5: State Parks I

Due: Noon October 1

Description

A common pattern for displaying content such as a series of articles in a magazine is to allow scrolling between articles and also scrolling between pages within a single article. This kind of interaction can be accomplished via an arrangement of scroll views in which the pages of a single article are laid out in a column (first page at the top) on a single scroll view and the articles (scroll views) are laid out side by side on a "master" scroll view. To make navigation easy, horizontal scrolling between articles can only be done while viewing the first pages of articles. No diagonal scrolling is allowed and no jumping from one article to another while in the middle of one article. Scroll indicators (arrows) appear to inform the user in which direction they can currently scroll.

Instead of magazine articles we'll use photos from Pennsylvania state parks. Sample photos from a small number of parks will be provided (along with a property list). Extensive use of delegate methods will be required to give the desired scrolling behavior. The kind of coding required for this app is very typical of much app development - finding ways of obtaining a desired behavior given the particular framework methods provided.

Walk Through

Instead of a walk through, here are the essential elements your app should support.

1. Your app should build for both the iPad and iPhone platforms (Universal). Use Auto Layout.
2. Your app should use the Model-View-Controller pattern. The model should read in the property list containing all the park and photo information. Give the model a clean, simple interface that provides park and photo information. You'll be reusing this model in subsequent weeks, so give it a clean, flexible interface.
3. The view should consist of an arrangement of park scroll views (vertical scrolling only) on a main scroll view (horizontal scrolling only). The app should always be showing a single photo.
4. Horizontal scrolling (switching parks) should only be allowed when displaying the first page of a park's photos (i.e., at the top its column of photos).
5. The first page of each park's photos should display the name of the park.
6. Vertical scrolling should be allowed to view all the photos in each park. Scrolling should not be allowed beyond the last photo in a park.
7. No diagonal scrolling at any time.
8. Scroll indicators in the form of four arrows should appear to indicate in which direction scrolling is allowed at any time when the user interacts with the app. They should disappear when a photo is showing and the user is not interacting with the app.
9. Zooming in and out on an individual photo should be supported using the traditional pinch gesture. A minimum scale of 1.0 should be used. A maximum scale of 10.0 is reasonable. Panning around a photo that has been scaled up should be supported. Scrolling to another photo should be disabled until the zoom scale of the photo has returned to 1.0.

10. Your app does not have to support rotation. Set the allowed orientation in the project editor to restrict the app to portrait only.

Testing

Be sure to thoroughly test all possible interactions with your app.

Hints

1. Your model and controller must allow for any number of parks and any number of photos in each park. Make no assumptions in your code. Let the data (plist) that you read in dictate these numbers.
2. Your app will need to make extensive use of the `UIScrollViewDelegate` protocol. Review the reference documentation for this protocol so you have an idea of the events to which your app can respond.
3. Remember to set the `delegate` and `contentSize` of scrollViews.
4. Zooming in and out of an individual photo requires more trickery to fool the user. If your app simply tells your scrollview to zoom in on the current photo being displayed you should notice some strange, unwanted behavior, as the photo will scale, but the other photos will be visible as you pan around. There are probably several ways to support zooming that appear natural. One way is to place each UIImageView on its own UIScrollView (which is then placed on the main UIScrollView) and then carefully make sure the correct scrollView receives the appropriate gestures, perhaps disabling zooming or panning as necessary. I chose to use a separate scroll view just for zooming (and panning) on a photo. It appears only when a pinch gesture (on the main view) is detected, responds to the pinch gesture (which has a scale property), and disappears when the scale factor equals 1 and the gesture has ended. The user never detects this slight of hand because when this scroll view appears and disappears, its content (an imageView) is identical to the content visible on the main scroll view at that time. Getting this to work perfectly is difficult. Don't worry if panning is possible beyond the edge of a photo, but do your best. Getting zooming to look just right also takes some doing, all done with delegate methods to achieve a desired behavior

Troubleshooting

1. Be sure to set the delegate property of your scrollview(s) and their contentSize property. Set min and max zooming as appropriate and implement delegate method viewForZooming.

Submission

Your submission should be pushed on the master branch. Be sure to verify that your project builds and remove all cruft.