Picasso

[Трансформация](http://developer.alexanderklimov.ru/android/library/picasso.php#transformation)

Как известно, знаменитый художник Пабло Пикассо любил котов. Есть фотография, подтверждающая его любовь.



Художник нарисовал несколько картин с котом. Одна из них.



А также есть библиотека **Picasso**, предназначенная для асинхронной подгрузки изображений котов из сети, ресурсов или файловой системы, их кэширования и отображения.

Сам проект на [Гитхабе](https://github.com/square/picasso).

[Официальная страница](http://square.github.io/picasso/) библиотеки от авторов, где можно найти ссылки на документацию и примеры.

В Android Studio прописываем зависимость в Gradle (проверяйте номер свежей версии).

dependencies {

compile 'com.squareup.picasso:picasso:2.5.2'

}

Вот как просто загрузить картинку из сети:

Picasso.with(context)

.load(url)

.placeholder(R.drawable.user\_placeholder)

.error(R.drawable.user\_placeholder\_error)

.into(imageView);

Вы указываете адрес картинки (url), заглушку (placeholder), заглушку для ошибки после трёх неудачных попыток загрузки (error) и в методе **into()** указываете компонент **ImageView**, в который загружаете изображение.

При загрузке картинка кэшируется и при повторном запросе на скачивание библиотека может достать картинку из кэша, а не скачивать из интернета, что ускоряет работу приложения. Если кэш будет переполнен или удалён пользователем, то картина снова скачается из сети. Очень удобно.

Если вы храните большие картинки в ресурсах или на внешнем накопителе, то рекомендуется использовать отдельный процесс для загрузки. Библиотека уже настроена на работу в асинхронном режиме, поэтому вы можете использовать её и в этих случаях.

// из ресурсов

Picasso.with(context).load(R.drawable.landing\_screen).into(imageView1);

// из внешнего накопителя

Picasso.with(context).load(new File(...)).into(imageView2);

Не забывайте про метод библиотеки **fit()**, который уменьшает размер картинки перед размещением в **ImageView**. Это полезно для экономии ресурсов, если вам в реальности нужна маленькая картинка, а не оригинал.

Трансформация

У библиотеки есть специальный метод **transform()** для манипуляций с изображениями. Вам нужно создать собственную реализацию и передать её в метод. Например, на сайте разработчика приведён пример кадрирования изображения, когда создаётся квадратная картинка по наименьшей стороне.

Сначала создаётся отдельный класс с интерфейсом **Transformation**, который требует реализовать два метода **transform()** и **key()**.

package ru.alexanderklimov.testapplication;

import android.graphics.Bitmap;

import com.squareup.picasso.Transformation;

public class CropSquareTransformation implements Transformation {

@Override public Bitmap transform(Bitmap source) {

int size = Math.min(source.getWidth(), source.getHeight());

int x = (source.getWidth() - size) / 2;

int y = (source.getHeight() - size) / 2;

Bitmap result = Bitmap.createBitmap(source, x, y, size, size);

if (result != source) {

source.recycle();

}

return result;

}

@Override public String key() { return "square()"; }

}

Для наглядности разместим два компонента **ImageView** и посмотрим на результат после нажатия на кнопку.

public class MainActivity extends ActionBarActivity {

private ImageView mDrawableTransformedImage;

private ImageView mDrawableTransformedImage2;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mDrawableTransformedImage = (ImageView) findViewById(R.id.imageView);

mDrawableTransformedImage2 = (ImageView) findViewById(R.id.imageView2);

}

public void onClick(View v) {

Picasso.with(this)

.load(R.drawable.cat\_bottom)

.transform(new CropSquareTransformation())

.into(mDrawableTransformedImage);

Picasso.with(this)

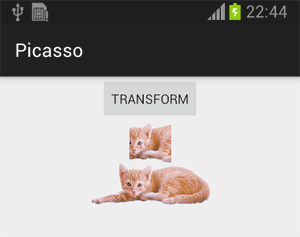
.load(R.drawable.cat\_bottom)

//.transform(new CropSquareTransformation())

.into(mDrawableTransformedImage2);

}

}



Другой пример трансформации - перекраска изображения. В Android 5.0 появилась возможность перекрашивать значки встроенными средствами. Для старых устройств можно воспользоваться библиотекой (взято [отсюда](http://ptrprograms.blogspot.ru/2014/08/programmatically-coloring-drawables.html)). Создадим новый класс:

package ru.alexanderklimov.testapplication;

import android.content.Context;

import android.content.res.Resources;

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.PorterDuff;

import android.graphics.drawable.BitmapDrawable;

import com.squareup.picasso.Transformation;

public class ColorTransformation implements Transformation {

private int color = 0;

public ColorTransformation() {

}

public ColorTransformation(int color) {

setColor(color);

}

public void setColor(int color) {

this.color = color;

}

public void setColorFromRes(Context context, int colorResId) {

setColor(context.getResources().getColor(colorResId));

}

public int getColor() {

return color;

}

@Override

public Bitmap transform(Bitmap source) {

if (color == 0) {

return source;

}

BitmapDrawable drawable = new BitmapDrawable(Resources.getSystem(), source);

Bitmap result = Bitmap.createBitmap(drawable.getIntrinsicWidth(),

drawable.getIntrinsicHeight(), Bitmap.Config.ARGB\_8888);

Canvas canvas = new Canvas(result);

drawable.setBounds(0, 0, drawable.getIntrinsicWidth(), drawable.getIntrinsicHeight());

drawable.setColorFilter(color, PorterDuff.Mode.SRC\_IN);

drawable.draw(canvas);

drawable.setColorFilter(null);

drawable.setCallback(null);

if (result != source) {

source.recycle();

}

return result;

}

@Override

public String key() {

return "DrawableColor:" + color;

}

}

Пробуем на этой же картинке.

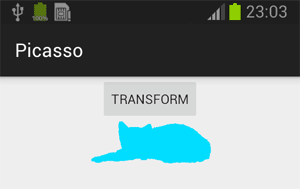
Picasso.with(this)

.load(R.drawable.cat\_bottom)

.transform(new ColorTransformation(getResources()

.getColor(android.R.color.holo\_blue\_bright)))

.into(mDrawableTransformedImage);



Круглый аватар

Ещё один пример, позволяющий создавать круглые картинки, которые можно использовать в качестве аватаров.

package ru.alexanderklimov.as21;

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.PorterDuff;

import android.graphics.PorterDuffXfermode;

import android.graphics.Rect;

import com.squareup.picasso.Transformation;

public class CircularTransformation implements Transformation {

private int mRadius = 10;

public CircularTransformation(final int radius) {

this.mRadius = radius;

}

@Override

public Bitmap transform(Bitmap source) {

Bitmap output = Bitmap.createBitmap(source.getWidth(), source.getHeight(),

Bitmap.Config.ARGB\_8888);

Canvas canvas = new Canvas(output);

final Paint paint = new Paint();

final Rect rect = new Rect(0, 0, source.getWidth(), source.getHeight());

paint.setAntiAlias(true);

paint.setFilterBitmap(true);

paint.setDither(true);

canvas.drawARGB(0, 0, 0, 0);

paint.setColor(Color.parseColor("#BAB399"));

if (mRadius == 0) {

canvas.drawCircle(source.getWidth() / 2 + 0.7f, source.getHeight() / 2 + 0.7f,

source.getWidth() / 2 - 1.1f, paint);

} else {

canvas.drawCircle(source.getWidth() / 2 + 0.7f, source.getHeight() / 2 + 0.7f,

mRadius, paint);

}

paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC\_IN));

canvas.drawBitmap(source, rect, rect, paint);

if (source != output) {

source.recycle();

}

return output;

}

@Override

public String key() {

return "circular" + String.valueOf(mRadius);

}

}

Разместим на экране **ImageView** и зададим размер аватара 150 пикселей.

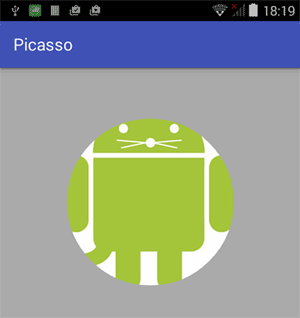
ImageView avatarImageView = (ImageView)findViewById(R.id.imageView);

Picasso.with(this)

.load("http://developer.alexanderklimov.ru/android/images/android\_cat.jpg")

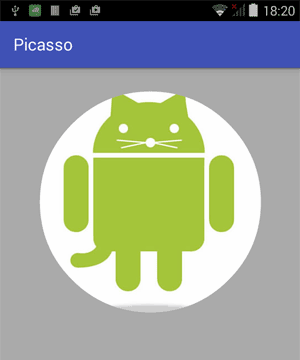
.transform(new CircularTransformation(150))

.into(avatarImageView);



Если задать нулевой радиус, то будут использоваться размеры картинки для вычисления нужных размеров.

.transform(new CircularTransformation(0))



Техника, используемая в примере, не совсем эффективна. Для подобных целей рекомендуется использовать шейдеры. Переписанный пример класса.

package ru.alexanderklimov.as21;

import android.graphics.Bitmap;

import android.graphics.BitmapShader;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.PorterDuff;

import android.graphics.PorterDuffXfermode;

import android.graphics.Rect;

import android.graphics.Shader;

import com.squareup.picasso.Transformation;

public class CircularTransformation implements Transformation {

public CircularTransformation() {

}

@Override

public Bitmap transform(final Bitmap source) {

final Paint paint = new Paint();

paint.setAntiAlias(true);

paint.setShader(new BitmapShader(source, Shader.TileMode.CLAMP, Shader.TileMode.CLAMP));

final Bitmap output = Bitmap.createBitmap(source.getWidth(), source.getHeight(), Bitmap.Config.ARGB\_8888);

final Canvas canvas = new Canvas(output);

canvas.drawCircle(source.getWidth() / 2, source.getHeight() / 2, source.getWidth() / 2, paint);

if (source != output)

source.recycle();

return output;

}

@Override

public String key() {

return "circle";

}

}

В этом примере используется пустой конструктор без параметров, который равносилен значению 0 из предыдущего примера. Вы можете доработать пример, добавив возможность указания радиуса, как это реализовано в том примере.

Новые возможности

В [статье](http://lucasr.org/2014/09/23/new-features-in-picasso/) описаны новые возможности для версии 2.4.

Появилась возможность встроить собственные обработчики запросов, если у вас используется нестандартная схема вместо http, res и т.п.

Появилась возможность управления приоритетами при запросах, чтобы скачать какое-то изображение раньше других.

[Picasso для Palette](http://jakewharton.com/coercing-picasso-to-play-with-palette/)

[Extracting colors from images: Integrating Picasso and Palette](https://medium.com/android-news/extracting-colors-from-images-integrating-picasso-and-palette-b9ba45c9c418#.fh83p3jz2)

Чтобы протестировать библиотеку в боевых условиях, я написал приложение [Коты Василия Ложкина](https://play.google.com/store/apps/details?id=ru.vasya_lozhkin.cats). Результатом доволен.