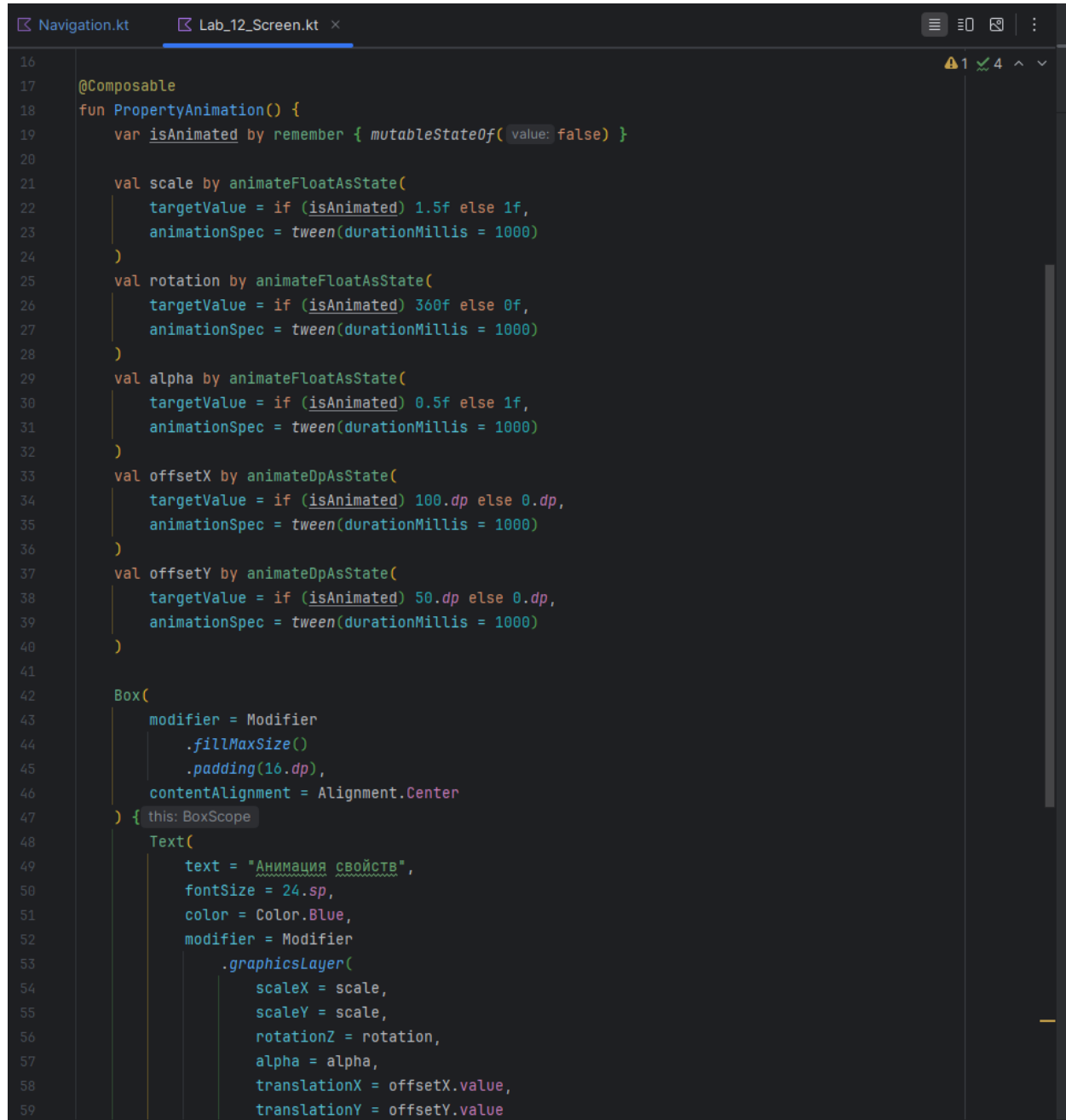


Лабораторная работа №12

АНИМАЦИЯ СВОЙСТВ

Цель работы: создать приложение с возможностью воспроизведения анимации свойств.

A screenshot of an IDE window showing two tabs: 'Navigation.kt' and 'Lab_12_Screen.kt'. The 'Lab_12_Screen.kt' tab is active, displaying Kotlin code for a composable function named 'PropertyAnimation'. The code uses Compose's animation APIs like 'animateFloatAsState' and 'animateDpAsState' to animate various properties of a text box. The text box contains the text 'Анимация свойств' and is styled with a blue color and a font size of 24.sp. The animation is triggered by a state variable 'isAnimated' which is managed by 'remember' and 'mutableStateOf'. The animation duration is set to 1000 milliseconds for all properties. The code is line-numbered from 16 to 59.

```
16
17 @Composable
18 fun PropertyAnimation() {
19     var isAnimated by remember { mutableStateOf( value: false) }
20
21     val scale by animateFloatAsState(
22         |   targetValue = if (isAnimated) 1.5f else 1f,
23         |   animationSpec = tween(durationMillis = 1000)
24         | )
25     val rotation by animateFloatAsState(
26         |   targetValue = if (isAnimated) 360f else 0f,
27         |   animationSpec = tween(durationMillis = 1000)
28         | )
29     val alpha by animateFloatAsState(
30         |   targetValue = if (isAnimated) 0.5f else 1f,
31         |   animationSpec = tween(durationMillis = 1000)
32         | )
33     val offsetX by animateDpAsState(
34         |   targetValue = if (isAnimated) 100.dp else 0.dp,
35         |   animationSpec = tween(durationMillis = 1000)
36         | )
37     val offsetY by animateDpAsState(
38         |   targetValue = if (isAnimated) 50.dp else 0.dp,
39         |   animationSpec = tween(durationMillis = 1000)
40         | )
41
42     Box(
43         |   modifier = Modifier
44         |       .fillMaxSize()
45         |       .padding(16.dp),
46         |   contentAlignment = Alignment.Center
47     ) { this: BoxScope
48         Text(
49             |   text = "Анимация свойств",
50             |   fontSize = 24.sp,
51             |   color = Color.Blue,
52             |   modifier = Modifier
53             |       .graphicsLayer(
54             |           |   scaleX = scale,
55             |           |   scaleY = scale,
56             |           |   rotationZ = rotation,
57             |           |   alpha = alpha,
58             |           |   translationX = offsetX.value,
59             |           |   translationY = offsetY.value
```

Рис 12.1. Код лабораторной работы

```
Navigation.kt  Lab_12_Screen.kt x
18 fun PropertyAnimation() {
42     Box(
43         modifier = Modifier
44             .fillMaxSize()
45             .padding(16.dp),
46         contentAlignment = Alignment.Center
47     ) { this: BoxScope
48         Text(
49             text = "АНИМАЦИЯ СВОЙСТВ",
50             fontSize = 24.sp,
51             color = Color.Blue,
52             modifier = Modifier
53                 .graphicsLayer(
54                     scaleX = scale,
55                     scaleY = scale,
56                     rotationZ = rotation,
57                     alpha = alpha,
58                     translationX = offsetX.value,
59                     translationY = offsetY.value
60                 )
61                 .background(Color.LightGray)
62                 .padding(16.dp)
63         )
64
65         Spacer(modifier = Modifier.height(16.dp))
66
67         Button(
68             onClick = { isAnimated = !isAnimated },
69             modifier = Modifier.align(Alignment.BottomCenter)
70         ) { this: RowScope
71             Text(if (isAnimated) "Остановить" else "Запустить")
72         }
73     }
74 }
75
76 @Composable
77 fun Lab_12_Screen(navController: NavController) {
78     PropertyAnimation()
79 }
```

Рис 12.2. Код лабораторной работы

Результаты работы программы:

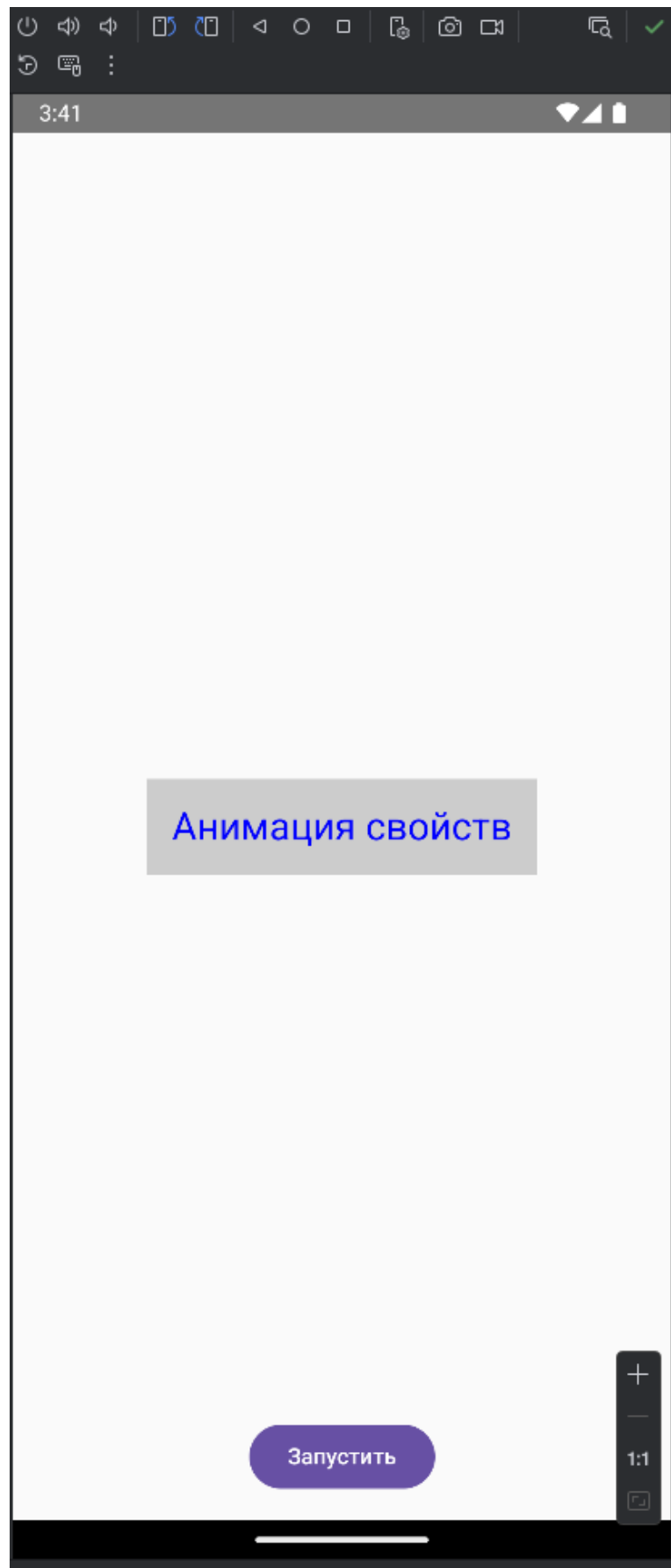


Рис 12.3. Изначальное состояние

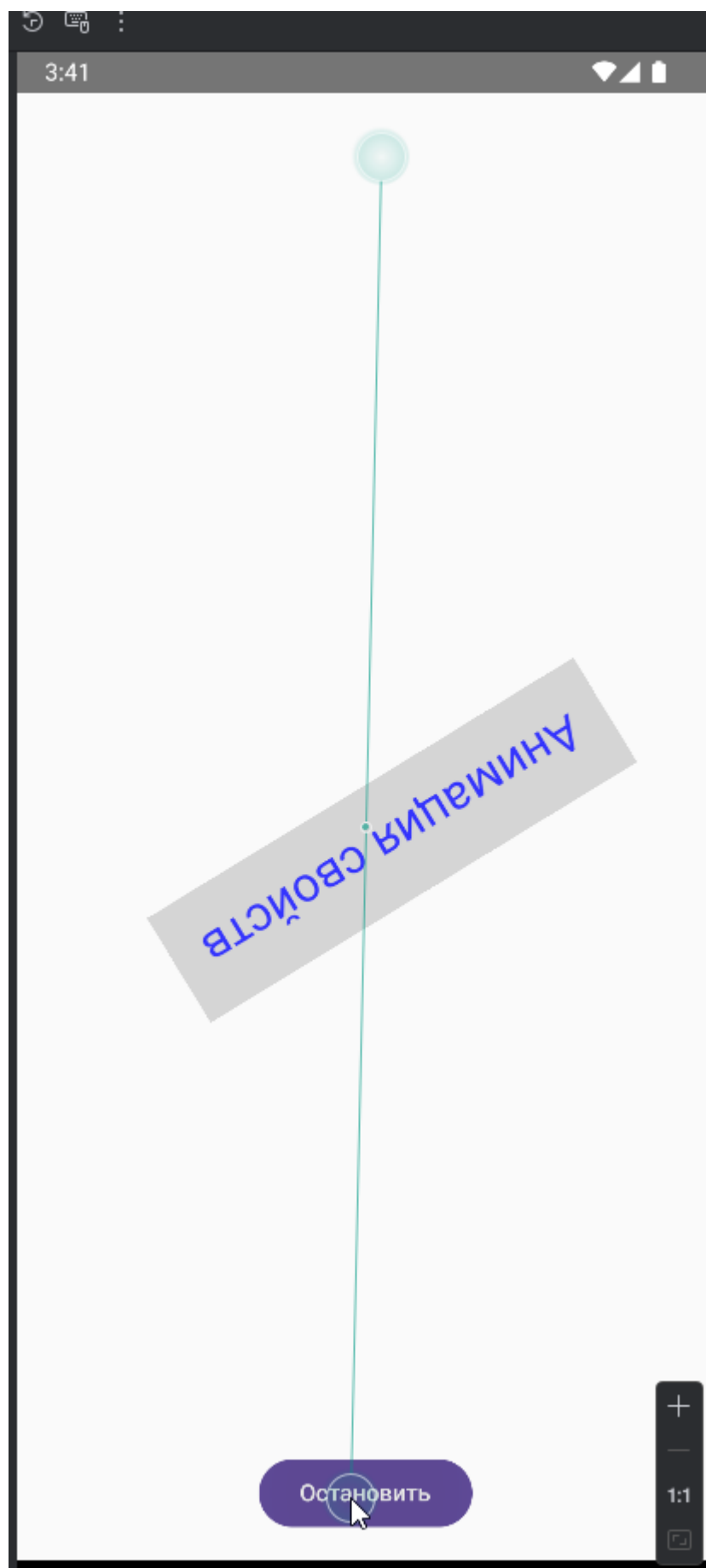


Рис 12.4. Процесс анимации

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