Topic for discussion

Introduction to Compose Animation

Workshop hosted by **Hiren**
(Android Engineer)

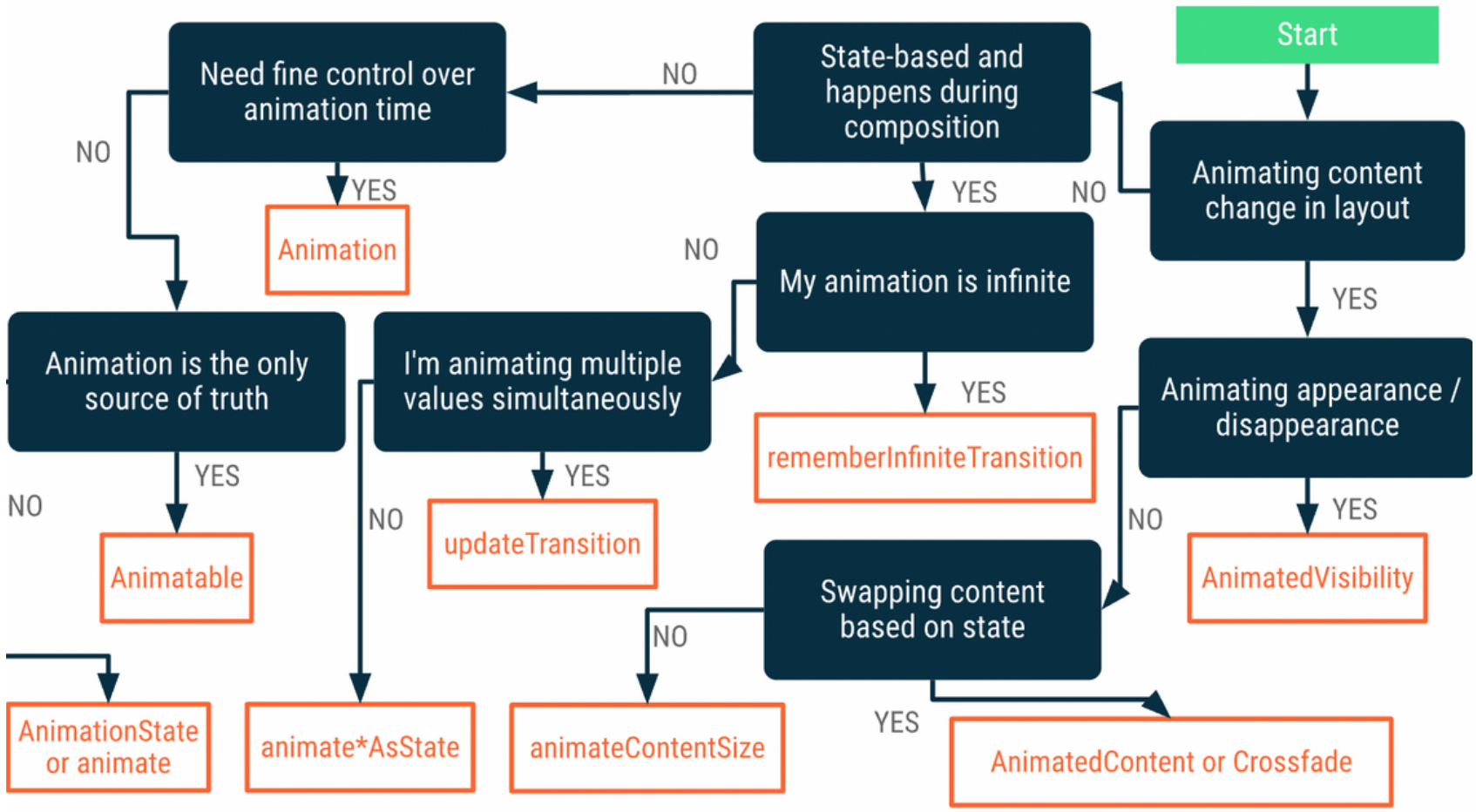
Animation Reimagined
Compose Animation APIs

- Declarative and interruptible
- Easy to use
- Tooling support

Declaring Dependencies

```
dependencies {
    implementation("androidx.compose.animation:animation:1.2.0-beta03")
}

android {
    buildFeatures {
        compose = true
    }

    composeOptions {
        kotlinCompilerExtensionVersion = "1.2.0-beta03"
    }

    kotlinOptions {
        jvmTarget = "1.8"
    }
}
```

The diagram below helps you decide what API to use to implement your animation.



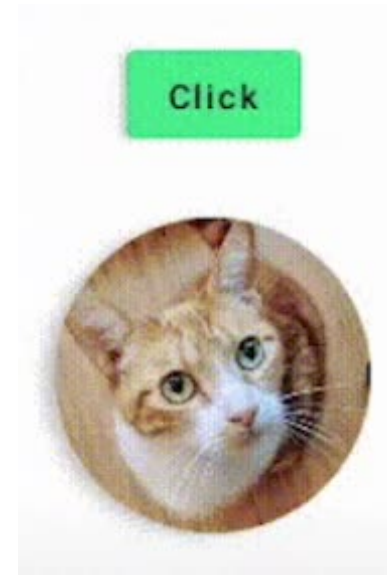
- For Appearance and disappearance use AnimatedVisibility
- Swapping content based on state:
  - If you are crossfading content: Crossfade
  - Otherwise use AnimatedContent
- If the animation is infinite rememberInfiniteTransition

Toggling Visibility - Enter and Exit

```
var visible by remember { mutableStateOf(true) }

Column {
    Button(onClick = { visible = !visible }) {
        Text("Click")
    }

    if (visible) {
        CatIcon()
    }
}
```

Recomposed on state changes

```
var visible by remember { mutableStateOf(true) }

Column {
    Button(onClick = { visible = !visible }) {
        Text("Click")
    }

    AnimatedVisibility (visible) {
        CatIcon()
    }
}
```



AnimatedVisibility & AnimatedContent

AnimatedVisibility

Enter and exit if its child

AnimatedContent

Transaction between content changes.

EnterTransition

fadeIn()

slideIn()

scaleIn()

ExitTransition

fadeOut()

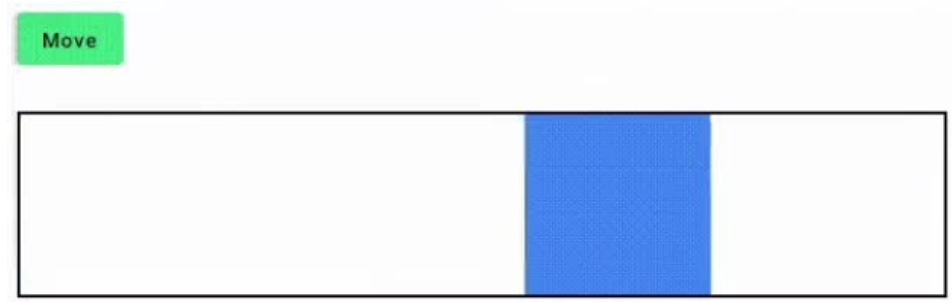
slideOut()

scaleOut()

animate*AsState Animate a single value

Animate a single value

```
val offsetX by animateDpAsState(
    if (isOn) 512.dp else 0.dp
)
```



Interruptible Animation