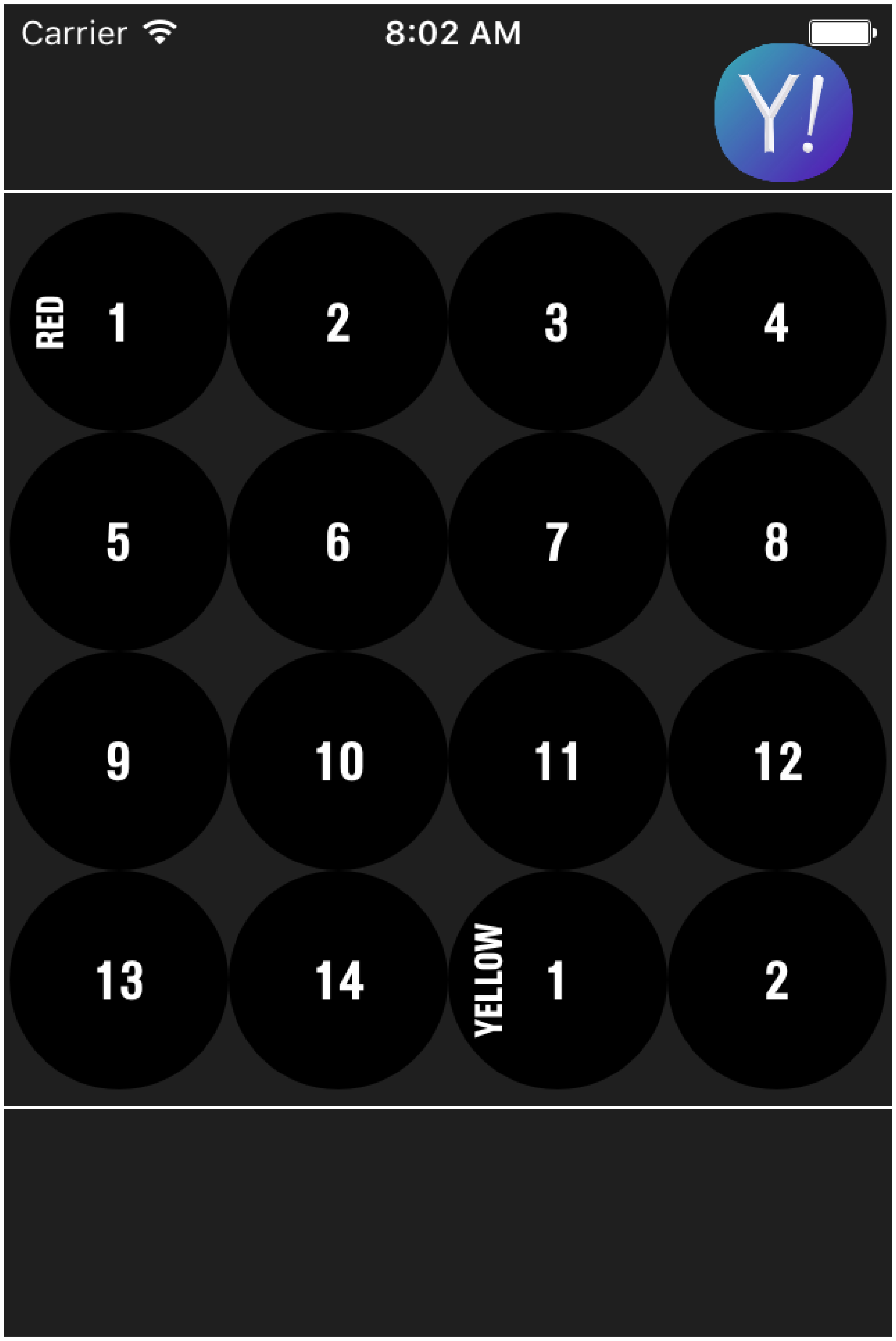
# HooView

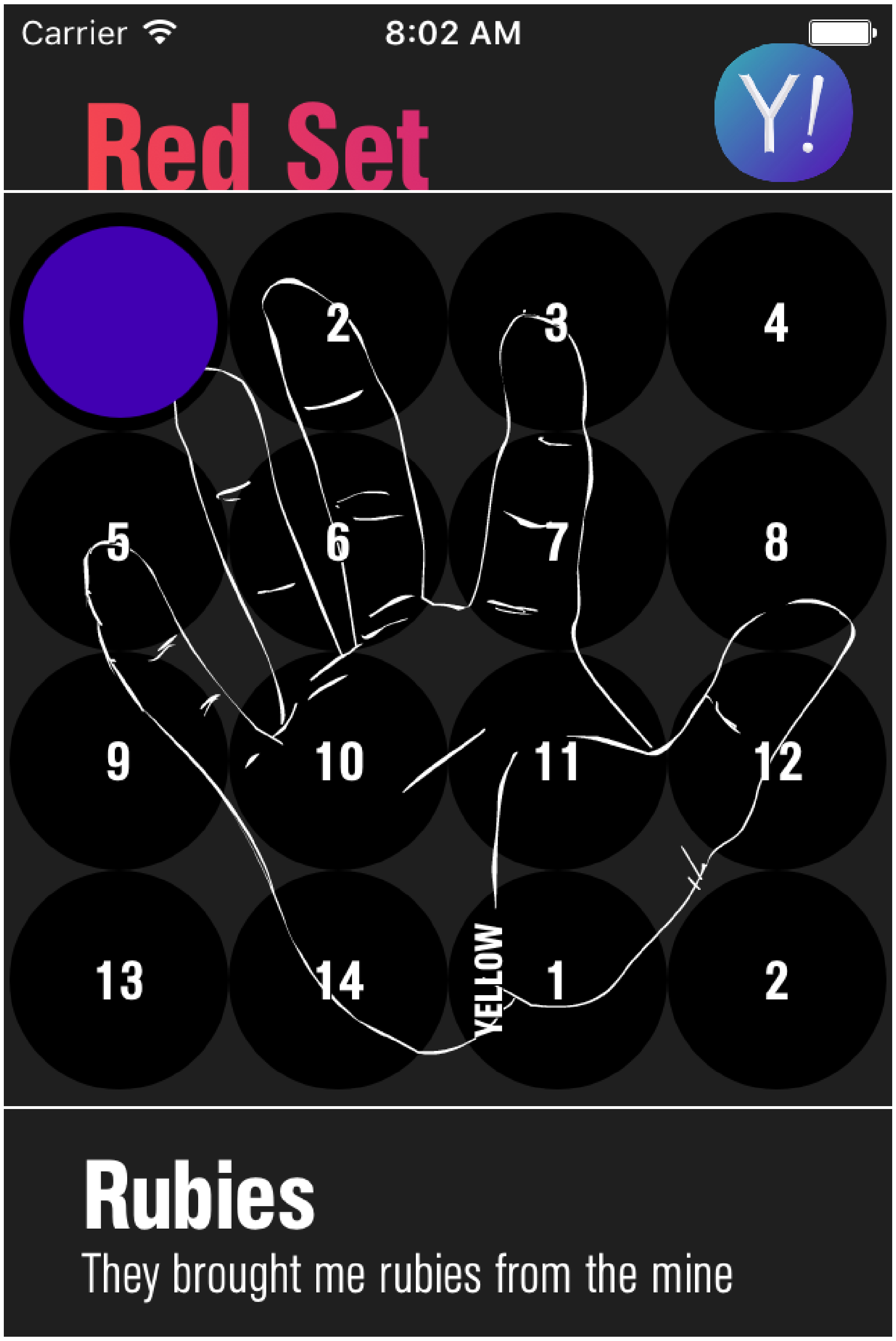
This will be your first "View at the Hoo" for your Yahoo interview.

## Spec

This app's UI is simple: present all sets of "Hoo"s in a scrollable grid. Every set of Hoos is associated with a color, and every Hoo contains an image, a title, and a poem.



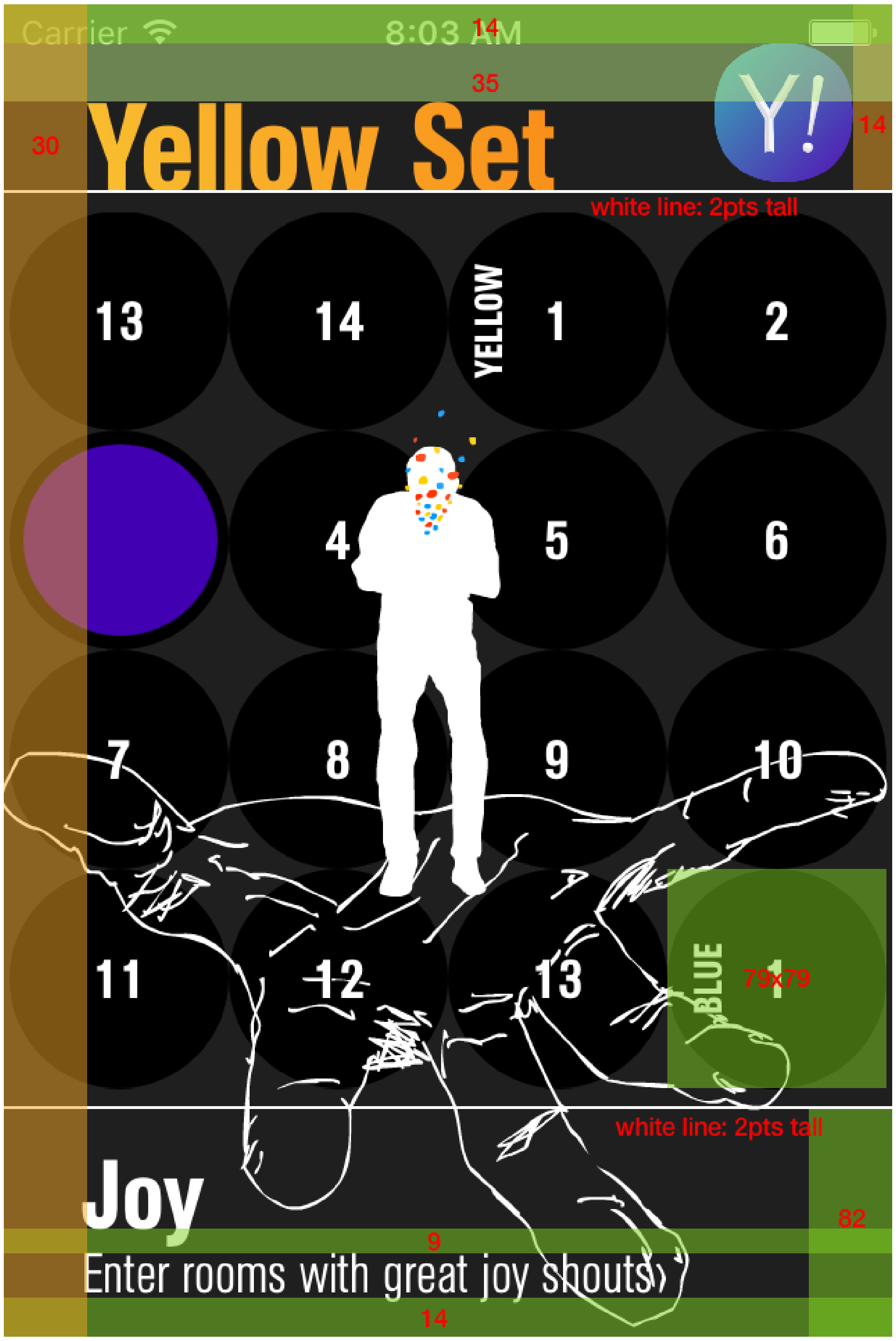
When the user taps on a Hoo in the grid, the selection disc, along with the Hoo metadata, should be presented like so:



Red Set

**Rubies**

They brought me rubies from the mine



The server team is backed up as usual, and doesn't have time to ship a RESTful server API; all they can do is publish blobs of resources on a web server. To save the user's bandwidth, the resources are also zipped. Luckily, one of the Messenger devs was kind enough to create a downloader already so that the ##.zip files appear in the app's main bundle under "HooSets". The sets may or may not be unzipped already; if they're not, you have to make an unzipper from scratch. JUST KIDDING! That's what the CocoaPod SSZipArchive is for. In case you've never used it before, the `NSFileManager` class can help a great deal in examining the contents of the local file system. The resourses are organized like so:

```
## (Hoo Set 1 directory)
|
|____ header.png (also has the @2x and @3x versions)
|
|____ hoo_set.json (provides color and hoo count meta data)
|
|____ ## (Hoo 1 directory)
|
|____ ## (Hoo 2 directory)
```

```
|
|____ ## (Hoo 3 directory)
      |
      |___hoo.json (provides title and poem meta data)
      |
      |___hoo.png
|
|____ ...

## (Hoo set 2)
|
|____ ## (Hoo 1)
|
|____ ...
```

**BONUS:** Ideally, in addition to being able to tap on the Hoos, the user should also be able to scrub (pan) across the screen to activate different Hoos. But wait!? Won't that interfere with the way the ScrollView works? This is where you can be creative and try to come up with a fun user experience to allow the user to scroll also.

## Workspace

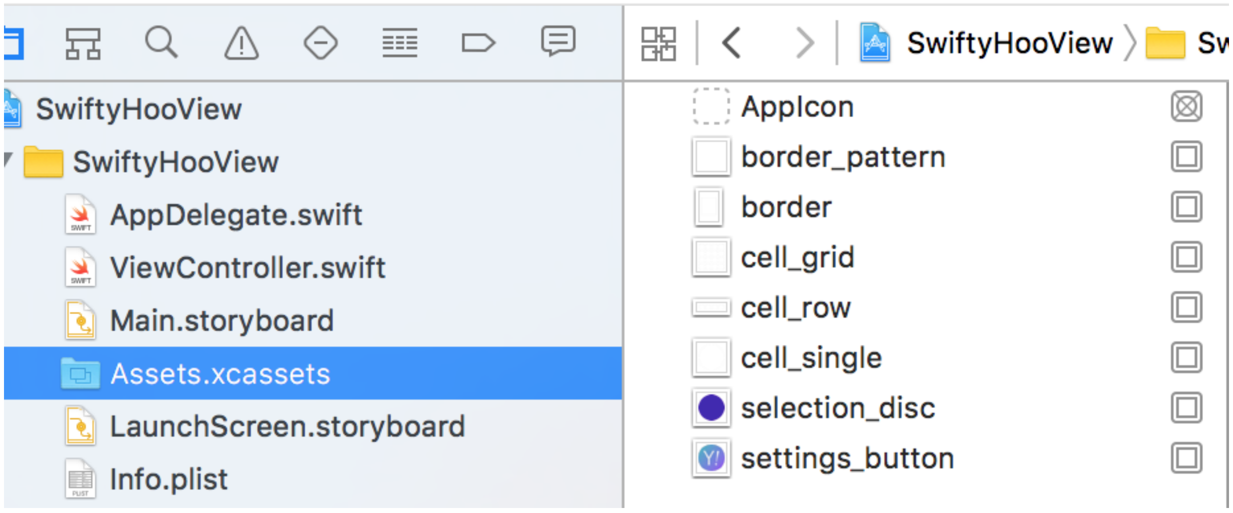
**The works space is jump started to provide all the assets, fonts and tools you'll need.**

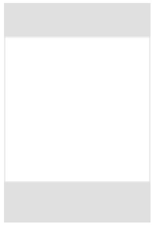
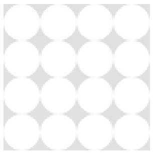
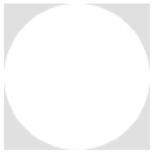
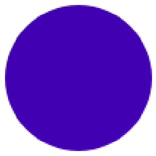
- The `ViewController` class goes as far as unzipping the resources for you and makes them available via an NSArray of NSURL objects which provide the file system paths:

```
- (void)viewDidLoad {
    [super viewDidLoad];

    NSArray<NSURL *> *hooSets = [ViewController unzipHooSets];
    NSLog(@"%@", hooSets);

    // Read in the files and create objects for them here
}

/// Returns a dictionary Hoo Set directory URL values keyed by the name of the Hoo Set directory.
+ (NSArray<NSURL *> *)unzipHooSets;
```

```
override func viewDidLoad() {
    super.viewDidLoad()

    let hooSets = try? ViewController.unzipHooSets()
    print("\(hooSets)")

    // Read in the files and create objects for them here
}

static func unzipHooSets() throws -> [NSURL]
```

- All of the assets you'll need are located in the asset catalog. You don't need to use them all, and in fact there are different assets available depending on how you implement your views:



- border_pattern: A 2x2 pixel image that can be used for tiling the area surrounding the grid. 10% white png.
- border: An image that can be stretched to the area surrounding the grid. 10% white png. Instead, you can use a set of UIViews to achieve the same thing. Just make the backgroud color 10% white: 
- cell_grid: 
- cell_row:
- cell_single: 
- selection_disc: 
- settings_button:

- The font has already been integrated, you simply need to use the constants defined for you:

```
static NSString *kAGBookProLightFontName = @"AGBookPro-LightCnd";
static NSString *kAGBookProMediumFontName = @"AGBookPro-MediumCnd";
```

```
struct Constants {
    static let AGBookProLightFontName = "AGBookPro-LightCnd"
    static let AGBookProMediumFontName = "AGBookPro-MediumCnd"
}
```

## Tips

- We love Git. Yahoo has one of the largest installations of Enterprise Github in the world, so we use it a lot. If you think you're at a good stopping point -- no matter how much code you've written -- commit it! It'll help us see your thought process better, and help you keep your workspace neat and tidy.

## Goals

- Gauge your expertise of UIKit and Auto Layout
- Evaluate your code organization and object oriented design skills
- Assess how you balance code quality, feature development, and completeness given the time constraint

**GOOD LUCK!**