В каких случаях нужно использовать SurfaceView

SurfaceView  используется точно таким же образом, как любые произ- водные от View классы. Вы можете применять анимацию и размещать  их внутри разметки  так же, как и другие Представления.

Объект  Surface, который  лежит  в основе SurfaceView,  поддерживает рисование  с помощью  большинства стандартных  методов Canvas,  опи- санных ранее, а также может использовать все возможности библиотеки OpenGL  ES.

Применяя OpenGL,  вы можете рисовать  на Surface любые поддержи- ваемые двумерные  или трехмерные  объекты, получая  при этом все выго- ды от аппаратного  ускорения (если  таковое имеется).  Таким  образом, вы значительно повышаете  производительность, если сравнивать  с теми же операциями, выполненными на двумерном Canvas.

Объекты SurfaceView особенно пригодятся для отображения динамиче- ских трехмерных изображений, к примеру, в интерактивных играх, их можно назвать  лучшим  выбором  для отображения предварительного просмотра видеопотоков  с камеры в режиме реального времени.

**Создание нового объекта SurfaceView**

Чтобы создать данный тип Представления, наследуйте класс SurfaceView и реализуйте интерфейс SurfaceHolder.Callback, описывающий функцию обратного вызова. Он уведомляет  Представление о том, что исходный объ- ект Surface был создан/уничтожен/модифицирован и передает в объект SurfaceHolder ссылку, содержащую допустимый экземпляр Surface.

Типичный шаблон проектирования SurfaceView предусматривает клас- сы, производные от Thread, которые принимают ссылку на текущий объект SurfaceHolder и немедленно его обновляют.

В листинге  15.23 показана реализация SurfaceView, рисование  с помо- щью объекта Canvas. Новые производные от Thread классы создаются вну- три элемента SurfaceView, и все обновления пользовательского интерфейса происходят  в этом новом классе.

**Листинг 15.23.**Каркас для реализации класса SurfaceView

import android.content.Context;

import android.graphics.Canvas;

import android.view.SurfaceHolder;

import android.view.SurfaceView;

public class MySurfaceView extends SurfaceView implements

SurfaceHolder.Callback {

private SurfaceHolder holder;

private MySurfaceViewThread mySurfaceViewThread;

private boolean hasSurface;

MySurfaceView(Context context) {

super(context);

init();

}

private void init() {

// Создайте новый объект SurfaceHolder  и определите данный

// класс в качестве его функции обратного вызова. holder = getHolder();

holder.addCallback(this);

hasSurface = false;

}

public void resume() {

// Создайте и запустите поток для обновления UI. if (mySurfaceViewThread == null) {

mySurfaceViewThread = new MySurfaceViewThread();

if (hasSurface == true)

mySurfaceViewThread.start();

}

}

public void pause() {

// Уничтожьте поток для обновления UI. if (mySurfaceViewThread != null) {

mySurfaceViewThread.requestExitAndWait();

mySurfaceViewThread = null;

}

}

public void surfaceCreated(SurfaceHolder holder) {

hasSurface = true;

if (mySurfaceViewThread != null)

mySurfaceViewThread.start();

}

public void surfaceDestroyed(SurfaceHolder holder) {

hasSurface = false;

pause();

}

public void surfaceChanged(SurfaceHolder holder, int format,

int w, int h) {

if (mySurfaceViewThread != null)

mySurfaceViewThread.onWindowResize(w, h);

}

class MySurfaceViewThread extends Thread {

private boolean done;

MySurfaceViewThread() {

super();

done = false;

}

@Override

public void run() {

SurfaceHolder surfaceHolder = holder;

// Повторяйте цикл перерисовки, пока поток не будет остановлен. while (!done) {

// Заблокируйте Surface и верните Canvas, на котором можно

// рисовать.

Canvas canvas = surfaceHolder.lockCanvas();

// TODO: Процесс рисования на объекте Canvas!

// Разблокируйте Canvas и отобразите текущее изображение. surfaceHolder.unlockCanvasAndPost(canvas);

}

}

public void requestExitAndWait() {

// Пометьте этот поток как завершенный и соедините

// его с главным потоком приложения. done = true;

try {

join();

} catch (InterruptedException ex) { }

}

public void onWindowResize(int w, int h) {

// Обработайте изменение доступных размеров поверхности.

}

}

}