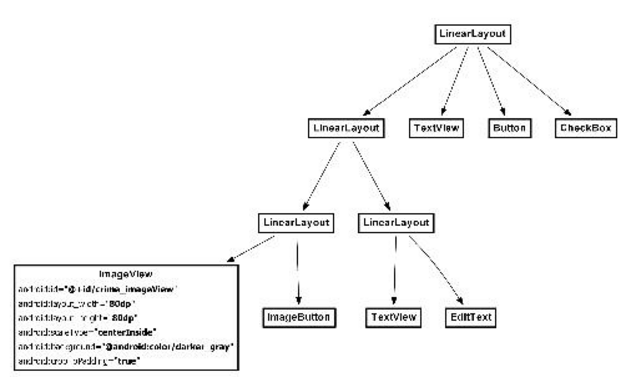
Обновление уровня модели    **343**



**Рис. 20.7.**CrimeFragment с новым виджетом ImageView

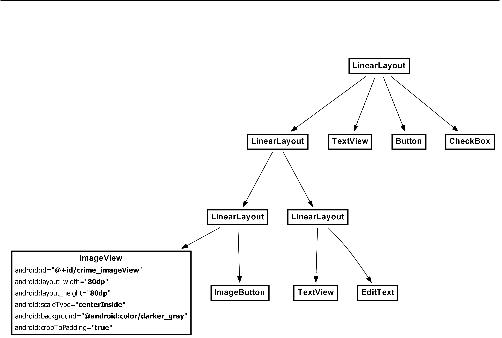
**Добавление ImageView**

Вернитесь к файлу layout/fragment\_crime.xml и добавьте виджетImageView, показанный на рис. 20.8.



**Рис. 20.8.**Макет CrimeFragment с ImageView (layout/fragment\_crime.xml)

Также нам понадобится виджет ImageView в альбомном макете (рис. 20.9).



**Рис. 20.9.**Альбомный макет с ImageView(layout-land/fragment\_crime.xml)

В файле CrimeFragment.java определите новое поле и получите ссылку наImageView

в методе onCreateView(…).

**Листинг 20.11.**Подготовка ImageView (CrimeFragment.java)

public class CrimeFragment extends Fragment {

...

private ImageButton mPhotoButton;**private ImageView mPhotoView;**

...

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup parent,

Bundle savedInstanceState) {

...

mPhotoButton.setOnClickListener(new View.OnClickListener() { public void onClick(View v) {

// Launch the camera activity

Intent i = new Intent(getActivity(), CrimeCameraActivity.class); startActivityForResult(i, REQUEST\_PHOTO);

}

});

**mPhotoView = (ImageView)v.findViewById(R.id.crime\_imageView);**

...

}

Чтобы убедиться в том, что виджет ImageView находится на положенном месте, просмотрите макет или запустите приложение CriminalIntent.

http://www.studfiles.ru/html/2706/1137/html_K2MGkhC3f0.fGLx/htmlconvd-G_3cW2345x1.jpg

**Обработка изображения**

Для вывода изображения на виджете ImageView потребуется некоторая предварительная обработка, потому что файлы, полученные с камеры, могут быть просто огромными. С каждым годомфирмы-производителиустанавливают на своих телефонах все большие и современные камеры. Технический прогресс радует пользователей, но создает проблемы программистам.

На момент написания книги самые современные телефоны на платформе Android оснащались 8-мегапиксельнымикамерами. Изображения такого размера мгновенно израсходуют всю доступную память, поэтому нам понадобится код, который будет масштабировать изображение перед загрузкой, а также код стирания ненужных изображений.

**Добавление масштабированных фотографий**

**вImageView**

Впакете com.bignerdranch.android.criminalintentсоздайте новый класс с именем

PictureUtils. Добавьте в файлPictureUtils.java метод, масштабирующий изображение по размеру стандартного экрана устройства.

**Листинг 20.12.**Добавление класса PictureUtils (PictureUtils.java)

public class PictureUtils {

**/\*\***

**\*Получение объекта BitmapDrawable по данным локального файла,**

**\*масштабированного по текущим размерам окна.**

**\*/**

**@SuppressWarnings("deprecation")**

**public static BitmapDrawable getScaledDrawable(Activity a, String path) { Display display = a.getWindowManager().getDefaultDisplay();**

**float destWidth = display.getWidth(); float destHeight = display.getHeight();**

**// Чтение размеров изображения на диске**

**BitmapFactory.Options options = new BitmapFactory.Options(); options.inJustDecodeBounds = true; BitmapFactory.decodeFile(path, options);**

**float srcWidth = options.outWidth; float srcHeight = options.outHeight;**

**int inSampleSize = 1;**

**if (srcHeight > destHeight || srcWidth > destWidth) { if (srcWidth > srcHeight) {**

**inSampleSize = Math.round(srcHeight / destHeight); } else {**

**inSampleSize = Math.round(srcWidth / destWidth);**

**}**

**}**

**options = new BitmapFactory.Options(); options.inSampleSize = inSampleSize;**

**Bitmap bitmap = BitmapFactory.decodeFile(path, options); return new BitmapDrawable(a.getResources(), bitmap);**

**}**

}

Методы Display.getWidth() иDisplay.getHeight() считаются устаревшими (deprecated). Эта тема более подробно рассматривается в конце главы.

В идеале изображение стоило бы масштабировать так, чтобы оно точно соответствовало размерам ImageView. Однако размер представления, в котором выводится представление, часто бывает недоступен в нужный момент. Например, в методеonCreateView(…) мы не можем получить размерImageView. Для надежности изображение масштабируется по размерам текущего экрана устройства, который доступен всегда. Представление, в котором будет выводиться фотография, может быть меньше стандартного размера экрана, но больше быть не может.

Затем добавьте в CrimeFragment закрытый метод, который связывает масштабированную версию изображения сImageView.

**Листинг 20.13.**Добавление showPhoto() (CrimeFragment.java)

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle savedInstanceState) {

...

}

**private void showPhoto() {**

**// Назначение изображения, полученного на основе фотографии**

**Photo p = mCrime.getPhoto(); BitmapDrawable b = null;**

**if (p != null) {**

**String path = getActivity()**

**.getFileStreamPath(p.getFilename()).getAbsolutePath(); b = PictureUtils.getScaledDrawable(getActivity(), path);**

**}**

**mPhotoView.setImageDrawable(b);**

**}**

Включите в файл CrimeFragment.java реализациюonStart() с вызовомshowPhoto(), чтобы фотография была готова к тому моменту, как пользователь увидитCrimeFragment.

**Листинг 20.14.**Загрузка изображения (CrimeFragment.java)

...

private void showPhoto() {

// Назначение изображения, полученного на основе фотографии

Photo p = mCrime.getPhoto(); BitmapDrawable b = null;

if (p != null) {

String path = getActivity()

.getFileStreamPath(p.getFilename()).getAbsolutePath(); b = PictureUtils.getScaledDrawable(getActivity(), path);

}

http://www.studfiles.ru/html/2706/1137/html_K2MGkhC3f0.fGLx/htmlconvd-G_3cW2347x1.jpg

mPhotoButton.setImageDrawable(b);

}

**@Override**

**public void onStart() { super.onStart(); showPhoto();**

**}**

Включите в метод CrimeFragment.onActivityResult(…)вызов showPhoto(), чтобы изображение было видимым при возвращении пользователя из CrimeCameraActivity.

**Листинг 20.15.**Вызов showPhoto() в onActivityResult(…) (CrimeFragment.java)

@Override

public void onActivityResult(int requestCode, int resultCode, Intent data) { if (resultCode != Activity.RESULT\_OK) return;

if (requestCode == REQUEST\_PHOTO) {

// Создание нового объекта Photo и связывание его с Crime

String filename = data

.getStringExtra(CrimeCameraFragment.EXTRA\_PHOTO\_FILENAME); if (filename != null) {

Photo p = new Photo(filename); mCrime.setPhoto(p); **showPhoto();**

**~~Log.i(TAG, "Crime: " + mCrime.getTitle() + "has a photo");~~**

}

}

}

**Выгрузка изображения**

Включите в класс PictureUtils метод для уничтожения экземпляраBitmapDrawable, связанного сImageView (если он существует).

**Листинг 20.16.**Очистка данных (PictureUtils.java)

public class PictureUtils { /\*\*

\* ...

\*/

@SuppressWarnings("deprecation")

public static BitmapDrawable getScaledDrawable(Activity a, String path) {

...

}

**public static void cleanImageView(ImageView imageView) {**

**if (!(imageView.getDrawable() instanceof BitmapDrawable)) return;**

**// Стирание изображения для экономии памяти**

**BitmapDrawable b = (BitmapDrawable)imageView.getDrawable(); b.getBitmap().recycle();**

**imageView.setImageDrawable(null);**

**}**

}

http://www.studfiles.ru/html/2706/1137/html_K2MGkhC3f0.fGLx/htmlconvd-G_3cW2348x1.jpg

О вызове Bitmap.recycle() стоит рассказать подробнее. В документации говорится, что вызывать этот метод не обязательно, но это не так.Bitmap.recycle() освобождает системную (native) память, занимаемую растровым изображением. Это большая часть содержания объекта растрового изображения. (Системная память может содержать больший или меньший объем данных в зависимости от версии Android. До Honeycomb в ней хранились все данные объектов JavaBitmap.)

Если не освободить память явным вызовом recycle(), она в конечном итоге все равно будет освобождена. Однако освобождение произойдеткогда-тов будущем в*завершителе*(finalizer), а не при уничтожении самого изображения в ходе уборки мусора. Соответственно возникает вероятность исчерпания свободной памяти до вызова завершителя.

Момент выполнения завершителя заранее неизвестен, поэтому такие ошибки сложно отслеживать и воспроизводить. Таким образом, при большом размере изображений (как в нашем случае) лучше вызвать recycle() для предотвращения неприятных ошибок памяти.

Включите в CrimeFragmentреализацию onStop()с вызовом cleanImageView(…).

**Листинг 20.17.**Выгрузка изображения (CrimeFragment.java)

@Override

public void onStart() { super.onStart(); showPhoto();

}

**@Override**

**public void onStop() { super.onStop();**

**PictureUtils.cleanImageView(mPhotoView);**

**}**

Загрузка изображений в onStart() с выгрузкой вonStop() — полезная практика. Эти методы отмечают точки, в которых активность может быть видна пользователю. Если же выполнять загрузку и выгрузку вonResume() иonPause(), результат может оказаться неожиданным для пользователя.

Приостановленная активность все равно может быть частично видимой — если, например, поверх нее открывается активность, не занимающая весь экран. Если использовать onResume() иonPause(), изображения в таких ситуациях будут исчезать и появляться. Лучше загружать изображения, как только ваша активность становится доступной, и откладывать выгрузку до того момента, когда активность гарантированно не видна на экране.

Запустите приложение CriminalIntent. Сделайте снимок и убедитесь в том, что он появился в ImageView. Закройте CriminalIntent и запустите приложение заново. Убедитесь в том, что при возвращении к тому же элементу списка преступлений фотография отображается, как и положено.

Ориентация CrimeCameraActivity подсказывает пользователю, что фотографии следует делать в альбомной ориентации. Если фотография будет сделана в книжной ориентации, то изображение может быть неправильно расположено на кнопке.

Этот недостаток будет исправлен в первом упражнении.



|  |  |
| --- | --- |
|  |  |
|  |  |

**Отображение полноразмерных изображений в DialogFragment**

Наша работа над поддержкой камеры завершится отображением увеличенных версий фотографий Crime.



**Рис. 20.10.**DialogFragment с увеличенным изображением

Создайте новый класс в пакете com.bignerdranch.android.criminalintent. Присвойте классу имяImageFragment; назначьте его субклассомDialogFragment.

В аргументах фрагмента ImageFragment должен передаваться путь к файлу фотографии дляCrime. В файлеImageFragment.java добавьте методnewInstance(String), который получает путь к файлу и помещает его в пакет аргументов, как показано в листинге 20.18.

**Листинг 20.18.**Создание ImageFragment (ImageFragment.java)

public class ImageFragment extends DialogFragment { **public static final String EXTRA\_IMAGE\_PATH =**

**"com.bignerdranch.android.criminalintent.image\_path";**

**public static ImageFragment newInstance(String imagePath) { Bundle args = new Bundle();**

**args.putSerializable(EXTRA\_IMAGE\_PATH, imagePath);**

**ImageFragment fragment = new ImageFragment(); fragment.setArguments(args); fragment.setStyle(DialogFragment.STYLE\_NO\_TITLE, 0);**

**return fragment;**

**}**

}

Фрагменту назначается стиль DialogFragment.STYLE\_NO\_TITLE, с которым фрагмент имеет минималистское оформление, показанное на рис. 20.10.

ImageFragment не нужны заголовок и кнопки, предоставляемыеAlertDialog. Если ваш фрагмент может обойтись без них, лучше использовать более элегантное, быстрое и гибкое решение с переопределениемonCreateView(…) и использованием простого объектаView (вместо переопределенияonCreateDialog(…) и использования Dialog).

В файле ImageFragment.java переопределите методonCreateView(…), чтобы он создавал объектImageView «с нуля» с получением пути к файлу из своих аргументов. Затем получите масштабированную версию изображения и свяжите ее сImageView.

Также переопределите onDestroyView(), который должен освобождать память, когда надобность в изображении отпадает.

**Листинг 20.19.**Создание ImageFragment (ImageFragment.java)

public class ImageFragment extends DialogFragment {

public static final String EXTRA\_IMAGE\_PATH = "com.bignerdranch.android.criminalintent.image\_path";

public static ImageFragment newInstance(String imagePath) {

...

}

**private ImageView mImageView;**

**@Override**

**public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle savedInstanceState) {**

**mImageView = new ImageView(getActivity());**

**String path = (String)getArguments().getSerializable(EXTRA\_IMAGE\_PATH); BitmapDrawable image = PictureUtils.getScaledDrawable(getActivity(), path);**

**mImageView.setImageDrawable(image);**

**return mImageView;**

**}**

**@Override**

**public void onDestroyView() { super.onDestroyView(); PictureUtils.cleanImageView(mImageView);**

**}**

}

Наконец, необходимо вывести это диалоговое окно из CrimeFragment. В файлеCrimeFragment.java добавьте слушателя дляmPhotoView. В его реализации создайте экземплярImageFragment и добавьте его в экземплярFragmentManager активности

CrimePagerActivity вызовом метода show(…)для ImageFragment. Также понадобится строковая константа для идентификации ImageFragmentв FragmentManager.

**Листинг 20.20.**Отображение ImageFragment (CrimeFragment.java)public class CrimeFragment extends Fragment {

...

**private static final String DIALOG\_IMAGE = "image";**

...

@Override

@TargetApi(11)

public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle savedInstanceState) {

...

mPhotoView = (ImageView)v.findViewById(R.id.crime\_imageView);**mPhotoView.setOnClickListener(new View.OnClickListener() {**

**public void onClick(View v) { Photo p = mCrime.getPhoto(); if (p == null)**

**return;**

**FragmentManager fm = getActivity()**

**.getSupportFragmentManager(); String path = getActivity()**

**.getFileStreamPath(p.getFilename()).getAbsolutePath(); ImageFragment.newInstance(path)**

**.show(fm, DIALOG\_IMAGE);**

**}**

**});**

...

}

Запустите приложение CriminalIntent. Сделайте снимок и убедитесь в том, что теперь приложение позволяет просмотреть фотографию сцены преступления во всех подробностях.

**Упражнение. Ориентация изображения в Crime**

Иногда пользователь хочет сделать снимок в книжной ориентации. Найдите в документации API информацию о том, как обнаружить текущую ориентацию. Сохраните правильную ориентацию в Photo и используйте ее для поворота изображения

в CrimeFragmentи ImageFragment.

**Упражнение. Удаление фотографий**

В текущей версии приложения можно заменить фотографию Crime, но при этом старый файл остается и занимает место на диске. Добавьте в метод onActivityResult(int,int, Intent) класса CrimeFragment код проверки существующей фотографии и удаления соответствующего файла с диска.

В качестве дополнительного задания предоставьте пользователю возможность удаления фотографии без замены. Реализуйте в CrimeFragment контекстное меню и/или режим контекстных действий, активизируемый долгим нажатием на миниатюре изображения. Команда меню Delete Photo удаляет фотографию с диска, из модели и ImageView.

**Для любознательных: устаревшие конструкции в Android**

В главе 19, при назначении размера области предварительного просмотра камеры мы использовали устаревший метод и устаревшую константу. В этой главе тоже использовались устаревшие методы. У наблюдательного читателя может возникнуть вопрос: «Что это вообще значит?»

Начнем с того, что же понимать под самим термином. Если некоторая часть API считается устаревшей, это означает, что она более не является необходимой. Иногда устаревание происходит из-затого, что выполняемая операция стала ненужной —

как в случае с методом SurfaceHolder.setType(int)и константой SurfaceHolder. SURFACE\_TYPE\_PUSH\_BUFFERS, использованными в главе 19.

Встарых версиях Android экземпляр SurfaceHolder должен был настраиваться в соответствии с предполагаемым использованием. Сейчас это уже не нужно, поэтому методsetType(…) стал бесполезным.

Вдругих случаях это означает, что метод был заменен новым методом, который покакой-то причине является предпочтительным. Например, классBitmapDrawableсодержит устаревший конструктор BitmapDrawable(Bitmap), при использовании которого часто допускались ошибки. А может быть, новый метод «чище» с точки зрения архитектуры, как, например, методView.setBackgroundDrawable(Drawable). Также можно вспомнить методы Display.getWidth()иDisplay.getHeight(), которые использовались ранее. Сейчас они заменены одним методомgetSize(Point), который предотвращает ошибки, возникавшие при последовательном вызове get-

Width() и getHeight().

Проблема устаревания кода решается по-разномув зависимости от того, на какой платформе вы работаете. Две крайности воплощены в двух подходах, которые вам, вероятно, знакомы — подходе Microsoft и подходе Apple.

В подходе Microsoft части API устаревают, но никогда не удаляются. Это связано с тем, что в мировоззрении Microsoft важнейшим фактором является работоспособность наибольшего количества программ в любой версии ОС. Когда Microsoft представляет общедоступный API, он всегда поддерживается. Порой доходит до сохранения ошибочного, недокументированного поведения для обеспечения обратной совместимости. Из-заэтого мир Microsoft порой выглядит довольно странно.

С другой стороны, в мире Apple API удаляются из ОС вскоре после того, как они были объявлены устаревшими. Руководители мира Apple стремятся иметь чистую,