6: State Parks II

Due: Noon, October 8

Description

In Part II of this app we will reuse the model, but instead use both Table Views and Collection Views to display the photos. We'll use a different property list that includes captions for each photo.

Walk Through

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

1. Your app should present the user with both a table view and a collection view of the park photos. A tab bar should be used to allow the user to switch between these two views. If you designed your model well it should remain relatively unchanged.
2. The table view should contain all the photos, with a section for each park. Section titles should be the park names. Cells should contain the photo caption on the left and a thumbnail-size image on the right. This requires a custom subclass of UITableViewCell.
3. The collection view should also use sections (with appropriate titles) for each park and each cell should display just the photo (no captions). Everything should be laid out/spaced in an appealing way.
4. Tapping on a cell (in either the table or collection views) should bring up an image containing the photo. The appearance of this photo should animate from the thumbnail size image in the table/collection view. I.e., it should appear to the user that the thumbnail image expands to "full size."
5. The full size image should support zooming and panning. Tapping on the image when its scale is 1.0 should dismiss the image, again animating "back" to the thumbnail image.
6. Table view sections should support an expand/collapse pattern. By default, all the rows in each section should be displayed. Tapping on a section title should "collapse" the section and hide all the rows using a pleasing animation. Tapping on it again should make the rows reappear using a pleasing animation.
7. Your app should support device rotations. This includes when the app is displaying a photo full screen.

Testing

Be sure to thoroughly test all possible interactions with your app. Tapping, scrolling, zooming, collapsing/expanding in all combinations possible.

Hints

1. Though not required, I recommend adding a new storyboard for this project. You can specify which storyboard should be the initial one in the project editor. Do not create a new project.
2. Be sure to include icon images on each of the tabs.

6: State Parks II

Due: Noon, October 8

3. Animating the full size photos into place requires some of the same coordinate manipulation that we saw in the Pentominoes app and a little trickery to fool the user. Simple UIView animations are sufficient.
4. The collapse/expand pattern of table view sections is common and can make your table views more user friendly. You'll need a control view (probably a button) in a custom section header view. Review the UITableViewDataSource protocol to find the method for providing custom views for section headers. You will need to modify the table view data source to provide the correct number of rows for a section depending upon whether the section is collapsed or not. You might think about having the model maintain this status, but really the model shouldn't care about whether a section is collapsed or not. Let the controller maintain this status.
5. Don't forget to set Reuse Identifiers on your prototype cells and headers.
6. Use appropriate animation for the collapsing/expanding of sections. Find the appropriate UITableView method for reloading a section with an animation.
7. Use a single shared instance of the model, shared by both controllers.
8. You should need only two view controllers, one a subclass of UITableViewController and one a subclass of UICollectionViewController. When creating your view in IB, drag these controllers out of the object library.
9. Be sure to use appropriate Auto Layout constraints, especially on any custom cells and headers created in IB.
10. You shouldn't need any special code to support rotations if you used Auto Layout correctly. The only place you might need customization is for your custom section headers for the table view.
11. As always, use a good MVC structure.

Troubleshooting

1. Be sure delegates are set for the relevant controllers and views. Remember to set the contentSize on any scrollviews.
2. Be sure to use the correct dequeue method in your collection view controller's `cellForItemAtIndexPath`.

Submission

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