w_left_triangle.dart

소스 코드

전체 코드

```dart
// RoundedLeftTriangleWidget 클래스는 그대로 유지
import 'package:flutter/material.dart';

class RoundedLeftTriangleWidget extends StatelessWidget {
  final double width;
  final double height;
  final double cornerDepth; // 왼쪽 둥근 부분의 x축 깊이
  final double cornerSpread; // 왼쪽 둥근 부분이 y축으로 퍼지는 정도 (높이의 절반 기준)
  final Color color;

  const RoundedLeftTriangleWidget({
    Key? key,
    required this.width,
    required this.height,
    this.cornerDepth = 10.0, // 기본 깊이
    this.cornerSpread = 8.0, // 기본 퍼짐 정도
    this.color = Colors.blue,
  })  : assert(cornerSpread * 2 <= height, "cornerSpread * 2 cannot exceed height"),
        super(key: key);

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      width: width,
      height: height,
      child: CustomPaint(
        painter: _RoundedLeftTrianglePainter(
          color: color,
          cornerDepth: cornerDepth,
          cornerSpread: cornerSpread,
        ),
      ),
    );
  }
}


class _RoundedLeftTrianglePainter extends CustomPainter {
  final Color color;
  final double cornerDepth;  // x축 방향으로 얼마나 들어갈지
  final double cornerSpread; // y축 방향으로 얼마나 펼쳐질지 (중심에서부터의 거리)

  _RoundedLeftTrianglePainter({
    required this.color,
    required this.cornerDepth,
```

```dart
    required this.cornerSpread,
  });

  @override
  void paint(Canvas canvas, Size size) {
    final Paint paint = Paint()..color = color;
    final Path path = Path();

    final double width = size.width;
    final double height = size.height;

    // 왼쪽 둥근 부분이 시작되고 끝나는 y 좌표
    final double arcTopY = height / 2 - cornerSpread;
    final double arcBottomY = height / 2 + cornerSpread;

    // 경로 그리기 시작: 오른쪽 상단 (뾰족)
    path.moveTo(width, 0);

    // 오른쪽 상단에서 오른쪽 하단으로 직선 (뾰족)
    path.lineTo(width, height);

    // 오른쪽 하단에서 왼쪽 둥근 부분의 하단 시작점으로 직선 (뾰족한 연결)
    path.lineTo(cornerDepth, arcBottomY); // x 좌표는 cornerDepth, y는 arcBottomY

    // 왼쪽 둥근 부분: quadratic Bezier 곡선 사용
    // 제어점은 (0, height / 2) - 즉, 가장 왼쪽 중앙점
    // 끝점은 (cornerDepth, arcTopY)
    path.quadraticBezierTo(
        0, height / 2, // 제어점 (x1, y1) - 가장 뾰족해야 할 지점
        cornerDepth, arcTopY  // 끝점 (x2, y2)
    );

    // 왼쪽 둥근 부분의 상단 끝점에서 시작점(오른쪽 상단)으로 직선을 그어 경로를 닫음
    path.lineTo(width, 0);

    canvas.drawPath(path, paint);
  }

  @override
  bool shouldRepaint(covariant _RoundedLeftTrianglePainter oldDelegate) {
    return oldDelegate.color != color ||
        oldDelegate.cornerDepth != cornerDepth ||
        oldDelegate.cornerSpread != cornerSpread;
  }
}

```

1 파일 개요

파일명: `lib/view/widget/step1/w_left_triangle.dart`
역할: 왼쪽 둥근 삼각형(Rounded Left Triangle) 위젯 정의
주요 목적:

- UI 장식용 또는 강조용 **커스텀 삼각형 위젯**
- 왼쪽이 둥글고 오른쪽이 뾰족한 비대칭 삼각형 생성
- 색상, 크기, 둥근 깊이, 퍼짐 정도를 매개변수로 조정 가능

사용 맥락 예시:

- 카드, 배너, 단계 표시 등 UI 장식
- 버튼 배경, 강조 표시, 진행 표시용 삼각형

2️⃣ 주요 기능

1. **크기 지정**
   - `width`, `height` 매개변수로 위젯 크기 조절
   - `SizedBox` 내부에 `CustomPaint` 배치
2. **왼쪽 둥근 처리**
   - `cornerDepth`: 왼쪽 둥근 부분 x축 깊이
   - `cornerSpread`: y축 퍼짐 정도 (높이 기준)
   - `_RoundedLeftTrianglePainter`에서 `quadraticBezierTo`로 구현
3. **색상 지정**
   - `color` 매개변수로 색상 설정 가능
   - 기본값: `Colors.blue`
4. **재사용성**
   - StatelessWidget 구조로 다양한 화면에서 반복 사용 가능
   - 매개변수 변경만으로 모양과 크기 변경 가능

3️⃣ 구조 분석

```
RoundedLeftTriangleWidget (StatelessWidget)
 ├─ width, height
 ├─ cornerDepth, cornerSpread
 ├─ color
 └─ build()
       └─ SizedBox(width, height)
             └─ CustomPaint(painter: _RoundedLeftTrianglePainter)

_RoundedLeftTrianglePainter (CustomPainter)
 ├─ color, cornerDepth, cornerSpread
 ├─ paint(Canvas canvas, Size size)
 |    ├─ Path 생성:
 |    |    ├─ moveTo(width, 0) → 오른쪽 상단
 |    |    ├─ lineTo(width, height) → 오른쪽 하단
 |    |    ├─ lineTo(cornerDepth, arcBottomY) → 왼쪽 둥근 시작
 |    |    ├─ quadraticBezierTo(0, height/2, cornerDepth, arcTopY) → 둥근 곡선
 |    |    └─ lineTo(width, 0) → 경로 닫기
```

```
|     └ canvas.drawPath(path, paint)
└ shouldRepaint() → color, cornerDepth, cornerSpread 변경 시 repaint
```

- `CustomPainter`를 사용하여 벡터 기반 렌더링
- 좌측 곡선 + 우측 뾰족 구조

---

4 동작 흐름

1. `RoundedLeftTriangleWidget(width: 100, height: 50)` 호출
2. `SizedBox`에 크기 지정 후 `CustomPaint` 위젯 생성
3. `_RoundedLeftTrianglePainter.paint()` 호출
    - 오른쪽 상단 → 오른쪽 하단 → 왼쪽 둥근 시작 → 곡선 → 오른쪽 상단으로 닫기
4. Bezier 곡선을 통해 부드러운 좌측 곡선 표현
5. `canvas.drawPath()`로 화면 렌더링

---

5 장점

- **커스텀 UI 구현 가능**: Flutter 기본 위젯만으로는 어려운 비대칭 삼각형
- **매개변수 조절 가능**: 크기, 색상, 곡선 깊이/퍼짐 자유
- **벡터 기반**: 해상도 독립적, 선명한 렌더링
- **재사용성 높음**: StatelessWidget 구조

---

6 단점 / 개선점

1. **Bezier 제어점 고정**
    - 현재 중앙 제어점 `(0, height/2)`만 사용 → 곡선 형태가 고정됨
    - 개선: 제어점을 매개변수화하거나 곡선 곡률 조정 가능
2. **반응형 디자인 미지원**
    - width, height, cornerDepth, cornerSpread 모두 고정값 → 화면 크기 변화에 대응 어려움
    - 개선: MediaQuery, LayoutBuilder 활용
3. **애니메이션/상호작용 미지원**
    - 현재는 단순 그리기용
    - 개선: 색상 변화, 크기 변화, 터치 시 반응 애니메이션 추가 가능

---

7 결론

- 이 파일은 `RoundedLeftTriangleWidget`을 정의하는 전형적인 **커스텀 UI 위젯**입니다.
- 비대칭 삼각형 + 좌측 둥근 처리라는 특징이 있어, 단계 표시, 강조, 장식용 UI에 유용합니다.
- 개선하면 **반응형, 매개변수 확장, 애니메이션**까지 적용 가능하여 다양한 화면에서 재사용성을 높일 수 있습니다.