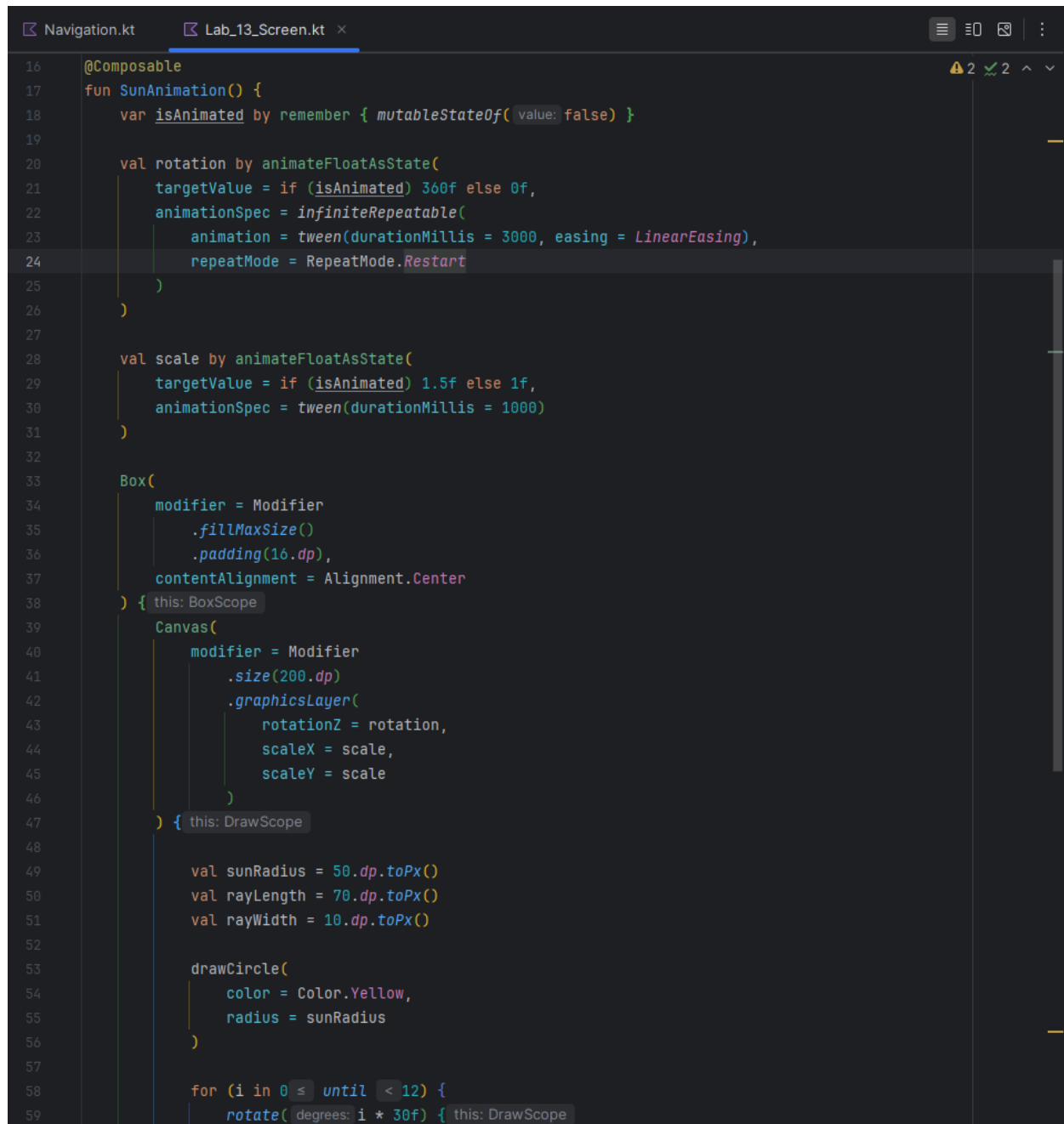


Лабораторная работа №13

АНИМАЦИЯ ГЕОМЕТРИЧЕСКИХ ОБЪЕКТОВ

Цель работы: создать приложение с возможностью воспроизведения анимации геометрических объектов в двумерном и трёхмерном пространствах.

A screenshot of an IDE window titled 'Lab_13_Screen.kt'. The code is written in Kotlin and implements a sun animation. It starts with a @Composable function 'SunAnimation' that uses 'remember' to store a 'mutableStateOf' for 'isAnimated'. It then uses 'animateFloatAsState' to animate 'rotation' and 'scale'. The 'rotation' animation is infinite and repeatable with a duration of 3000ms and 'LinearEasing'. The 'scale' animation has a duration of 1000ms. The UI is composed of a 'Box' containing a 'Canvas'. The 'Canvas' is sized to 200.dp and contains a 'graphicsLayer' with 'rotationZ', 'scaleX', and 'scaleY' properties. Below the canvas, there are constants for 'sunRadius', 'rayLength', and 'rayWidth', and a 'drawCircle' call with 'Color.Yellow'. Finally, a 'for' loop rotates the canvas 12 times, each time by 30 degrees.

```
16 @Composable
17 fun SunAnimation() {
18     var isAnimated by remember { mutableStateOf(value: false) }
19
20     val rotation by animateFloatAsState(
21         targetValue = if (isAnimated) 360f else 0f,
22         animationSpec = infiniteRepeatable(
23             animation = tween(durationMillis = 3000, easing = LinearEasing),
24             repeatMode = RepeatMode.Restart
25         )
26     )
27
28     val scale by animateFloatAsState(
29         targetValue = if (isAnimated) 1.5f else 1f,
30         animationSpec = tween(durationMillis = 1000)
31     )
32
33     Box(
34         modifier = Modifier
35             .fillMaxSize()
36             .padding(16.dp),
37         contentAlignment = Alignment.Center
38     ) { this: BoxScope
39         Canvas(
40             modifier = Modifier
41                 .size(200.dp)
42                 .graphicsLayer(
43                     rotationZ = rotation,
44                     scaleX = scale,
45                     scaleY = scale
46                 )
47         ) { this: DrawScope
48
49             val sunRadius = 50.dp.toPx()
50             val rayLength = 70.dp.toPx()
51             val rayWidth = 10.dp.toPx()
52
53             drawCircle(
54                 color = Color.Yellow,
55                 radius = sunRadius
56             )
57
58             for (i in 0 until 12) {
59                 rotate(degrees: i * 30f) { this: DrawScope
```

Рис 13.1. Код лабораторной работы

```
Navigation.kt Lab_13_Screen.kt x
17 fun SunAnimation() {
38     ) { this: BoxScope
47         ) { this: DrawScope
49             val sunRadius = 50.dp.toPx()
50             val rayLength = 70.dp.toPx()
51             val rayWidth = 10.dp.toPx()
52
53             drawCircle(
54                 color = Color.Yellow,
55                 radius = sunRadius
56             )
57
58             for (i in 0 until 12) {
59                 rotate(degrees: i * 30f) { this: DrawScope
60                     drawLine(
61                         color = Color.Yellow,
62                         start = Offset(sunRadius, y: 0f),
63                         end = Offset(x: sunRadius + rayLength, y: 0f),
64                         strokeWidth = rayWidth
65                     )
66                 }
67             }
68         }
69
70         Spacer(modifier = Modifier.height(16.dp))
71
72         Button(
73             onClick = { isAnimated = !isAnimated },
74             modifier = Modifier.align(Alignment.BottomCenter)
75         ) { this: RowScope
76             Text(if (isAnimated) "Остановить" else "Запустить")
77         }
78     }
79 }
80
81 @Composable
82 fun Lab_13_Screen(navController: NavController) {
83     SunAnimation()
84 }
```

Рис 13.2. Код лабораторной работы

Результаты работы программы:

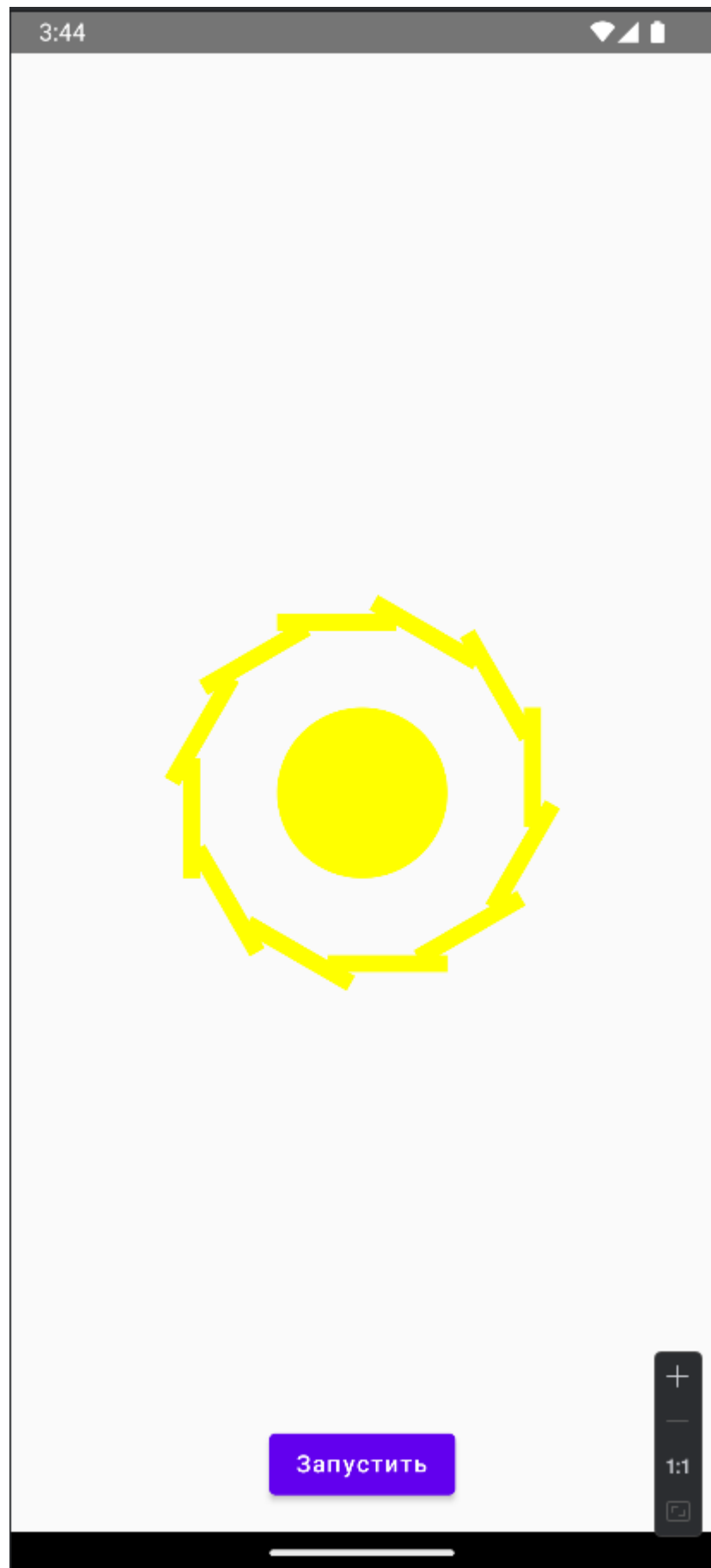


Рис 13.3. Изначальное состояние

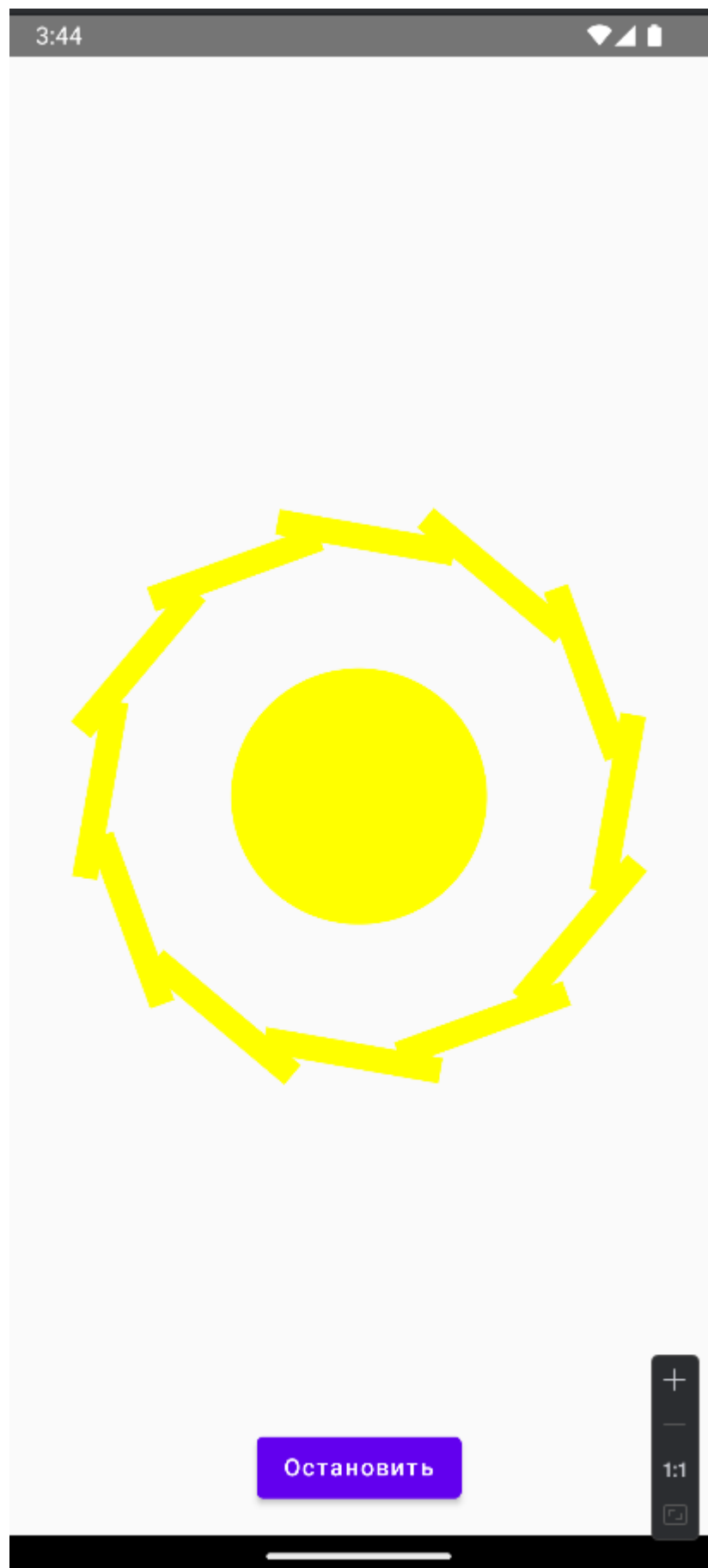


Рис 13.4. Процесс анимации

Выполнил	Баранько Д.А. 090303-ПИА-о22
Проверил	Елкин Н.С.