    _drawEye(canvas, radius);
  }

  void _drawHead(Canvas canvas, Size size) {
    var rect = Rect.fromCenter(
        center: Offset(0, 0), height: size.width, width: size.height);
    var a = tween.evaluate(repaint).value / 180 * pi;
    canvas.drawArc(rect, a, 2 * pi - a.abs() * 2, true,
        _paint..color = tween.evaluate(repaint).color??Colors.black);
  }

  void _drawEye(Canvas canvas, double radius) {
    canvas.drawCircle(Offset(radius * 0.15, -radius * 0.6), radius * 0.12,
        _paint..color = Colors.white);
  }

  @override
  bool shouldRepaint(covariant PicManPainter oldDelegate) =>
      oldDelegate.repaint != repaint;
}
```

> 这样动画的使用方法就基本介绍完了，下面将介绍一下 动画的变化速率 和 动画控制器中 的其他方法。

全部评论（13）

风二中 LV.1 JY.4 | Android&Flutter | 2月前
打卡

点赞 回复

衿璃 JY.4 | flutter | 3月前
😩转了一圈又回来了，立志【看完】（划掉）看懂 张老师所有的博客
点赞 1

衿璃 | 2月前
🐱，摸鱼一个月，接着学
点赞 回复

IAM17 LV.4 JY.5 | Flutter @ 前端 | 9月前
CustomPainter 中有一个 _repaint 的 Listenable 对象。这个对象发出通知时，在 shouldRepaint 允许时会触发重绘。这句的意思是shouldRepaint 方法必须返回 true才会重绘？实际上shouldRepaint 方法返回 false 也会重绘。
点赞 2

张风捷特烈（作者） | 9月前
已修正：_repaint 方式，必定触发重绘。shouldRepaint 是用于外界对画板更新时，是否重绘。在这篇文章有详细探索：[Flutter 绘制探索 3 | 深入分析 CustomPainter 类] juejin.cn
点赞 回复

用户56787... 回复 张风捷特烈 | 16小时前
我也正有此疑问，重复本章例程时，我将 shouldRepaint 固定返回 false，也能实现动画
"已修正：_repaint 方式，必定触发重绘。shouldRepaint 是用于外界对画板更新时... "
点赞 回复

卡卡休 JY.1 | 1年前
创建画布 shouldRepaint方式里的参数写错了 不是PicMan
点赞 1

张风捷特烈（作者） | 1年前
感谢提醒，已处理 😉
点赞 回复

旺仔139 JY.1 | 2年前
66
点赞 回复

伞菌 LV.2 JY.1 | Flutter | 2年前
虽然无关紧要，但吃豆人的英文是Pacman吧😂
2 回复

elilai JY.4 | FE | 2年前
begin: ColorDouble(color: Colors.blue, value: 10),
end: ColorDouble(color:Colors.red, value:10));
两个value都是10，嘴不会动了~
点赞 1

张风捷特烈（作者） | 2年前
已修改
点赞 回复

煮一壶月光下酒 JY.1 | 酒徒 | 2年前
mark
点赞 回复