

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI TP. HỒ CHÍ MINH
BỘ MÔN CÔNG NGHỆ THÔNG TIN**



BÁO CÁO TIÊU LUẬN ANDROID

ĐỀ TÀI:

TÌM HIỂU VỀ ANIMATION TRONG ANDROID

Sinh viên thực hiện: LÊ THỊ NGỌC MAI

VÕ SỸ KHÁ

NGUYỄN THANH QUI

Lớp : CQ.59.CNTT

Khoá : KHÓA 59

Tp. Hồ Chí Minh, năm 2021

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI TP. HỒ CHÍ MINH
BỘ MÔN CÔNG NGHỆ THÔNG TIN**



BÁO CÁO TIÊU LUẬN ANDROID

ĐỀ TÀI:

TÌM HIỂU VỀ ANIMATION TRONG ANDROID

Sinh viên thực hiện: LÊ THỊ NGỌC MAI

VÕ SỸ KHÁ

NGUYỄN THANH QUI

Lớp : CQ.59.CNTT

Khoa : KHÓA 59

Tp. Hồ Chí Minh, năm 2021

THIẾT KẾ TỔNG QUAN ĐỀ TÀI

Họ tên SV: Lê Thị Ngọc Mai – Võ Sỹ Khá – Nguyễn Thanh Qui

Khóa: Khóa 59

Lớp: CQ.59.CNTT

1. Tên đề tài: Tìm hiểu về Animation trong Android.

2. Mục đích, yêu cầu:

a. Mục đích

Nhằm mục đích cho chúng ta hiểu rõ hơn về các Animation trong Android

b. Yêu cầu

- Demo về các Animation

3. Nội dung và phạm vi đề tài

a. Nội dung

- Tổng quan về các Animation trong Android

- Định nghĩa các Animation

+ View Animation

+ Drawable Animation

+ Property Animation

+ Ripple Effect / Touch Feedback

+ Reveal Effect

+ Transition Animation

+ Animate View State Changes

+ AnimatedVectorDrawable

- Chương trình DEMO

b. Phạm vi đề tài

- Nghiên cứu công cụ lập trình Android Studio.

c. Kết luận

4. Công nghệ, công cụ và ngôn ngữ lập trình:

Công cụ hỗ trợ: Android Studio.

5. Các kết quả chính dự kiến sẽ đạt được và ứng dụng: Hoàn thành chương trình DEMO
Cho các hiệu ứng

LỜI CẢM ƠN

Lời đầu tiên em xin gửi lời cảm ơn chân thành đến quý thầy, cô giáo trong Bộ môn Công nghệ thông tin – Phân hiệu Trường Đại học Giao thông vận tải.

Những người đã truyền dạy, đã trang bị cho em kho tàng kiến thức về bầu trời công nghệ thông tin rộng lớn.

Ở đây, em không chỉ học được kiến thức về sách vở mà em còn học được các bài học, kỹ năng sống trước khi tạm biệt mái trường đại học thân yêu này và tiến ra biển đời mênh mông rộng lớn. Đặc biệt, em xin gửi lời cảm ơn chân thành và sâu sắc đến **Thầy Nguyễn Quang Phúc**, người đã đồng hành cùng em trong suốt quá trình làm thực tập chuyên môn, người đã bỏ thời gian quý báu, thậm chí là thời gian nghỉ ngơi để hướng dẫn, để định hướng đường đi nước bước cho em. Em thật chẳng biết dùng lời nào để diễn tả được công lao của thầy.

Trong quá trình học tập và tìm hiểu em đã nỗ lực rất nhiều với mong muốn hoàn thành đồ án một cách tốt nhất, nhưng đời người sẽ có những thiếu sót không thể tránh khỏi, và với những người chưa chững chạc và trưởng thành như em thì sai lầm là không thể không mắc phải. Em mong thầy, cô bộ môn có thể thông cảm và cho em những ý kiến, đóng góp để em có thể hoàn thành đồ án của mình.

Sau cùng, em xin kính chúc Quý Thầy Cô trong Bộ môn Công nghệ thông tin lời chúc sức khoẻ, luôn hạnh phúc và thành công hơn nữa trong công việc cũng như trong cuộc sống.

Em xin chân thành cảm ơn!

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

*Tp. Hồ Chí Minh, ngày tháng năm
Giảng viên hướng dẫn*

MỤC LỤC

I. TỔNG QUAN ĐỀ TÀI

1	Tổng quan về các Animation	7
1.1	Animation là gì?	7
II.	Định nghĩa các Animation	7
1.	View Animation	7
2.	Drawable Animations	8
3.	Property Animations	10
4.	Ripple Efect / Touch Feedback	12
5.	Reveal Effect	13
6.	Transition Animation.....	14
7.	Animate View State Changes	15
8.	AnimatedVectorDrawable.....	16
9.	Constraint Animation / ConstraintSet	20
III.	CHƯƠNG TRÌNH DEMO	22
1.	Giao diện main.....	22
2.	View Animation.....	23
3.	Drawable Animation.....	24
4.	Property Animation	25
5.	Ripple Efect / Touch Feedback	26
6.	Reveal Effect	27
7.	Transition Animation.....	28
8.	Animate View State Changes	29
9.	AnimatedVectorDrawable.....	30
	TÀI LIỆU THAM KHẢO.....	31

I. TỔNG QUAN ĐỀ TÀI

1 Tổng quan về các Animation

1.1 Animation là gì?

-Định nghĩa cơ bản: Animation is a basic illusion of movement (tạm dịch : một ảo tưởng cơ bản về chuyển động).

-Animation được biết đến là nghệ thuật diễn hoạt hình ảnh có trong các chương trình quảng cáo, phim hoạt hình, các trò chơi điện tử... - đó là nghệ thuật làm cho hình ảnh được xuất hiện và chuyển động một cách sống động trên màn hình, dựa theo một câu chuyện, nội dung cụ thể nào đó. Người làm việc trong lĩnh vực Animation được gọi là Animator. Là ảnh động đánh mốc sự thay đổi vì đã đem màu sắc sinh động hơn, đa dạng hơn trong ứng dụng Android.

-Định nghĩa chuyên ngành: Animation là các phép biến đổi màu sắc, vị trí, kích thước hay hướng của một đối tượng trên màn hình theo thời gian. Android cung cấp các API hỗ trợ rất tốt cho việc áp dụng animation cho các thành phần UI và vẽ đồ họa 2D, 3D.

- Làm thế nào để tạo ra Animation :

- Frame by frame: tạo ảnh cho mỗi khung chuyển động
- Tweened animation: chỉ cần tạo ảnh cho khung bắt đầu và khung kết thúc, Flash sẽ tạo thêm nhiều khung ảnh ở giữa khoảng này (thay đổi các thuộc tính ảnh, kích thước, màu sắc...).

II. Định nghĩa các Animation

1. View Animation

- Chúng được sử dụng để làm các animations đơn giản như thay đổi kích thước, vị trí, xoay và kiểm soát độ trong suốt. Nó tương đối dễ dàng để thiết lập và cung cấp đủ khả năng để đáp ứng nhu cầu của nhiều ứng dụng, tuy nhiên nó vẫn có những hạn chế riêng.

- Một chuỗi hướng dẫn hoạt ảnh xác định hoạt ảnh tween, được xác định bằng mã XML hoặc Android. Cũng như việc xác định bối cảnh, tệp XML được khuyến nghị vì tệp này dễ đọc, có thể sử dụng lại và có thể hoán đổi hơn là mã hóa cứng hoạt ảnh. Trong ví dụ dưới đây, chúng tôi sử dụng XML. (Để tìm hiểu thêm về cách xác định hoạt ảnh trong mã ứng dụng của bạn, thay vì XML, hãy tham khảo AnimationSet lớp và các Animation lớp con khác.)

- Tập XML hoạt ảnh nằm trong res/anim/thư mục của dự án Android của bạn. Các tập tin phải có một phần tử gốc duy nhất: đây sẽ là một trong hai điều đơn `<alpha>`, `<scale>`, `<translate>`, `<rotate>`, yếu tố xen vào, hoặc `<set>` nguyên tố có chứa nhóm của các yếu tố này (có thể bao gồm một `<set>`).
- Một số giá trị, chẳng hạn như pivotX, có thể được chỉ định liên quan đến bản thân đối tượng hoặc liên quan đến phụ huynh. Đảm bảo sử dụng định dạng phù hợp cho những gì bạn muốn ("50" cho 50% so với gốc hoặc "50%" cho 50% so với chính nó).
- Bạn có thể xác định cách một phép chuyển đổi được áp dụng theo thời gian bằng cách gán một Interpolator. Android bao gồm một số lớp con Interpolator chỉ định các đường cong tốc độ khác nhau

2. Drawable Animations

- Chúng được sử dụng để làm các animations bằng các Drawables resource. Phương thức này rất hữu ích nếu bạn muốn tạo ra những điều dễ dàng hơn với Drawable resource và bitmaps
- Cách đơn giản nhất để tạo hoạt ảnh từng khung là xác định hoạt ảnh trong tệp XML, được đặt trong thư mục res / drawable / và đặt nó làm nền cho đối tượng View. Sau đó, gọi start() để chạy hoạt ảnh.
- Một AnimationDrawable được định nghĩa trong XML bao gồm một `<animation-list>` phần tử duy nhất và một loạt các `<item>` thẻ lồng nhau .

Thuộc tính XML	
<u>android:drawable</u>	Tham chiếu đến tài nguyên có thể vẽ được để sử dụng cho khung.
<u>android:duration</u>	Lượng thời gian (tính bằng mili giây) để hiển thị khung này.
<u>android:oneshot</u>	Nếu đúng, hoạt ảnh sẽ chỉ chạy một lần duy nhất và sau đó dừng lại.
<u>android:variablePadding</u>	Nếu đúng, hãy cho phép vùng đệm của có thể vẽ thay đổi dựa trên trạng thái hiện tại được chọn.

<u>android:visible</u>	Cung cấp trạng thái hiển thị ban đầu của vật có thể vẽ được; giá trị mặc định là false.
Phương pháp công khai	
void	<u>addFrame(Drawable frame, int duration)</u> Thêm khung vào hoạt ảnh
int	<u>getDuration(int i)</u>
<u>Drawable</u>	<u>getFrame(int index)</u>
int	<u>getNumberOfFrames()</u>
void	<u>inflate(Resources r, XmlPullParser parser, AttributeSet attrs, Resources.Theme theme)</u> Phóng to Bản vẽ này từ một tài nguyên XML được tạo kiểu tùy chọn theo chủ đề.
boolean	<u>isOneShot()</u>
boolean	<u>isRunning()</u> Cho biết hoạt ảnh hiện đang chạy hay không.
<u>Drawable</u>	<u>mutate()</u> Làm cho cái này có thể thay đổi được.
void	<u>run()</u> Phương thức này chỉ tồn tại cho mục đích triển khai và không nên được gọi trực tiếp.
void	<u>setOneShot(boolean oneShot)</u> Đặt hoạt ảnh sẽ phát một lần hay lặp lại.
boolean	<u>setVisible(boolean visible, boolean restart)</u> Đặt xem AnimationDrawable này có hiển thị hay không.
void	<u>start()</u> Bắt đầu hoạt ảnh từ khung hình đầu tiên, lặp lại nếu cần.
void	<u>stop()</u> Dừng hoạt ảnh ở khung hiện tại.
void	<u>unscheduleSelf(Runnable what)</u>

	Sử dụng cách <u>Callback</u> triển khai hiện tại để không lên lịch cho Drawable này.
--	--

3. Property Animations

- Được giới thiệu trong Android 3.0 (API level 11). Chúng được sử dụng để thay đổi thuộc tính của các đối tượng (View or non view objects). Chúng ta có thể xác định rõ các thuộc tính nhất định như translateX, TextScaleX của các đối tượng và thay đổi chúng. Các đặc tính khác của animations có thể thiết lập thời gian cho các animation, cho dù nó đảo ngược và bao nhiêu lần chúng ta muốn lặp lại nó.
- **Hoạt ảnh**

Lớp	Sự miêu tả
<u>ValueAnimator</u>	Công cụ định thời gian chính cho hoạt ảnh thuộc tính cũng tính toán các giá trị cho thuộc tính được hoạt ảnh. Nó có tất cả các chức năng cốt lõi là tính toán các giá trị hoạt ảnh và chứa chi tiết thời gian của từng hoạt ảnh, thông tin về việc liệu hoạt ảnh có lặp lại hay không, trình nghe nhận các sự kiện cập nhật và khả năng đặt các loại tùy chỉnh để đánh giá. Có hai phần đối với các thuộc tính hoạt ảnh: tính toán các giá trị hoạt ảnh và đặt các giá trị đó trên đối tượng và thuộc tính đang được làm động. <u>ValueAnimator</u> không thực hiện phần thứ hai, vì vậy bạn phải lắng nghe các cập nhật cho các giá trị được tính toán bởi <u>ValueAnimator</u> và sửa đổi các đối tượng mà bạn muốn tạo hoạt ảnh bằng logic của riêng bạn. Xem phần về <u>Tạo hoạt ảnh với ValueAnimator</u> để biết thêm thông tin.
<u>ObjectAnimator</u>	Một lớp con của <u>ValueAnimator</u> cho phép bạn thiết lập một đối tượng đích và thuộc tính đối tượng để tạo hoạt ảnh. Lớp này cập nhật thuộc tính tương ứng khi nó tính giá trị mới cho hoạt ảnh. Bạn muốn sử dụng <u>ObjectAnimator</u> hầu hết thời gian, vì nó làm cho quá trình tạo hiệu ứng các giá trị trên các đối tượng mục tiêu dễ dàng hơn nhiều. Tuy nhiên, đôi khi bạn muốn sử dụng <u>ValueAnimator</u> trực tiếp vì <u>ObjectAnimator</u> có một số hạn chế hơn, chẳng hạn như yêu cầu các phương thức truy cập cụ thể phải có mặt trên đối tượng đích.
<u>AnimatorSet</u>	Cung cấp cơ chế nhóm các hoạt ảnh lại với nhau để chúng chạy liên quan đến nhau. Bạn có thể đặt các hoạt ảnh để phát cùng nhau, tuân tự hoặc sau một khoảng thời gian cụ thể. Xem phần về <u>Biên đạo nhiều hoạt ảnh với Bộ hoạt hình</u> để biết thêm thông tin.

Lớp / Giao diện	Sự miêu tả
<u>IntEvaluator</u>	Trình đánh giá mặc định để tính toán các giá trị cho int các thuộc tính.
<u>FloatEvaluator</u>	Trình đánh giá mặc định để tính toán các giá trị cho float các thuộc tính.
<u>ArgbEvaluator</u>	Bộ đánh giá mặc định để tính toán các giá trị cho các thuộc tính màu được biểu thị dưới dạng giá trị thập lục phân.
<u>TypeEvaluator</u>	Một giao diện cho phép bạn tạo người đánh giá của riêng mình. Nếu bạn đang tạo hiệu ứng động một tài sản đối tượng đó là <i>không</i> một int, float hoặc màu sắc, bạn phải thực hiện các <u>TypeEvaluator</u> giao diện để xác định làm thế nào để tính toán giá trị dưới hình dạng một đối tượng sở hữu của. Bạn cũng có thể chỉ định một tùy chỉnh <u>TypeEvaluator</u> cho int, float và các giá trị màu là tốt, nếu bạn muốn để xử lý các loại khác với hành vi mặc định. Xem phần về <u>Sử dụng TypeEvaluator</u> để biết thêm thông tin về cách viết một đánh giá tùy chỉnh.
<u>AccelerateDecelerateInterpolator</u>	Một bộ nội suy có tốc độ thay đổi bắt đầu và kết thúc chậm nhưng tăng tốc ở giữa.
<u>AccelerateInterpolator</u>	Một bộ nội suy có tốc độ thay đổi bắt đầu chậm và sau đó tăng tốc.
<u>AnticipateInterpolator</u>	Một bộ nội suy có sự thay đổi bắt đầu lùi lại sau đó di chuyển về phía trước.
<u>AnticipateOvershootInterpolator</u>	Một bộ nội suy có sự thay đổi bắt đầu lùi lại, di chuyển về phía trước và vượt quá giá trị mục tiêu, sau đó cuối cùng quay trở lại giá trị cuối cùng.

<u>BounceInterpolator</u>	Một bộ nội suy có thay đổi bị trả lại ở cuối.
<u>CycleInterpolator</u>	Bộ nội suy có hoạt ảnh lặp lại trong một số chu kỳ xác định.
<u>DecelerateInterpolator</u>	Một bộ nội suy có tốc độ thay đổi bắt đầu nhanh chóng và sau đó giảm tốc.
<u>LinearInterpolator</u>	Một bộ nội suy có tốc độ thay đổi là không đổi.
<u>OvershootInterpolator</u>	Một bộ nội suy mà sự thay đổi của nó di chuyển về phía trước và vượt quá giá trị cuối cùng sau đó quay trở lại.
<u>TimeInterpolator</u>	Một giao diện cho phép bạn triển khai bộ nội suy của riêng mình.

4. Ripple Effect / Touch Feedback

- Ripple là gì?

Ripple là một làn sóng nhỏ giống như hoa văn, thường được hình thành trên bề mặt của một số chất lỏng khi bạn thả một thứ gì đó lên nó.

Hiệu ứng Ripple trong Android là gì?

Hiệu ứng Ripple cung cấp xác nhận trực quan tức thời tại điểm tiếp xúc khi người dùng tương tác với các phần tử giao diện người dùng.

Các phần tử giao diện người dùng này có thể là bất kỳ phần tử View nào.

Giống như - Bố cục, Nút, Chế độ xem văn bản, Chế độ xem danh sách, v.v.

Bất cứ khi nào người dùng nhấp hoặc chạm vào bất kỳ phần tử giao diện người dùng nào như nút, người dùng cần cung cấp một số loại xác nhận trực quan để người dùng biết rằng thao tác chạm hoặc nhấp của họ đã thành công. Tương tác đáp ứng khuyến khích khám phá sâu hơn về một ứng dụng bằng cách tạo ra các phản ứng màn hình kịp thời, hợp lý và thú vị với đầu vào của người dùng. Tuy nhiên, sự tương tác này không được gây mất tập trung cho người dùng. Theo truyền thống trong Android, điều này đã được xử lý bằng

cách sử dụng các ngăn kéo trong danh sách trạng thái để cung cấp các ngăn kéo có màu hoặc bóng mờ khác nhau cho biết rằng một điều khiển đang ở trạng thái chạm hoặc nhấn. Với Android Lollipop, một cơ chế phản hồi chạm mới đã được giới thiệu để cung cấp phản hồi chạm này và dựa trên khái niệm về gợn sóng trên ẩn dụ thẻ, đặc điểm nổi bật trong thiết kế Material design. Những gợn sóng này thực sự rất dễ thực hiện.

Làm thế nào để đạt được hiệu ứng Ripple?

Nó có thể đạt được bằng 2 cách:

- (a) Theo chương trình - bằng cách sử dụng lớp RippleDrawable .
- (b) Bằng XML - bằng cách sử dụng XML có thể vẽ cho thấy hiệu ứng gợn sóng để đáp ứng với các thay đổi trạng thái của Chế độ xem.

5. Reveal Effect

- Sơ lược

Trong các phiên bản Android 5.0 trở lên, một hiệu ứng hình ảnh động mới được thêm vào, tức là hiển thị hoạt ảnh. Hiệu ứng hiển thị rất phổ biến trong hệ thống, tương tự như hiệu ứng của gợn sóng. Nó mở rộng từ một điểm nhất định ra xung quanh hoặc tổng hợp từ xung quanh đến một điểm nhất định. Hiệu quả đạt được trong bài viết này được hiển thị bên dưới. Có thể được sử dụng trong hiệu ứng hoạt hình Chế độ xem trong Hoạt động hoặc Được sử dụng trong hoạt ảnh chuyển tiếp bước nhảy Hoạt động, như thể hiện trong hình sau, biểu đồ hiệu ứng hiển thị và ẩn của Chế độ xem được sử dụng:

- Sử dụng

Rất đơn giản để sử dụng hoạt ảnh tiết lộ. Android Sdk đã cung cấp cho chúng ta một lớp công cụ ViewAnimationUtils để tạo hoạt ảnh tiết lộ. Chỉ có một phương thức tĩnh trong ViewAnimationUtils, phương thức này createCircularReveal(View view, int centerX, int centerY, float startRadius, float endRadius) trả về một đối tượng hoạt hình Animator. ViewAnimationUtils.createCircularReveal() Phương pháp này có thể thêm hoạt ảnh vào vùng đã cắt để hiển thị hoặc ẩn chế độ xem. Chúng tôi chủ yếu sử dụng phương thức createCircularReveal, có bốn tham số:

- Tham số đầu tiên là dạng xem View thực hiện hoạt ảnh tiết lộ
- Tham số thứ hai là tọa độ x của tâm hình tròn hoạt ảnh so với hệ tọa độ của khung nhìn.
- Tham số thứ ba là tọa độ y của tâm vòng tròn hoạt ảnh so với hệ tọa độ của khung nhìn.
- Tham số thứ tư là bán kính bắt đầu của hình tròn hoạt ảnh và tham số thứ năm là bán kính kết thúc của hình tròn hoạt hình.

6. Transition Animation

- Các chuyển đổi Activity và Fragment trong Android 5.0 dựa trên một tính năng mới của Android có tên là Transitions. Khung chuyển tiếp này, ra đời từ 4.4, cung cấp một API rất thuận tiện để tạo hiệu ứng hoạt hình giữa các trạng thái giao diện người dùng khác nhau.

Khung chủ yếu dựa trên hai khái niệm: cảnh và chuyển tiếp.

Cảnh xác định trạng thái giao diện người dùng hiện tại

Chuyển tiếp xác định quá trình thay đổi hoạt ảnh giữa các cảnh khác nhau.

Mặc dù việc dịch chuyển đổi thành chuyển đổi có vẻ rất chính xác, nhưng tôi vẫn cảm thấy rằng việc sử dụng chuyển tiếp là không trực quan. Để hiểu rõ hơn về các địa điểm riêng lẻ sau đây, hãy trực tiếp sử dụng chuyển tiếp để biểu diễn chuyển đổi.

Khi một cảnh thay đổi, quá trình chuyển đổi chủ yếu chịu trách nhiệm về:

(1) Ghi lại trạng thái của từng Chế độ xem trong cảnh bắt đầu và cảnh kết thúc. (2) Tạo Animator dựa trên sự khác biệt của chế độ xem chuyển từ cảnh này sang cảnh khác.

Trong Android 5.0, Transition có thể được sử dụng để đạt được các hiệu ứng hoạt ảnh cực kỳ phức tạp khi chuyển đổi giữa Activity hoặc Fragment. Mặc dù trong các phiên bản trước, bạn có thể sử dụng Activity's overridePendingTransition () và FragmentTransaction's setCustomAnimation () để chuyển đổi giữa hoạt ảnh Activity hoặc Fragment, nhưng chúng chỉ giới hạn trong việc chuyển đổi hoạt ảnh của toàn bộ chế độ

xem. API Lollipop mới tiến thêm một bước nữa, cho phép một chế độ xem hoạt động khi vào hoặc thoát khỏi vùng chứa bộ cục của nó và thậm chí nó có thể chia sẻ một chế độ xem trong các hoạt động / phân đoạn khác nhau.

- Khung chuyển đổi chức năng chính hoặc kịch bản ứng dụng

- Bạn có thể thêm hoạt ảnh khi chuyển giữa các hoạt động
- Hoạt động chuyển tiếp giữa các yếu tố được chia sẻ
- Hoạt ảnh chuyển tiếp của các thành phần bộ cục trong hoạt động.

7. Animate View State Changes

-Là hoạt ảnh động trạng thái , với chế độ View để xem thực hiện hiệu ứng khi trạng thái thay đổi.

-Nó cũng giống như trước khi chúng ta đặt hiệu ứng nền ở các trạng thái khác nhau cho các nút (Button) thông qua các bộ chọn lọc (Selector)

State List chính là chìa khóa để bạn tạo ra các Trạng thái khác nhau cho các widget đó bằng cách định nghĩa vào một resource drawable như các bài đã học. Sau đó ứng dụng được thực thi, tùy vào ngữ cảnh lúc đó mà hệ thống sẽ quyết định dùng đến Trạng thái tương ứng cho widget mà bạn đã khai báo.

Có 2 dạng State List: Drawable State List và Color State List

-Bởi vì hoạt ảnh trạng thái chế độ View là hiệu ứng hoạt ảnh được tạo ra ở các trạng thái khác nhau của chế độ View, trước tiên chúng hãy xem trạng thái của nó bao gồm như sau:

State(trạng thái)	Mô tả
Android:constantSize	Kích thước bên trong của drawable sẽ không thay đổi; kích thước là tối đa của tất cả các trạng thái
Android:state_activated	Giá trị trạng thái cho StateListDrawable, định nghĩa trạng thái widget đó đang được kích hoạt hay không
Android:state_active	Giá trị trạng thái cho StateListDrawable
Android:state_checkable	Định danh trạng thái chỉ ra rằng đối tượng có thể hiện thị một con dấu
Android:state_checked	Định danh trạng thái cho biết đối tượng hiện đang được kiểm tra
android:state_enabled	Giá trị trạng thái cho StateListDrawable, được thiết lập khi chế độ xem được bật hay dùng để biểu diễn một widget có đang cho phép tương tác hay không

android:state_first	Giá trị trạng thái cho StateListDrawable, được đặt khi một chế độ xem hoặc có thể vẽ ở vị trí đầu tiên trong một tập hợp có thứ tự.
android:state_focused	Giá trị trạng thái cho StateListDrawable, được đặt khi một dạng xem có tiêu điểm đầu vào.
Android:state_last	Giá trị trạng thái cho StateListDrawable, được đặt khi một chế độ xem hoặc có thể vẽ ở vị trí cuối cùng trong một tập hợp có thứ tự.
android:state_middle	Giá trị trạng thái cho StateListDrawable, được đặt khi một chế độ xem hoặc có thể vẽ ở vị trí giữa trong một tập hợp có thứ tự.
android:state_pressed	Giá trị trạng thái cho StateListDrawable, được đặt khi người dùng nhấn xuống trong một dạng xem.
Android:state_selected	Giá trị trạng thái cho StateListDrawable, được đặt khi một chế độ xem (hoặc một trong các chế độ xem) hiện được chọn
android:state_single	Giá trị trạng thái cho StateListDrawable, được đặt khi một chế độ xem hoặc có thể vẽ được máy chủ của nó coi là “đơn”
android:state_window_focused	Giá trị trạng thái cho StateListDrawable, được đặt khi cửa sổ của chế độ xem có tiêu điểm đầu vào.
Android:visible	Cho biết liệu ban đầu có thể nhìn thấy phần có thể vẽ hay không.

8. AnimatedVectorDrawable

- Để hiểu cặn kẽ về hoạt ảnh vectơ trong Android, trước tiên, cần hiểu những kiến thức liên quan về SVG.

-SVG (Scalable Vector Graphics) là định dạng ảnh vector dùng để thể hiện các đối tượng đồ họa hai chiều và có hỗ trợ tương tác từ phía người dùng cũng như ảnh động. Vì là hình ảnh dạng vector nên chúng ta có thể hiển thị, co giãn (scale) thoái mái mà không làm giảm chất lượng hình ảnh.

-Hay dễ hiểu hơn, SVG là một ngôn ngữ sử dụng XML để mô tả các chương trình vẽ và đồ họa hai chiều, và định nghĩa của nó tuân theo các tiêu chuẩn W3C

SVG thuộc tiêu chuẩn mở và được quản lý bởi tổ chức World Wide Web Consortium, một tổ chức quản lý nhiều chuẩn khác như HTML, XHTML... Các tập tin có đuôi “.svg” được mặc định hiểu là tập tin SVG. SVG có thể phóng to thu nhỏ mọi kích cỡ mà không giảm chất lượng hình ảnh. Vì thế, nó được dùng nhiều trong các bản đồ, sơ đồ.

Điều gì làm cho đồ họa vector trở nên hấp dẫn đến mức rất nhiều công ty lớn trong ngành công nghệ cũng như hãng phần mềm thiết kế nhảy vào cùng phát triển? Bạn có thể tưởng

tượng rằng trong đồ họa vector, mọi đường thẳng, đường cong, hình tròn, hình chữ nhật... đều được vẽ ra đều dựa vào các điểm tọa độ. Các điểm này sẽ được nối với nhau trong không gian hai chiều để tạo nên các hình ảnh thực sự. Bởi vì tọa độ này chỉ mang tính tương đối so với hệ trục tại thời điểm vẽ nên 1 đơn vị trong đồ họa vector có thể là 10 pixel, 20 pixel hay 100 pixel.

Ưu điểm của SVG

Kích thước file nhỏ, dễ nén

Hình ảnh SVG, XML, chứa nhiều mảnh lặp đi lặp lại của văn bản, vì vậy chúng rất thích hợp cho các thuật toán nén lossless dữ liệu. Khi một hình ảnh SVG đã được nén bằng thuật toán tiêu chuẩn gzip, nó được gọi là một hình ảnh "svgz" và sử dụng phần mở rộng tên tập tin .svgz tương ứng.

Hiển thị đẹp trên màn hình retina

SVGs giống với tất cả các đồ họa vector, có thể được thu nhỏ đến kích thước bất kỳ mà không mất đi sự rõ ràng (trừ rất nhỏ). Nói cách khác, bạn có thể phóng to để một SVG tất cả các bạn muốn và họ sẽ luôn luôn nhìn sắc nét. Vì vậy, bạn không còn phải tạo ra một phiên bản 2x Retina phiên bản cho logo hình ảnh của bạn.

Có thể làm ảnh động

Sử dụng thẻ SVG để nhúng các hình ảnh trên trang web cho phép chúng ta định dạng một cách dễ dàng thông qua CSS, giống như cách làm với thẻ HTML thông thường. Ta có thể thay đổi thuộc tính đối tượng như màu nền, độ mờ đục, vị trí, chiều rộng,... Ngoài ra, ta cũng có thể thêm các hiệu ứng hình ảnh động ấn tượng bằng cách sử dụng sự kết hợp của các thư viện JS và CSS.

Hỗ trợ đầy đủ

Sau nhiều năm không tương thích trình duyệt, SVGs cuối cùng đã có thể. Chúng được hỗ trợ trong tất cả các trình duyệt hiện đại bao gồm IE9. Bạn thậm chí có thể sử dụng Fallbacks nếu bạn vẫn còn quan tâm đến IE8.

Thời gian tải tốt hơn

SVGs tuyệt vời cho thiết kế web, vì nó có độ phân giải vô hạn và kích thước file rất nhỏ. Nó có thể được nhúng trực tiếp vào một tài liệu HTML với thẻ svg, do trình duyệt không cần phải tải về đồ họa. Điều này có nghĩa rằng trang web của bạn sẽ được tải nhanh hơn!

Nhược điểm của SVG

SVG là ngôn ngữ không được thiết kế để sửa chữa trực tiếp trên mã nguồn. Để tạo ra các hình ảnh SVG nói chung, cần dùng các công cụ hỗ trợ.

Nói về Android SVG Animation, AnimatedVectorDrawable sử dụng ObjectAnimator và AnimatorSet để thúc đẩy gradient của thuộc tính VectorDrawable để tạo hiệu ứng hoạt hình.

Thông thường bạn có thể xác định được hình ảnh vector thông qua các tệp XML ở res/drawable hoặc res/anim. Thêm hoạt ảnh vào hình ảnh vector để thêm hoạt ảnh vào các thuộc tính của và các phần tử. Phần tử xác định tập hợp đường dẫn hoặc nhóm con và phần tử xác định đường dẫn sẽ được vẽ.

VectorDrawable nói chung là một tệp XML được xác định cho thẻ gốc. Các phần tử vector, group , clip-path, path và các phần tử có các thuộc tính riêng có thể phát hoạt ảnh.

Vector	Mô tả
android:name	Tên vector
android:width	Chiều rộng bên trong của một sơ đồ vector, hỗ trợ tất cả các kích thước bởi hệ thống Android, thông thường sử dụng dp
android:height	Chiều cao bên trong của biểu đồ vector
android:viewportWidth	Chiều rộng của chế độ xem vector, chế độ xem canvas ảo được vẽ bởi dữ liệu đường dẫn vector
android:viewportHeight	Chiều cao của chế độ xem vector
android:tint	Minh họa vector về màu sắc. Mặc định là không có màu
android:tintMode	Chế độ hòa trộn Porter-Duff của màu vector. Giá trị mặc định là src_in.
android:autoMirrored	Thiết lập có tự động phản chiếu ảnh khi hệ thống bố trí từ phải sang trái hay không
android:alpha	Độ trong suốt của ảnh

Group	Đặc điểm
android:name	Xác định tên của group
android:rotation	Xác định xem đường dẫn của nhóm được xoay bao nhiêu độ
android:pivotX	Xác định điểm tham chiếu X khi mở rộng và xoay nhóm. Giá trị này được chỉ liên quan đến giá trị khung nhìn của vector
android:pivotY	Xác định điểm tham chiếu Y khi mở rộng và xoay nhóm. Giá trị được chỉ định
android:scaleX	Xác định hệ số thu phóng của trục X
android:scaleY	Xác định hệ số thu phóng của trục Y
android:translateX	Xác định độ dời của trục X chuyển động. Nó được chỉ định liên quan đến giá trị khung nhìn của vector
android:translateY	Xác định độ dời của trục Y chuyển động. Nó được chỉ định liên quan đến giá trị khung nhìn của vector

<path>	Mô tả
android:name	Xác định tên của path để đường dẫn có thể được tham chiếu theo tên ở những nơi khác
android:pathData	Thông tin đường dẫn giống như phần tử d trong SVG
android:fillColor	Xác định màu của path đã tô, nếu không được xác định, thì path này sẽ không được tô
android:strokeColor	Xác định cách vẽ đường viền của path
android:strokeWidth	Xác định độ dày của đường viền của path
android:strokeAlpha	Xác định độ trong suốt của đường viền của path
android:fillAlpha	Xác định độ trong suốt của màu đường tô màu
android:trimPathStart	Tỷ lệ cắt bớt đường dẫn từ đầu đường dẫn, phạm vi giá trị là từ 0 đến 1; lưu ý rằng hoạt ảnh từ một nửa đến đầu là từ-0,5-đến-0
android:trimPathEnd	Tỷ lệ cắt bớt đường dẫn từ cuối đường dẫn, phạm vi giá trị là từ 0 đến 1; lưu ý rằng hoạt ảnh từ một nửa đến cuối là từ-0,5-đến-1,0
android:trimPathOffset	Đặt phạm vi chặn đường dẫn, phạm vi giá trị từ 0 đến 1
android:strokeLineCap	Đặt hình dạng của nắp đường dẫn, giá trị là round, square
android:strokeLineJoin	Đặt ché độ kết nối tại điểm giao nhau của đường dẫn, giá trị là miter, round, bevel.
android:strokeMiterLimit	Đặt giới hạn trên của góc xiên

clip-path	Mô tả
android:name	Xác định tên của đường dẫn clip-path
android:pathData	Thông tin của clip-path

9. Constraint Animation / ConstraintSet

- Là layout mặc định kể từ Android Studio 3.0, ConstraintLayout giúp cho việc thiết kế các layouts phức tạp trở nên đơn giản hơn bằng cách cho phép các views kết nối với nhau thông qua các ràng buộc (constraints) dựa trên mối quan hệ giữa các views khác nhau, và quan trọng hơn, ConstraintLayout hướng tới việc thiết kế giảm các views lồng nhau – điều này sẽ làm tăng hiệu suất thực thi cho các tập tin layout.

Định vị các views

ConstraintLayout cung cấp các thuộc tính cho phép chúng ta định vị view hiện tại một cách dễ dàng. Các thuộc tính được mô tả như bảng sau:

Các thuộc tính	Mô tả
<i>layout_constraintTop_toTopOf</i>	Ràng buộc phần trên (top) của view hiện tại đến phần trên của view khác.
<i>layout_constraintTop_toBottomOf</i>	Ràng buộc phần trên của view hiện tại đến phần dưới (bottom) của view khác.
<i>layout_constraintBottom_toTopOf</i>	Ràng buộc phần dưới của view hiện tại đến phần trên của view khác.
<i>layout_constraintBottom_toBottomOf</i>	Ràng buộc phần dưới của view hiện tại đến phần dưới của view khác.
<i>layout_constraintLeft_toTopOf</i>	Ràng buộc bên trái (left) của view hiện tại đến phần trên của view khác.

<i>layout_constraintLeft_toBottomOf</i>	Ràng buộc bên trái của view hiện tại đến phần dưới của view khác.
<i>layout_constraintLeft_toLeftOf</i>	Ràng buộc bên trái của view hiện tại đến bên trái của view khác.
<i>layout_constraintLeft_toRightOf</i>	Ràng buộc bên trái của view hiện tại đến bên phải (right) của view khác.
<i>layout_constraintRight_toTopOf</i>	Ràng buộc bên phải của view hiện tại đến phần trên của view khác.
<i>layout_constraintRight_toBottomOf</i>	Ràng buộc bên phải của view hiện tại đến phần dưới của view khác.
<i>layout_constraintRight_toLeftOf</i>	Ràng buộc bên phải của view hiện tại đến bên trái của view khác.
<i>layout_constraintRight_toRightOf</i>	Ràng buộc bên phải của view hiện tại đến bên phải của view khác.
<i>Start, End</i>	<p>Chúng ta có thể dùng <i>Start</i> để thay thế cho <i>Left</i> và dùng <i>End</i> để thay thế cho <i>Right</i>. Lưu ý rằng, khi dùng <i>Start</i> thì phải dùng <i>Start</i> hay <i>End</i> tương ứng, không được dùng <i>Start</i> kết hợp với <i>Left</i> hay <i>Right</i>.</p> <p>Ví dụ <i>layout_constraintRight_toLeftOf</i> tương đương với <i>layout_constraintEnd_toStartOf</i></p>
<i>layout_constraintHorizontal_bias</i>	Định vị view theo trục ngang màn hình.
<i>layout_constraintVertical_bias</i>	Định vị view theo trục dọc màn hình.

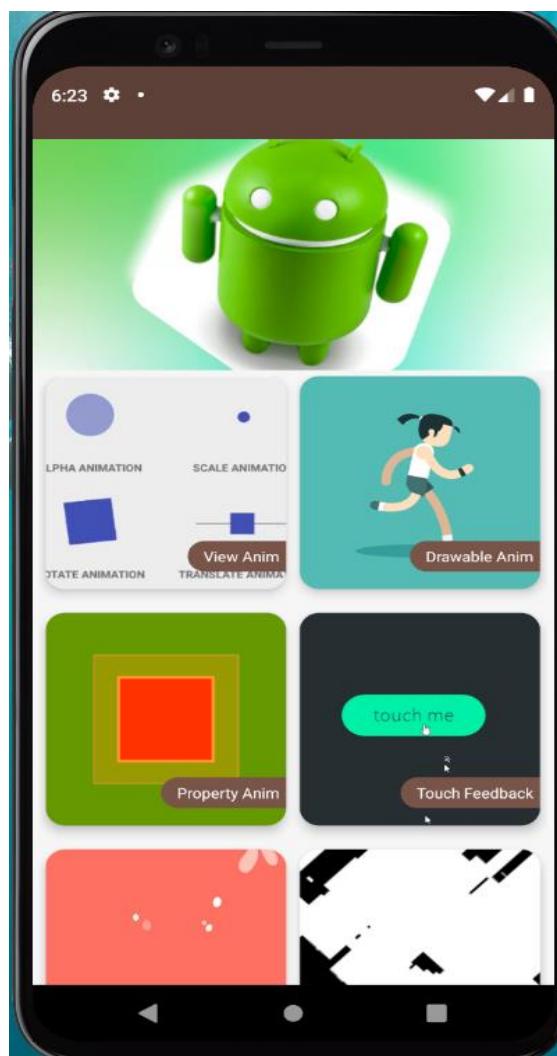
Kích cỡ các views

Giống như các views, ConstraintLayout cung cấp hai thuộc tính là *layout_width* và *layout_height* cho phép chúng ta điều khiển kích cỡ của ConstraintLayout.

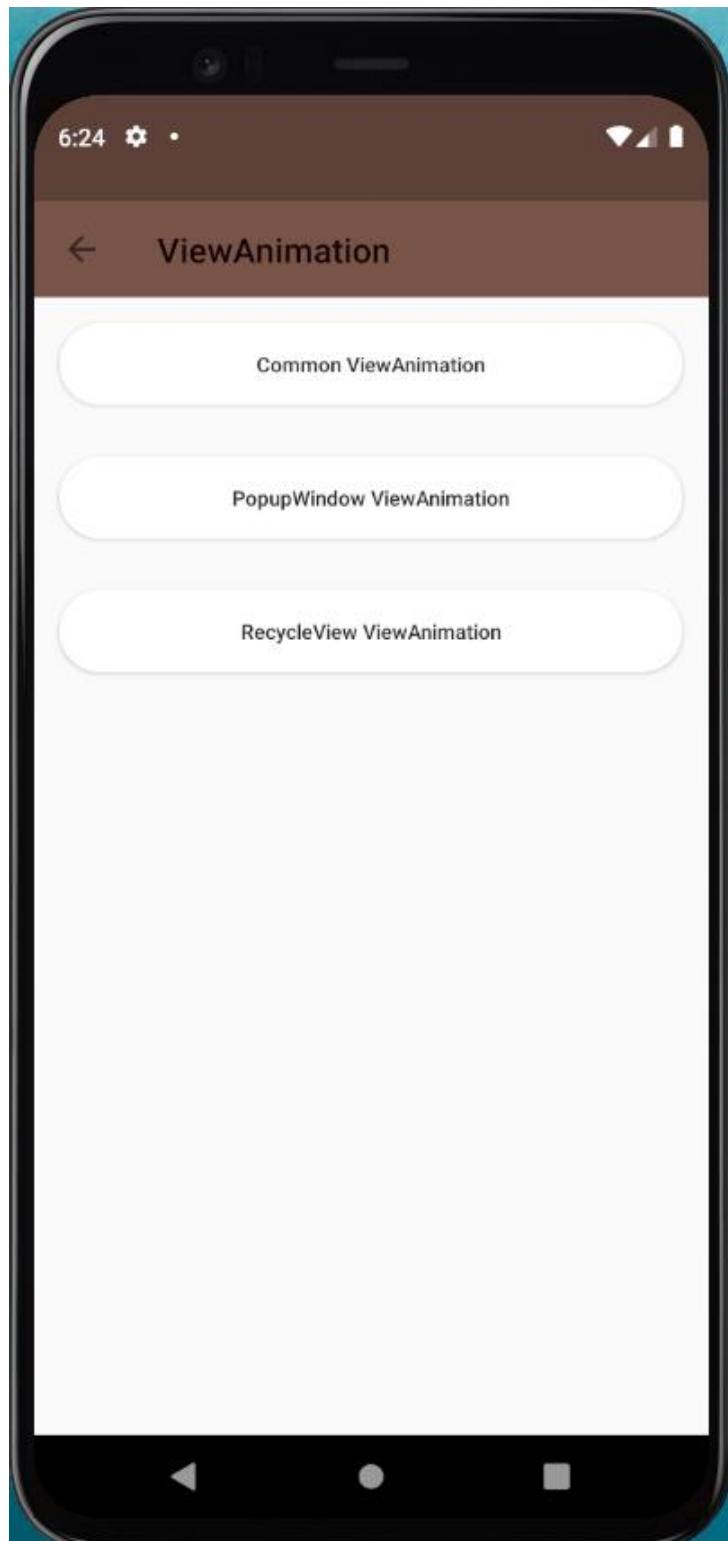
- Có 3 kiểu giá trị phổ biến dùng cho *layout_width* và *layout_height* như sau:
 - *Tùy ý, ví 18dp hay 133dp*: kích cỡ layout là cố định theo giá trị cho trước
 - *wrap_content*: điều chỉnh kích cỡ layout vừa khớp với nội dung bên trong nó
 - *match_parent*: mở rộng kích cỡ layout khớp kích cỡ của view cha hay view chứa nó

III. CHƯƠNG TRÌNH DEMO

1. Giao diện main



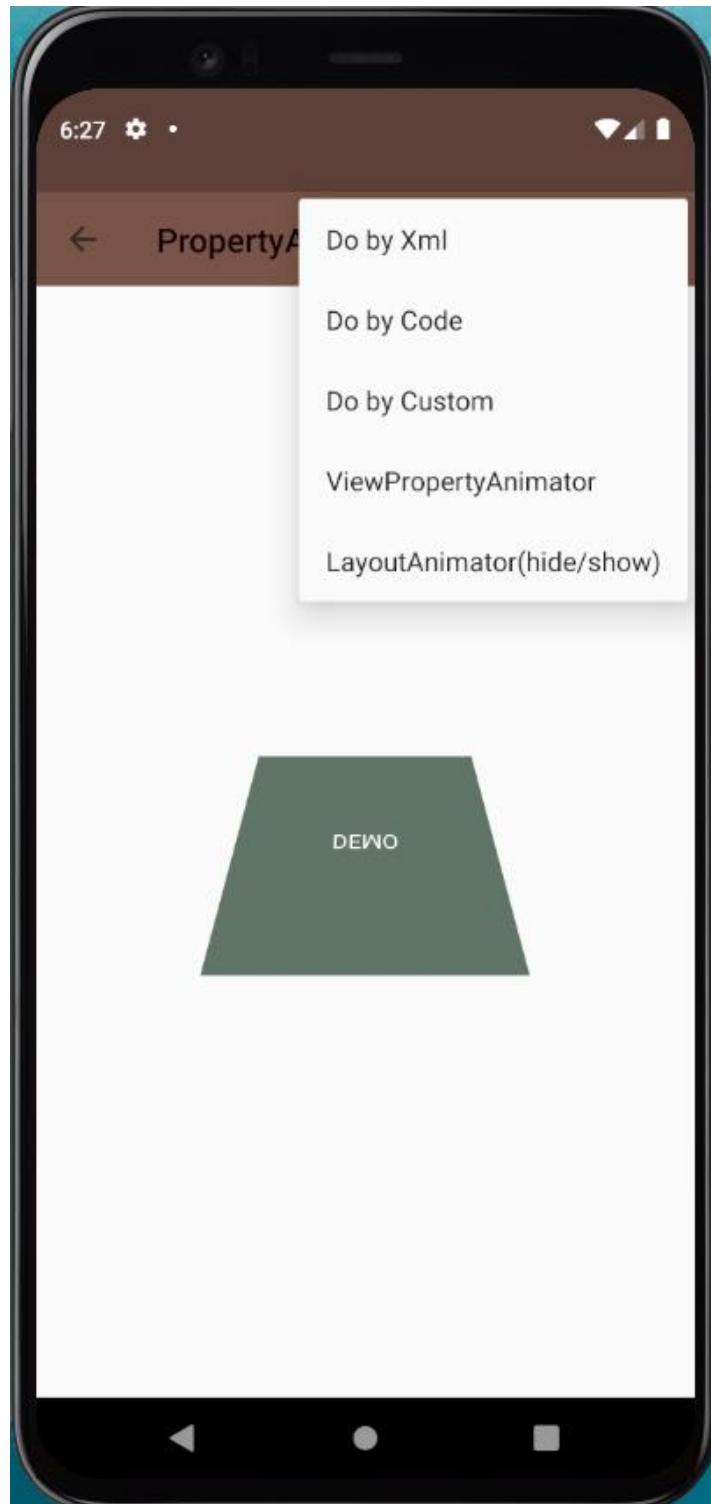
2. View Animation



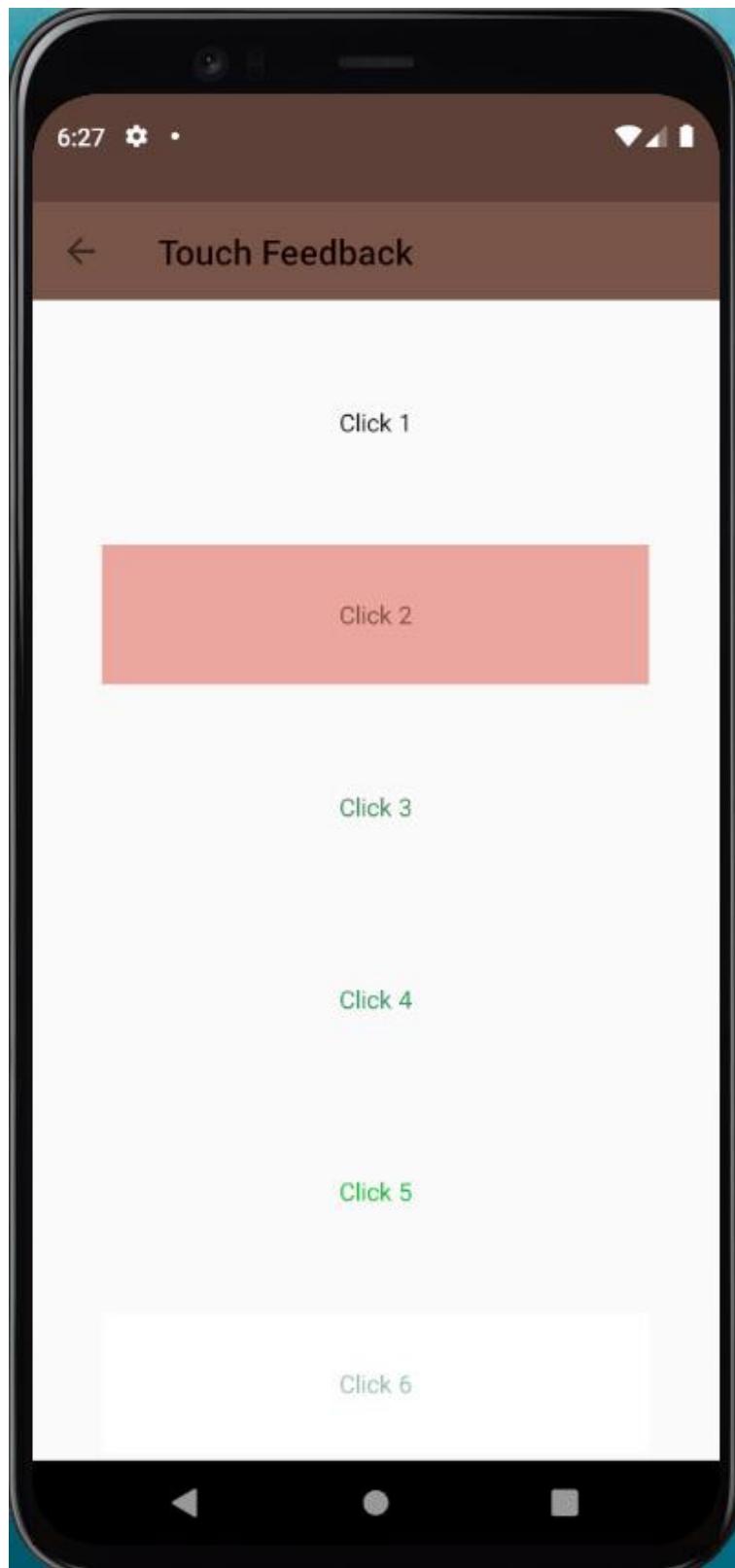
3. Drawable Animation



4. Property Animation



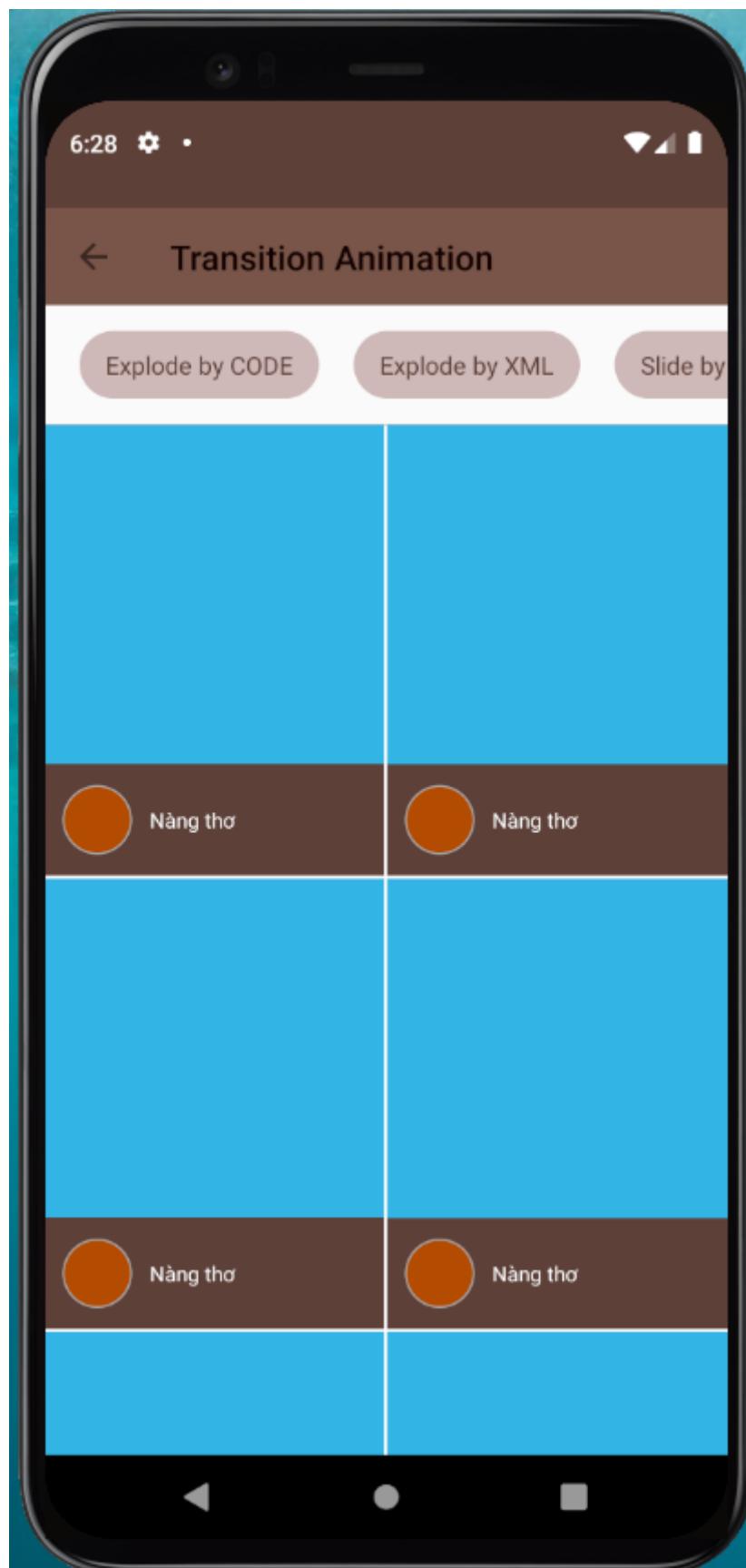
5. Ripple Effect / Touch Feedback



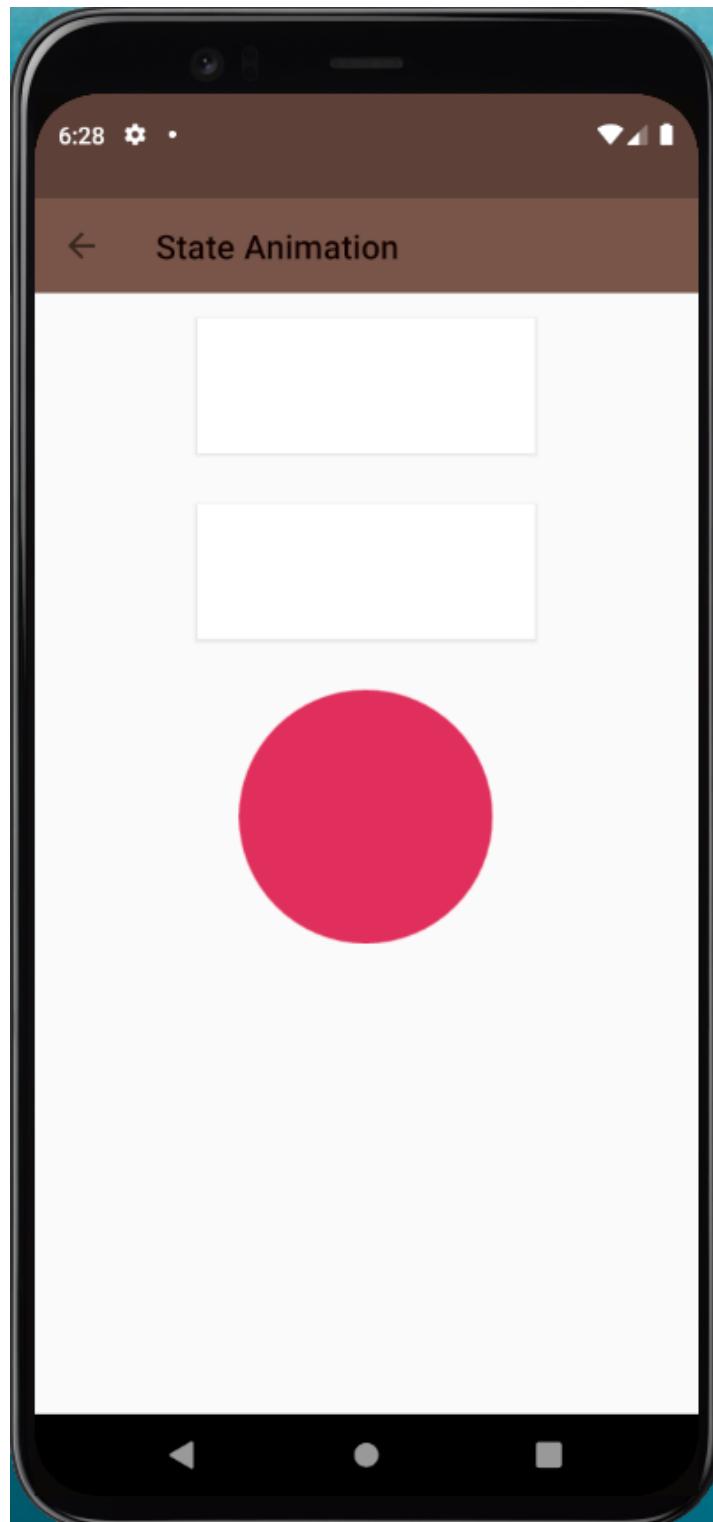
6. Reveal Effect



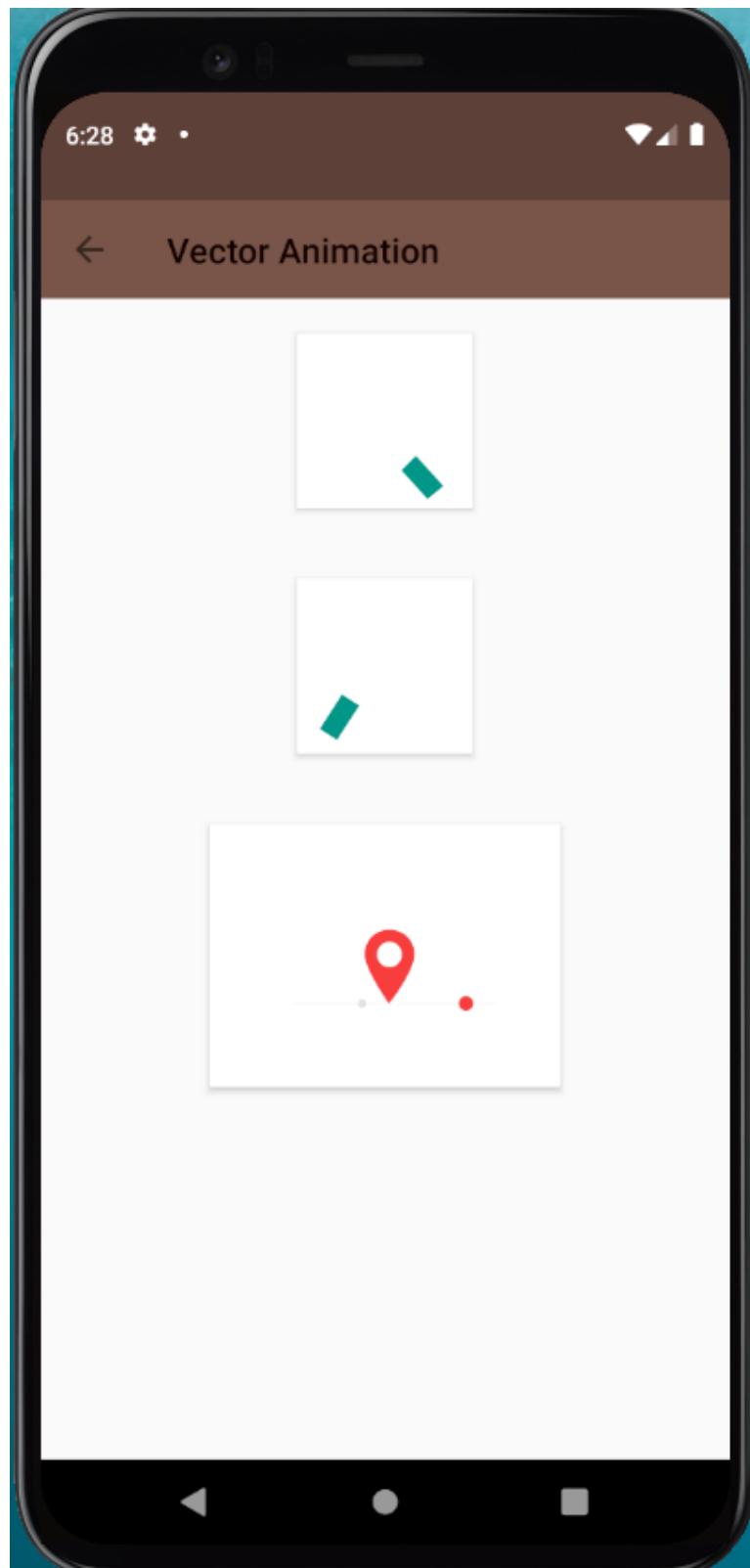
7. Transition Animation



8. Animate View State Changes



9. AnimatedVectorDrawable



TÀI LIỆU THAM KHẢO

1. <https://viblo.asia/p/tat-tan-tat-ve-animation-trong-android-63vKjQrV52R>
2. youtube.com/watch?v=2Mpm3BaWubQ&t=1325s
3. link git : [khaheo2528/AnimationLTTBDD \(github.com\)](https://github.com/khaheo2528/AnimationLTTBDD)

Phân công công việc:

Sinh viên	Công việc
Nguyễn Thanh Qui	+ Ripple Effect / Touch Feedback + Reveal Effect + Transition Animation + Làm báo cáo hoàn chỉnh
Lê Thị Ngọc Mai	+ View Animation + Drawable Animation + Property Animation +Tìm hiểu về tổng quát, kiểm tra source code
Võ Sỹ Khá	+ Animate View State Changes +AnimatedVectorDrawable +Thiết kế demo