CS260 Intro to Java & Android
09.AndroidAdvUI (Part I)

Winter 2015
Creating TicTacToe for Android

- We are going to begin to use everything we’ve learned thus far to produce the shell of an Android TicTacToe game
TicTacToe for Android
Step #1
Create Initial App

- Create TicTacToeAndroid using the default settings for an Android project
- Delete the Hello World textView widget
- Copy over the UI from your Game Skeleton application which should look like the following screen
TicTacToe for Android

Step #1
Create Initial App
TicTacToe for Android
Step #2
Add an Array Resource

- Create a string array in strings.xml where the string-array name is gameDifficulty and the items are:
  - Easy
  - Medium
  - Hard
TicTacToe for Android
Step #3
Create Ask Difficulty Dialog

```java
private void newGameDialog ()
{
    new AlertDialog.Builder (this)
        .setTitle (R.string.sGameDifficulty)
        .setItems (R.array.gameDifficulty, new DialogInterface.OnClickListener()
        {
            public void onClick (DialogInterface dialog, int difficultyLevel)
            {
                startGame (difficultyLevel);
            }
        }).show ();
}

private void startGame (int difficultyLevel)
{
}
```
Start the game

- In the private method startGame that accepts the difficulty level, use the Log.d method to display the difficulty level in the logcat window based on the difficulty level the user selected.

- Hook up the New Game button so that when the button is pressed:
  - the Game Difficulty dialogue appears allowing the user to enter Easy (0), Medium (1), or Hard (2)
  - after the user enters their selection, the difficulty value shows up in the logcat window
Graphics

- We will explore still graphics and then graphic animation
- Android provides libraries to perform 2D and 3D graphics
- android.graphics provides low-level graphics drawing tools such as
  - canvases
  - color filters
  - points
  - rectangles
Android colors

- Represented with four 8-bit numbers (ARGB)
  - alpha - measures transparency where 0 is completely transparent and 255 is completely opaque
  - red
  - green
  - blue

```java
int color = Color.BLACK;
int color = Color.argb (127, 0, 255, 255);
```
Accessing Color Information

- It is best to define your colors in an xml file
  ```xml
  <?xml version="1.0" encoding="UTF-8"?>
  <resources>
    <color name="steelblue">#4682b4</color>
    <color name="navy">#000080</color>
  </resources>
  ```

- Note: Four values is ARGB while three values is RGB where A is fully opaque

- Colors can be accessed by `int color = getResources().getColor(R.color.navy);`
Paint

- The Paint class holds information about
  - style and color
  - how to draw geometries, text, and bitmaps
- Before drawing a color, it is common to set the color with the method `setColor(Color.BLUE);` which is part of the Paint class
The canvas is a surface to draw on

Android framework APIs provide 2D drawing APIs to:

a) render your own graphics on a canvas - need to call onDraw() method passing a Canvas

b) modify (customizing) existing Views - draw graphics or animations into an existing View

a) is best for simple graphics with no animation

b) is best for video games with complex animation and frequent redraws
Basic Display

- An Activity provides the UI for the display screen
- An Activity hosts a View
- A View hosts a Canvas
- The `onDraw()` method can be overridden to perform custom drawing on the Canvas
public class CustomDrawableView extends View {
    private ShapeDrawable mDrawable;
    public CustomDrawableView (Context context) {
        super (context);
        int x = 10, y = 10, width = 300, height = 50;

        mDrawable = new ShapeDrawable (new OvalShape());
        mDrawable.getPaint ().setColor (0xff74AC23);
        mDrawable.setBounds (x, y, x + width, y + height);
    }
    protected void onDraw (Canvas canvas) {
        mDrawable.draw (canvas);
    }
Drawing Custom View

CustomDrawableView mCustomDrawableView;

protected void onCreate (Bundle savedInstanceState)
{
    super.onCreate (savedInstanceState);
    mCustomDrawableView = new CustomDrawableView (this);
    setContentView (mCustomDrawableView);
}

Let’s draw a simple graphic in our Android game before getting into more complicated graphics.

Create a new class called CustomDrawableView in your tictactoeandroid package. Place the previous CustomDrawableView code in this class.

For now, comment out displaying the TicTacToe main activity and simply write the code necessary to display a new CustomDrawableView instead.

You should end up with the UI on the following slide. If you get this, check your updated TicTacToe into subversion.
TicTacToe for Android
Step #4
CustomView Modifications

• You can do any kind of drawing in the onDraw of a custom view.

• The following slides allow you to draw the lines of a TicTacToe board

• See if you can get this to work
Add Instance Variables

private Paint mBackground = new Paint();
private Paint mDarkLines = new Paint();
private float mNUMBER_OF_RECTANGLES = 3f;
private int mRectangleHeight, mRectangleWidth;
Add onDraw Logic

```java
mBackground.setColor (getResources ().getColor (R.color.teal));
canvas.drawRect (0, 0, getWidth (), getHeight (), mBackground);
// Compute mRectangleHeight and mRectangleWidth here

mDarkLines.setColor (Color.BLACK);

for (int i = 0; i < (int) mNUMBER_OF_RECTANGLES; i++)
{
    canvas.drawLine (0, i * mRectangleHeight, getWidth (),
                    i * mRectangleHeight, mDarkLines);
    canvas.drawLine (i * mRectangleWidth, 0,
                    i * mRectangleWidth, getHeight (), mDarkLines);
}
```
Result