Dashboard Layout.Java

|  |
| --- |
| /\* |
|  | \* Copyright 2011 Google Inc. |
|  | \* |
|  | \* Licensed under the Apache License, Version 2.0 (the "License"); |
|  | \* you may not use this file except in compliance with the License. |
|  | \* You may obtain a copy of the License at |
|  | \* |
|  | \* http://www.apache.org/licenses/LICENSE-2.0 |
|  | \* |
|  | \* Unless required by applicable law or agreed to in writing, software |
|  | \* distributed under the License is distributed on an "AS IS" BASIS, |
|  | \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. |
|  | \* See the License for the specific language governing permissions and |
|  | \* limitations under the License. |
|  | \*/ |
|  |  |
|  | package cubi.rafi; |
|  |  |
|  | import android.content.Context; |
|  | import android.util.AttributeSet; |
|  | import android.view.View; |
|  | import android.view.ViewGroup; |
|  |  |
|  | /\*\* |
|  | \* Custom layout that arranges children in a grid-like manner, optimizing for even horizontal and |
|  | \* vertical whitespace. |
|  | \*/ |
|  | public class DashboardLayout extends ViewGroup { |
|  |  |
|  | private static final int UNEVEN\_GRID\_PENALTY\_MULTIPLIER = 10; |
|  |  |
|  | private int mMaxChildWidth = 0; |
|  | private int mMaxChildHeight = 0; |
|  |  |
|  | public DashboardLayout(Context context) { |
|  | super(context, null); |
|  | } |
|  |  |
|  | public DashboardLayout(Context context, AttributeSet attrs) { |
|  | super(context, attrs, 0); |
|  | } |
|  |  |
|  | public DashboardLayout(Context context, AttributeSet attrs, int defStyle) { |
|  | super(context, attrs, defStyle); |
|  | } |
|  |  |
|  | @Override |
|  | protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) { |
|  | mMaxChildWidth = 0; |
|  | mMaxChildHeight = 0; |
|  |  |
|  | // Measure once to find the maximum child size. |
|  |  |
|  | int childWidthMeasureSpec = MeasureSpec.makeMeasureSpec( |
|  | MeasureSpec.getSize(widthMeasureSpec), MeasureSpec.AT\_MOST); |
|  | int childHeightMeasureSpec = MeasureSpec.makeMeasureSpec( |
|  | MeasureSpec.getSize(widthMeasureSpec), MeasureSpec.AT\_MOST); |
|  |  |
|  | final int count = getChildCount(); |
|  | for (int i = 0; i < count; i++) { |
|  | final View child = getChildAt(i); |
|  | if (child.getVisibility() == GONE) { |
|  | continue; |
|  | } |
|  |  |
|  | child.measure(childWidthMeasureSpec, childHeightMeasureSpec); |
|  |  |
|  | mMaxChildWidth = Math.max(mMaxChildWidth, child.getMeasuredWidth()); |
|  | mMaxChildHeight = Math.max(mMaxChildHeight, child.getMeasuredHeight()); |
|  | } |
|  |  |
|  | // Measure again for each child to be exactly the same size. |
|  |  |
|  | childWidthMeasureSpec = MeasureSpec.makeMeasureSpec( |
|  | mMaxChildWidth, MeasureSpec.EXACTLY); |
|  | childHeightMeasureSpec = MeasureSpec.makeMeasureSpec( |
|  | mMaxChildHeight, MeasureSpec.EXACTLY); |
|  |  |
|  | for (int i = 0; i < count; i++) { |
|  | final View child = getChildAt(i); |
|  | if (child.getVisibility() == GONE) { |
|  | continue; |
|  | } |
|  |  |
|  | child.measure(childWidthMeasureSpec, childHeightMeasureSpec); |
|  | } |
|  |  |
|  | setMeasuredDimension( |
|  | resolveSize(mMaxChildWidth, widthMeasureSpec), |
|  | resolveSize(mMaxChildHeight, heightMeasureSpec)); |
|  | } |
|  |  |
|  | @Override |
|  | protected void onLayout(boolean changed, int l, int t, int r, int b) { |
|  | int width = r - l; |
|  | int height = b - t; |
|  |  |
|  | final int count = getChildCount(); |
|  |  |
|  | // Calculate the number of visible children. |
|  | int visibleCount = 0; |
|  | for (int i = 0; i < count; i++) { |
|  | final View child = getChildAt(i); |
|  | if (child.getVisibility() == GONE) { |
|  | continue; |
|  | } |
|  | ++visibleCount; |
|  | } |
|  |  |
|  | if (visibleCount == 0) { |
|  | return; |
|  | } |
|  |  |
|  | // Calculate what number of rows and columns will optimize for even horizontal and |
|  | // vertical whitespace between items. Start with a 1 x N grid, then try 2 x N, and so on. |
|  | int bestSpaceDifference = Integer.MAX\_VALUE; |
|  | int spaceDifference; |
|  |  |
|  | // Horizontal and vertical space between items |
|  | int hSpace = 0; |
|  | int vSpace = 0; |
|  |  |
|  | int cols = 1; |
|  | int rows; |
|  |  |
|  | while (true) { |
|  | rows = (visibleCount - 1) / cols + 1; |
|  |  |
|  | hSpace = ((width - mMaxChildWidth \* cols) / (cols + 1)); |
|  | vSpace = ((height - mMaxChildHeight \* rows) / (rows + 1)); |
|  |  |
|  | spaceDifference = Math.abs(vSpace - hSpace); |
|  | if (rows \* cols != visibleCount) { |
|  | spaceDifference \*= UNEVEN\_GRID\_PENALTY\_MULTIPLIER; |
|  | } |
|  |  |
|  | if (spaceDifference < bestSpaceDifference) { |
|  | // Found a better whitespace squareness/ratio |
|  | bestSpaceDifference = spaceDifference; |
|  |  |
|  | // If we found a better whitespace squareness and there's only 1 row, this is |
|  | // the best we can do. |
|  | if (rows == 1) { |
|  | break; |
|  | } |
|  | } else { |
|  | // This is a worse whitespace ratio, use the previous value of cols and exit. |
|  | --cols; |
|  | rows = (visibleCount - 1) / cols + 1; |
|  | hSpace = ((width - mMaxChildWidth \* cols) / (cols + 1)); |
|  | vSpace = ((height - mMaxChildHeight \* rows) / (rows + 1)); |
|  | break; |
|  | } |
|  |  |
|  | ++cols; |
|  | } |
|  |  |
|  | // Lay out children based on calculated best-fit number of rows and cols. |
|  |  |
|  | // If we chose a layout that has negative horizontal or vertical space, force it to zero. |
|  | hSpace = Math.max(0, hSpace); |
|  | vSpace = Math.max(0, vSpace); |
|  |  |
|  | // Re-use width/height variables to be child width/height. |
|  | width = (width - hSpace \* (cols + 1)) / cols; |
|  | height = (height - vSpace \* (rows + 1)) / rows; |
|  |  |
|  | int left, top; |
|  | int col, row; |
|  | int visibleIndex = 0; |
|  | for (int i = 0; i < count; i++) { |
|  | final View child = getChildAt(i); |
|  | if (child.getVisibility() == GONE) { |
|  | continue; |
|  | } |
|  |  |
|  | row = visibleIndex / cols; |
|  | col = visibleIndex % cols; |
|  |  |
|  | left = hSpace \* (col + 1) + width \* col; |
|  | top = vSpace \* (row + 1) + height \* row; |
|  |  |
|  | child.layout(left, top, |
|  | (hSpace == 0 && col == cols - 1) ? r : (left + width), |
|  | (vSpace == 0 && row == rows - 1) ? b : (top + height)); |
|  | ++visibleIndex; |
|  | } |
|  | } |
|  | } |

Dimen.xml

|  |
| --- |
| <!-- |
|  | Copyright 2011 Google Inc. |
|  |  |
|  | Licensed under the Apache License, Version 2.0 (the "License"); |
|  | you may not use this file except in compliance with the License. |
|  | You may obtain a copy of the License at |
|  |  |
|  | http://www.apache.org/licenses/LICENSE-2.0 |
|  |  |
|  | Unless required by applicable law or agreed to in writing, software |
|  | distributed under the License is distributed on an "AS IS" BASIS, |
|  | WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. |
|  | See the License for the specific language governing permissions and |
|  | limitations under the License. |
|  | --> |
|  | <resources> |
|  |  |
|  | <dimen name="body\_padding\_medium">10dp</dimen> |
|  | <dimen name="body\_padding\_large">10dp</dimen> |
|  | <dimen name="text\_size\_small">14sp</dimen> |
|  | <dimen name="text\_size\_medium">18sp</dimen> |
|  | <dimen name="text\_size\_large">18sp</dimen> |
|  | <dimen name="text\_size\_xlarge">18sp</dimen> |
|  | <dimen name="speaker\_image\_size">36dp</dimen> |
|  | <dimen name="speaker\_image\_padding">8dp</dimen> |
|  | <dimen name="vendor\_image\_size">100dp</dimen> |
|  |  |
|  | </resources> |

Style.xml

|  |
| --- |
| <!-- |
|  | Copyright 2011 Google Inc. |
|  |  |
|  | Licensed under the Apache License, Version 2.0 (the "License"); |
|  | you may not use this file except in compliance with the License. |
|  | You may obtain a copy of the License at |
|  |  |
|  | http://www.apache.org/licenses/LICENSE-2.0 |
|  |  |
|  | Unless required by applicable law or agreed to in writing, software |
|  | distributed under the License is distributed on an "AS IS" BASIS, |
|  | WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. |
|  | See the License for the specific language governing permissions and |
|  | limitations under the License. |
|  | --> |
|  |  |
|  | <resources> |
|  |  |
|  | <style name="DashboardButton"> |
|  | <item name="android:layout\_gravity">center\_vertical</item> |
|  | <item name="android:layout\_width">wrap\_content</item> |
|  | <item name="android:layout\_height">wrap\_content</item> |
|  | <item name="android:gravity">center\_horizontal</item> |
|  | <item name="android:drawablePadding">2dp</item> |
|  | <item name="android:textStyle">bold</item> |
|  | <item name="android:background">@null</item> |
|  | <item name="android:textSize">@dimen/text\_size\_small</item> |
|  | </style> |
|  |  |
|  | </resources> |

Androidmanifest.xml

|  |
| --- |
| <!-- |
|  | Copyright 2011 Google Inc. |
|  |  |
|  | Licensed under the Apache License, Version 2.0 (the "License"); |
|  | you may not use this file except in compliance with the License. |
|  | You may obtain a copy of the License at |
|  |  |
|  | http://www.apache.org/licenses/LICENSE-2.0 |
|  |  |
|  | Unless required by applicable law or agreed to in writing, software |
|  | distributed under the License is distributed on an "AS IS" BASIS, |
|  | WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. |
|  | See the License for the specific language governing permissions and |
|  | limitations under the License. |
|  | --> |
|  |  |
|  | <resources> |
|  |  |
|  | <style name="DashboardButton"> |
|  | <item name="android:layout\_gravity">center\_vertical</item> |
|  | <item name="android:layout\_width">wrap\_content</item> |
|  | <item name="android:layout\_height">wrap\_content</item> |
|  | <item name="android:gravity">center\_horizontal</item> |
|  | <item name="android:drawablePadding">2dp</item> |
|  | <item name="android:textStyle">bold</item> |
|  | <item name="android:background">@null</item> |
|  | <item name="android:textSize">@dimen/text\_size\_small</item> |
|  | </style> |
|  |  |
|  | </resources> |