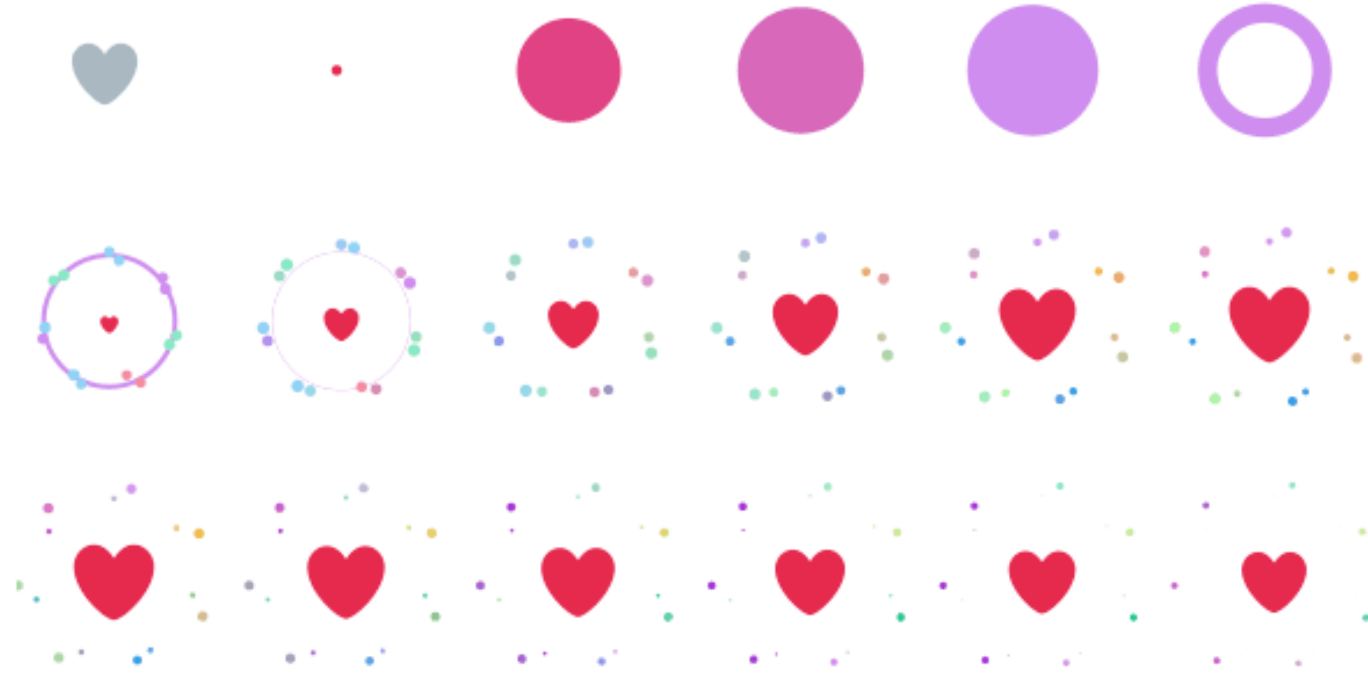


Jetpack Compose Animasyon Temelleri

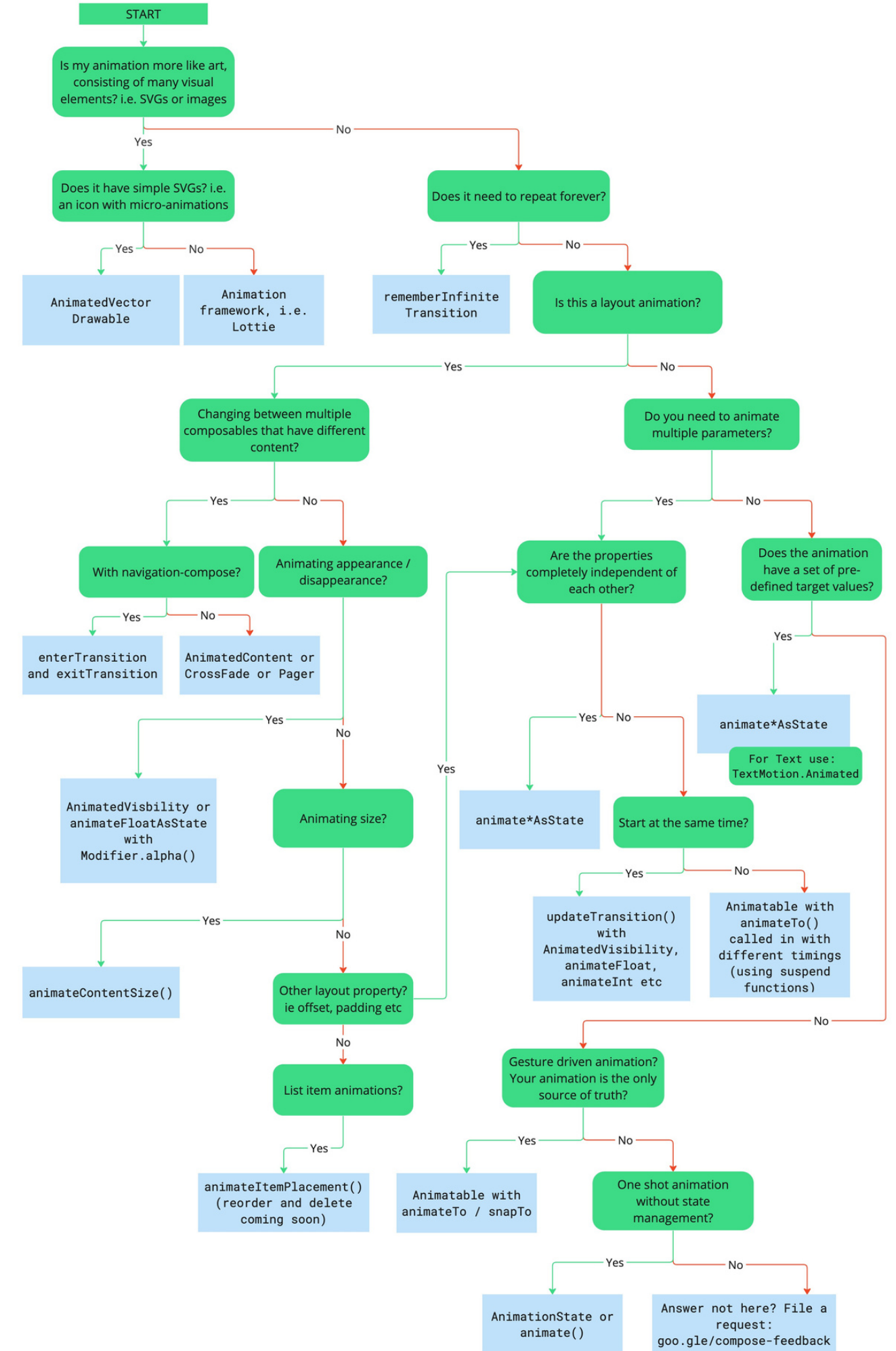
Animation Nedir?

Animasyon, bir veya birden fazla değerin her time frame'deki devamlı değişimi ve bu değişim sonucunda oluşan hareket(motion) illüzyonudur.



Api Listesi

Compose farklı animasyon caselerini karşılamak için farklı fonksiyonlar kullanıyor. Bu yüzden View sistemine kıyasla fonksiyon sayısı daha fazla. Bunları akılda tutmak epey zor olduğu için Google bize bir condition şeması ile yardımcı oluyor.



::animateXAsState

Tek bir değeri animate etmek için kullanılıyor. Her common type için ayrı bir fonksiyon var. (::animateFloatAsState, ::animateColorAsState)



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

val alpha: Float by animateFloatAsState(if (enabled) 1f else 0.5f)
Box(
    Modifier.fillMaxSize()
        .graphicsLayer(alpha = alpha)
        .background(Color.Red)
)
```

::updateTransition

Aynı değişime bağımlı birden fazla değeri animate etmek için kullanılıyor.

```
enum class BoxState {  
    Collapsed,  
    Expanded  
}  
  
var currentState by remember { mutableStateOf(BoxState.Collapsed) }  
val transition = updateTransition(currentState, label = "box state")  
  
val rect by transition.animateRect(label = "rectangle") { state ->  
    when (state) {  
        BoxState.Collapsed -> Rect(0f, 0f, 100f, 100f)  
        BoxState.Expanded -> Rect(100f, 100f, 300f, 300f)  
    }  
}  
  
val borderWidth by transition.animateDp(label = "border width") { state ->  
    when (state) {  
        BoxState.Collapsed -> 1.dp  
        BoxState.Expanded -> 0.dp  
    }  
}
```

AnimatedVisibility

Duruma göre composable ağacına giren veya çıkan
Composable için visibility animasyonu




```
var condition by remember { mutableStateOf(true) }
AnimatedVisibility(visible = condition) {
    Text(text = "Edit")
}

var condition by remember { mutableStateOf(true) }
AnimatedVisibility(
    visible = condition,
    enter = slideInVertically() + expandVertically() + fadeIn(),
    exit = slideOutVertically() + shrinkVertically() + fadeOut()
) {
    Text("Hello", Modifier.fillMaxWidth().height(200.dp))
}
```

animateItemPlacement

Lazy List item'larını animate etmek için **AnimateVisibility** kullanamıyoruz, çalışmıyor. Onun yerine bize bu **Modifier** sunulmuş. Bu Modifier kullanıldığında Lazy List item değişiklikleri ekrana animated şekilde yansıyor. Fakat şu an sadece item sıra değişimi için çalışıyor. Item eklendiğinde veya çıkarıldığında çalışmıyor.



```
LazyColumn {  
    items(books, key = { it.id }) {  
        Row(Modifier.animateItemPlacement()) {  
            // ...  
        }  
    }  
}
```

AnimationSpec

AnimationSpec, animasyonların süre, tekrar ve interpolation parametrelerini custom etmeye yarıyor. Neredeyse her animasyon fonksiyonu opsiyonel AnimationSpec parametresi kabul ediyor.

Üç adet temel hazır AnimationSpec var:

Tween: Interpolation için Linear, Curve gibi basit matematik denklemleri kullanan spec. Animasyonlarda en yaygın bu kullanılıyor.

Spring: Internal olarak yaylı mekaniği hesaplamaları içeren spec. Bu spec'te Interpolation'ı yönetme şeklimiz yaylının stiffness, damping ve mass değerlerini değiştirmek.

KeyframesSpec: Bu spec, karışık matematik denklemleri yazmaya gerek kalmadan belirli zaman noktalarında interpolasyona müdahale etmeye yarıyor.