ANDROID GRAPHICS & SOUNDS

Ho Dac Hung

Drawing

- The Android framework APIs provides a set of 2D-drawing APIs that allow you to render your own custom graphics onto a canvas or to modify existing Views to customize their look and feel.

Drawing into a View object

- Draw your graphics or animations into a View object from your layout. In this manner, the drawing of your graphics is handled by the system's normal View hierarchy drawing process — you simply define the graphics to go inside the View.

Drawing to a Canvas

- Draw your graphics directly to a Canvas. This way, you personally call the appropriate class's onDraw() method (passing it your Canvas), or one of the Canvas draw...() methods (like drawPicture()). In doing so, you are also in control of any animation.

Drawing to a Canvas

- A Canvas works for you as a pretense, or interface, to the actual surface upon which your graphics will be drawn — it holds all of your "draw" calls. Via the Canvas, your drawing is actually performed upon an underlying Bitmap, which is placed into the window.

Drawing to a Canvas

```
Bitmap b = Bitmap.createBitmap(100, 100, Bitmap.Config.ARGB_8888);

Canvas c = new Canvas(b);
```

Drawing to a Canvas on a View

- If your application does not require a significant amount of processing or frame-rate speed, then you should consider creating a custom View component and drawing with a Canvas in View.onDraw(). The most convenient aspect of doing so is that the Android framework will provide you with a pre-defined Canvas to which you will place your drawing calls.

Drawing to a Canvas on a SurfaceView

- The SurfaceView is a special subclass of View that offers a dedicated drawing surface within the View hierarchy. The aim is to offer this drawing surface to an application's secondary thread, so that the application isn't required to wait until the system's View hierarchy is ready to draw. Instead, a secondary thread that has reference to a SurfaceView can draw to its own Canvas at its own pace.

Canvas drawARGB

void drawARGB (int a, int r, int g, int b)

Parameters	
a	int: alpha component (0..255) of the color to draw onto the canvas
r	int: red component (0..255) of the color to draw onto the canvas
g	int: green component (0..255) of the color to draw onto the canvas
b	int: blue component (0..255) of the color to draw onto the canvas

Canvas
drawArc

void drawArc (float left, float top, float right, float bottom, float startAngle, float sweepAngle, boolean useCenter, Paint paint)

Parameters	
left	float
top	float
right	float
bottom	float
startAngle	float: starting angle where the arc begins
sweepAngle	float: sweep angle measured clockwise
useCenter	boolean: If true, include the center of the oval in the arc
paint	Paint: the paint used to draw the arc

Canvas drawCircle

void drawCircle (float cx, float cy, float radius, Paint paint)

Parameters	
cx	float: the x-coordinate of the center of the cirle to be drawn
cy	float: the y-coordinate of the center of the cirle to be drawn
radius	float: the radius of the cirle to be drawn
paint	Paint: the paint used to draw the circle

Canvas drawColor

void drawColor (int color)

Parameters	
color	int: the color to draw onto the canvas

Canvas
drawLine

void drawLine (float startX, float startY, float stopX, float stopY, Paint paint)

Parameters	
startX	float: the x-coordinate of the start point of the line
startY	float: the y-coordinate of the start point of the line
stopX	float: the x-coordinate of the end point of the line
stopY	float: the y-coordinate of the end point of the line
paint	Paint: the paint used to draw the line

Canvas
drawLines

void drawLines (float[] pts, int offset, int count, Paint paint)

Parameters	
pts	float[]: array of points to draw [x0 y0 x1 y1 x2 y2 ...]
offset	int: number of values in the array to skip before drawing.
count	int: the number of values in the array to process, after skipping "offset" of them. Since each line uses 4 values, the number of "lines" that are drawn is really (count >> 2).
paint	Paint: the paint used to draw the line

Canvas drawPicture

void drawPicture (Picture picture, Rect dst)

Parameters	
picture	Picture
dst	Rect

Canvas
drawRect

void drawRect (float left, float top, float right, float bottom, Paint paint)

Parameters	
left	float: the left side of the rectangle to be drawn
top	float: the top side of the rectangle to be drawn
right	float: the right side of the rectangle to be drawn
bottom	float: the bottom side of the rectangle to be drawn
paint	Paint: the paint used to draw the rect

Canvas drawRect

void drawRect (Rect r, Paint paint)

Parameters	
r	Rect: The rectangle to be drawn.
paint	Paint: The paint used to draw the rectangle

Canvas
drawText

void drawText (String text, float x, float y, Paint paint)

Parameters	
text	string: the text to be drawn
x	float: the x-coordinate of the origin of the text being drawn
y	float: the y-coordinate of the baseline of the text being drawn
paint	Paint: the paint used for the text (e.g. color, size, style)

Canvas
drawText

void drawText (String text, int start, int end, float x, float y, Paint paint)

Parameters	
text	String: The text to be drawn
start	int: The index of the first character in text to draw
end	int: (end - 1) is the index of the last character in text to draw
x	float: The x-coordinate of the origin of the text being drawn
y	float: The y-coordinate of the baseline of the text being drawn
paint	Paint: The paint used for the text (e.g. color, size, style)

Rect

Rect holds four integer coordinates for a rectangle. The rectangle is represented by the coordinates of its 4 edges (left, top, right bottom).

Fields	
text	
start	
end	
x	

Rect Constructors

Rect ()

Rect (int left,int top, int right, int bottom)

Rect (Rect r)

Rect
Contains

boolean contains (int x, int y)

Parameters	
x	int: The X coordinate of the point being tested for containment
y	int: The Y coordinate of the point being tested for containment
Returns	
boolean	true iff (x,y) are contained by the rectangle, where containment means left <= x < right and top <= y < bottom

Rect
Contains

boolean contains (int left, int top, int right, int bottom)

Parameters	
left	int: The left side of the rectangle being tested for containment
top	int: The top of the rectangle being tested for containment
right	int: The right side of the rectangle being tested for containment
bottom	int: The bottom of the rectangle being tested for containment
Returns	
boolean	true iff the the 4 specified sides of a rectangle are inside or equal to this rectangle

Rect
Intersect

boolean intersect (int left,int top,int right,int bottom)

Parameters	
left	int: The left side of the rectangle being tested for containment
top	int: The top of the rectangle being tested for containment
right	int: The right side of the rectangle being tested for containment
bottom	int: The bottom of the rectangle being tested for containment
Returns	
boolean	true if the specified rectangle and this rectangle intersect

Paint
setARGB

void setARGB (int a,int r,int g, int b)

Parameters	
a	int: The new alpha component (0..255) of the paint's color.
r	int: The new red component (0..255) of the paint's color.
g	int: The new green component (0..255) of the paint's color.
b	int: The new blue component (0..255) of the paint's color.

Paint
setAlpha

void setAlpha (int a)

Parameters	
a	int: set the alpha component [0..255] of the paint's color.

Paint
setColor

void setColor (int color)

Parameters	
color	int: The new color (including alpha) to set in the paint.

Paint setStrokeWidth

void setStrokeWidth (float width)

Parameters	
width	float: set the paint's stroke width, used whenever the paint's style is Stroke or StrokeAndFill.

Paint setTextSize

void setTextSize (float textSize)

Parameters	
textSize	float: set the paint's text size in pixel units.

Paint
setTypeface

Typeface setTypeface (Typeface typeface)

Parameters	
typeface	Typeface: May be null. The typeface to be installed in the paint
Returns	
Typeface	typeface

Playing sounds

- SoundPool
- MediaPlayer

SoundPool

- A SoundPool is a collection of samples that can be loaded into memory from a resource inside the APK or from a file in the file system.
- In addition to low-latency playback, SoundPool can also manage the number of audio streams being rendered at once.
- Sounds can be looped by setting a non-zero loop value. A value of -1 causes the sound to loop forever.

SoundPool

- The playback rate can also be changed. A playback rate of 1.0 causes the sound to play at its original frequency. A playback rate of 2.0 causes the sound to play at twice its original frequency, and a playback rate of 0.5 causes it to play at half its original frequency. The playback rate range is 0.5 to 2.0.
- Priority runs low to high, i.e. higher numbers are higher priority. Priority is used when a call to play() would cause the number of active streams to exceed the value established by the maxStreams parameter when the SoundPool was created.

SoundPool Constructor

SoundPool (int maxStreams, int streamType, int srcQuality)

Parameters	
maxStreams	int: the maximum number of simultaneous streams for this SoundPool object
streamType	int: the audio stream type as described in AudioManager For example, game applications will normally use STREAM_MUSIC.
srcQuality	int: the sample-rate converter quality. Currently has no effect. Use 0 for the default.

SoundPool autoPause

void autoPause ()

Pause all active streams. Pause all streams that are currently playing. This function iterates through all the active streams and pauses any that are playing. It also sets a flag so that any streams that are playing can be resumed by calling autoResume().

SoundPool autoResume

void autoResume ()

Resume all previously active streams. Automatically resumes all streams that were paused in previous calls to autoPause().

SoundPool
load

int load (Context context,int resId,int priority)

Parameters	
context	Context: the application context
resId	int: the resource ID
priority	int: the priority of the sound. Currently has no effect. Use a value of 1 for future compatibility.
Returns	
Int	a sound ID. This value can be used to play or unload the sound.

SoundPool
load

int load (String path, int priority)

Parameters	
path	String: the path to the audio file
priority	int: the priority of the sound. Currently has no effect. Use a value of 1 for future compatibility.
Returns	
Int	a sound ID. This value can be used to play or unload the sound.

SoundPool
load

int load (FileDescriptor fd, long offset, long length, int priority)

Parameters	
fd	FileDescriptor: a FileDescriptor object
offset	long: offset to the start of the sound
length	long: length of the sound
priority	int: the priority of the sound. Currently has no effect. Use a value of 1 for future compatibility.
Returns	
Int	a sound ID. This value can be used to play or unload the sound.

SoundPool
unload

boolean unload (int soundID)

Parameters	
soundID	int: a soundID returned by the load() function
Returns	
boolean	true if just unloaded, false if previously unloaded

SoundPool
pause

void pause (int streamID)

Parameters	
streamID	int: a streamID returned by the play() function

SoundPool
play

int play (int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)

Parameters	
soundID	int: a soundID returned by the load() function
leftVolume	float: left volume value (range = 0.0 to 1.0)
rightVolume	float: right volume value (range = 0.0 to 1.0)
priority	int: stream priority (0 = lowest priority)
loop	int: loop mode (0 = no loop, -1 = loop forever)
rate	float: playback rate (1.0 = normal playback, range 0.5 to 2.0)
Returns	
int	non-zero streamID if successful, zero if failed

SoundPool resume

void resume (int streamID)

Resume a playback stream. Resume the stream specified by the streamID. This is the value returned by the play() function. If the stream is paused, this will resume playback. If the stream was not previously paused, calling this function will have no effect.

Parameters	
streamID	int: a streamID returned by the play() function

SoundPool
stop

void stop (int streamID)

Stop a playback stream. Stop the stream specified by the streamID. This is the value returned by the play() function. If the stream is playing, it will be stopped. It also releases any native resources associated with this stream. If the stream is not playing, it will have no effect.

Parameters	
streamID	int: a streamID returned by the play() function

SoundPool
setLoop

void setLoop (int streamID,int loop)

Parameters	
streamID	int: a streamID returned by the play() function
loop	int: loop mode (0 = no loop, -1 = loop forever)

SoundPool setPriority

void setPriority (int streamID, int priority)

Parameters	
streamID	int: a streamID returned by the play() function
priority	int

SoundPool
setRate

void setRate (int streamID, float rate)

Parameters	
streamID	int: a streamID returned by the play() function
rate	float: playback rate (1.0 = normal playback, range 0.5 to 2.0)

SoundPool
setVolume

void setVolume (int streamID,

Parameters	
streamID	int: a streamID returned by the play() function
leftVolume	float: left volume value (range = 0.0 to 1.0)
rightVolume	float: right volume value (range = 0.0 to 1.0)

MediaPlayer

The Android multimedia framework includes support for playing variety of common media types, so that you can easily integrate audio, video and images into your applications. You can play audio or video from media files stored in your application's resources (raw resources), from standalone files in the filesystem, or from a data stream arriving over a network connection, all using MediaPlayer APIs.

MediaPlayer

MediaPlayer can fetch, decode, and play both audio and video with minimal setup. It supports several different media sources such as:

- Local resources
- Internal URIs, such as one you might obtain from a Content Resolver
- External URLs (streaming)