Dots – Android Improvements

Andrew Siegle and Tyler Hannan

# Context

The Dots game uses two important features of Android. The dots grid is created completely by an Android paint object. The GameDrawer is a view which is placed in a LinearLayout and contains all of the information needed to redraw the grid after every move. The other feature is the adaption of the layouts into landscape and large formats. The dots game can be played in a phone or table aspect, but the sizes and layouts need to differ for the different devices for proper user experience.

# Drawing

## Architecture

The architecture used in the Dots game is simple and all-containing. The system contains a GameDrawer class which is a simple View which inherits its bounds from a linear layout it resides in which is declared in the project’s layout XML. The drawer is a private instance of the Activity controlling it and the objects it needs to draw are explicitly passed to the GameDrawer when they are updated.

## Process

The GameDrawer overrides the onDraw method of an Android View object. The onDraw method is invoked every time the drawer is invalidated via the drawer.invalidate() method. The palette is cleared on every onDraw() and the GameDrawer draws the lines and squares which it has been given from the activity every time it redraws. This is useful because one of the features of the program is to change the color or the players and since each square is redrawn, each square updated with the new color every time the user changes colors.

## Caveats

It is wrong to assume that the view will redraw consistently. When the programmer wants to redraw a view, he or she must manually tell the view to invalidate. Another caveat is to ensure that the data in the Activity model is up-to-date in the drawer as well. The Dots program kept redundant data and it was imperative to (for example) add a line to the local model and the drawer model. In hind sight, the better way to do this would be to create the GameDrawer View and pass in the Activity and make its model accessible so that the view can draw directly from the up-to-date model in the Activity.

# Sizes and Layouts

## Architecture and Process

The standard Android layout xml resources were used in this process. The different XML files must be contained in specific folders pertaining to the size and orientation the user should experience. The naming scheme for the folders are is as follows:

|  |  |
| --- | --- |
| Folder | Used for |
| Res/layout | Normal layout (default) |
| Res/layout-land | Normal landscape layout |
| Res/layout-large | Screens larger than 4 in |
| Res/layout-large-land | Screens larger than 4 in and in landscape mode |

In the table above, “large” can be replaced for “small”, “medium”, “large”, or “xlarge” for screens larger than 2, 3, 4, and 7 inches respectively. It can also be replaced with “ldpi”, “mdpi”, “hdpi”, and “xdpi” for screens larger than 100, ~125, ~180, and ~275 dpi respectively. Android will automatically detect the dpi or the screen size of the device and will apply the correct layout. If there is not a folder for that specific device, it will default to the normal “layout” folder.

## Drawbacks

Each devices needs a completely different XML layout file for each device /orientation. While this is usually not a problem with small systems, some systems (like those that draw) may need to be designed differently to obtain the correct user experience.