Android
Layout Managers
Avoiding Most XML
LayoutManagers

How things get put on the screen

• Every Activity and Fragment has a layout manager
• Everything goes into a layout manager
  • including other layout managers!
• Note hierarchy View -> ViewGroup -> [LayoutManager]
  • a view is any viewable thing in android
    • every interface element
  • ViewGroup is a container of a collection of Views
• Since LayoutManagers inherit from View they are visible on screen
• Since they inherit from ViewGroup they can hold other Views
Layout Managers

• AbsoluteLayout
  • Do not use — deprecated since Android v3

• RelativeLayout
  • Fairly similar to CSS positions. Everything follows from everything else

• LinearLayout
  • One row (or one column at a time)

• FrameLayout — think picture frame
  • designed to block out an area on the screen to display a single item.

• ConstraintLayout
  • Allows better control than relative layout

• CoordinatorLayout
  • “Frame layout on steroids” —- I actually have no idea what this means!
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    // set up the initial layout
    RelativeLayout ll = new RelativeLayout(getContext());
    ll.setId(View.generateViewId());
    return ll;
}

private int randomColor() {
    Random r = new Random();
    return Color.rgb(r.nextInt(255), r.nextInt(255), r.nextInt(255));
}

public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    doRL((RelativeLayout)view);
}

private void doRL(RelativeLayout ll) {
    do1(ll);
    //do1i(ll);
    //doN(ll, 20);
    //doNRule(ll, 20);
    //doTurn(ll, 20);
}

void do1(RelativeLayout ll) {
    float density = getResources().getDisplayMetrics().density;
    TextView v = new TextView(getContext());
    v.setBackgroundColor(randomColor());
    v.setText("" + 0);
    v.setGravity(Gravity.CENTER);
    v.setId(View.generateViewId());
    RelativeLayout.LayoutParams lp = new RelativeLayout.LayoutParams((int) (100 * density), (int) (100 * density));
    ll.addView(v, lp);
}
What went wrong

• My square is not square and the 0 did not appear

• Problem, the layout goes behind the menu bar, so either need to get rid of menu bar or ensure that top of layout is below menu bar
Put many items

void doN(RelativeLayout rl, int maxx) {
    // unchanged code not shown
    for (int i = 0; i < maxx; i++) {
        TextView v = new TextView(getContext());
        v.setBackgroundColor(randomColor());
        v.setText("" + i);
        v.setGravity(Gravity.CENTER);
        v.setId(View.generateViewId());
        RelativeLayout.LayoutParams lp = new
        RelativeLayout.LayoutParams((int) (100 * density),
        (int) (100 * density));
        rl.addView(v, lp);
    }
}

• Why are the 20 items not showing?
• They are, they are just stacked on top of each other
• By default, relative layout put everything in top corner
• Can be very useful — if you want to stack interface elements
Getting Creative

```java
void doTurn(RelativeLayout rl, int maxx) {
    float density = getResources().getDisplayMetrics().density;
    TypedValue tv = new TypedValue();
    if (getContext().getTheme().resolveAttribute(android.R.attr.actionBarSize, tv, true)) {
        int actionBarHeight = TypedValue.complexToDimensionPixelSize(tv.data, getResources().getDisplayMetrics());
        rl.setPadding(0, actionBarHeight, 0, 0);
    }
    View pp = null;
    for (int i = 0; i < maxx; i++) {
        TextView v = new TextView(getContext());
        v.setBackgroundColor(randomColor());
        v.setText("" + i);
        v.setGravity(Gravity.CENTER);
        v.setId(View.generateViewId());
        RelativeLayout.LayoutParams lp = new RelativeLayout.LayoutParams((int) (100 * density), (int) (100 * density));
        if (pp != null) {
            if (i < 6)
                lp.addRule(RelativeLayout.BELOW, pp.getId());
            else {
                lp.addRule(RelativeLayout.END_OF, pp.getId());
                if (i == 6) lp.addRule(RelativeLayout.BELOW, pp.getId());
            }
        }
        rl.addView(v, lp);
        pp = v;
    }
}
```
Show Many items

• Rules
  • define where things should appear either
    • with respect to the entire layout (e.g. top-right)
    • some previously positioned element
      • hence the name RelativeLayout

RelativeLayout.LayoutParams lp = new RelativeLayout.LayoutParams((int) (100 * density), (int) (100 * density));
if (pp!=null) {
  lp.addRule(RelativeLayout.ABOVE, pp.getId());
}
else {
  lp.addRule(RelativeLayout.ASSIGN_PARENT_BOTTOM);
  lp.addRule(RelativeLayout.ASSIGN_PARENT_RIGHT);
}
rl.addView(v, lp);
• Most of previous examples could have been done in XML. These, not so much

• How do you do the one on left?
  • Only rule you need is
    ```java
    lp.addRule(RelativeLayout.CENTER_IN_PARENT);
    ```

• Consider just putting in the innermost square with one surrounding square.
If you absolutely must

• AbsoluteLayout was deprecated because using it is normally a bad idea.

• But sometimes needs must

• for instance, the button on all of the previous slides

• By manipulating margins and rules in relative layout you get absolute positioning

```java
FloatingActionButton fab = new FloatingActionButton(this);
fab.setImageResource(R.drawable.arrow_down_float);
fab.show();
RelativeLayout.LayoutParams clp = new RelativeLayout.LayoutParams(
    ViewGroup.LayoutParams.WRAP_CONTENT,
    ViewGroup.LayoutParams.WRAP_CONTENT);
clp.addRule(RelativeLayout.ALIGN_PARENT_BOTTOM);
clp.addRule(RelativeLayout.ALIGN_PARENT_END);
float density = getResources().getDisplayMetrics().density;
clp.setMargins(0, 0, (int)(10*density), (int)(10*density));
vv.addView(fab, clp);
```
LinearLayout
Because relative layouts can be a pain

- Align in a row or a column
- Items are added left to right or top to bottom

```java
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    System.out.println("Linear Create");
    LinearLayout ll = new LinearLayout(getContext());
    ll.setId(View.generateViewId());
    ll.setOrientation(LinearLayout.VERTICAL);
    return ll;
}
```

```java
public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    LinearLayout ll = (LinearLayout) view;
    TypedValue tv = new TypedValue();
    if (getContext().getTheme().resolveAttribute(android.R.attr.actionBarSize, tv, true)) {
        int actionBarHeight = TypedValue.complexToDimensionPixelSize(tv.data, getResources().getDisplayMetrics());
        ll.setPadding(0, actionBarHeight, 0, 0);
    }
    doVertical(ll, 5);
}
```

```java
public void doVertical(LinearLayout ll, int numComponents) {
    ll.setOrientation(LinearLayout.VERTICAL);
    for (int i = 1; i <= numComponents; i++) {
        TextView v = new TextView(getContext());
        v.setBackgroundColor(randomColor());
        v.setText("" + i);
        v.setId(View.generateViewId());
        LinearLayout.LayoutParams lp = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, 0, i);
        ll.addView(v, lp);
    }
}
```

MATCH_PARENT means make item the same dimension as parent
WRAP_CONTENT means make item as small as possible to fit content

LinearLayout allows third param, the weight. In vertical layout this allows layout to expand to fill vertical space; sharing to items according to relative
LinearLayout
Horizontally

• Can mix & match weights and absolutes

```java
public void doHorizontal(LinearLayout ll, int numComponents) {
    ll.setOrientation(LinearLayout.HORIZONTAL);
    float density = getResources().getDisplayMetrics().density;
    for (int i = 1; i <= numComponents; i++) {
        TextView v = new TextView(getContext());
        v.setBackgroundColor(randomColor());
        v.setText("" + i);
        v.setGravity(Gravity.CENTER);
        v.setId(View.generateViewId());
        LinearLayout.LayoutParams lp;
        if (i%2==0)
            lp = new LinearLayout.LayoutParams(0, ViewGroup.LayoutParams.MATCH_PARENT, i);
        else
            lp = new LinearLayout.LayoutParams((int)(40*density), ViewGroup.LayoutParams.MATCH_PARENT, 0);
        ll.addView(v, lp);
    }
}
```

Set R2L or T2B. Here L2R

Since L2R, and using weights to determine widths, need to set width to 0, and height to MATCH_PARENT.

What would this look like if WRAP_CONTENT rather than MATCH_PARENT?
Getting Really Ugly

Use both Horizontal and Vertical layouts

```java
public void doVandH(LinearLayout ll, int rows) {
    ll.setOrientation(LinearLayout.VERTICAL);
    for (int i=1; i<rows+1; i++) {
        LinearLayout hll = new LinearLayout(getContext());
        ll.addView(hll, new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, 0, i));
        doHorizontal(hll, 5+i);
    }
}
```

• Another interface design not practical to write in XML
• But should this ever be used?
  • Probably not.
Doing everything in Code

• Why Not?
  • can slow development as you have to restart process and navigate to screen
  • can make for difficult to read code
  • difficult to hand off to a someone who expects XML layouts
• But
  • some interfaces cannot easily be rendered in XML
  • In small shops you only have to learn one thing
  • dynamic / user-defined interfaces pretty much require it.
From LabSix base to all code

• Everything in the fragments we have already discussed.
• FloatingActionButton is in Main Activity .. discussed how to do that .. mostly
• So
  • Initialize
  • Add Toolbar

Use XML in values directory. Could even get rid of this by defining a “constants” class

```java
public static int MAIN_ACTIVITY_ID = 22346;
public static int MAIN_ACTIVITY_TOOLBAR = 22347;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    RelativeLayout vv = new RelativeLayout(this);
    vv.setId(MAIN_ACTIVITY_ID);
    vv.setLayoutParams(new ViewGroup.LayoutParams(getResources().getDisplayMetrics().widthPixels, getResources().getDisplayMetrics().heightPixels));
    setContentView(vv);

    TypedValue tv = new TypedValue();
    int actionBarHeight = 40;
    if (this.getTheme().resolveAttribute(android.R.attr.actionBarSize, tv, true)) {
        actionBarHeight = TypedValue.complexToDimensionPixelSize(tv.data, getResources().getDisplayMetrics());
    }

    Toolbar toolbar = new Toolbar(this);
    toolbar.setId(MAIN_ACTIVITY_TOOLBAR);
    ViewGroup.LayoutParams layoutParams = new ViewGroup.LayoutParams(MATCH_PARENT, actionBarHeight);
    toolbar.setLayoutParams(layoutParams);
    toolbar.setBackgroundColor(ContextCompat.getColor(this, R.color.purple_200));
    toolbar.setTitle(R.string.app_name + "!!!");
    toolbar.setVisibility(View.VISIBLE);
    vv.addView(toolbar, 0);
    setSupportActionBar(toolbar);
```
All Code

```java
FragmentManager fragmentManager = this.getSupportFragmentManager();
FragmentTransaction transaction = fragmentManager.beginTransaction();
transaction.replace(MAIN_ACTIVITY_ID, new RelativeFragment(), null);
transaction.addToBackStack(null);
transaction.commit();

int tt = fragmentManager.getBackStackEntryCount();
for(int i = 0; i < tt; ++i) {
    fragmentManager.popBackStack();
}

public void onBackPressed() {
    fragmentManager = this.getSupportFragmentManager();
    int tt = fragmentManager.getBackStackEntryCount();
    Log.i("THIS", "Back stack " + tt);
    if (tt>1)
        super.onBackPressed();
}

public boolean onCreateOptionsMenu(Menu menu){
    MenuItem edit_item = menu.add(0, 1, 1, "Hello");
    edit_item setShowAsActionFlags(MenuItem.SHOW_AS_ACTION_ALWAYS);

    MenuItem delete_item = menu.add(0, 2, 2, "Geoff");
    delete_item setShowAsActionFlags(MenuItem.SHOW_AS_ACTION_ALWAYS);

    return super.onCreateOptionsMenu(menu);
}
```

• Navigation

• Handling the Android Back Button

• Items in the action bar