Animacije

Jetpack Compose animacije

prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

Razvoj mobilnih aplikacija i servisa
Računarstvo i informatika

2024.

Animacije

- Predstavljaju tranzicije u okviru korisničkog interfejsa
- Vrše se iz jednog stanja u drugo
- Jetpack Compose kao UI biblioteka nudi gotov API za formiranje animacija
- Animacije su vezane za state promenljive unutar Composable funkcija

Animations API

- Pruža funkcije za rad sa animacijama
- Ima ih mnogo i postoje pravila koje se funkcije koriste u zavisnosti od objekta koji treba animirati
- Grafički prikaz: https://developer.android.com/static/develop/ui/compose/images/animations/compose_animation_decision_tree_v2.jpg

Animations API

- Beskonačne animacije (*rememberInfiniteTransition*)
- Enter/exit animacije (*enterTransition*, *exitTransition*)
- Animiranje sadržaja elementa (*AnimatedContent*)
- Animiranje vidljivosti elementa (*AnimatedVisibility*)
- Animiranje veličine elementa (*Modifier.animateContentSize*)
- Animiranje pozicije elementa (*animateItemPlacement*)
- Animiranje property-ja elemenata (*updateTransition*, *animate*AsState*)
  - * = *Float, Color, Size, Offset, Int, Rect*, ...

Animiranje vidljivosti

- AnimatedVisibility(Boolean) – composable koji prima boolean parametar koji označava da li je sadržaj u okviru composable vidljiv

```
AnimatedVisibility(
    visible = shown,
    enter = slideInVertically(),
    exit = slideOutVertically()
)
```

Animiranje vrednosti

- animate*AsState – funkcija koja omogućava animaciju vrednosti promenljive na osnovu nekih uslova

```
val backgroundColor by animateColorAsState(
        targetValue = if (tabPage == TabPage.Home) Seashell
                      else GreenLight,
        label = "background color")
```

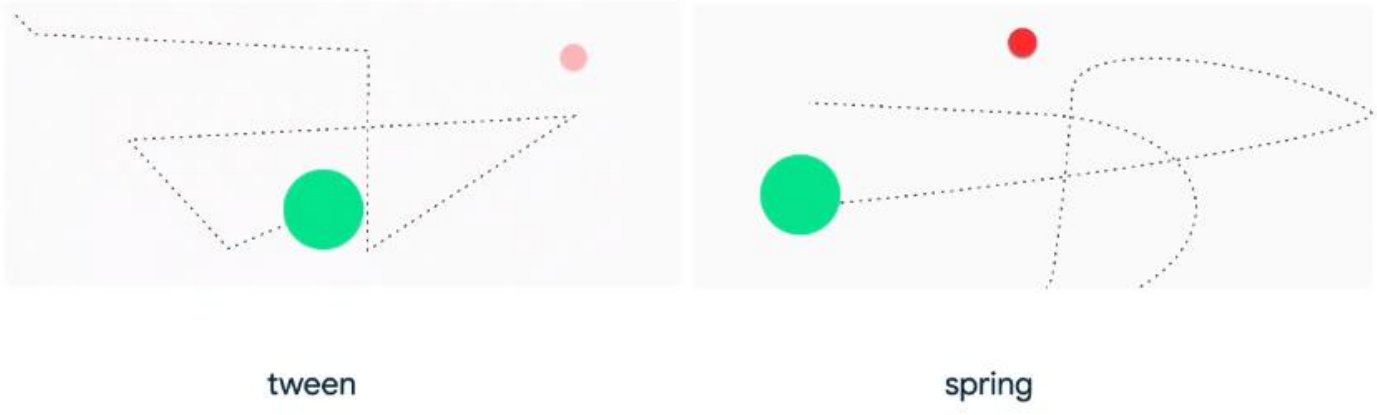
Animiranje vrednosti

```
var visible by remember {
    mutableStateOf(true)
}
val animatedAlpha by animateFloatAsState(
    targetValue = if (visible) 1.0f else 0f,
    label = "alpha"
)
```

```
Box(
    modifier = Modifier
        .size(200.dp)
        .graphicsLayer {
            alpha = animatedAlpha
        }
        .clip(RoundedCornerShape(8.dp))
        .background(colorGreen)
        .align(Alignment.TopCenter)
) { }
```

Tween i spring

- Dva načina tranzicije između animacija
- Spring je lagan i podržava tečne prelaske između animacija
- Tween animira između dve vrednosti (be*tween*)



Spring

```
val value by animateFloatAsState(
    targetValue = 1f,
    animationSpec = spring(
        dampingRatio = Spring.DampingRatioHighBouncy,
        stiffness = Spring.StiffnessMedium
    )
)
```

Tween

```
val value by animateFloatAsState(
    targetValue = 1f,
    animationSpec = tween(
        durationMillis = 300,
        delayMillis = 50,
        easing = LinearOutSlowInEasing
    )
)
```

Keyframes

- Keyframes dozvoljavaju specificiranje ključnih tačaka u animaciji

```
val value by animateFloatAsState(
    targetValue = 1f,
    animationSpec = keyframes {
        durationMillis = 375
        0.0f at 0 with LinearOutSlowInEasing // for 0-15 ms
        0.2f at 15 with FastOutLinearInEasing // for 15-75 ms
        0.4f at 75 // ms
        0.4f at 225 // ms
    }
)
```

Repeatable animacija

- Animacija koja se ponavlja specificirani broj puta

```
val value by animateFloatAsState(
    targetValue = 1f,
    animationSpec = repeatable(
        iterations = 3,
        animation = tween(durationMillis = 300),
        repeatMode = RepeatMode.Reverse
    )
)
```

InfiniteRepeatable animacija

- Animacija koja se ponavlja beskonačan broj puta

```
val value by animateFloatAsState(
    targetValue = 1f,
    animationSpec = infiniteRepeatable(
        animation = tween(durationMillis = 300),
        repeatMode = RepeatMode.Reverse
    )
)
```

Literatura

- Animations - Android Docs
- Animations Codelab 1
- Animations Codelab 2

Hvala na pažnji!