This weekend was Hack Princeton, our school’s biannual hackathon run by Hack Princeton (find out more here, including the list of winners!). Unfortunately, a commitment on Saturday precluded my full-fledged participation; however, I managed to pump out a small tool on Friday: a dynamic graphing (note: **graphing**, not graphical) interface for iOS.

I’m not exactly sure how I’m going to use it, but the essence of the class is that it allows the client to create 3-D bar graphs (similar to Excel’s 3-D graphing modes) and feed the graphs data over time; as the data is received, the graphs resize themselves and adjust the axes as necessary.

All drawing is done with Core Graphics: first, the bar is drawn as a rectangle and, subsequently, the darker components are drawn around the initial bar to create perspective. The most challenging aspect of the project was implementing appropriate resizing: for example, if a bar grows, you can’t just increase its height—if you do, it’ll stretch out and, in particular, the pieces of the bar used to generate perspective (especially the dark bit at the very top) will get stretched out and distorted. Instead, I used dynamic resizing with cap insets, i.e., the following steps:

1. Create the bar as a generic UIView
2. Casted it to a UIImage with this code:

UIGraphicsBeginImageContext(bar.bounds.size);

[bar.layer renderInContext:UIGraphicsGetCurrentContext()];

UIImage \*viewImage = UIGraphicsGetImageFromCurrentImageContext();

UIGraphicsEndImageContext();

UIImage \*barImage = [viewImage resizableImageWithCapInsets:UIEdgeInsetsMake(bar.backdrop\_size + 5, 0, bar.backdrop\_size + 5, 0)];

1. Ensure proper cap insets (as above): in this case, make sure that you avoid stretching out a bar’s perspective on the top and bottom, so pad it with cap insets above and below—this enforces that only the middle section of the bar will be stretched out whenever the frame expands.
2. Cast to a UIImageView and add as a subview

I’ve open-sourced the project thus far; hopefully I’ll get around to adding line graphs and some other features—or, better yet, finding a use for the class within an application!

Here’s a demo: