Paint Logic Tutorial

Kobe Taylor

Displaying the Canvas

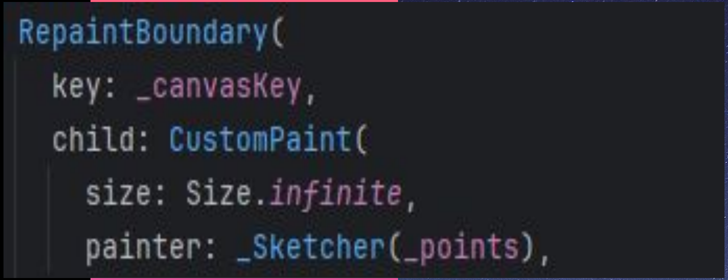
RepaintBoundary

- Isolate this widget's painting from the rest of the ui

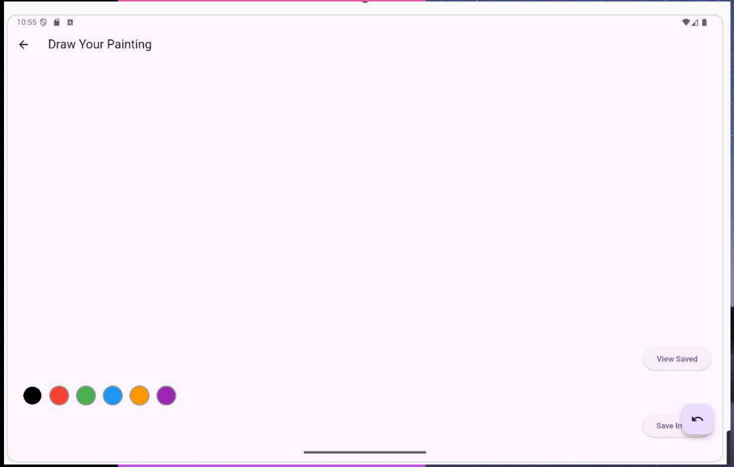
  We use this so we can save the canvas as an image

size: Size.infinite

  Makes the canvas fill as much space as possible.

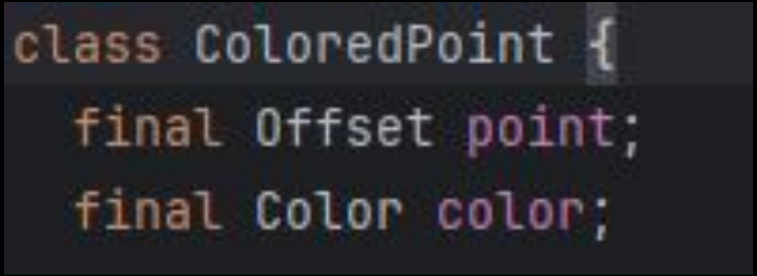


```
RepaintBoundary(
  key: _canvasKey,
  child: CustomPaint(
    size: Size.infinite,
    painter: _Sketcher(_points),
```



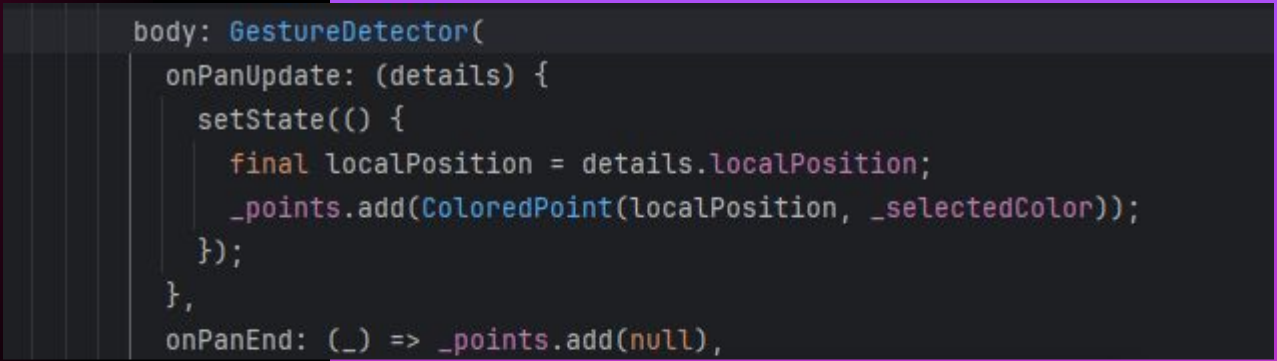
Data Model for Drawing

01	Offset Point	x/y position of the stroke - tells where the user touches the screen
02	Color color	Keeps track of the color used at that point



```
class ColoredPoint {
  final Offset point;
  final Color color;
```

Gesture Detection



```
body: GestureDetector(
  onPanUpdate: (details) {
    setState(() {
      final localPosition = details.localPosition;
      _points.add(ColoredPoint(localPosition, _selectedColor));
    });
  },
  onPanEnd: (_) => _points.add(null),
```

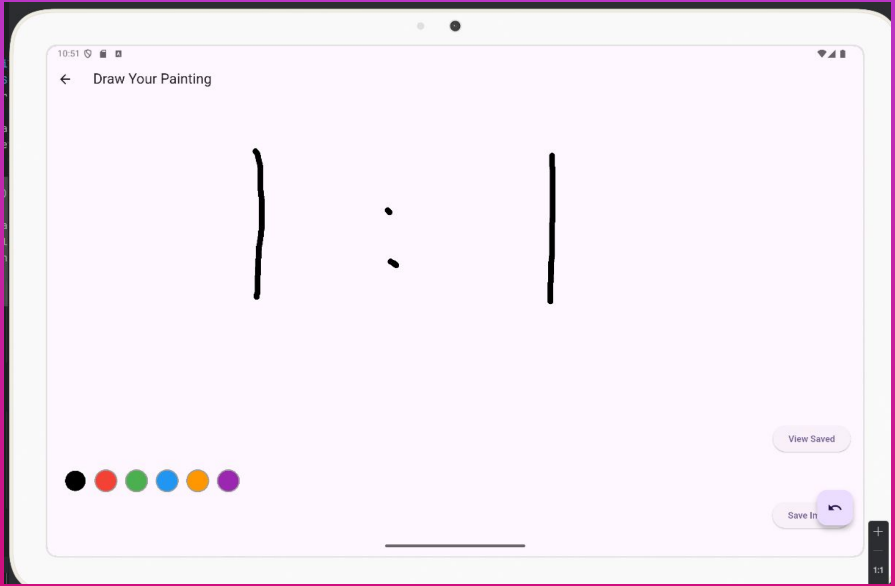
GestureDetector

- Flutter Widget that detects gestures like taps and drags

We use an onPanUpdate, which registers when a cursor or “finger” is dragged across a screen

onPanEnd: (_) => _points.add(null),

- When a user draws, the points will connect until the user lifts their finger
- Adds null to _points to indicate the end of a stroke.

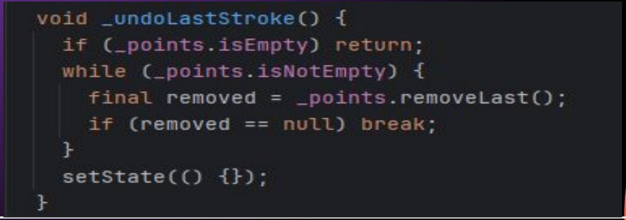


→ final removed = _points.removeLast();

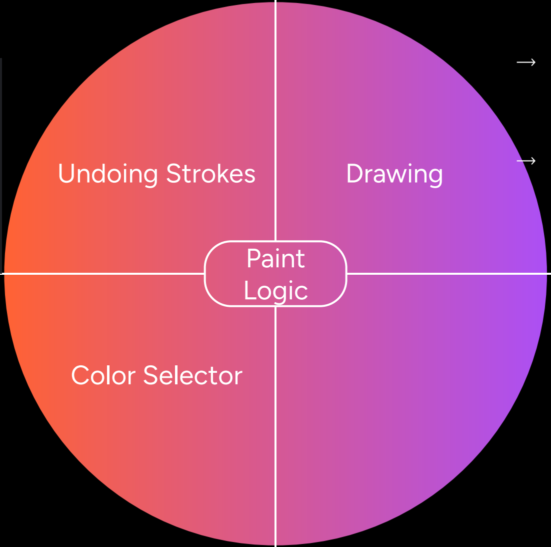
- Takes the last item in the list and removes it.
- Saves it in a variable to check if it's null.

→ if (removed == null) break;

- When we reach a null, that means we've reached the end of the last stroke.
- We stop removing points once we've erased that stroke.



```
void _undoLastStroke() {
  if (_points.isEmpty) return;
  while (_points.isNotEmpty) {
    final removed = _points.removeLast();
    if (removed == null) break;
  }
  setState(() {});
}
```



→ final paint = Paint()

→ ..color = p1.color

- Sets the color was active when the point was created

→ ..strokeWidth = 9.0

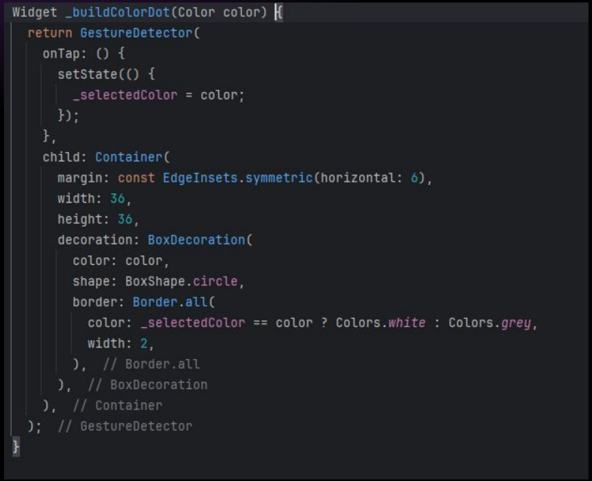
- The thickness of the brush

→ ..strokeCap = StrokeCap.round;

- Makes the line smooth and rounded

→ canvas.drawLine(p1.point, p2.point, paint);

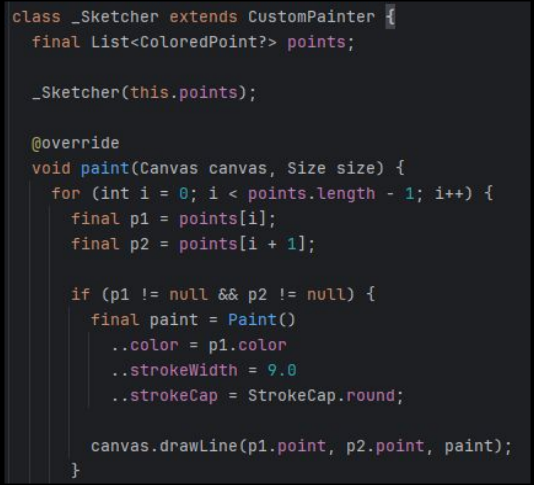
- Draws a straight line on the canvas between the two points using the defined paint style



```
Widget _buildColorDot(Color color) {
  return GestureDetector(
    onTap: () {
      setState(() {
        _selectedColor = color;
      });
    },
    child: Container(
      margin: const EdgeInsets.symmetric(horizontal: 6),
      width: 36,
      height: 36,
      decoration: BoxDecoration(
        color: color,
        shape: BoxShape.circle,
        border: Border.all(
          color: _selectedColor == color ? Colors.white : Colors.grey,
          width: 2,
        ), // Border.all
      ), // BoxDecoration
    ), // Container
  ); // GestureDetector
}
```

→ Renders tappable color circles

→ Tapping changes brush color



```
class _Sketcher extends CustomPainter {
  final List<ColoredPoint?> points;

  _Sketcher(this.points);

  @override
  void paint(Canvas canvas, Size size) {
    for (int i = 0; i < points.length - 1; i++) {
      final p1 = points[i];
      final p2 = points[i + 1];

      if (p1 != null && p2 != null) {
        final paint = Paint()
          ..color = p1.color
          ..strokeWidth = 9.0
          ..strokeCap = StrokeCap.round;

        canvas.drawLine(p1.point, p2.point, paint);
      }
```

THANK YOU!