今では、アニメーションをまったく使用しない優れたアプリを手に入れるのは困難です。

すべてのアプリが携帯電話の改善されたリソースを利用しており、ユーザーインターフェイスにポップやジャンプ、回転などのアニメーションを与えているようです。アプリにアニメーションを追加します。そのため、次のいくつかのレッスンでは、Flutterでアニメーションを作成する方法に焦点を当て、これらのアニメーションをFlutterアプリに組み込むことでユーザーに楽しい体験を作成できるようにします。

AndroidやiOSなどのネイティブAPIを使用してアニメーションを構築しているとき、多くのSDKはアニメーションがほとんど後付けのような時代に構築されたため、最初のAndroidとiPhoneにはできるリソースがありませんでした物事を簡単にアニメーション化します。

したがって、これらのアニメーションライブラリは実際にはかなり後のことであり、iOSまたはAndroidでアニメーションを作成する方法のチュートリアルを見てみると、特に曲線や異なるタイミングにもう少し複雑なものを見ると、実際には通常非常に多くのコードが含まれます。しかし、Flutterは最近になって構築されたため、最新のアプリにとってアニメーションがどれほど重要かを認識しています。

したがって、Flutter SDKを使用してアニメーションを実装する場合、Flutterアプリにアニメーションを追加するのは実際には非常に簡単であることがわかります。だからあなたを遠ざける理由はありません。

このレッスンでは、Flutterアプリで最も簡単で最も一般的に使用されるアニメーションの1つについて説明します。これらはヒーローアニメーションです。

それでは、ヒーローアニメーションはどのように見えますか？

まあそれは通常、画面遷移で行われます。

そのため、画面のページ1から画面のページ2に移動すると、共有要素がある場合、その要素はページ1とページ2の間で遷移します。

そのため、画像やアイコンなどの要素は両方のページに存在する必要があり、ユーザーがアプリのページ1からページ2に移動するときに継続的に遷移します。共有要素を持つという要件があるため、Androidから来た場合は、このアニメーションを共有要素の移行として知っている可能性があり、実際のFlutterアプリではこのように見えます。

そのため、1つの画面から別の画面に移行しており、その共有要素は2つの画面間でトゥイーンまたはアニメーション化されています。このレッスンでは、ヒーローアニメーションをアプリに実装します。

このように、ユーザーがようこそ画面からログイン画面に移動すると、小さな稲妻アイコンがこのようなスムーズな移行で2番目の画面に移行するため、これを実装します。それを行うには、まずヒーローアニメーションについて少し読む必要があります。しかし、ご想像のとおり、ウィジェットを使用することになり、そのウィジェットはヒーローウィジェットと呼ばれます。

また、ヒーローウィジェットアニメーションを実装するために必要なのは、3つの要素だけです。

最初の要素は2つのヒーローウィジェットで、1つは最初の画面に、もう1つは2つ目の画面にあります。

そして、それらのヒーローウィジェットはタグプロパティを共有する必要があるため、両方ともタグプロパティを持ち、両方のヒーローウィジェットで同じ値を持つように設定する必要があります。

そして最後に、プッシュまたはポップなどのナビゲーターベースの画面遷移を使用して、1番目と2番目の画面間を移動します。そして、ヒーローアニメーションを実装するために必要なのはこれだけです。

それでは、先に進んで、アプリでそれを行いましょう。

したがって、ウェルカム画面では、ロゴを示すこの画像、つまり稲妻の画像があることがわかります。

ここで、コンテナを単に沼地の標準的な不動コンテナにする代わりに、ヒーローウィジェット内にラップします。

したがって、それはHeroと呼ばれ、このウィジェットには1つの必須プロパティのみがあり、それがタグです。タグがないと、ヒーローウィジェットはアニメーション化する他のヒーローウィジェットを判別できません。

それでは、ヒーローウィジェットに含まれているもの、つまりロゴ画像がわかるように、タグを付けてこのロゴを呼び出しましょう。

この最初のヒーローウィジェットでは、コンテナの高さが60ピクセルであることに注意してください。

登録ボタンをクリックすることに決めたら、ナビゲータを使用して登録画面に移行します。

したがって、registration\_screen.dartに移動して、同じロゴの画像である共有要素がある場所を見つけます。

この場合、コンテナの高さはわずかに異なります。

はるかに大きく、60ではなく200ピクセルです。

したがって、このコンテナをヒーローウィジェット内にラップし、タグを付けて、タグが以前のヒーローウィジェットの以前のタグと正確に一致することを確認しましょう。

これが開始ヒーローウィジェットであり、これが最終的な終了ヒーローウィジェットです。

そして、それらは同じタグを持っているため、Flutterはこれが最後の外観であり、これが最初の外観であることを知っています。

そのため、画像の高さまたはサイズのみが変更されることに注意してください。

ヒーローアニメーションを実装するために必要なのはそれだけです。

そして、先に進んでアプリを実行すると、実際に非常に洗練された外観のアニメーションが、今やったほんの少しの作業で得られることがわかります。

私のヒーローウィジェットは登録画面にあるので、先に進み、登録ボタンをクリックします。この小さな画像、ロゴ画像をここで見たいと思います。

小さい画像からはるかに大きな画像に移行することに注意してください。

そして、この最初の画面が隠されていても、その画像は常に画面上にあります。

そのため、その画像はアニメーション化され、次の画面に移行するときに空中でほとんどホバリングしています。

だから今ここにあなたのための挑戦があります。

ようこそ画面から登録画面に移動するヒーローウィジェットを実装しました。

ログイン画面に移動したときにそのヒーローアニメーションを実行するために、その画像のログイン画面をクリックしても同じ効果が得られるかどうか試してみてください。

そのため、ビデオを一時停止して、このチャレンジを完了してみてください。

これで、ようこそ画面にヒーローアニメーションの出発点ができました。

そして、同じタグでヒーローウィジェットを作成する限り、任意の画面に遷移できます。

そのため、ヒーローウィジェットを作成し、その新しいヒーローウィジェット内のログイン画面上の画像でコンテナをラップし、ウェルカム画面にあるものと同じタグを与えることができます。

このタグがログイン画面、登録画面、ウェルカム画面の3つの場所に存在することに注目してください。

同じタグで持つことができるヒーロー要素の数に制限はありません。

しかし、これは、ようこそ画面からログイン画面に移動すると、Flutterは同じタグ、ここのロゴ、ここのロゴを持つヒーローウィジェットを見つけることができることを意味します。

つまり、アプリを再度実行すると、ログインをクリックしたときと登録をクリックしたときの両方でヒーローアニメーションを実装できるようになります。必要なのは、たった2行のコードで、非常に簡単でした。次のレッスンでは、さらにいくつかのカスタムアニメーションを見ていきます。アニメーションコントローラーを使用して、アニメーションをどのくらいの時間続けたいかなど、より複雑なアニメーションを定義できるようにします。またはアニメーションをどのように展開するか。

それで、それ以上については、次のレッスンで説明します。