CS378 - Mobile Computing

Responsiveness
An App Idea

• From Nifty Assignments
• Draw a picture use randomness
• Pick an equation at random
• Operators in the equation have the following property:
  Given an input between -1 and 1 the output is also between -1 and 1
• sin and cos scaled to pi / 2, multiply, add, average, remainder (except for 0)
Random Art

• The color at any given point is based on the x and y coordinates of that point scaled to between -1 and 1
• Feed the x and y coordinates into the equation
• Pick equations at random, keep the good pictures, throws away the boring ones
Random Art

Color of this pixel?
Assume large frame is 400 by 300.
Assume this pixel is at 100, 20
x = 100 / 400 = 0.25 -> scaled to -1 to 1 = -0.5
y = 30 / 300 = 0.1 -> scaled to -1 to 1 = -0.2
Plug these values into random equation:
Assume equation is $yxASCSySSxCyCACMMSCSSCC$
postfix, A = Average, S = Sin, C = Cos, M = Multiply
Assume answer is 0.75. Scale to number of colors.
Assume 256 shades of gray.
Color at that pixel is 224$^{th}$ shade of gray (224, 224, 224)
Result  yxASCSSySSxCyCACMMMSCSSCC
Result

xxACSSxCAYCyxAySASCAyCCAYyyAAxMS
xCxCAxySMMMCMCMCSMCS
Result $yCCSxxMSSAS$
Results
Results
Results
RandomArt Application

• Create a subclass of View that does the computation and draws the graphics

• More on 2d graphics later in term
  – but we simply override the onDraw(Canvas) method and draw want we want
  – colors via Paint objects
  – Canvas.drawPoint(x, y, Paint) method

• add click listener to the View so click results in new expression and a redraw
  – invalidate() -> leads to onDraw(Canvas)
The Problem

- Neat idea but computationally expensive
- 480 by 800 screen on Galaxy Nexus
- 384,000 pixels
- depending on the expressions, tens of millions of computations, plus the rendering
Responsiveness

• user's threshold of pain? 1 second? 2?
  – Android dev documents claim 100 to 200 milliseconds (0.1 to 0.2 seconds)

• The Android Systems has its own threshold of pain
  – if the systems determines an application has become unresponsive it displays the Application Not Responding dialog

• ANR occurs if app not responsive to user input
Android System

• The Activity Manager and Window Manager system services monitor applications for responsiveness

• ANR dialog displayed if:
  – No response to an input event such as a key press or screen touch in 5 seconds
  – A BroadcastReceiver doesn't finish executing in 10 seconds
Typical Blocking Operations

- complex calculations or rendering
- looking at data set of unknown size
- parsing a data set
- processing multimedia files
- accessing network resources
- accessing location based services
- access a content provider
- accessing a local database
- accessing a file
The UI Thread

• For application that consist of Activity (or Activities) it is vital to not block the UI thread (main thread of execution)

• AND on API level 11 and later certain operations must be moved off the main UI thread
  – code that accesses resources over a network
  – for example, HTTPrequests on the main UI thread result in a NetworkOnMainThreadException
  – discover StrictMode

Enabling Responsiveness

• move time consuming operations to child threads
  – Android AsyncTask
  – Java Thread
  – Service?
• provide progress bar for worker threads
• big setups -> use a splash screen or render main view as quickly as possible and filling in information asynchronously
• assume the network is SLOW
• don't access the Android UI toolkit from outside the UI thread
Asking for Trouble

• Loading image from network may be slow and block the main (UI) thread of the application

```java
private void setImage() {
    Bitmap b = loadImageFromNetwork("http://www.utexas.edu/sites/default/ImageView.png");
    mImageView.setImageBitmap(b);
}

private Bitmap loadImageFromNetwork(String imageUrl){
    Bitmap bitmap = null;
    try {
        URL url = new URL(imageUrl);
        bitmap = BitmapFactory.decodeStream((InputStream) url.getContent());
    } catch (IOException e) {
        Log.d(TAG, "problem: ");
    }
    return bitmap;
}
```
AsyncTask

• Android class to handle simple threading for operations that take a few seconds
• Removes some of the complexities of Java Thread class and Android Handler class
• Result *published* to the UI thread
AsyncTask

• Three Generic Parameters
  – data type of *Parameter* to task
  – data type of *Progress*
  – data type of *Result*

• four steps in carrying out task
  – onPreExecute()
  – doInBackground(Param... params)
  – onProgressUpdate(Progress values)
  – onPostExecute(Result result)
Methods

• `onPreExecute()` runs on UI thread before background processing begins

• `doInBackground(Param... params)` runs on a background thread and won't block UI thread

• `publishProgress(Progress... values)` method invoked by `doInBackground` and results in call to `onProgressUpdate()` method on UI thread

• `onPostExecute(Result result)` runs on UI thread once `doInBackground` is done
public void onClick(View v) {
    new DownloadImageTask().execute("http://www.utexas
}

private class DownloadImageTask
    extends AsyncTask<String, Void, Bitmap> {
/** The system calls this to
 * perform work in a worker thread and
 * delivers it the parameters given
 * to AsyncTask.execute() */
    protected Bitmap doInBackground(String... urls) {
        return loadImageFromNetwork(urls[0]);
    }

    /** The system calls this to perform
 * work in the UI thread and delivers
 * the result from doInBackground() */
    protected void onPostExecute(Bitmap result) {
        mImageView.setImageBitmap(result);
    }
}
Random Art with AsyncTask

• Add progress bar and button for new art
• create a Bitmap and draw to that
• <Integer, Integer, Bitmap>
Just One More
Loaders

- Loader classes introduced in API 11
- Help asynchronously load data from content provider or network for Activity or Fragment
- monitor data source and deliver new results when content changes
- multiple classes to work with