有关墙纸的工作

shecenon@gmail.com

October 31, 2011

Contents

Part I
Android Wallpaper

1 Android Wallpaper Framework

frameworks/base/core/java/android/app/WallpaperManager.java
frameworks/base/core/java/android/service/wallpaper/WallpaperService.java
frameworks/base/services/java/com/android/server/WindowManagerService.java
WallpaperManager.java define
public static final String COMMAND_DROP = ”android.home.drop”;

2 Wallpaper interactive

2.1 Launcher code 的代码

(packages/apps/Launcher2/src/com/android/launcher2/)
CellLayout.java call

```
mWallpaperManager.sendWallpaperCommand(getWindowToken(),
                                       "android.home.drop",
                                        cellXY[0] + childLeft + lp.width / 2,
                                       cellXY[1] + childTop + lp.height / 2, 0, null);
```

点击产生水波，，由Launcher

packages/apps/Launcher2/src/com/android/launcher2/Workspace.java

```
onInterceptTouchEvent()
          ......
       case MotionEvent.ACTION_CANCEL:
       case MotionEvent.ACTION_UP:
            if (mTouchState != TOUCH_STATE_SCROLLING) {
                final CellLayout currentScreen = (CellLayout)
                    getChildAt(mCurrentScreen);
                if (!currentScreen.lastDownOnOccupiedCell())
                    {
                    getLocationOnScreen(mTempCell);
                    // Send a tap to the wallpaper if the
                        last down was on empty space
                    final int pointerIndex = ev.
                        findPointerIndex(mActivePointerId);
                    mWallpaperManager.sendWallpaperCommand(
                        getWindowToken(),
                              "android.wallpaper.tap",
                              mTempCell[0] + (int) ev.getX(
                                  pointerIndex),
                              mTempCell[1] + (int) ev.getY(
                                  pointerIndex), 0, null);
                }
            }


    private void updateWallpaperOffset(int scrollRange) {
        IBinder token = getWindowToken();
        if (token != null) {
            mWallpaperManager.setWallpaperOffsetSteps(1.0f
                / (getChildCount() - 1), 0 );
            mWallpaperManager.setWallpaperOffsets(
                getWindowToken(),
                    Math.max(0.f, Math.min(mScrollX/(float)
                        scrollRange, 1.f)), 0);
        }
    }
```

2.2 Wallpaper code

```
packages/wallpapers/Basic/src/com/android/wallpaper/fall/
    FallRS.java
     @Override
     public Bundle onCommand(String action, int x, int y,
         int z, Bundle extras,
             boolean resultRequested) {
         if (WallpaperManager.COMMAND_TAP.equals(action)) {
             addDrop(x + (mWorldState.rotate == 0 ? (
                 mWorldState.width * mWorldState.xOffset) :
                 0), y);
         } else if (WallpaperManager.COMMAND_DROP.equals(
             action)) {
             addDrop(x + (mWorldState.rotate == 0 ? (
                 mWorldState.width * mWorldState.xOffset) :
                 0), y);
         }
         return null;
     }
```

2.3 系统壁纸的存放

```
packages/apps/Launcher2/res/values-hdpi/wallpapers.xml:
            <item>wallpaper_goldengate</item>
device/sibrary/m970/overlay/packages/apps/Launcher2/res/
    values-hdpi/wallpapers.xml:          <item>
    wallpaper_goldengate</item>
packages/apps/Launcher2/res/drawable-mdpi/wallpaper_beach.
    jpg
packages/apps/Launcher2/res/drawable-mdpi/
    wallpaper_beach_small.jpg
packages/apps/Launcher2/res/drawable-hdpi/
    wallpaper_goldengate.jpg
packages/apps/Launcher2/res/drawable-hdpi/
    wallpaper_goldengate_small.jpg
device/sibrary/m970/overlay/packages/apps/Launcher2/res/
    drawable-hdpi/wallpaper_shuttle_small.jpg
device/sibrary/m970/overlay/packages/apps/Launcher2/res/
    drawable-hdpi/wallpaper_shuttle.jpg加入要加入一个壁纸，首先
    在文件
```

```

wallpapers.添加其名，然后，在中放入两xmlwallpaper_foodrawable张图片
:wallpaper_foo.及jpgwallpaper_foo_small.，后面的小图不能省略。jpg

device/sibrary/m970/overlay/frameworks/base/core/res/res/values/config.xml
frameworks/base/core/res/res/values/config.xml
```

android之壁纸机制
1.涉及核心类:
1>ImageWallpaper.java(IW):继承WallpaperService主要负责静态壁纸的draw处理;
2>WallpaperManager.java(WM):主要负责壁纸的存取方法管理(可能会多个实例);
3>WallpaperManagerService(WMS).java:主要是对WalllpaperManager一些核心方法提供,及一些初始参数的保存(服务);
4>iWallpaperManager.aidl(IWM):负责WallpaperManager与WallpaperManagerService之间的通信;
5>IWallpaperManagerCallback(IMC).aidl:负责WallpaperManager与WallpaperManagerSer
间的通信,这是一个回调机制与前面不同;
6>WallpaperService.java(WS):设置壁纸的引擎机制(包括动态与静态);//这类工作时没有修改过,所以个人了解不是很清楚,希望有朋友补充.
7>launcher.java(LC)设置壁纸初始化值,带到壁纸机制的转屏.

2. 壁纸从设置存到取流程:
1>首先WM.setBitmap(bitmap)
```
public void setBitmap(Bitmap bitmap) throws IOException {
        try {
            ParcelFileDescriptor fd = sGlobals.mService.setWallpaper(null);
            if (fd == null) {
                return;
            }
            FileOutputStream fos = null;
            try {
                fos = new ParcelFileDescriptor.AutoCloseOutputStream(fd);
                bitmap.compress(Bitmap.CompressFormat.PNG, 90, fos);
            } finally {
                if (fos != null) {
                    fos.close();
                }
            }
        } catch (RemoteException e) {
        }
    }
2>然后WMS.setWallpaper(null),设置成功会写入到壁纸相应文件里,文件
监听类此时会触发
private final FileObserver mWallpaperObserver = new FileObserver(
            WALLPAPER_DIR.getAbsolutePath(), CREATE | CLOSE_WRITE | DELETE | DELETE_SELF
                @Override
                public void onEvent(int event, String path) {
                    if (path == null) {
                        return;
                    }
                    synchronized (mLock) {
                        // changing the wallpaper means we'll need to back up the new or
                        long origId = Binder.clearCallingIdentity();
                        BackupManager bm = new BackupManager(mContext);
                        bm.dataChanged();
                        Binder.restoreCallingIdentity(origId);

                        File changedFile = new File(WALLPAPER_DIR, path);
                        if (WALLPAPER_FILE.equals(changedFile)) {
                            notifyCallbacksLocked();//会发出广播
与调用回调方法
                        }
                    }
                }
            };

//notifyCallbacksLocked()做两件事情
   private void notifyCallbacksLocked() {
        final int n = mCallbacks.beginBroadcast();
        for (int i = 0; i < n; i++) {
            try {
                mCallbacks.getBroadcastItem(i).onWallpaperChanged();//回
调机制在ＷＭ实现onWallpaperChanged()
            } catch (RemoteException e) {

                // The RemoteCallbackList will take care of removing
                // the dead object for us.
            }
        }
        mCallbacks.finishBroadcast();
        final Intent intent = new Intent(Intent.ACTION_WALLPAPER_CHANGED);//壁
纸变化的广播意图
        mContext.sendBroadcast(intent);//ＩＷ会接收到此意图,Ｉ
```

W.updateWallpaper()去获取新壁纸.
    }

3>WM.onWallpaperChanged()通过handler机制清除壁纸缓存

```
   private final Handler mHandler;

        Globals(Looper looper) {
            IBinder b = ServiceManager.getService(Context.WALLPAPER_SERVICE);
            mService = IWallpaperManager.Stub.asInterface(b);
            mHandler = new Handler(looper) {
                @Override
                public void handleMessage(Message msg) {
                    switch (msg.what) {
                        case MSG_CLEAR_WALLPAPER:
                            synchronized (this) {
                                mWallpaper = null;//用户自定义壁
纸
                                mDefaultWallpaper = null;//系统
默认壁纸
                            }
                            break;
                    }
                }
            };
        }

        public void onWallpaperChanged() {
            /* The wallpaper has changed but we shouldn't eagerly load the
             * wallpaper as that would be inefficient. Reset the cached wallpape
             * to null so if the user requests the wallpaper again then we'll
             * fetch it.
             */
            mHandler.sendEmptyMessage(MSG_CLEAR_WALLPAPER);
        }
```

//IW.updateWallpaper()

```
  void updateWallpaper() {
            synchronized (mLock) {
                try {
                    mBackground = mWallpaperManager.getFastDrawable();//W
M去获取壁纸
                } catch (RuntimeException e) {
                    Log.w("ImageWallpaper", "Unable to load wallpaper!", e);
                }
            }
        }
//收到壁纸变换广播
     class WallpaperObserver extends BroadcastReceiver {
            public void onReceive(Context context, Intent intent) {
                updateWallpaper();//调用
                drawFrame();
                // Assume we are the only one using the wallpaper in this
                // process, and force a GC now to release the old wallpaper.
                System.gc();
            }
        }

        @Override
        public void onCreate(SurfaceHolder surfaceHolder) {
            super.onCreate(surfaceHolder);
            IntentFilter filter = new IntentFilter(Intent.ACTION_WALLPAPER_CHANGED);//
            mReceiver = new WallpaperObserver();
            registerReceiver(mReceiver, filter);//注册
            updateWallpaper();
            surfaceHolder.setSizeFromLayout();
        }
```

4 >mWallpaperManager.getFastDrawable();//WM去获取壁纸

```
a.  public Drawable getFastDrawable() {
        Bitmap bm = sGlobals.peekWallpaperBitmap(mContext, true);//获
取壁纸总方法
        if (bm != null) {
            Drawable dr = new FastBitmapDrawable(bm);
            return dr;
        }
        return null;
    }

b. public Bitmap peekWallpaperBitmap(Context context, boolean returnDefault) {
            synchronized (this) {
                if (mWallpaper != null) {
                    return mWallpaper;
                }
                if (mDefaultWallpaper != null) {
                    return mDefaultWallpaper;
                }
                mWallpaper = null;
                try {
                    mWallpaper = getCurrentWallpaperLocked(context);//调
用获取用户自定义壁纸方法
                } catch (OutOfMemoryError e) {
                    Log.w(TAG, "No memory load current wallpaper", e);
                }
                if (mWallpaper == null && returnDefault) {
                    mDefaultWallpaper = getDefaultWallpaperLocked(context);调
用默认壁纸方法
                    return mDefaultWallpaper;
                }
                return mWallpaper;
            }
        }
```

c. 两方法分析

```
private Bitmap getCurrentWallpaperLocked(Context context) {
            try {
                Bundle params = new Bundle();
                ParcelFileDescriptor fd = mService.getWallpaper(this, params);/
ＭＳ.getWallpaper(this, params),params是out型表示参数传送是从Ｗ
ＭＳ传到ＷＭ,是与平时java编码不合适习惯的这android一特性也是aidl机
制的一部分;这里留个问题就ＷＭＳ是如何获取到的params参数呢?
                if (fd != null) {
                    int width = params.getInt("width", 0);
                    int height = params.getInt("height", 0);

                    if (width <= 0 || height <= 0) {
                        // Degenerate case: no size requested, just load
                        // bitmap as-is.
                        Bitmap bm = null;
                        try {
                            bm = BitmapFactory.decodeFileDescriptor(
                                   fd.getFileDescriptor(), null, null);
                        } catch (OutOfMemoryError e) {
                            Log.w(TAG, "Can’t decode file", e);
                        }
                        try {
                            fd.close();
                        } catch (IOException e) {
                        }
                        if (bm != null) {
                            bm.setDensity(DisplayMetrics.DENSITY_DEVICE);
                        }
                        return bm;
                    }

                    // Load the bitmap with full color depth, to preserve
                    // quality for later processing.
                    BitmapFactory.Options options = new BitmapFactory.Options();
                    options.inDither = false;
                    options.inPreferredConfig = Bitmap.Config.ARGB_8888;
                    Bitmap bm = BitmapFactory.decodeFileDescriptor(
                            fd.getFileDescriptor(), null, options);
                    try {
                        fd.close();
                    } catch (IOException e) {
                    }

                    return generateBitmap(context, bm, width, height);
                }
            } catch (RemoteException e) {
            }
            return null;
        }

        private Bitmap getDefaultWallpaperLocked(Context context) {
            try {
                InputStream is = context.getResources().openRawResource(
                        com.android.internal.R.drawable.default_wallpaper);
                if (is != null) {
                    int width = mService.getWidthHint();
                    int height = mService.getHeightHint();

                    if (width <= 0 || height <= 0) {
                        // Degenerate case: no size requested, just load
                        // bitmap as-is.
                        Bitmap bm = null;
                        try {
                            bm = BitmapFactory.decodeStream(is, null, null);
                        } catch (OutOfMemoryError e) {
                            Log.w(TAG, "Can't decode stream", e);
                        }
                        try {
                            is.close();
                        } catch (IOException e) {
                        }
                        if (bm != null) {
```

```
                            bm.setDensity(DisplayMetrics.DENSITY_DEVICE);
                        }
                        return bm;
                    }
```
5 >W M S .getWallpaper(this, params)
```
    public ParcelFileDescriptor getWallpaper(IWallpaperManagerCallback cb,
            Bundle outParams) {
        synchronized (mLock) {
            try {
                if (outParams != null) {
                    outParams.putInt("width", mWidth);
                    outParams.putInt("height", mHeight);
                }
                mCallbacks.register(cb);
                File f = WALLPAPER_FILE;
                if (!f.exists()) {
                    return null;
                }
                return ParcelFileDescriptor.open(f, MODE_READ_ONLY);//ParcelFil
```
定义的句柄具有安全性,对它所属的流具体保护性,否会图像丢失出现花屏
情况.我对它了解也不深,不遇到类似的bug.
```
            } catch (FileNotFoundException e) {
                /* Shouldn't happen as we check to see if the file exists */
                Slog.w(TAG, "Error getting wallpaper", e);
            }
            return null;
        }
    }
```

6 >呵呵呵,该画了 I W.draw(),考虑有很多东西要跟大家分享下就把 I W类
粘贴出来
```
package com.android.internal.service.wallpaper;

import com.android.internal.view.WindowManagerPolicyThread;

import android.app.WallpaperManager;
import android.graphics.Canvas;
import android.graphics.Rect;
import android.graphics.Region.Op;
import android.graphics.drawable.Drawable;
import android.os.HandlerThread;
import android.os.Looper;
import android.os.Process;
import android.service.wallpaper.WallpaperService;
import android.util.Log;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.content.Context;
import android.content.IntentFilter;
import android.content.Intent;
import android.content.BroadcastReceiver;

/**
* Default built-in wallpaper that simply shows a static image.
*/
public class ImageWallpaper extends WallpaperService {
    WallpaperManager mWallpaperManager;
    private HandlerThread mThread;

    @Override
    public void onCreate() {
        super.onCreate();
        mWallpaperManager = (WallpaperManager) getSystemService(WALLPAPER_SERVICE);
        Looper looper = WindowManagerPolicyThread.getLooper();
        if (looper != null) {
            setCallbackLooper(looper);
        } else {
            mThread = new HandlerThread("Wallpaper", Process.THREAD_PRIORITY_FOREGROUND
            mThread.start();
            setCallbackLooper(mThread.getLooper());
        }
    }

    public Engine onCreateEngine() {
        return new DrawableEngine();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mThread != null) {
            mThread.quit();
        }
    }

    class DrawableEngine extends Engine {
        private final Object mLock = new Object();
        private WallpaperObserver mReceiver;
        Drawable mBackground;
        float mXOffset;
        float mYOffset;

        class WallpaperObserver extends BroadcastReceiver {
            public void onReceive(Context context, Intent intent) {
                updateWallpaper();
                drawFrame();
                // Assume we are the only one using the wallpaper in this
                // process, and force a GC now to release the old wallpaper.
                System.gc();
            }
        }

        @Override
        public void onCreate(SurfaceHolder surfaceHolder) {
            super.onCreate(surfaceHolder);
            IntentFilter filter = new IntentFilter(Intent.ACTION_WALLPAPER_CHAN
            mReceiver = new WallpaperObserver();
            registerReceiver(mReceiver, filter);
            updateWallpaper();
            surfaceHolder.setSizeFromLayout();
        }

        @Override
        public void onDestroy() {
            super.onDestroy();
            unregisterReceiver(mReceiver);
        }

        @Override
        public void onVisibilityChanged(boolean visible) {//亮屏
时会执行
            drawFrame();
        }

        @Override
        public void onTouchEvent(MotionEvent event) {
            super.onTouchEvent(event);
        }

        @Override
        public void onOffsetsChanged(float xOffset, float yOffset,
                float xOffsetStep, float yOffsetStep,
                int xPixels, int yPixels) {//滑动壁纸时会执行
            mXOffset = xOffset;
            mYOffset = yOffset;
            drawFrame();
        }

        @Override
        public void onSurfaceChanged(SurfaceHolder holder, int format, int width, int he
机和转屏时会执行
            super.onSurfaceChanged(holder, format, width, height);
            drawFrame();
        }

        @Override
        public void onSurfaceCreated(SurfaceHolder holder) {
            super.onSurfaceCreated(holder);
        }

        @Override
        public void onSurfaceDestroyed(SurfaceHolder holder) {
            super.onSurfaceDestroyed(holder);
        }

        void drawFrame() {
            SurfaceHolder sh = getSurfaceHolder();
            Canvas c = sh.lockCanvas();//锁住canvas
            if (c != null) {
                final Rect frame = sh.getSurfaceFrame();
                synchronized (mLock) {
                    final Drawable background = mBackground;
                    final int dw = frame.width();
                    final int dh = frame.height();
                    final int bw = background != null ? background.getIntrinsicWidth() :
                    final int bh = background != null ? background.getIntrinsicHeight()
                    final int availw = dw-bw;
                    final int availh = dh-bh;
                    int xPixels = availw < 0 ? (int)(availw*mXOffset+.5f) : (availw/2);
                    int yPixels = availh < 0 ? (int)(availh*mYOffset+.5f) : (availh/2);

                    c.translate(xPixels, yPixels);//滑动后计算到
壁纸画起点位置
                    if (availw<0 || availh<0) {
                        c.save(Canvas.CLIP_SAVE_FLAG);
                        c.clipRect(0, 0, bw, bh, Op.DIFFERENCE);
                        c.drawColor(0xff000000);//出现壁纸尺寸
异常或是转屏延迟就会画黑
                        c.restore();
                    }
                    if (background != null) {
                        background.draw(c);
                    }
                }
                sh.unlockCanvasAndPost(c);//解锁canvas并提交
            }
        }

        void updateWallpaper() {
            synchronized (mLock) {
                try {
                    mBackground = mWallpaperManager.getFastDrawable();
                } catch (RuntimeException e) {
                    Log.w("ImageWallpaper", "Unable to load wallpaper!", e);
                }
            }
        }
    }
}
```

３.壁纸长宽初始化值及转屏处理

```
1>LC.setWallpaperDimension()
private void setWallpaperDimension() {
        WallpaperManager wpm = (WallpaperManager)getSystemService(WALLPAPER_SER

        Display display = getWindowManager().getDefaultDisplay();
        boolean isPortrait = display.getWidth() < display.getHeight();

        final int width = isPortrait ? display.getWidth() : display.getHeight()
        final int height = isPortrait ? display.getHeight() : display.getWidth(
        wpm.suggestDesiredDimensions(width * WALLPAPER_SCREENS_SPAN, height);//
置长宽
    }
```
２>
```
   public void suggestDesiredDimensions(int minimumWidth, int minimumHeight) {
        try {
            sGlobals.mService.setDimensionHints(minimumWidth, minimumHeight);//
        } catch (RemoteException e) {
        }
    }
```

```
3>
    public void setDimensionHints(int width, int height) throws RemoteException {
        checkPermission(android.Manifest.permission.SET_WALLPAPER_HINTS);

        if (width <= 0 || height <= 0) {
            throw new IllegalArgumentException("width and height must be > 0");
        }

        synchronized (mLock) {
            if (width != mWidth || height != mHeight) {
                mWidth = width;
                mHeight = height;
                saveSettingsLocked();//将值以xml形式存储,开机时
候会调用loadSettingsLocked()读取
                if (mWallpaperConnection != null) {
                    if (mWallpaperConnection.mEngine != null) {
                        try {
                            mWallpaperConnection.mEngine.setDesiredSize(
                                    width, height);
                        } catch (RemoteException e) {
                        }
                        notifyCallbacksLocked();//通知壁纸有变
化(包括换壁纸与横竖转换).
                    }
                }
            }
        }
    }
```

４>android的壁纸机制用得ＩＰＣ机制,aidl机制,广播机制,这里我们就不再介绍.不太清楚google吧.另外我将aidl内容贴出来,希望对大家的理解有帮助.

```
/** @hide */
１>IWallpaperManager.aidl
interface IWallpaperManager {

    /**
     * Set the wallpaper.
     */
    ParcelFileDescriptor setWallpaper(String name);

    /**
     * Set the live wallpaper.
     */
    void setWallpaperComponent(in ComponentName name);

    /**
     * Get the wallpaper.
     */
    ParcelFileDescriptor getWallpaper(IWallpaperManagerCallback cb,
            out Bundle outParams);

    /**
     * Get information about a live wallpaper.
     */
    WallpaperInfo getWallpaperInfo();

    /**
     * Clear the wallpaper.
     */
    void clearWallpaper();

    /**
     * Sets the dimension hint for the wallpaper. These hints indicate the desi
     * minimum width and height for the wallpaper.
     */
    void setDimensionHints(in int width, in int height);

    /**
     * Returns the desired minimum width for the wallpaper.
     */
    int getWidthHint();

    /**
     * Returns the desired minimum height for the wallpaper.
     */
    int getHeightHint();
}
```

2 >IWallpaperManagerCallback.aidl

```
oneway interface IWallpaperManagerCallback {
    /**
     * Called when the wallpaper has changed
     */
    void onWallpaperChanged();
}
```

which app to handle file.

~/.local/share/applications/mimeapps.list

how to enable auto login gconf-editor apps gdm simple-greater disable_user_list