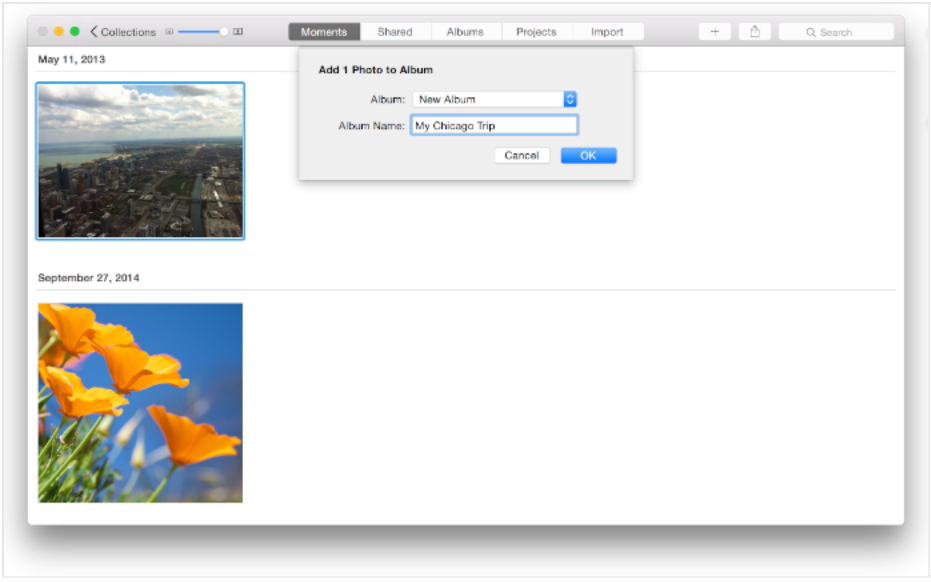
App Styles and Anatomy

Broadly speaking, there are three main styles of OS X apps:

Single-window utility. A single-window utility app, such as Calculator or Dictionary, helps users perform the primary task within one window. Although a single-window utility app might also open an additional window—such as a preferences window—the user remains focused on the main window.

Single-window "shoebox." A single-window shoebox app tends to give users an app-specific way to view and manage their content. For example, Photos users don't find or organize their photos in the Finder; instead, they manage their photo collections entirely within the app.



Multiwindow document-based. A multiwindow document-based app, such as Pages, opens a new window for each document the user creates or views. This style of app does not need a main window (although it might open a preferences or other auxiliary window).

Regardless of style, an OS X app presents content in one or more windows and app-specific commands in the systemwide menu bar.

The AppKit framework defines the windows, menus, controls, and other objects that you use to present your app's UI, and it supports many app-level features, such as gesture recognition, font management,

image handling, accessibility, and speech synthesis and recognition. AppKit also defines the light and dark vibrant appearances in Yosemite, in addition to a visual effect view you can use to create custom vibrant views (to learn more about this view, see `NSVisualEffectView`).

Setting aside programmatic inheritance, you can think of the UI objects that AppKit provides as belonging in the following broad conceptual categories:

- **Windows.** A window provides the frame in which the app's content is displayed.
- **Menus.** A menu, such as File, Edit, or Window, contains commands that people use to control the app.
- **Content views.** A content view, such as a table view, text view, or popover, displays the user's content and can contain controls that people use to manage the content.
- **Controls.** People use controls, such as a buttons, sliders, and checkboxes, to provide input and perform tasks in the app.

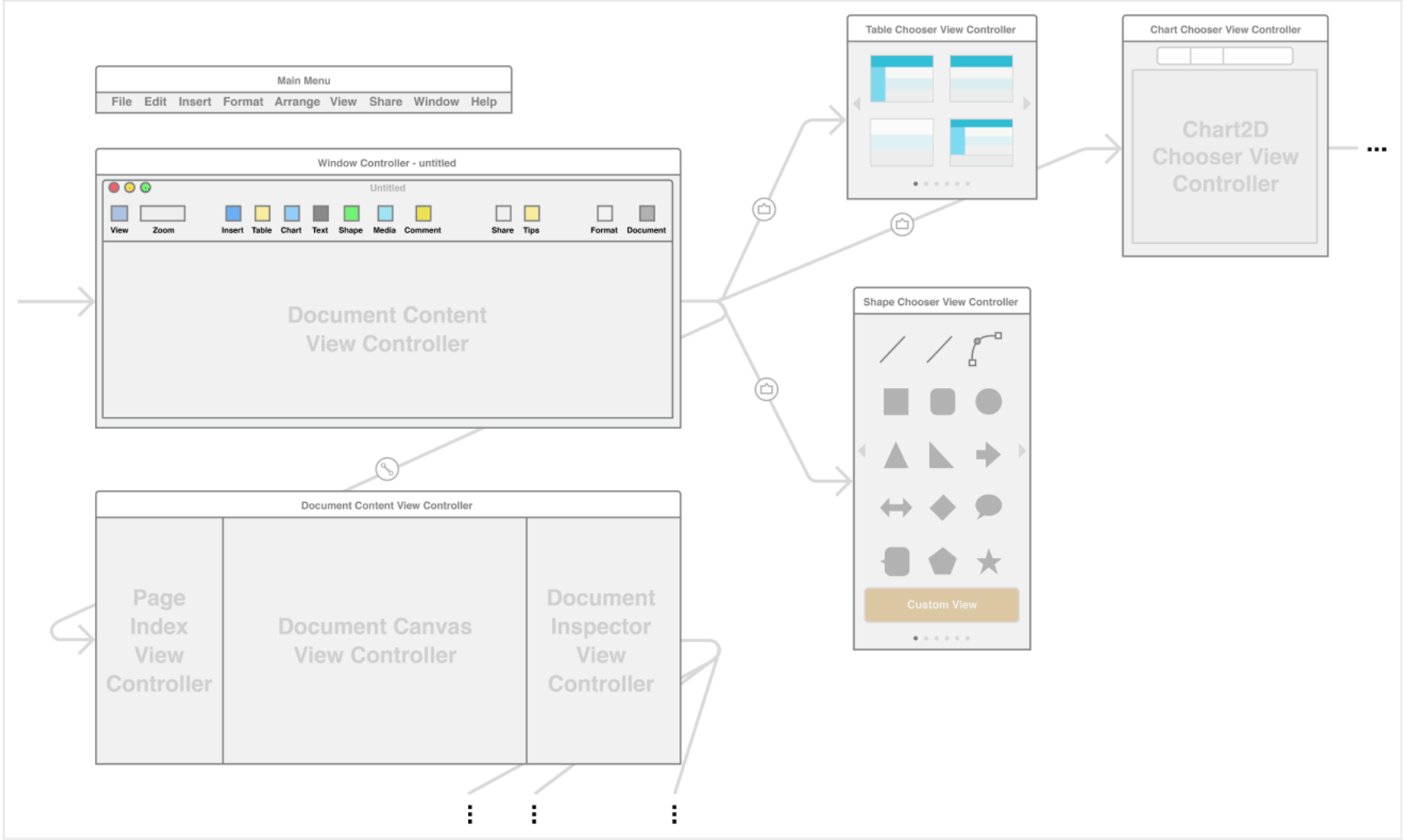
To manage many of these UI objects programmatically, you use various types of controllers, such as a window controller, a tab view controller, or a split view controller. Typically, a window controller contains one or more view controllers, each of which manages a set of views and controls.

A storyboard is a great way to visualize the basic anatomy of an app. In a storyboard, a *scene* represents a controller and the views it manages, and a *segue* represents a relationship between two scenes. Two scenes can be related by containment (a window controller can contain a tab view controller) or by presentation (a view controller can present a popover).

Note

If you've used storyboards to create an iOS app, you might notice that OS X apps tend to have fewer presentation segues, and many more containment segues, than iOS apps do. The difference in segue usage reflects the fact that OS X apps have much more screen space available to them, which means they have less need to present one view controller on top of another.

Viewing an app in a storyboard makes it easy to grasp its anatomy. Here, in the storyboard for an app similar to Pages, you can see the views contained in various view controllers and the relationships between the view controllers.



Designing for Yosemite

Starting and Stopping