/\*

\* Copyright 2012 The Android Open Source Project

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package com.example.android.animationsdemo;

import android.content.Context;

import android.content.res.TypedArray;

import android.graphics.Canvas;

import android.graphics.Rect;

import android.graphics.drawable.Drawable;

import android.util.AttributeSet;

import android.widget.ImageButton;

/\*\*

\* An image button that uses a blue highlight (@link android.R.attr.selectableItemBackground} to

\* indicate pressed and focused states.

\*/

public class TouchHighlightImageButton extends ImageButton {

/\*\*

\* The highlight drawable. This generally a {@link android.graphics.drawable.StateListDrawable}

\* that's transparent in the default state, and contains a semi-transparent overlay

\* for the focused and pressed states.

\*/

private Drawable mForegroundDrawable;

/\*\*

\* The cached bounds of the view.

\*/

private Rect mCachedBounds = new Rect();

public TouchHighlightImageButton(Context context) {

super(context);

init();

}

public TouchHighlightImageButton(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public TouchHighlightImageButton(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

init();

}

/\*\*

\* General view initialization used common to all constructors of the view.

\*/

private void init() {

// Reset default ImageButton background and padding.

setBackgroundColor(0);

setPadding(0, 0, 0, 0);

// Retrieve the drawable resource assigned to the android.R.attr.selectableItemBackground

// theme attribute from the current theme.

TypedArray a = getContext()

.obtainStyledAttributes(new int[]{android.R.attr.selectableItemBackground});

mForegroundDrawable = a.getDrawable(0);

mForegroundDrawable.setCallback(this);

a.recycle();

}

@Override

protected void drawableStateChanged() {

super.drawableStateChanged();

// Update the state of the highlight drawable to match

// the state of the button.

if (mForegroundDrawable.isStateful()) {

mForegroundDrawable.setState(getDrawableState());

}

// Trigger a redraw.

invalidate();

}

@Override

protected void onDraw(Canvas canvas) {

// First draw the image.

super.onDraw(canvas);

// Then draw the highlight on top of it. If the button is neither focused

// nor pressed, the drawable will be transparent, so just the image

// will be drawn.

mForegroundDrawable.setBounds(mCachedBounds);

mForegroundDrawable.draw(canvas);

}

@Override

protected void onSizeChanged(int w, int h, int oldw, int oldh) {

super.onSizeChanged(w, h, oldw, oldh);

// Cache the view bounds.

mCachedBounds.set(0, 0, w, h);

}

}