CS260 Intro to Java & Android
05.Android UI

Fall 2011

User Interface

- UIs in Android are built using View and ViewGroup objects
- A View is the base class for subclasses called “widgets”
- widget is a fully implemented UI object
- widget examples include
  - text field
  - button
  - textbox

View Class

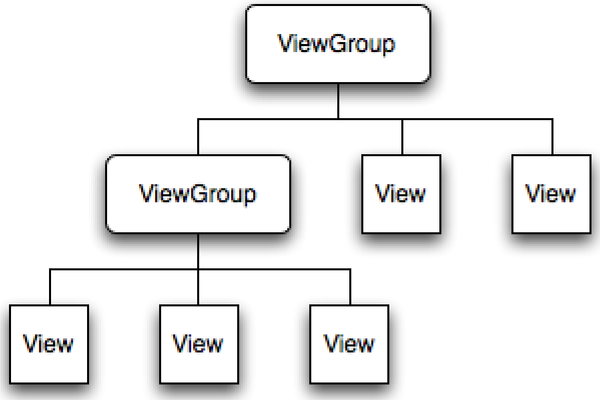
- A View class is the basic building block for UI components
- A View
  - occupies a rectangular area on the screen
  - has measurement information
  - has layout information
  - has drawing information
  - handles events such as scrolling & key interactions

ViewGroup Class

- A ViewGroup
  - extends a View
  - can contain other views (called children)
  - is the base class for layouts and view containers

View Hierarchy

- An Activity's UI is defined using View and View group nodes
- The hierarchy tree can be complex or simple
- Design before implementing your UI



Using Views

- Views in a window are arranges in a single tree
- Views can be added
  - from code
  - from a view in an XML layout file
- Common operations on a tree of views
  - set properties (e.g. set the text of a TextView)
  - set the focus of a particular view
  - set up listeners for when something happens to a view object
  - set the visibility of a view object

setContentView

- The setContentView () method attaches the view hierarchy tree to the screen for rendering
- The root node requests that each child node draw itself
- Each ViewGroup requests that each child node draw itself

More View Hierarchy Facts

- children can make certain requests (e.g. size, location, …), but the parent has the final say
- Views are instantiated from the root node down the tree
- If elements overlap, the last element drawn is displayed

Android User Interfaces

- We are going to create the UI for a generic game
- The game will have:
  1. A title (string)
  2. New Game (button)
  3. Continue (button)
  4. Rules (button)
  5. About (button)
  6. Exit (button)

Game Project

- Using Eclipse, create a game project
  - Project name: **Game**
  - Build Target: **Android 1.5**
  - Application name: **Game**
  - Package name: **edu.pacificu.cs.Game**
- Build the project
- Run the application in the AVD1.5 emulator

JIT Java - Files & Packages

- Java class definition
  - A Java class definition must be fully enclosed within a single file
  - The file name must match exactly the class definition name
  - Class definitions are usually in their own files
  - Only one public class is allowed per file
- Java package
  - a package is essentially a library
  - a package contains related class files in the same directory

JIT Java - Import

- The import directive tells the compiler where to look for class definitions

```
package edu.pacificu.cs.Game;

import android.app.Activity;
import android.os.Bundle;

public class Game extends Activity
{
  /** Called when the activity is first created. */
  @Override
  public void onCreate (Bundle savedInstanceState)
  {
      super.onCreate (savedInstanceState);
      setContentView (R.layout.main);
  }
}
```

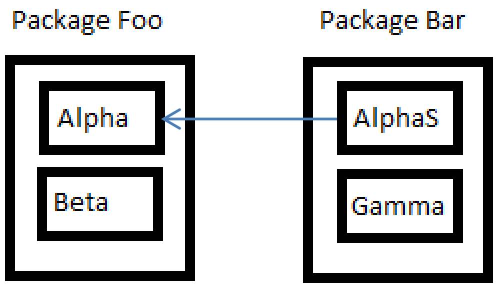
JIT Java - Class Access & Packages

- class - if declared public, the class is visible to all classes. If no modifier exists, the class is visible to all classes within the same package
- class "member" access by other classes. A class member can have a modifier of: public, protected, no modifier, or private

Access Levels				
Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

JIT Java - Package Example

Consider the following figure:



The table below shows where the members of the class Alpha are visible for each of the access modifiers that can be applied

		Visibility		
Modifier	Alpha	Beta	AlphaS	Gamma
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

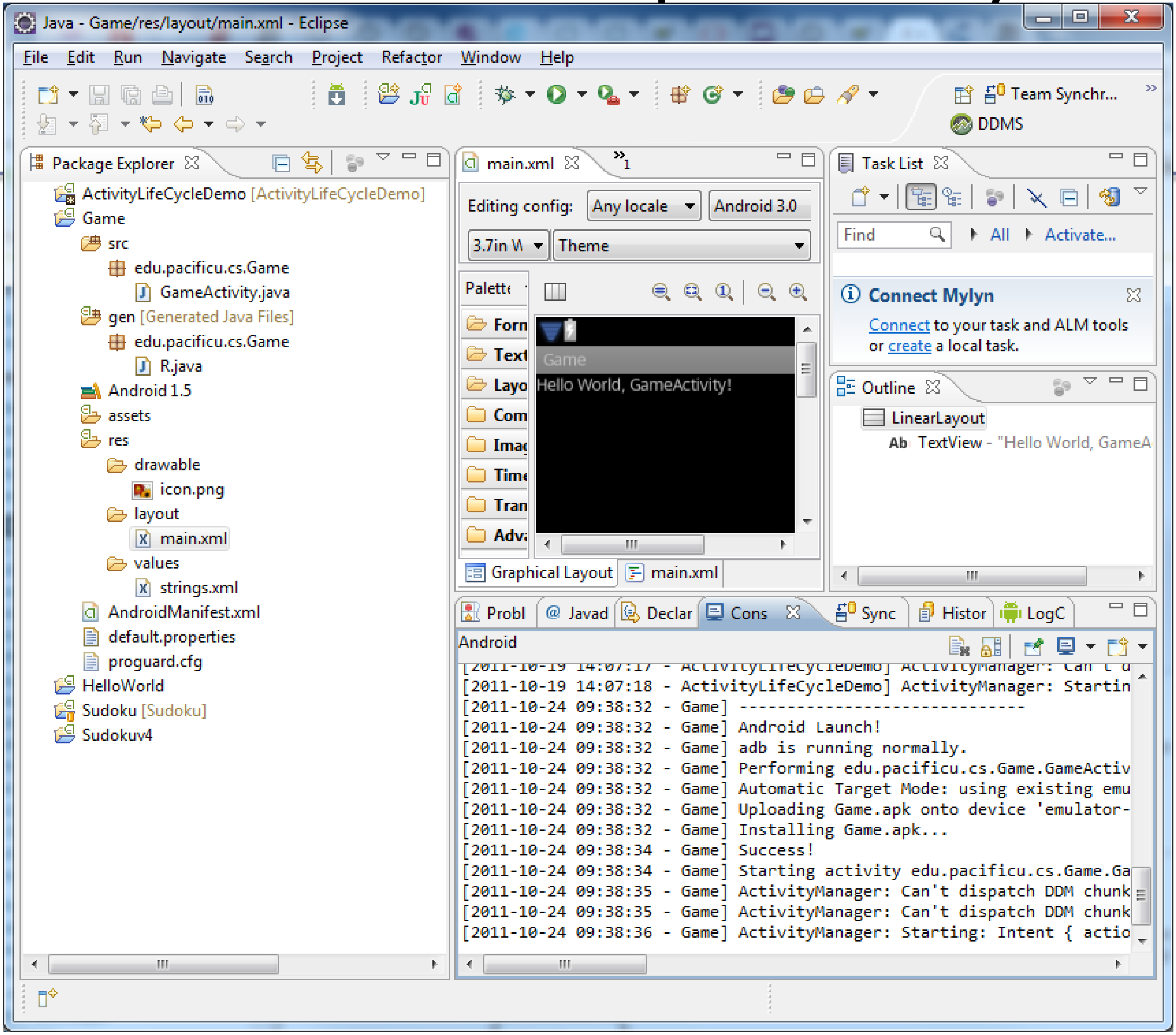
UI Design

- UIs can be designed in one of two ways
  - procedurally - meaning “ in code”
  - declaratively - meaning using some descriptive language (e.g. html, xml, …) an no code
- Our initial game will use a declarative approach

Game Activity

```
package edu.pacificu.cs.Game;

import android.app.Activity;
import android.os.Bundle;

public class Game extends Activity
{
  /** Called when the activity is first created. */
  @Override
  public void onCreate (Bundle savedInstanceState)
  {
      super.onCreate (savedInstanceState);
      setContentView (R.layout.main);
  }
}
```

main.xml in Graphical Layout



```
[2011-10-19 14:07:17 - ActivityLifeCycleDemo] ActivityManager: Can't d
[2011-10-19 14:07:18 - ActivityLifeCycleDemo] ActivityManager: Startin
[2011-10-24 09:38:32 - Game] ------------------------------
[2011-10-24 09:38:32 - Game] Android Launch!
[2011-10-24 09:38:32 - Game] adb is running normally.
[2011-10-24 09:38:32 - Game] Performing edu.pacificu.cs.Game.GameActiv
[2011-10-24 09:38:32 - Game] Automatic Target Mode: using existing emu
[2011-10-24 09:38:32 - Game] Uploading Game.apk onto device 'emulator-
[2011-10-24 09:38:32 - Game] Installing Game.apk...
[2011-10-24 09:38:34 - Game] Success!
[2011-10-24 09:38:34 - Game] Starting activity edu.pacificu.cs.Game.Ga
[2011-10-24 09:38:35 - Game] ActivityManager: Can't dispatch DDM chunk
[2011-10-24 09:38:35 - Game] ActivityManager: Can't dispatch DDM chunk
[2011-10-24 09:38:36 - Game] ActivityManager: Starting: Intent { actio
```

main.xml as xml text

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

Android's Use of XML

- XML is used when writing Android applications
- Android resource compiler (aapt) compiles xml code into a compressed binary format
- Compressed binary format stored on device, not xml code
- xml code (as compressed binary format) is instantiated (inflated) when necessary

What is this XML

```
<?xml version="1.0" encoding="utf-8"?>
```

- `<?xml` beginning of XML declaration
- `version="1.0"` document is written for version 1.0 XML parser
- `encoding="utf-8"` uses utf-8 which is basically ASCII with escape characters
- `?>` end of XML tag

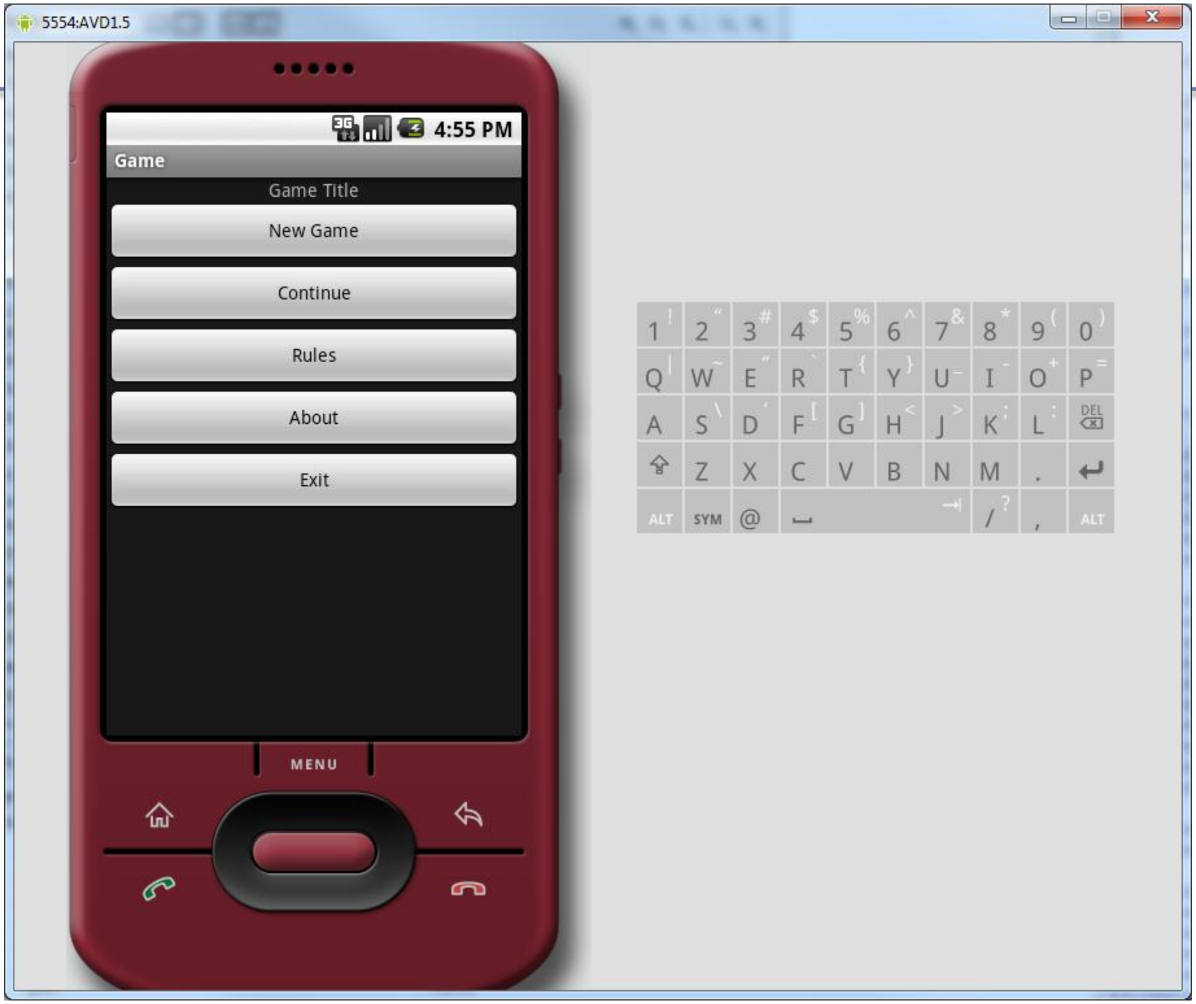
Linear Layout

- What is a layout?
  - container for one or more child objects
  - behavior to position child objects on the screen
- Common layouts
  - FrameLayout
  - LinearLayout
  - RelativeLayout
  - TableLayout

Parameters Common to Layouts

```
xmlns:android="http://schemas.android.com/apk/res/android“ defines
Android’s XML namespace

xmlns:android="http://schemas.android.com/apk/res/android“

android:orientation="vertical“

android:layout_width="fill_parent“

android:layout_height="fill_parent"
```

Create the following UI



UI Design Specifics

1. Button names are btnNewGame, btnContinue, btnRules, btnAbout, and btnExit
2. String name & values are:
   - sNewGame is New Game
   - sContinue is Continue
   - sRules is Rules
   - sAbout is About
   - sExit is Exit

More XML

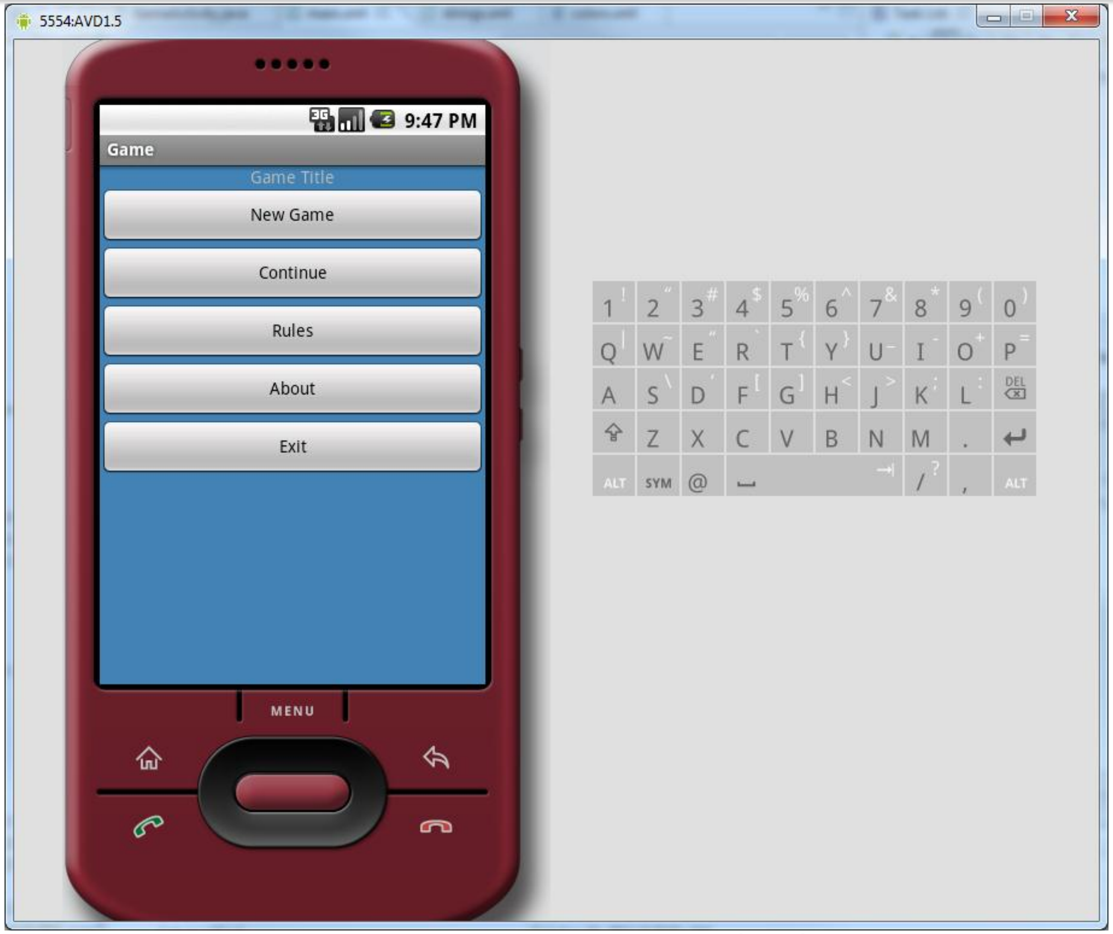
- What if we want to change the background color?

1. Create an xml color definition resource in the values folder called **colors.xml** as follows:

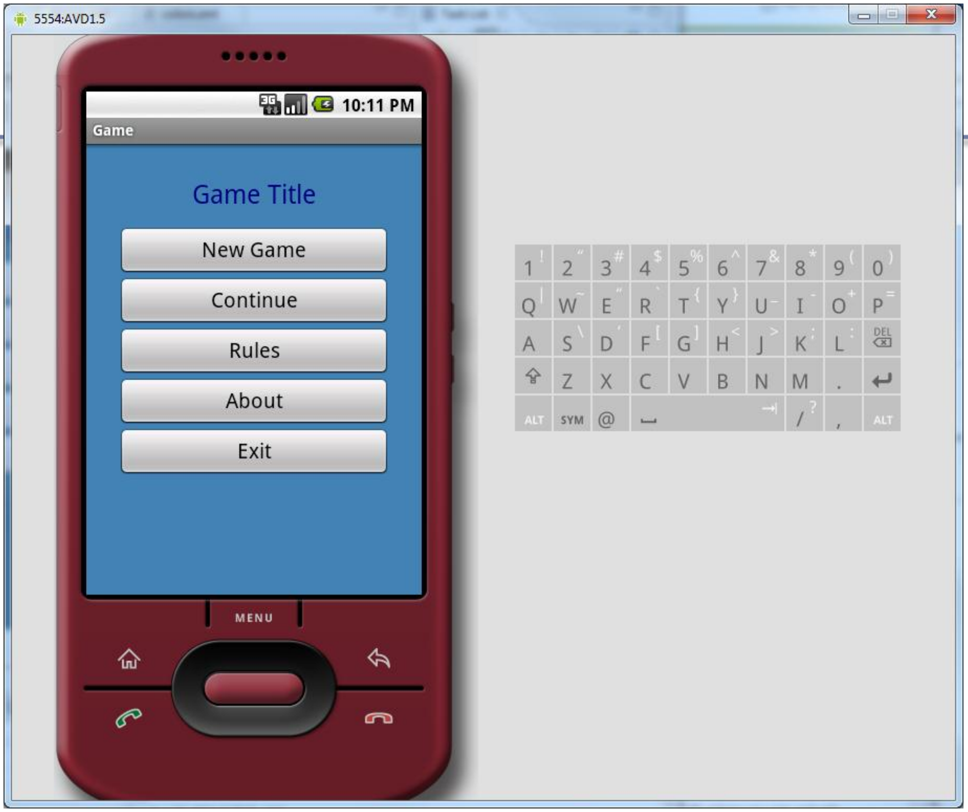
```xml
<?xml version="1.0" encoding="UTF-8"?>
<resources>
<color name="steelblue">#4682b4</color>
<color name="navy">#000080</color>
</resources>
```

2. Change the background of the LinearLayout to reference the background color
3. The next slide shows the game using a steelblue background

Game Using Colored Background

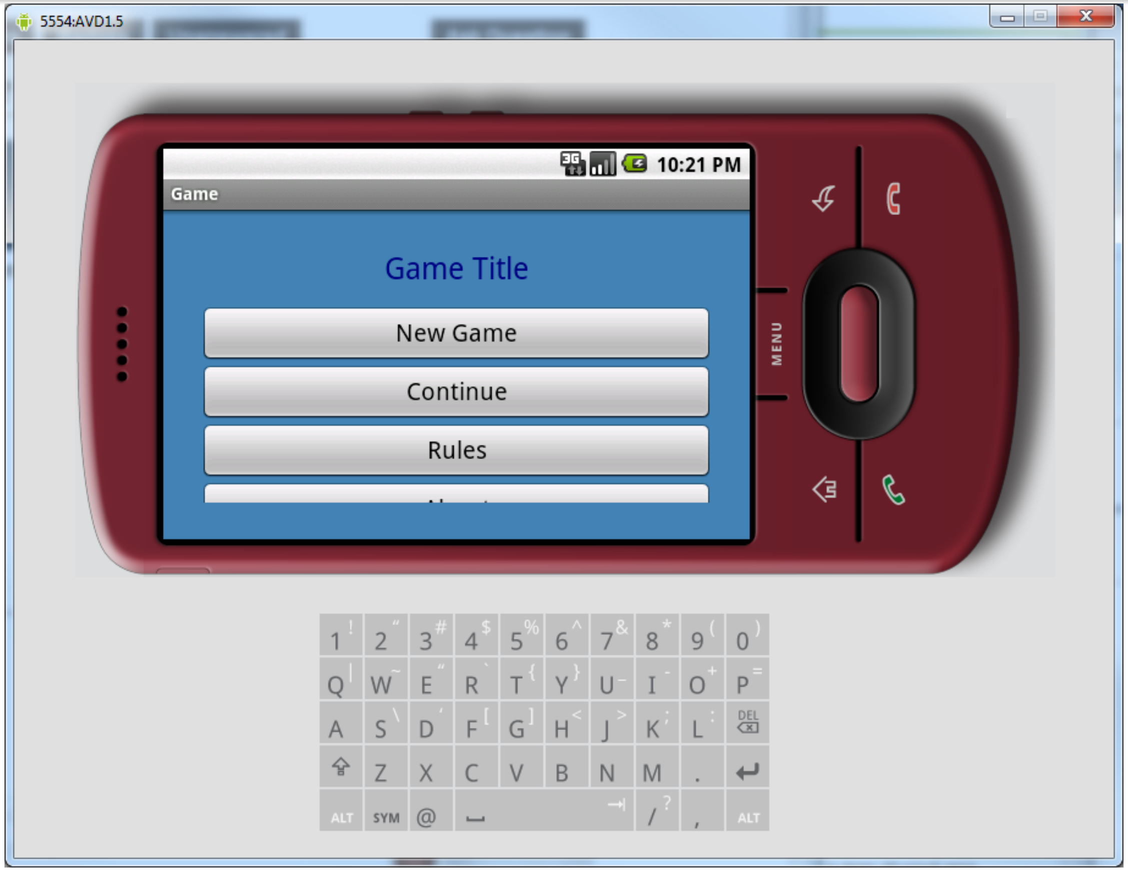


Modify Application



- Hint: Read Supporting Multiple Screens … specifically dp (density-independent pixels) and sp (scale-independent pixels)

Switch to Portrait


5554:AVD1.5
10:21 PM
Game
Game Title
New Game
Continue
Rules
MENU