

circular_prograss.dart

소스 코드

전체 코드

```
1  import 'dart:math' as math;
2  import 'package:flutter/material.dart';
3
4  class RotatingImageCircularProgressBar extends StatefulWidget {
5    final double progress; // 0.0 ~ 1.0
6    final double size;
7    final String imagePath; // 회전할 이미지 경로
8
9    final Duration progressAnimationDuration;
10   final Duration rotationAnimationDuration; // 이미지 회전 애니메이션 한 바퀴 도는 시간
11
12   const RotatingImageCircularProgressBar({
13     Key? key,
14     required this.progress,
15     required this.imagePath,
16     this.size = 150.0,
17     this.progressAnimationDuration = const Duration(milliseconds: 300),
18     this.rotationAnimationDuration = const Duration(seconds: 2),
19   }) : super(key: key);
20
21   @override
22   _RotatingImageCircularProgressBarState createState() =>
23   _RotatingImageCircularProgressBarState();
24
25   class _RotatingImageCircularProgressBarState extends
26   State<RotatingImageCircularProgressBar>
27   with SingleTickerProviderStateMixin { // 이미지 회전 애니메이션을 위해 TickerProvider
28     필요
29     late AnimationController _rotationController;
30
31     @override
32     void initState() {
33       super.initState();
34       _rotationController = AnimationController(
35         duration: widget.rotationAnimationDuration,
36         vsync: this,
37       )..repeat(); // 애니메이션 시작 및 반복
38     }
39
40     @override
41     void dispose() {
42       _rotationController.dispose();
43       super.dispose();
44     }
45   }
```

```

43
44 @override
45 widget build(BuildContext context) {
46     final double imageActualSize = widget.size * 1;
47
48     return SizedBox(
49         width: widget.size,
50         height: widget.size,
51         child: TweenAnimationBuilder<double>(
52             tween: Tween<double>(begin: 0.0, end: widget.progress.clamp(0.0, 1.0)),
53             duration: widget.progressAnimationDuration,
54             builder: (context, animatedProgress, _) {
55                 return Stack(
56                     alignment: Alignment.center,
57                     children: [
58                         RotationTransition(
59                             turns: _rotationController, // AnimationController 사용
60                             child: Image.asset(
61                                 widget.imagePath,
62                                 width: imageActualSize,
63                                 height: imageActualSize,
64                                 fit: BoxFit.contain, // 이미지 비율 유지하며 채움
65                                 // 이미지 로딩 중 오류 발생 시 처리 (선택 사항)
66                                 errorBuilder: (context, error, stackTrace) {
67                                     print("Error loading image: $error");
68                                     return Icon(Icons.broken_image, size: imageActualSize, color:
Colors.grey);
69                                 },
70                             ),
71                         ),
72                     ],
73                 );
74             },
75         ),
76     );
77 }
78 }
79
80 class GradientCircularProgressPainter extends CustomPainter {
81     final double progress; // 0.0 ~ 1.0
82     final Color startColor;
83     final Color endColor;
84     final double strokeWidth;
85     final Color? backgroundColor; // 배경 원의 색상 (선택 사항)
86
87     GradientCircularProgressPainter({
88         required this.progress,
89         required this.startColor,
90         required this.endColor,
91         this.strokeWidth = 10.0,
92         this.backgroundColor,
93     });
94

```

```

95  @override
96  void paint(Canvas canvas, Size size) {
97      final Offset center = Offset(size.width / 2, size.height / 2);
98      final double radius = math.min(size.width / 2, size.height / 2) - strokeWidth / 2;
99      // 그라데이션 영역을 위한 Rect, strokeWidth의 절반을 고려하여 stroke가 중앙에 오도록
100     final Rect rect = Rect.fromCircle(center: center, radius: radius);
101
102     // 배경 원 그리기 (선택 사항)
103     if (backgroundColor != null) {
104         final backgroundPaint = Paint()
105             ..color = backgroundColor!
106             ..style = PaintingStyle.stroke
107             ..strokeWidth = strokeWidth;
108         canvas.drawCircle(center, radius, backgroundPaint);
109     }
110
111     // 프로그레스 아크(arc) 그리기
112     final progressPaint = Paint()
113     // SweepGradient를 사용하여 원형 그라데이션 적용
114     ..shader = SweepGradient(
115         colors: [startColor, endColor],
116         startAngle: -math.pi / 2, // 12시 방향에서 시작
117         // endAngle은 progress에 따라 동적으로 계산되어야 하지만,
118         // SweepGradient 자체는 전체 원에 대한 그라데이션을 정의하고,
119         // drawArc에서 그리는 각도로 실제 표시되는 부분을 제어합니다.
120         // 하지만 그라데이션이 progress에 따라 변하게 하려면 아래처럼 stops나 transform을 조
121         절해야 할 수 있음.
122         // 여기서는 간단하게 전체 그라데이션을 만들고 drawArc로 자르는 방식을 사용.
123         // 또는, endAngle을 progress에 맞추고 tileMode를 clamp로 하여 progress 부분만 그라
124         데이션이 적용되게 할 수도 있습니다.
125         // 더 정확한 방법은 transform 또는 stops를 조절하는 것입니다.
126         // 여기서는 간단히 하기 위해, 전체 원에 그라데이션을 적용하고 drawArc로 해당 부분만 잘라
127         냅니다.
128         // 하지만 더 나은 효과를 위해 그라데이션 자체도 progress에 따라 변하도록 할 수 있습니
129         다. (아래 transform 참고)
130         stops: [0.0, 1.0], // 그라데이션 색상 정지 지점
131         tileMode: TileMode.clamp,
132         // 그라데이션 시작점을 12시 방향으로 맞추기 위한 변환
133         // transform: GradientRotation(-math.pi / 2), // 이 방법도 가능
134     ).createShader(rect) // createShader에는 그라데이션이 적용될 영역(rect)을 전달
135     ..style = PaintingStyle.stroke
136     ..strokeWidth = strokeWidth
137     ..strokeCap = StrokeCap.round; // 선의 끝을 둥글게
138
139     // progress가 0보다 클 때만 그립니다.
140     if (progress > 0.0) { // 0일때는 그리지 않도록
141         // drawArc의 sweepAngle을 사용하여 progress 만큼만 호를 그립니다.
142         // 시작 각도는 -math.pi / 2 (12시 방향)
143         // sweepAngle은 progress * 2 * math.pi (전체 원의 각도에 progress 비율을 곱함)
144         canvas.drawArc(
145             rect, // 아크를 그릴 사각형 영역
146             -math.pi / 2, // 시작 각도 (12시 방향)
147             math.pi * 2 * progress.clamp(0.0, 1.0), // 진행 각도 (0.0 ~ 1.0 사이 값으로 제한)

```

```

144         false, // 호의 중심과 연결하지 않음 (false = 호만 그림, true = 파이 조각)
145         progressPaint,
146     );
147 }
148 }
149
150 @override
151 bool shouldRepaint(covariant GradientCircularProgressPainter oldDelegate) {
152     return progress != oldDelegate.progress ||
153         startColor != oldDelegate.startColor ||
154         endColor != oldDelegate.endColor ||
155         strokeWidth != oldDelegate.strokeWidth ||
156         backgroundColor != oldDelegate.backgroundColor;
157 }
158 }

```

- 이미지가 회전하는 애니메이션은 정상 작동
- 하지만 원형 프로그레스 바는 Stack에 포함되지 않아 화면에 나타나지 않음

✅ 해결: 원형 Circular Progress 추가된 최종 버전

아래는 기존 구조를 유지하면서 원형 그라데이션 프로그레스를 정상적으로 표시하도록 수정한 버전입니다.

🔧 핵심 변경점

1. CustomPaint 를 Stack의 가장 아래에 추가
2. animatedProgress 를 Painter에 전달
3. strokeWidth는 기본값 유지 또는 필요 시 옵션화 가능
4. 이미지가 원형 프로그레스 중앙에 위치하도록 구조 정돈

📌 완성된 build() 버전

```

1 import 'package:flutter/material.dart';
2 import 'dart:math' as math;
3
4
5 class GradientCircularProgressPainter extends CustomPainter {
6     final double progress; // 0.0 ~ 1.0
7     final Color startColor;
8     final Color endColor;
9     final double strokeWidth;
10    final Color? backgroundColor; // 배경 원 색상 (선택)
11
12    GradientCircularProgressPainter({
13        required this.progress,
14        required this.startColor,
15        required this.endColor,
16        this.strokeWidth = 10.0,

```

```

17     this.backgroundColor,
18 });
19
20 @override
21 void paint(Canvas canvas, Size size) {
22     final Offset center = Offset(size.width / 2, size.height / 2);
23     final double radius = math.min(size.width / 2, size.height / 2) - strokeWidth / 2;
24     final Rect rect = Rect.fromCircle(center: center, radius: radius);
25
26     if (backgroundColor != null) {
27         final backgroundPaint = Paint()
28             ..color = backgroundColor!
29             ..style = PaintingStyle.stroke
30             ..strokeWidth = strokeWidth;
31         canvas.drawCircle(center, radius, backgroundPaint);
32     }
33
34     final progressPaint = Paint()
35         ..shader = SweepGradient(
36             colors: [startColor, endColor],
37             startAngle: -math.pi / 2,
38             stops: [0.0, 1.0],
39             tileMode: TileMode.clamp,
40         ).createShader(rect)
41         ..style = PaintingStyle.stroke
42         ..strokeWidth = strokeWidth
43         ..strokeCap = StrokeCap.round;
44
45     if (progress > 0.0) {
46         canvas.drawArc(
47             rect,
48             -math.pi / 2,
49             math.pi * 2 * progress.clamp(0.0, 1.0),
50             false,
51             progressPaint,
52         );
53     }
54 }
55
56 @override
57 bool shouldRepaint(covariant GradientCircularProgressPainter oldDelegate) {
58     return progress != oldDelegate.progress ||
59         startColor != oldDelegate.startColor ||
60         endColor != oldDelegate.endColor ||
61         strokeWidth != oldDelegate.strokeWidth ||
62         backgroundColor != oldDelegate.backgroundColor;
63 }
64 }
65
66 class CircularProgresswithImage extends StatefulWidget {
67     final double size;
68     final double progress; // 0.0 ~ 1.0
69     final Duration progressAnimationDuration;

```

```

70   final String imagePath;
71
72   const CircularProgressWithImage({
73     super.key,
74     required this.size,
75     required this.progress,
76     required this.imagePath,
77     this.progressAnimationDuration = const Duration(milliseconds: 500),
78   });
79
80   @override
81   State<CircularProgressWithImage> createState() => _CircularProgressWithImageState();
82 }
83
84 class _CircularProgressWithImageState extends State<CircularProgressWithImage>
85   with SingleTickerProviderStateMixin {
86   late final AnimationController _rotationController;
87
88   @override
89   void initState() {
90     super.initState();
91     _rotationController = AnimationController(
92       vsync: this,
93       duration: const Duration(seconds: 2),
94     )..repeat();
95   }
96
97   @override
98   void dispose() {
99     _rotationController.dispose();
100    super.dispose();
101  }
102
103  @override
104  Widget build(BuildContext context) {
105    final double imageActualSize = widget.size * 0.55;
106
107    return SizedBox(
108      width: widget.size,
109      height: widget.size,
110      child: TweenAnimationBuilder<double>(
111        tween: Tween<double>(begin: 0.0, end: widget.progress.clamp(0.0, 1.0)),
112        duration: widget.progressAnimationDuration,
113        builder: (context, animatedProgress, _) {
114          return Stack(
115            alignment: Alignment.center,
116            children: [
117              CustomPaint(
118                size: Size(widget.size, widget.size),
119                painter: GradientCircularProgressPainter(
120                  progress: animatedProgress,
121                  startColor: Colors.blueAccent,
122                  endColor: Colors.lightBlueAccent,

```

```

123         strokeWidth: 10,
124         backgroundColor: Colors.grey[300],
125     ),
126 ),
127 RotationTransition(
128     turns: _rotationController,
129     child: Image.asset(
130         widget.imagePath,
131         width: imageActualSize,
132         height: imageActualSize,
133         fit: BoxFit.contain,
134         errorBuilder: (context, error, stackTrace) {
135             print("Error loading image: $error");
136             return Icon(Icons.broken_image,
137                 size: imageActualSize, color: Colors.grey);
138         },
139     ),
140 ),
141 ],
142 );
143 },
144 ),
145 );
146 }
147 }
148

```

기대 동작

기능	동작 여부
이미지 회전	✓ 정상 (AnimationController.repeat)
프로그레스 바 표시	✓ Stack 아래에 CustomPaint 추가
progress 애니메이션	✓ TweenAnimationBuilder 연결
그라데이션 원형 표현	✓ GradientCircularProgressPainter 적용
중앙 이미지 크기 조절	✓ 이미지가 CircularProgress보다 작게 조정

화면 구성 미리 설명

```

1  +-----+
2  |           Progress Bar           | <-- GradientCircularProgressPainter
3  |           +-----+             |
4  |           | Rotating Img |       | <-- RotationTransition + Image.asset
5  |           +-----+             |
6  +-----+

```

