International Journal of Computer (IJC)

ISSN 2307-4523 (Print & Online)

https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

Creating Custom Animations Using Motionlayout in Android

Chike Mgbemena[*]

Mobile Software Engineer,Lagos, Nigeria

Email:chikespott@gmail.com

Abstract

This article discusses the process of creating custom animations using Motion Layout in Android. Motion Layout, as an extension of ConstraintLayout, provides developers with a powerful tool for managing animations and transitions between layouts. The main focus is on describing the features of Motion Layout, such as creating smooth transitions and complex animation effects with a minimum amount of code, as well as integration with various user interactions. An overview of the key components, including MotionScene and ConstraintSet, that provide flexibility and power in animation development is provided. Practical code examples and recommendations for using Motion Layout to improve the user interface of mobile applications are considered. The article also focuses on the relevance and demand for the use of animations in modern mobile applications, supporting this with statistical data on the growth of the mobile device market and user expectations.

Keywords: custom animations; animations; using Motion Layout; Motion Layout; android applications; IT;programming.

Introduction

Animations play a key role in creating engaging and user-friendly interfaces in mobile applications. According to a Statista study, the number of smartphone users exceeded 6.3 billion in 2023, highlighting the importance of mobile applications in people's daily lives. It is projected that by 2024, the mobile app market will reach 935 billion US dollars [1]. One of the technologies contributing to the creation of high-quality animations is Motion Layout, which provides developers with powerful tools for creating complex and intuitive animations. Motion Layout not only enhances user experience but also significantly improves the visual appeal of applications, making them more attractive to users.Numerous studies confirm the significance and effectiveness of Motion Layout in interface animation for mobile applications. For example, a study by Li and his colleagues [7], demonstrated the efficiency of adaptive animation of human movements for e-learning.

Received: 7/15/2024
Accepted: 9/15/2024
Published: 9/25/2024

----

* Corresponding author.

Their method allowed extracting human motion data at various levels of detail, which improved the realism and engagement of users. Another example is the work by Ye and his colleagues [8], where the ARAnimator system was studied. This system allows creating animations of virtual characters in augmented reality using gestures, significantly simplifying the process of creating animations in real environments.In addition, the study by Reitsma and Pollard [9] analyzed the use of motion graphs for creating realistic character animations. The results showed that the quality of animation depends on the complexity of the target environment and tasks, highlighting the importance of carefully evaluating animation methods for various applications. Meanwhile, Aberman and his colleagues [10] presented a new method for motion retargeting captured on video between different performers without the need for explicit 3D pose and camera parameter reconstruction. This significantly simplified the animation process and improved the quality of the final result.Another significant study was conducted by Mayer and his colleagues [11], who demonstrated the use of the MotionViz mobile application for artistic visualization of human motion. The application uses augmented reality technologies and expressive graphics to create animations, allowing users to customize visual effects and enhance the perception of movements.These studies emphasize the significance of Motion Layout as a tool for creating complex and interactive animations in mobile applications. The use of Motion Layout allows developers to create intuitive and realistic animations, improving user interaction with applications and increasing their satisfaction. Given the current trends in the mobile application market and the growing demands of users for interface quality, mastering and using Motion Layout becomes an essential skill for modern developers.

1. General Characteristics of Animations in Android

Animation is the process of adding motion to any type of image, object, or text. It can not only bring an image to life but also change its shape. In Android, animation is used to create visually appealing user interfaces. The main types of animations and their descriptions are presented in Table 1.

Table 1: Main Types of Animations

Type of Animation	**Description of Animation Type**
Property Animation	Property Animation is a powerful tool that allows animating almost any object. Introduced in Android 3.0, this platform offers high flexibility and reliability. Property animation can be applied to various interface elements such as CheckBox, RadioButton, and other widgets, providing a wide range of possibilities for implementing complex animation effects.
View Animation	View Animation is designed for animating individual interface elements. This type of animation allows changing parameters such as size, rotation, and start and end points of objects. Although view animation is less flexible and slower compared to property animation, it is excellent for implementing specific visual effects, such as expanding layouts. For example, view animation can be seen in an Expandable RecyclerView, where list elements smoothly expand.
Drawable Animation	Drawable Animation is used to create animations by sequentially displaying a series of images. This method provides an easy way to understand the principle of animation by showing one image after another, creating the illusion of movement. Examples of drawable animation are often found in various applications, such as app logo splash screens or loading sequences displaying a series of images to create a dynamic visual effect.

From the data presented in Table 1, it is evident that animation in Android offers a variety of tools to enhance user experience, allowing developers to create rich and interactive interfaces [2].The main methods of animations will be presented in Table 2.

Table 2: Animation Methods

Methods	Description
startAnimation()	This method will start the animation.
clearAnimation()	This method will clear the animation started in a specific view.

For animating raster images such as icons or illustrations, use the Drawable Animation API. These animations are typically defined statically using Drawable resources, but their behavior can also be set at runtime. For example, an effective way to indicate the relationship between two actions is to animate a play button transforming into a pause button when pressed [3].

When changing the visibility or position of elements in a layout, it is advisable to use subtle animation so that users can easily track interface changes. To move, show, or hide elements within the current layout, use the property animation system from the android.animation package, available from Android 3.0 (API level 11) and above. These APIs allow updating the properties of View objects over a specified duration, continuously redrawing the element as the properties change. For example, changing position properties will move an element across the screen, while changing the alpha property will make it appear or disappear.

To simply create such animations, you can enable layout animation so that changes in element visibility are automatically accompanied by animation. For more information, see “Animating Layout Changes” [4].

2. Using MotionLayout

Motion Layout is an extension of ConstraintLayout designed for creating complex user interface animations in Android applications. Conceptually, Motion Layout can be seen as a blend of a layout system and an animation mechanism, allowing developers to define not only the static positioning of elements but also their dynamic behavior.

The principle of Motion Layout is based on a declarative approach to describing animations. Developers define the initial and final states of the interface and the transition rules between them. Motion Layout automatically generates intermediate frames, ensuring smooth animation.

Compared to traditional animation methods in Android, such as ObjectAnimator or ViewPropertyAnimator, Motion Layout offers a higher level of abstraction. This allows for creating complex animations with less code and a more intuitive description.

The architecture of Motion Layout consists of several key components:

1. MotionScene: The central element that describes all aspects of the animation. MotionScene is defined in

a separate XML file and contains information about states, transitions, and keyframes.

2. ConstraintSet: A set of constraints defining the position and properties of Views in a specific state. Typically, two main ConstraintSets are defined: for the initial and final states.

3. Transition: Describes the rules for transitioning between states, including animation duration, interpolators, and event handlers.

4. KeyFrames: Allow defining intermediate states of the animation, providing finer control over the trajectory of motion and changes in View properties.

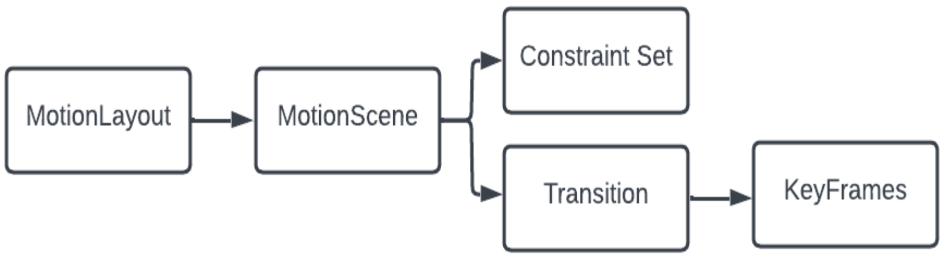

Figure 1: Motion Layout Component Interaction Diagram

Motion Layout offers a wide range of capabilities for creating user interface animations:

1. Managing View Position and Size: Motion Layout allows animating changes in position, size, and rotation of interface elements. This is achieved by modifying the corresponding attributes in the ConstraintSet.

2. Animating Attributes: In addition to geometric properties, Motion Layout can animate any View attributes, including color, transparency, and custom properties. This is implemented using CustomAttribute in the ConstraintSet.

3. Working with Motion Paths: Motion Layout supports defining arbitrary motion paths for Views using KeyPosition. This enables creating non-linear movement animations following a specified path.

4. Coordinating Multiple Animations: One of the key advantages of Motion Layout is the ability to synchronize animations of multiple Views within a single transition. This ensures consistency and integrity of the interface animation.

5. Responsiveness to User Gestures: Motion Layout integrates mechanisms for handling user gestures, allowing the creation of interactive animations controlled by swipes or drags.

6. Support for ConstraintHelper: Motion Layout is compatible with ConstraintHelper, extending animation

capabilities for groups of Views and enabling more complex effects.

The efficiency of using Motion Layout can be illustrated by comparing the number of lines of code required to implement typical animations (Table 3).

Table 3: Animation code comparison

Type of Animation	Traditional Approach	Motion Layout
Simple movement	20-30 lines	10-15 lines
Resizing	25-35 lines	12-18 lines
Complex trajectory	50-70 lines	20-30 lines
Coordination of multiple Views	100+ lines	40-60 lines

It is important to note that the advantages of Motion Layout are especially evident when implementing complex animations, where traditional approaches require significantly more code and effort to coordinate individual animations.

In conclusion, the theoretical part emphasizes that Motion Layout is a powerful tool for creating complex user interface animations in Android. Its declarative approach and integration with the layout system allow developers to effectively implement dynamic user interfaces, reducing code volume and improving the readability and maintainability of the software.

3. Practical part

The basic steps to create animations with Motion Layout are:

1. Adding the ConstraintLayout dependency In the build.gradle file of your project, add the following dependency:

```
groovy
dependencies {
   implementation 'androidx.constraintlayout:constraintlayout:2.1.0'
}
```

Creating a MotionLayout. Convert an existing ConstraintLayout to MotionLayout in your main layout file (activity_main.xml):

```xml
xml
<androidx.constraintlayout.motion.widget.MotionLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layoutDescription="@xml/motion_scene"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Swipe me!" />
</androidx.constraintlayout.motion.widget.MotionLayout>
```

Creating a MotionScene file In the res/xml folder, create the motion_scene.xml file that will define the animation:

```xml
xml
<?xml version="1.0" encoding="utf-8"?>
<MotionScene xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:motion="http://schemas.android.com/apk/res-auto">
    <Transition
        motion:constraintSetStart="@+id/start"
        motion:constraintSetEnd="@+id/end"
        motion:duration="1000">
        <OnSwipe
            motion:dragDirection="dragRight"
            motion:touchAnchorId="@+id/button"
            motion:touchAnchorSide="right" />
    </Transition>
    <ConstraintSet android:id="@+id/start">
        <Constraint
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            motion:layout_constraintStart_toStartOf="parent"
            motion:layout_constraintTop_toTopOf="parent" />
    </ConstraintSet>
    <ConstraintSet android:id="@+id/end">
        <Constraint
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginTop="8dp"
            motion:layout_constraintEnd_toEndOf="parent"
            motion:layout_constraintTop_toTopOf="parent" />
    </ConstraintSet>
</MotionScene>
```

4. Launch and test the animation Launch the application on your device or Android emulator. When you swipe the button, it should move smoothly from the left side of the screen to the right side.

The example using Motion Layout demonstrates that this tool significantly simplifies the creation of complex animations through the declarative description of states and transitions between them. In the provided button animation, the amount of code was minimized, highlighting the high efficiency of Motion Layout in UI development. This efficiency allows developers to focus more on the creative aspects of UI design rather than on technical implementation.

Motion Layout enables developers not only to control the position and size of elements but also to animate properties such as color, transparency, and other View attributes. This is particularly useful when creating interactive interfaces that require a rich user experience, enhancing the visual appeal and engagement of mobile applications.

Despite its advantages, there are limitations to consider when using Motion Layout. One notable limitation is related to performance on older devices. Complex animations may cause performance issues such as lagging or increased memory usage, which can negatively impact the user experience. Therefore, developers should carefully assess the complexity of animations when targeting devices with lower hardware capabilities.Another limitation is that Motion Layout is most effective in applications where animations are central to the user experience. In simpler applications, where animation is not a critical element, the use of Motion Layout may be excessive in terms of both development effort and performance overhead.Thus, while Motion Layout provides a powerful framework for creating complex animations, its use should be aligned with the specific needs and constraints of the target application.

4.Conclusion

Using Motion Layout allows developers to create complex and intuitive animations, enhancing user interaction with applications. Its flexibility and ease of use significantly improve the visual perception of applications, ensuring smooth and attractive animations. However, the discussion of the research results has highlighted several key aspects that require further clarification.

First, Motion Layout has proven its effectiveness in reducing the amount of code needed to create complex animations. This allows developers to focus on the creative aspects of design without spending excessive time on technical implementation. Examples presented in this research demonstrate how Motion Layout can be easily integrated into mobile applications, minimizing development costs.

Second, it is essential to clearly define the limitations of this study. During experiments, performance issues were encountered on older devices, where complex animations could cause the application to slow down. Additionally, Motion Layout is most suitable for applications where animations play a key role in the user experience. In other cases, using Motion Layout may be unnecessary and unjustified in terms of performance and development costs.

Finally, previous studies reviewed indicate that while Motion Layout is a powerful tool, its application should be carefully considered depending on the usage context. The works of Li and his colleagues. [7], Ye and his colleagues. [8], Reitsma and Pollard [9], as well as Aberman and his colleagues. [10], confirm that the use of adaptive and interactive animations can significantly enhance the user experience if they are intelligently integrated into the overall design of the application

References

[1]. 89 Essential Mobile App Stats 2024 [FactsandTrends]. [Electronic resource] Mode of operation: https://thrivemyway.com/mobile-app-stats / (accessed 06.20.2024).

[2]. Implementing Animations and Transitions in Android. [Electronic resource] Mode of operation: https://nyamebismark12-nb.medium.com/animations-and-transitions-in-android-eeaf3bc7b28e (accessed 06.20.2024).

[3]. Animation in Android with Example. [Electronic resource] Mode of operation: https://www.geeksforgeeks.org/animation-in-android-with-example/(accessed 06.20.2024).

[4]. Introduction to animations. [Electronic resource] Mode of operation: https://developer.android.com/develop/ui/views/animations/overview (accessed 06.20.2024).

[5]. Getting Started with the Motion Editor in Android Studio 4.0. [Electronic resource] Mode of operation: https://riggaroo.dev/getting-started-with-the-motion-editor-in-android-studio-4-0 / (accessed 06.20.2024).

[6]. Introduction to MotionLayout (part I). [Electronic resource] Mode of operation: https://medium.com/google-developers/introduction-to-motionlayout-part-i-29208674b10d (accessed 06.20.2024).

[7]. Li F. W. B. et al. Adaptive animation of human motion for e-learning applications //International Journal of Distance Education Technologies (IJDET). – 2007. – Т. 5. – №. 2. – С. 74-85.

[8]. Ye H. et al. ARAnimator: In-situ character animation in mobile AR with user-defined motion gestures //ACM Transactions on Graphics (TOG). – 2020. – Т. 39. – №. 4. – С. 83: 1-83: 12.

[9]. Reitsma P. S. A., Pollard N. S. Evaluating motion graphs for character animation //ACM Transactions on Graphics (TOG). – 2007. – Т. 26. – №. 4. – С. 18-es.

[10]. Aberman K. et al. Learning character-agnostic motion for motion retargeting in 2d //arXiv preprint arXiv:1905.01680. – 2019.

[11]. Mayer M. et al. MotionViz: Artistic Visualization of Human Motion on Mobile Devices //ACM SIGGRAPH 2021 Appy Hour. – 2021. – С. 1-2.