Scroll View School

Hands-On Challenges

Scroll View School
Hands-On Challenges

Challenge A: True Center

If you're a perfectionist like I am sometimes, there are two things that may be bothering you with the centered image scroller project, as-is. Let's fix them one by one!

Part 1: Keeping the code centralized

The first problem with this project is currently you have your code to handle scrolling and zooming in two different places. You're using a subclass to center the content, but you are using the view controller to set the initial zoom scales based on the content. It would be better if the two sets of code were in one place.

To fix this, open **RWTViewController.m** and delete `setZoomScales` and `viewWillLayoutSubviews`. Also delete the line that calls it in viewDidLoad.

Then open **RWTCenteredScrollView.m** and this new method:

```
- (void)setZoomScales {

  if ([self.delegate
respondsToSelector:@selector(viewForZoomingInScrollView:)]) {

    UIView *viewToCenter = [self.delegate
viewForZoomingInScrollView:self];

    CGSize boundsSize = self.bounds.size;
    CGSize contentSize = viewToCenter.bounds.size;

    CGFloat xScale = boundsSize.width / contentSize.width;
    CGFloat yScale = boundsSize.height / contentSize.height;
    CGFloat minScale = MIN(xScale, yScale);

    self.minimumZoomScale = minScale;
    self.zoomScale = minScale;
    self.maximumZoomScale = 3.0;

  }
}
```

This is the same code as was in **RWTViewController.m** before, but slightly tweaked so it works in this subclass.

You only want to call this method the first time this view lays out its subviews. So add a new property to keep track if it's the first time at the top of the file:

```
@interface RWTCenteredScrollView()
@property (nonatomic, assign) BOOL firstTime;
@end
```

Initialize this to `YES` in `initWithFrame:`:

```
self.firstTime = YES;
```

Finally replace `layoutSubviews:` with the following:

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

  if (self.firstTime) {
    [self setZoomScales];
  }

  [self centerContent];

  if (self.firstTime) {
    self.firstTime = NO;
  }
}
```

This just calls `setZoomScales` the first time layoutSubviews is called. Build and run, and your app should work just as usual, except now the code is centralized!

Uber Haxx0r Challenge: Keeping the image centered

Open **RWTViewController.m** and add these lines to the bottom of `viewDidLoad`:

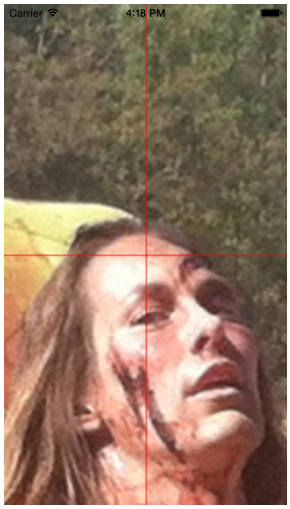
```
static CGFloat kLineWidth = 2.f;
UIView *vertLine = [[UIView alloc]
  initWithFrame:CGRectMake(CGRectGetMidX(self.view.bounds), 0,
  kLineWidth, CGRectGetHeight(self.view.bounds))];
vertLine.backgroundColor = [[UIColor redColor]
  colorWithAlphaComponent:0.5];
vertLine.autoresizingMask = UIViewAutoresizingFlexibleHeight |
  UIViewAutoresizingFlexibleRightMargin |
  UIViewAutoresizingFlexibleLeftMargin;
```

```
[self.view addSubview:vertLine];

UIView *horzLine = [[UIView alloc] initWithFrame:CGRectMake(0,
  CGRectGetMidY(self.view.bounds),
  CGRectGetWidth(self.view.bounds), kLineWidth)];
horzLine.backgroundColor = [[UIColor redColor]
  colorWithAlphaComponent:0.5];
horzLine.autoresizingMask = UIViewAutoresizingFlexibleWidth |
  UIViewAutoresizingFlexibleTopMargin |
  UIViewAutoresizingFlexibleBottomMargin;
[self.view addSubview:horzLine];
```

This creates two thin UIViews to act as "crosshairs" so it's easy to see what part of the image lines up to the center of the screen.

Run the project, then zoom into to give one of the zombies a headshot like this:



Rotate to landscape, and you'll see that your fine-tuned headshot no longer lines up!

As you can see, the visible center point on the image does not stay at the center of the scroll view after an orientation change. This is your uber haxx0r challenge – to add some code that fixes this.

Your general strategy will be the following:

- Keep track of the previous size of the scroll view. In `layoutSubviews`, if the current size is different from the last size, you know there's been an orientation change.
- Also keep track of the center of the view, in `UIImageView` coordinates. Upon an orientation change, convert the old center back to scroll view coordinates. This will likely no longer match up to the center of the scroll view's bounds, so figure out the difference, and adjust the content offset by that amount.

Let's get you started with this. Add two new properties to the top of **RWTCenteredScrollView.m**:

```
@property (nonatomic, assign) CGSize oldSize;
@property (nonatomic, assign) CGPoint oldCenterPoint;
```

This keeps track of the previous size of the scroll view, and the previous center point (in image view coordinates).

Next add these lines to the bottom of `layoutSubviews`:

```
else if (!CGSizeEqualToSize(self.bounds.size, self.oldSize)) {
  [self adjustInnerView];
  self.oldSize = self.bounds.size;
}

self.oldCenterPoint = [self centerPoint];
NSLog(@"Center point: %@",
  NSStringFromCGPoint(self.oldCenterPoint));
```

Next add this method to find the center of the scroll view in image view coordinates:

```
- (CGPoint)centerPoint {

  if ([self.delegate
respondsToSelector:@selector(viewForZoomingInScrollView:)]) {

    UIView *viewToCenter = [self.delegate
      viewForZoomingInScrollView:self];

    CGPoint scrollViewCenter =
      CGPointMake(CGRectGetMidX(self.bounds),
      CGRectGetMidY(self.bounds));
```

```
    CGPoint imageCenter = [self convertPoint:scrollViewCenter
```

```
      toView:viewToCenter];
    return imageCenter;

  }
  return CGPointZero;

}
```

Read this method carefully and make sure you understand it. Note the use of `convertPoint:toView:`, which converts a point from scroll view coordinates to image view coordinates. You will need to use this same method later to convert it back the other way.

Finally, add this method stub to the file:

```
- (void)adjustInnerView {

  if ([self.delegate
respondsToSelector:@selector(viewForZoomingInScrollView:)]) {

    UIView *innerView = [self.delegate
viewForZoomingInScrollView:self];

    CGPoint desiredCenter = // ???
    CGPoint actualCenter = // ???

    CGPoint contentOffset = [self contentOffset];
    contentOffset.x += // ???
    contentOffset.y += // ???
    self.contentOffset = contentOffset;

  }
}
```

Your challenge is to fill in the missing parts of this method to keep the view centered properly. `desiredCenter` should be the old center (in image coordinates) converted back into scroll view coordinates (the scroll view coordinates will be different due to the rotation). `actualCenter` should be the actual center of the view in scroll view coordinates.

If you get stuck, I encourage you to draw out the situation on paper, and/or debug with Reveal, LLDB, or log statements. Best of luck and happy zombie hunting!