

Submitted to- Mr. Manhar Kapoor

Submitted by- Abdul Sameer shaik(A00104315)

Table of Contents

[1. Creating Projects: 3](#_Toc77340359)

[2. Workspace window Overview: 3](#_Toc77340360)

[3. Navigating your workspace: 4](#_Toc77340361)

[4. Editing the files: 6](#_Toc77340362)

[5. Accessing Resources and Inspecting Elements: 8](#_Toc77340363)

[Using inspector area: 9](#_Toc77340364)

[Use the library access ready-to-use libraries: 9](#_Toc77340365)

[6. Using the workspace toolbar: 10](#_Toc77340366)

[7. Using multiple workspace windows: 11](#_Toc77340367)

[Using tabs: 12](#_Toc77340368)

[Using Interface Builder: 13](#_Toc77340369)

[8. Designing with Storyboards: 19](#_Toc77340370)

[Connecting the objects to code: 20](#_Toc77340371)

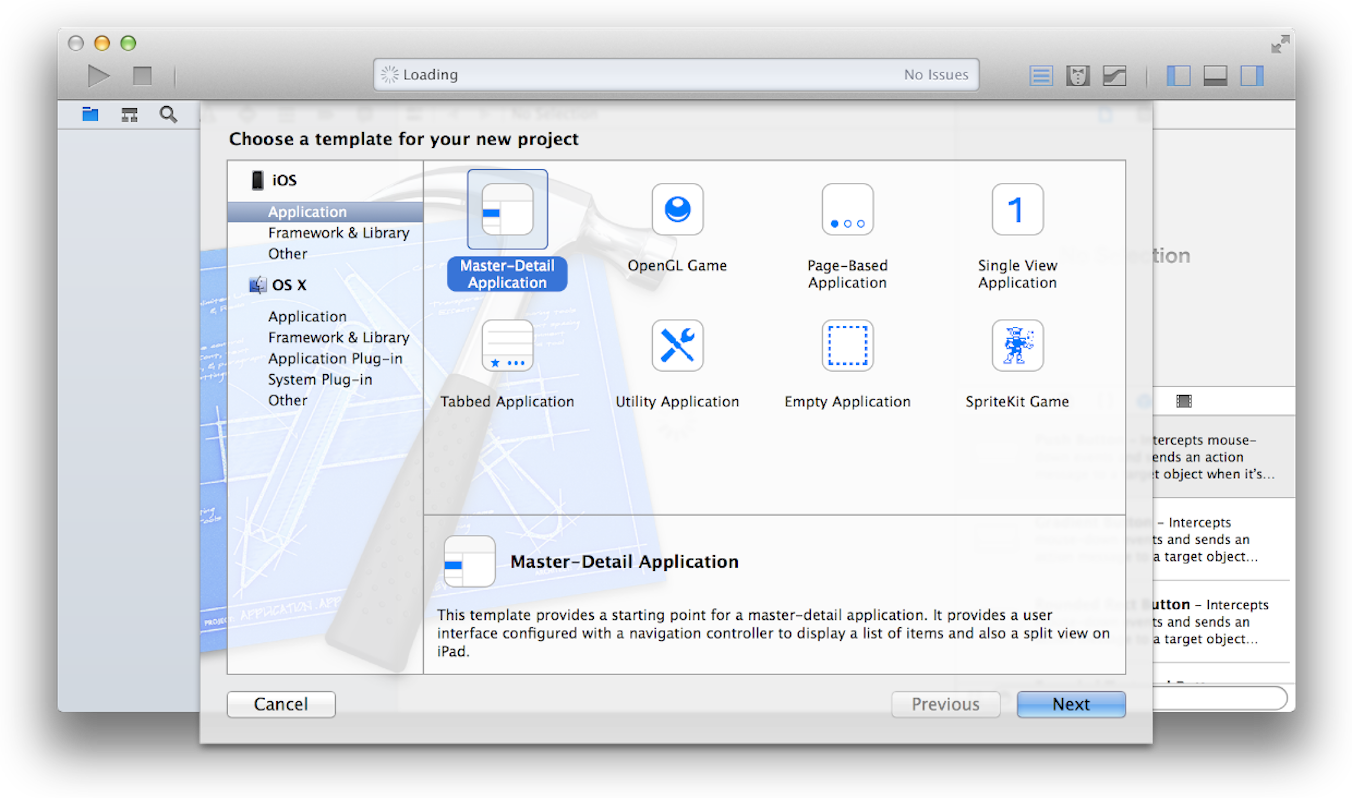
[Sending action Messages from a control to your code: 20](#_Toc77340372)

Write a technical document regarding Xcode; Guide to Xcode or Introduction to Xcode. By reading your guide, people will learn the xcode. Install or upgrade Xcode as needed. Explain Xcode and its capabilities. Use and explore the Xcode interface.

XCode- this is an integrated development environment for macOS, and this is user to develop software for iOS, iPadOS, WatchOS, and tvOS. Xcode was first released in 2003 but after improvising many things and again it is released on June 21, 2020 and this is available only for MacOS users.

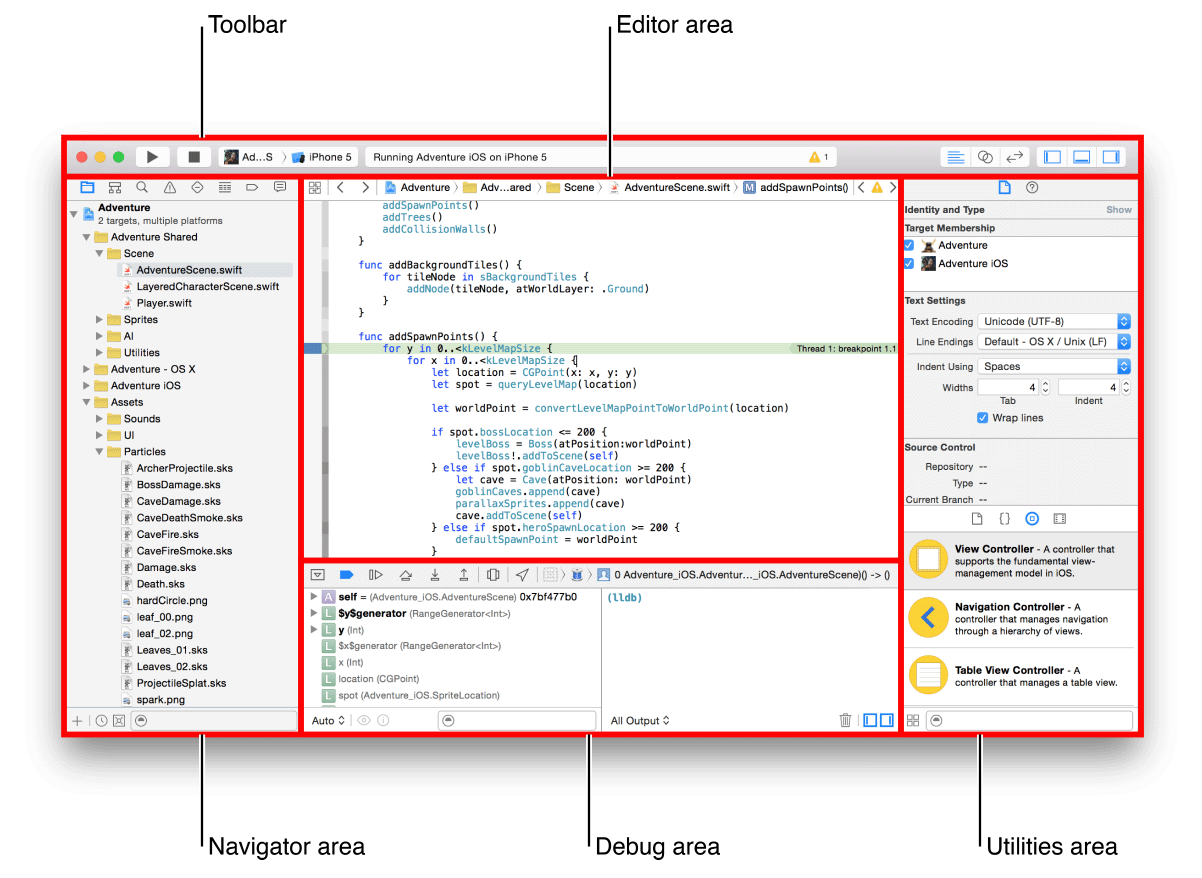
# Creating Projects:

Apps we create in Xcode requires a project which keeps the files and resources in organized. We can start the project by selecting choose file and New->New project then Xcode open a new workspace window and it will prompt to select a project template and we have options and we have to select the options what ever we want to do. And by selecting this templates it automatically includes all the project configuration files that help you start your development quickly.



# Workspace window Overview:

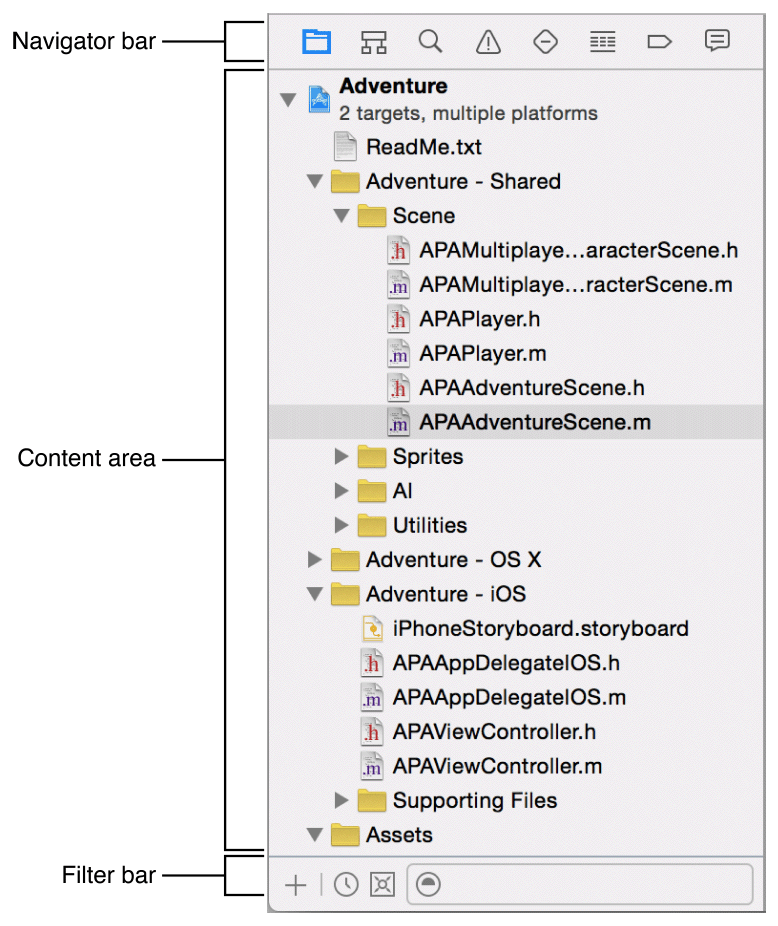
Here we can perform the core development tasks in the Xcode workspace window. This is the primary interface for creating and managing the projects because project is the main unit of development Xcode. This also have the all the elements need to build the app, framework and plug-in. workspace window also maintain the relationship between those elements.



* Show and hide the navigator area. Use this area for navigating all facets of your project including files, symbols, breakpoints, build issues, tests, breakpoints and build reports we can also search for any string in project.
* Show and hide the debug area. Use this area for viewing variables, interacting with the debugger console, and controlling the execution of your app.
* Show and hide the utilities area. Use this area to inspect or modify attributes of files graphical user interface elements, sprites and other elements in your project also use it to access a library of ready made resources.

# Navigating your workspace:

Here we can access files, symbols, unit tests, diagnostics and also other components of project from the navigator area. In navigator bar we can choose navigator that suits for the project and also each navigator bar allows to restrict the content that is displayed.



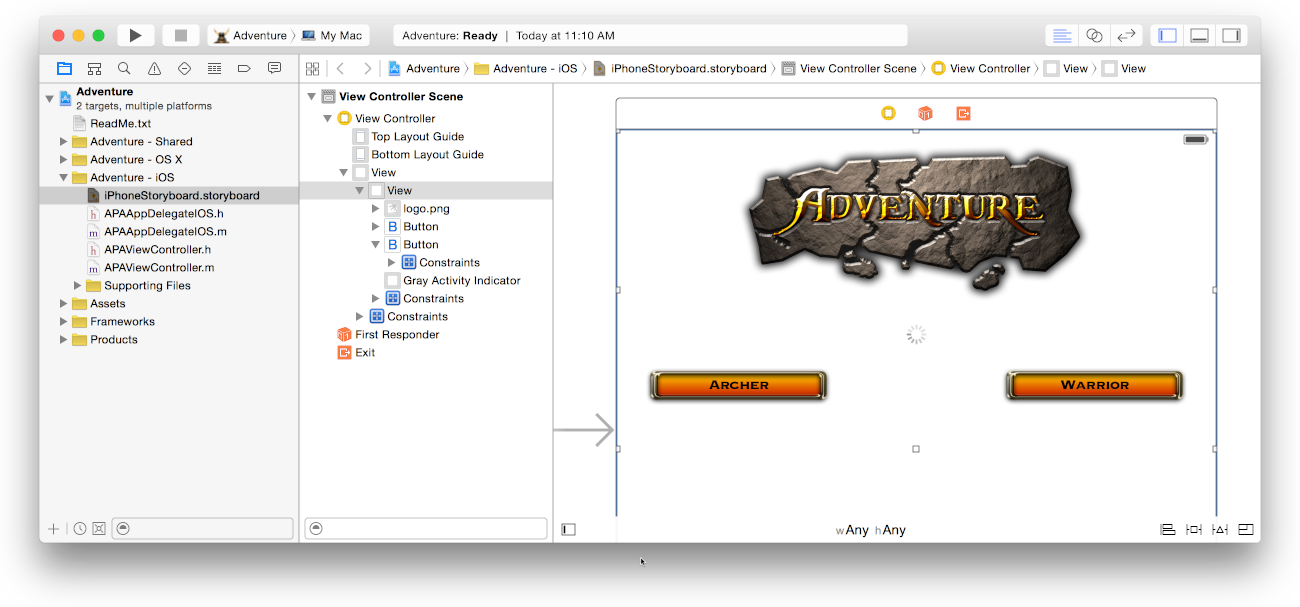
* **Project navigator.** Add, delete, group, and otherwise manage files in your project, or choose a file to view or edit its contents in the editor area.
* **Symbol navigator.**Browse the symbols in your project as a list or hierarchy. Buttons on the left of the filter bar let you limit the shown symbols to a combination of only classes and protocols, only symbols in your project, or only containers.
* **Find navigator.** Use search options and filters to quickly find any string within your project.
* **Issue navigator.** View issues such as diagnostics, warnings, and errors found when opening, analyzing, and building your project.
* **Test navigator.** Create, manage, run, and review unit tests.
* **Debug navigator.** Examine the running threads and associated stack information at a specified point or time during program execution.
* **Breakpoint navigator.** Fine-tune breakpoints by specifying characteristics such as triggering conditions.
* **Report navigator.** View the history of your build, run, debug, continuous integration, and source control tasks.

# Editing the files:

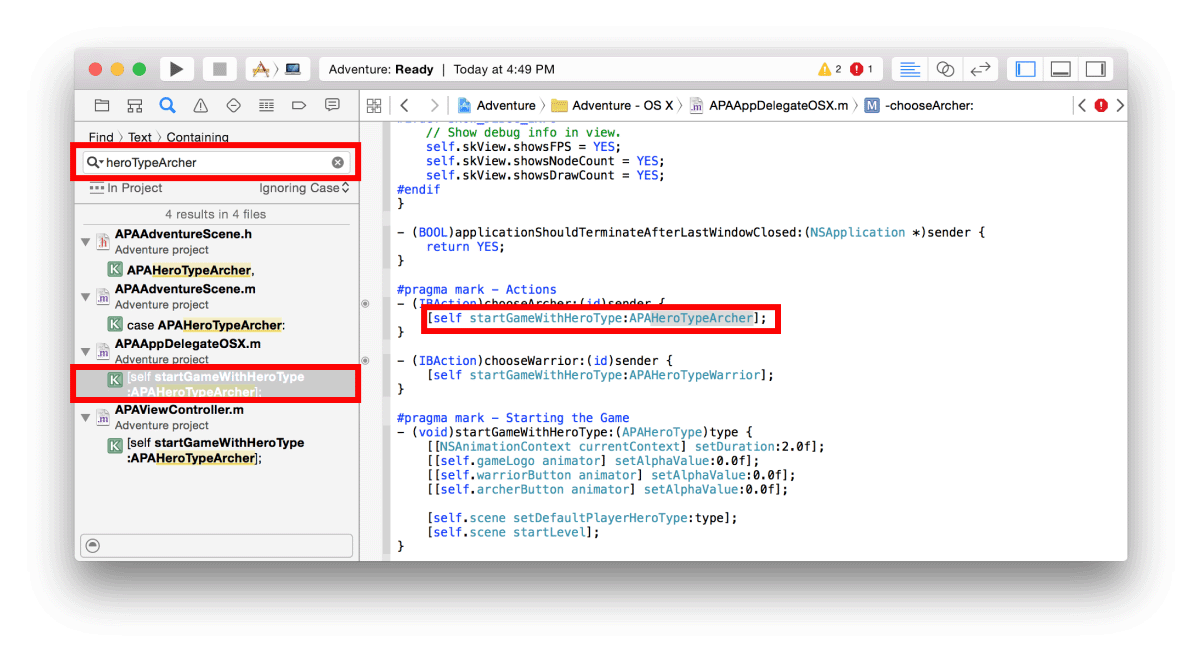
Basically this is the editor area where users can edit the code and this is always available in the workspace window

The most often used editors are :

* Source editor: here we can write and edit the source code
* Interface builder: here we can create and edit the user interface files
* Project editor: we can view and edit the how the project should be built by specifying the build options, targets, architectures and app entitlements.

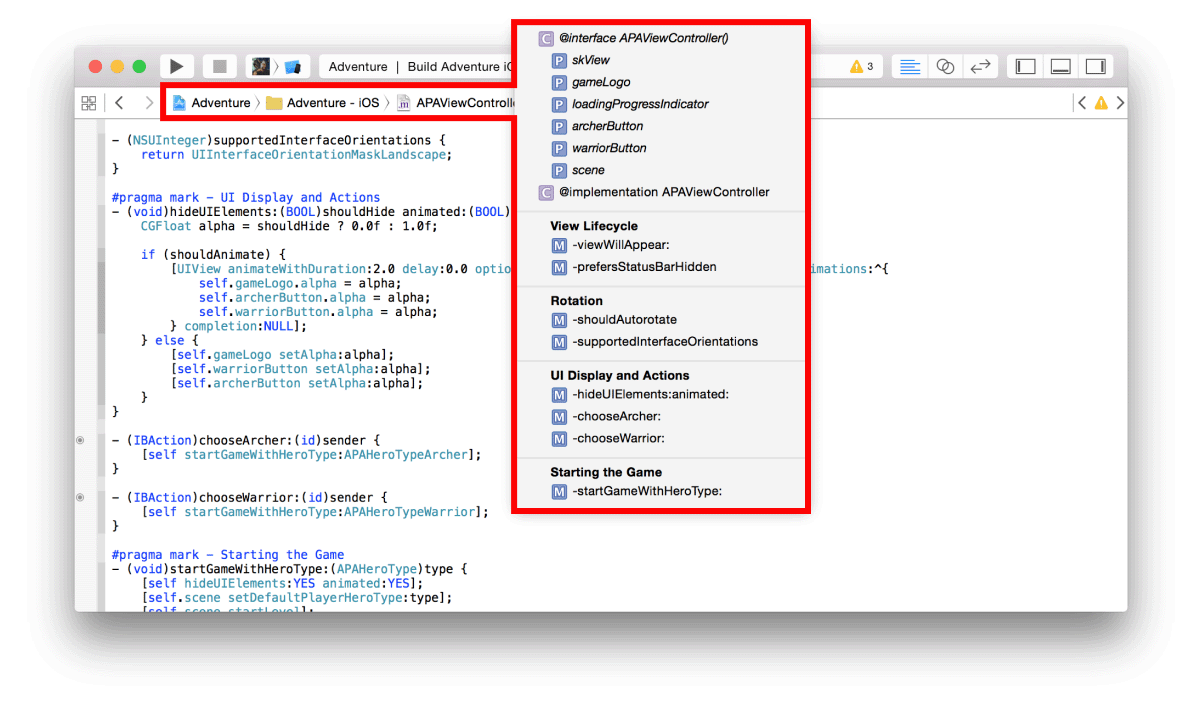


This is also one of the screenshot where we can find the number of results in the search area. This is also searches for symbols and other text in interface builder.



* **Standard editor.** Fills the editor area with the contents of the selected file.
* **Assistant editor.** Presents a separate editor pane with content logically related to content in the standard editor pane. You can also change the content.
* **Version editor.** Shows the differences between the selected file in one pane and another version of that same file in a second pane. This editor works only when your project is under source control.

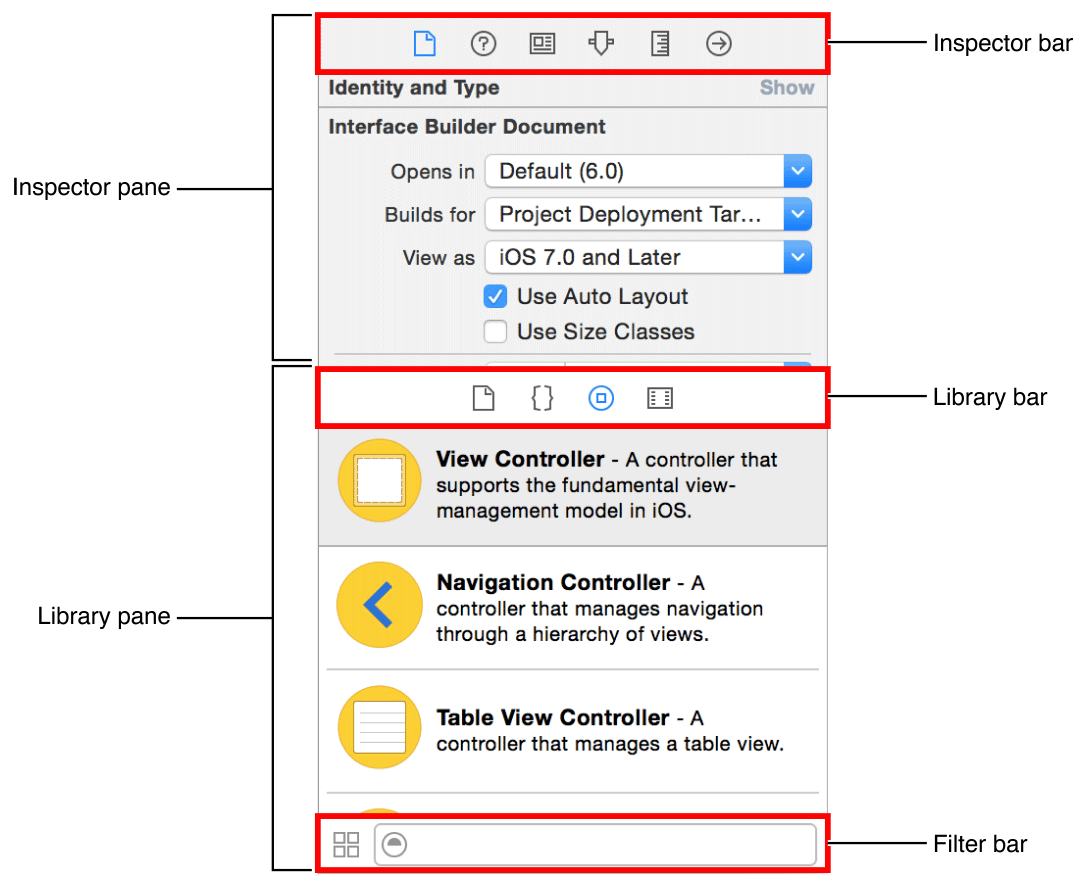
The following screenshot shows how to use the jump bar and also hierarchical mechanism for navigating the apps directly to items at any level in project.



# Accessing Resources and Inspecting Elements:

The utilities area on the right of workspace window gives the access to these accessing resources.

* Inspectors can view and modify the characteristics of the file in an editor.
* Here it has some ready-made libraries where we can use for the project.



## Using inspector area:

There are two ways how to use these inspector area and is explained below.

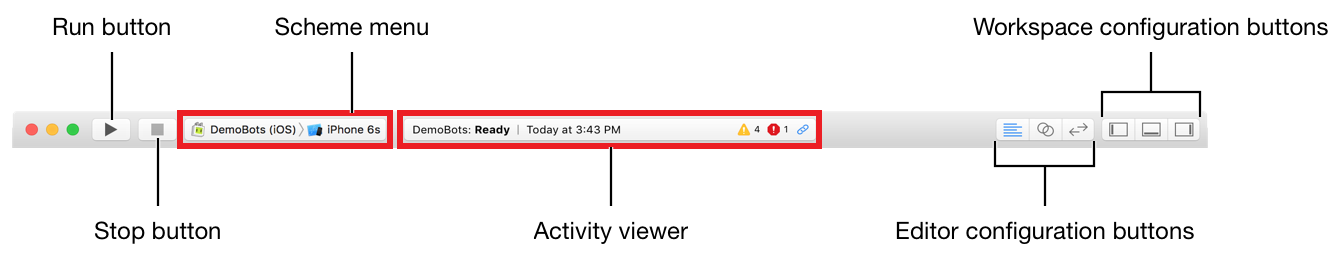
* **File Inspector**: we can view and manage the selected file and typically we will localize the storyboards and other media files and change settings for user interface files.
* **Quick Help**: in case we need any symbol or interface element or build setting in the file this can help displays a description method. Here where and how the method is declared its scope, the parameter it takes and platform and architecture availability.

## Use the library access ready-to-use libraries:

* **File Templates:** these are templates where they have common types of files and code constructs.
* **Code Snippets:** this is a short piece of code that we use in the software such as declarations, control flows, block declarations and templates for commonly used apple technologies.
* **Media:** this contains the graphics, icons, sound files and the like.
* **Objects:** items that we use for app interface.

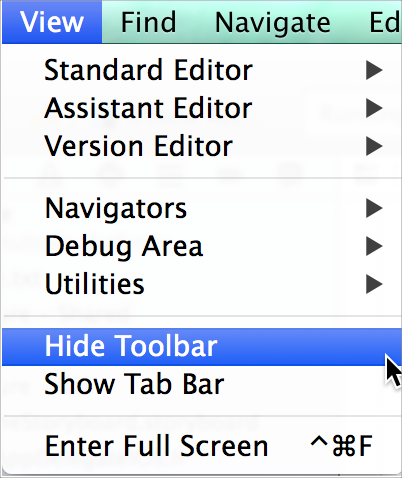
# Using the workspace toolbar:

This is the toolbar which will be at the top of the window which provides quick access to frequently used commands here the run buttons will appear if you click that button then code will build and runs. The activity viewer also shows the progress of tasks currently executing by displaying status message, build progress and other information about the project



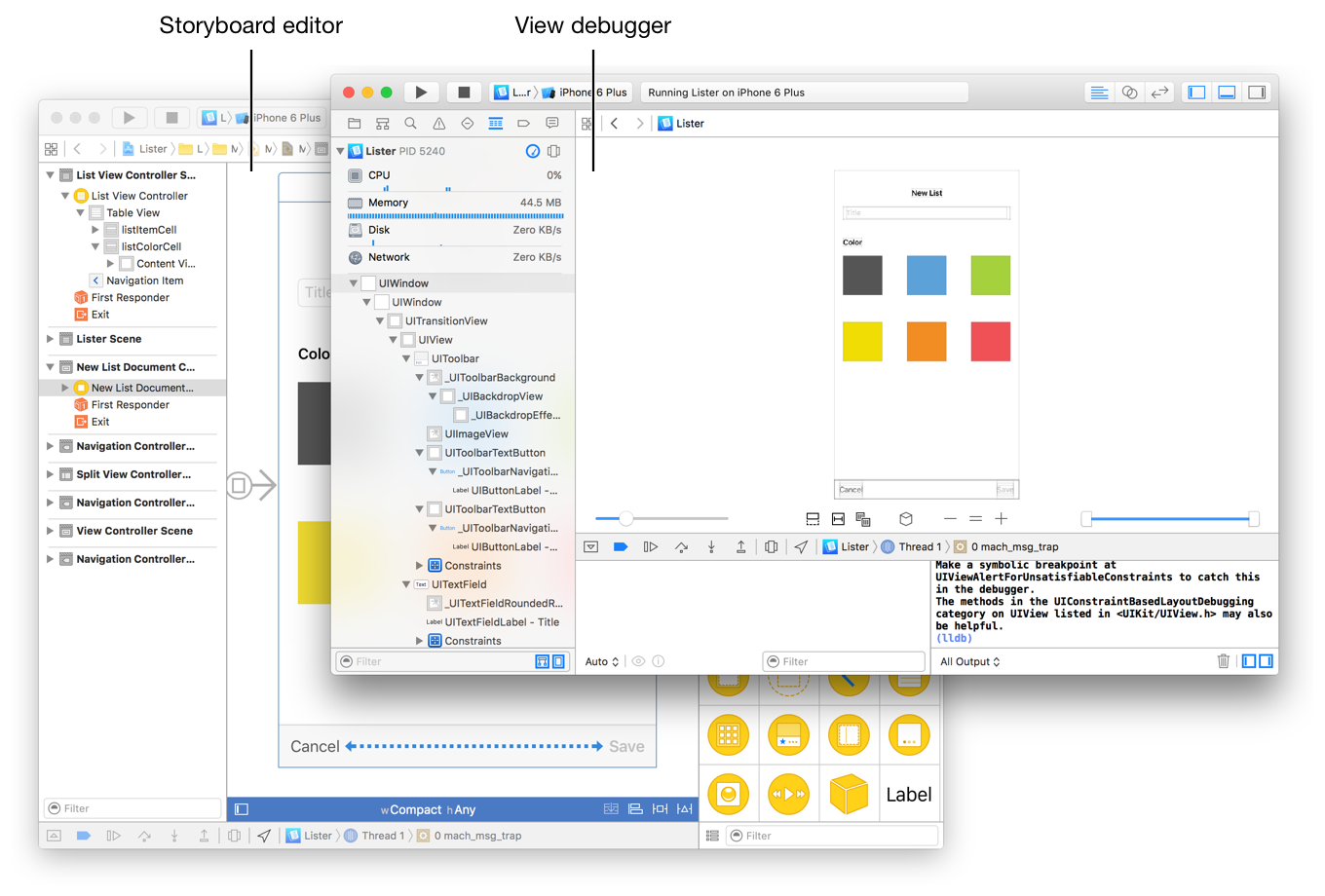
It also has the main window tool bar in Xcode for basics help

The view menu includes commands to hide and also to show the tool bar.



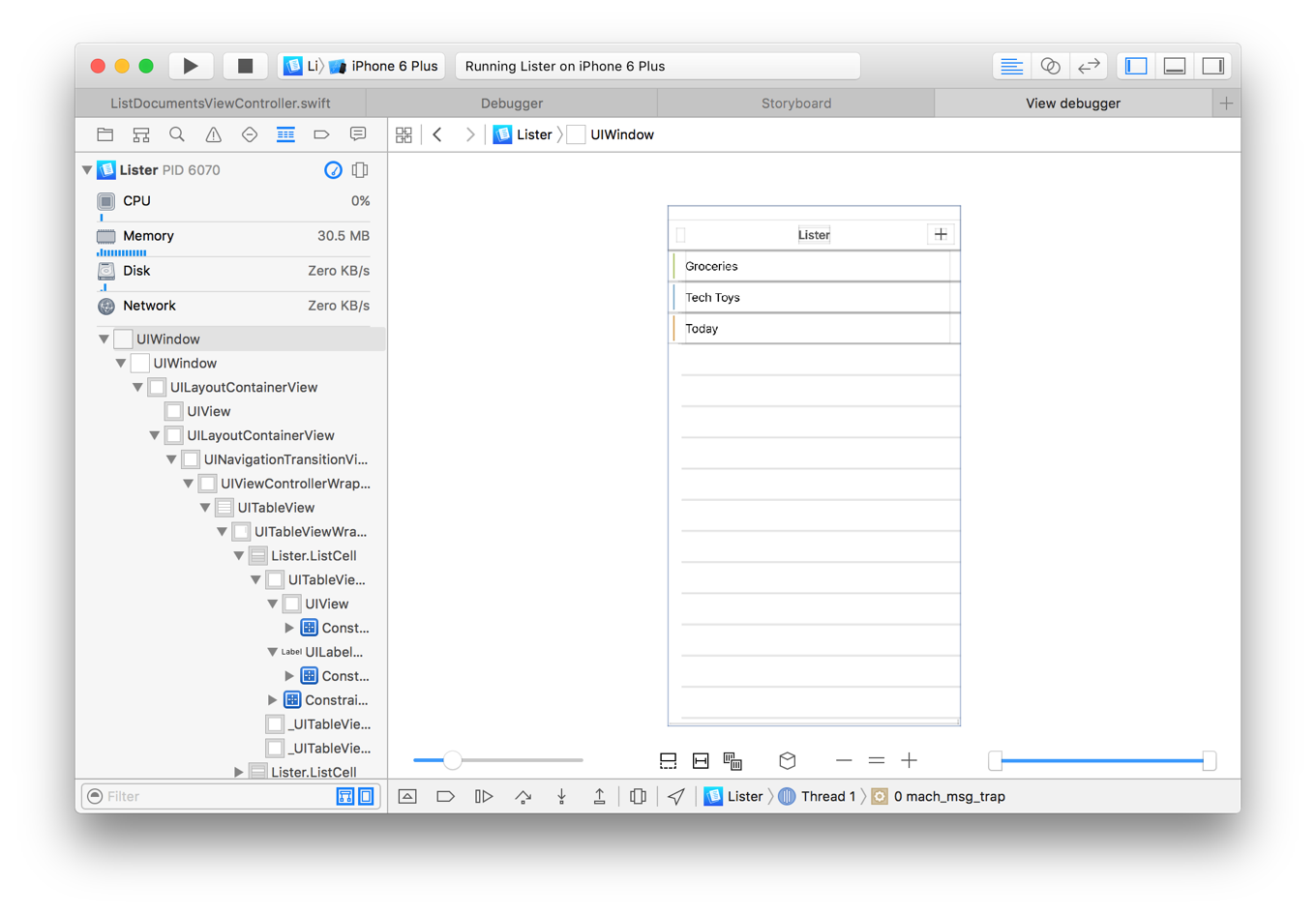
# Using multiple workspace windows:

We have to create the multiple workspace windows choosing File> New> Window and each window can be customized independently of the other.



## Using tabs:

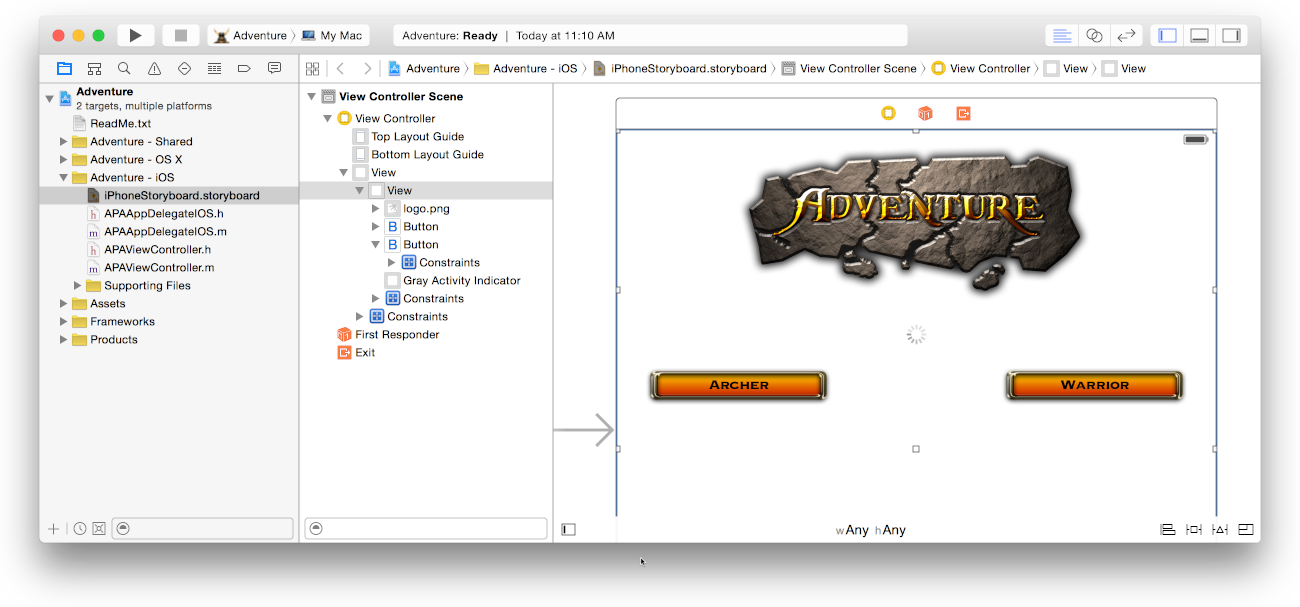
Add tabs to a workspace window by choosing File > New > Tab, or by clicking the Add (+) in the tab bar. Click a tab to make it the active workspace window layout.



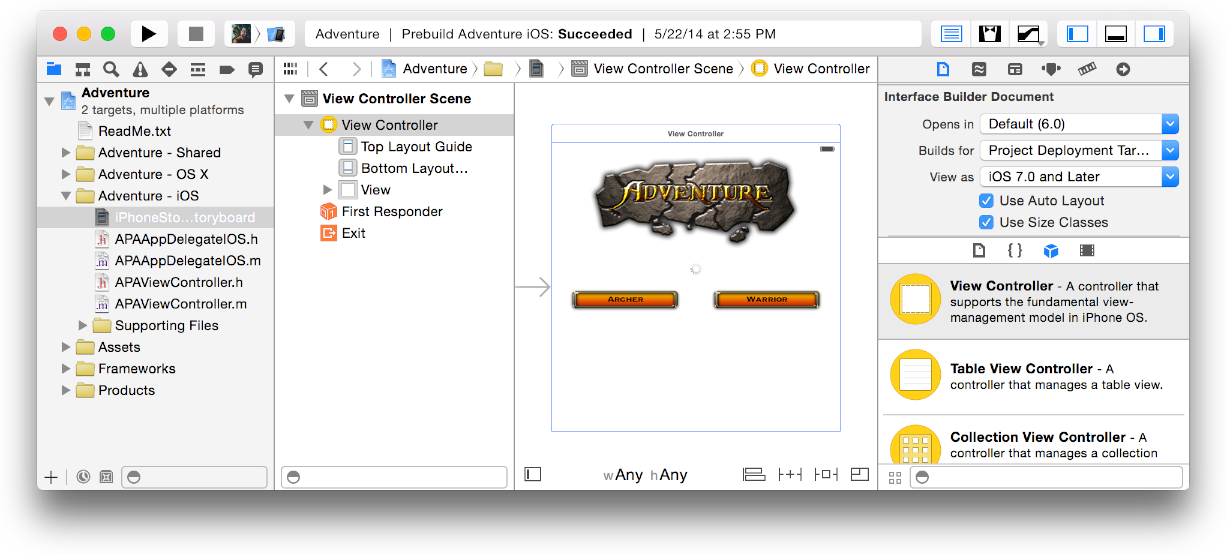
# Using Interface Builder:

We create the app user interface in interface builder. We can select a user interface file in the project navigator and file content’s is open in interface builder in the editor area of the work space window.

This has a file extension .storyboard or .xib. a xib file usually specifies on the view controller or menu bar. This specifies a set of view controller and segues between those controllers. A storyboard can also contain many view controllers and transitions between them.

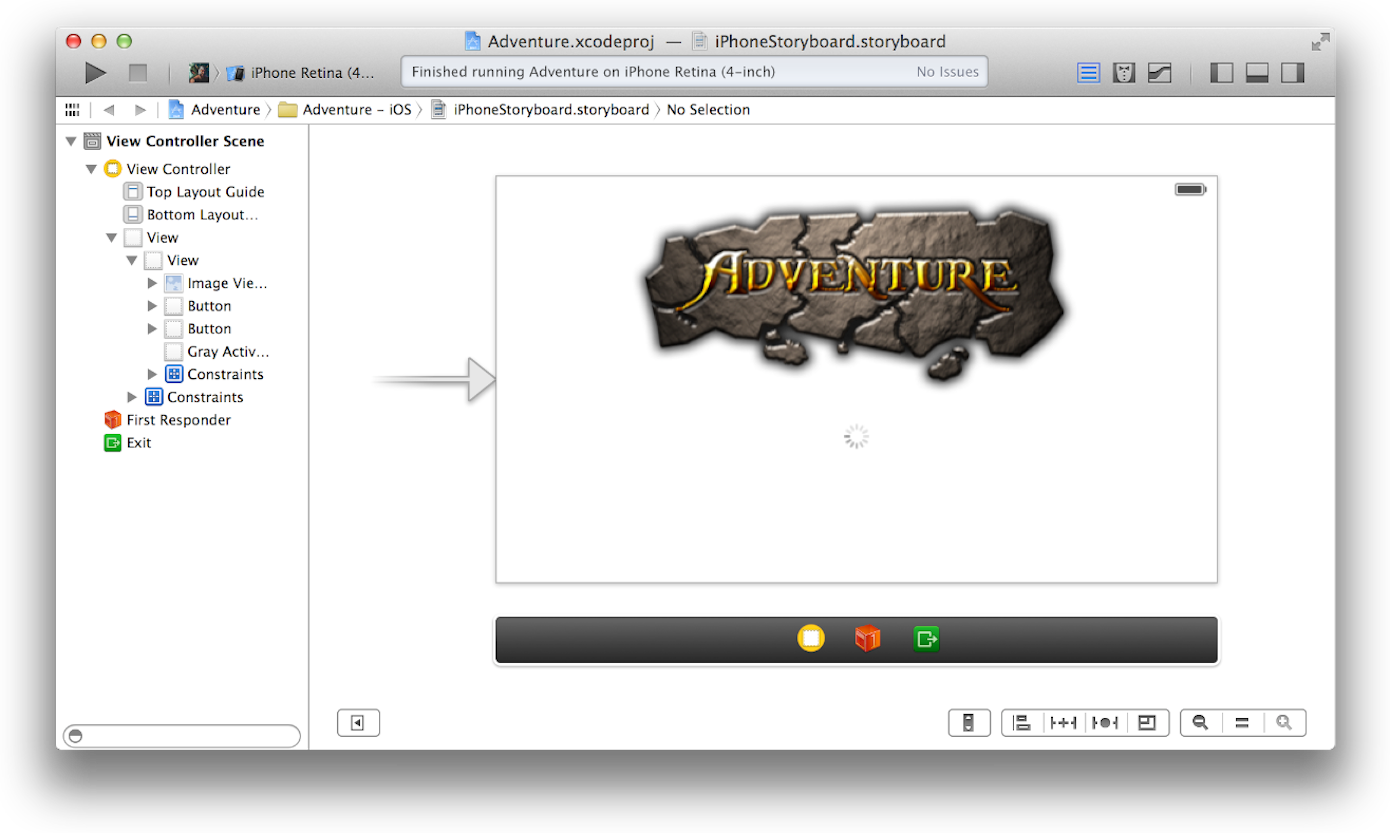


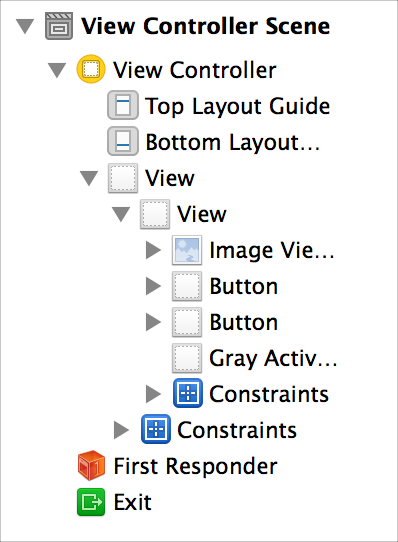
To add user interface elements, drag objects from the utilities area onto the interface builder canvas, where we can arrange the elements, set the attributes and establish connections between them and code in source files.

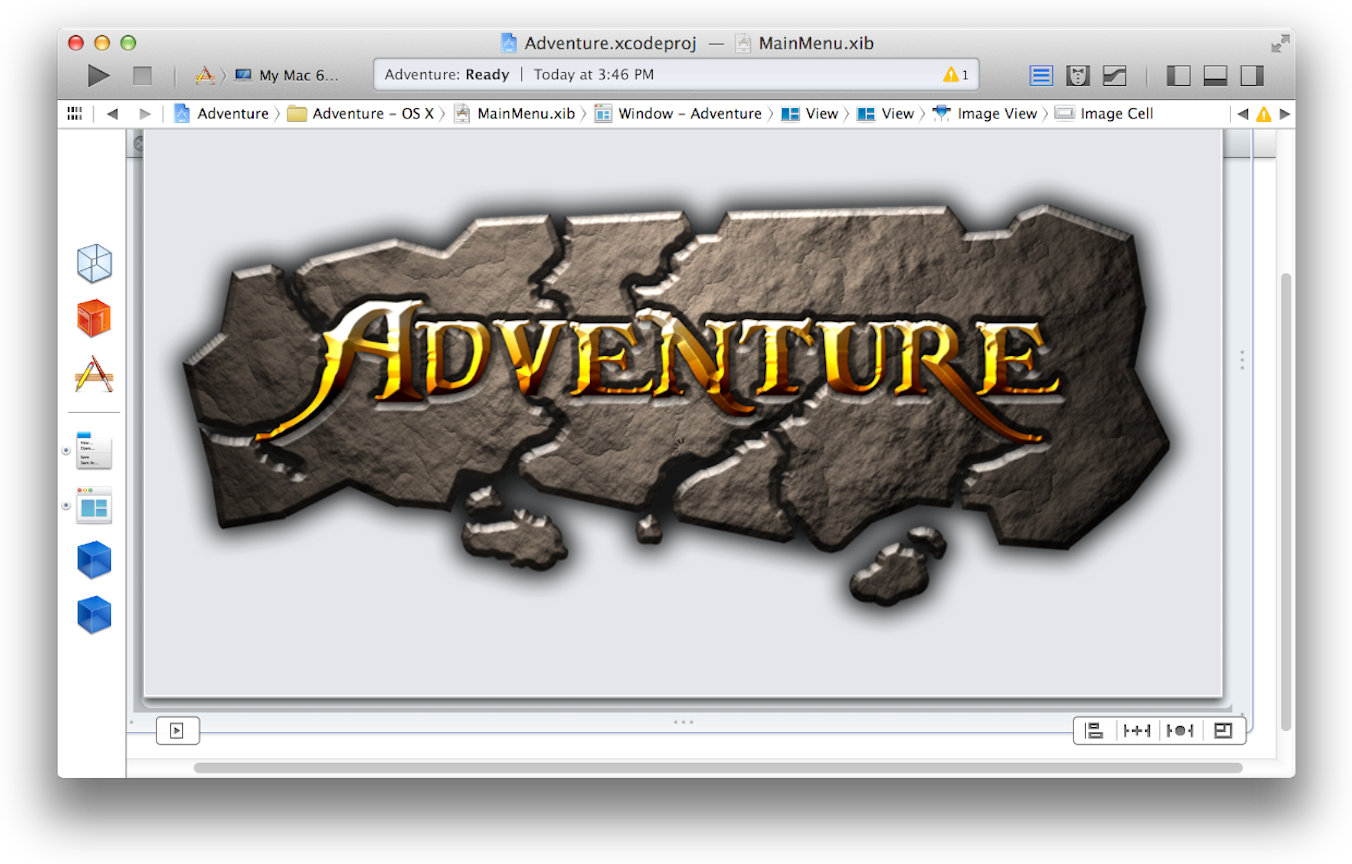


**Parts of interface builder:**

This has two major areas the dock on the left and the canvas on the right this dock list objects contained in the user interface file.

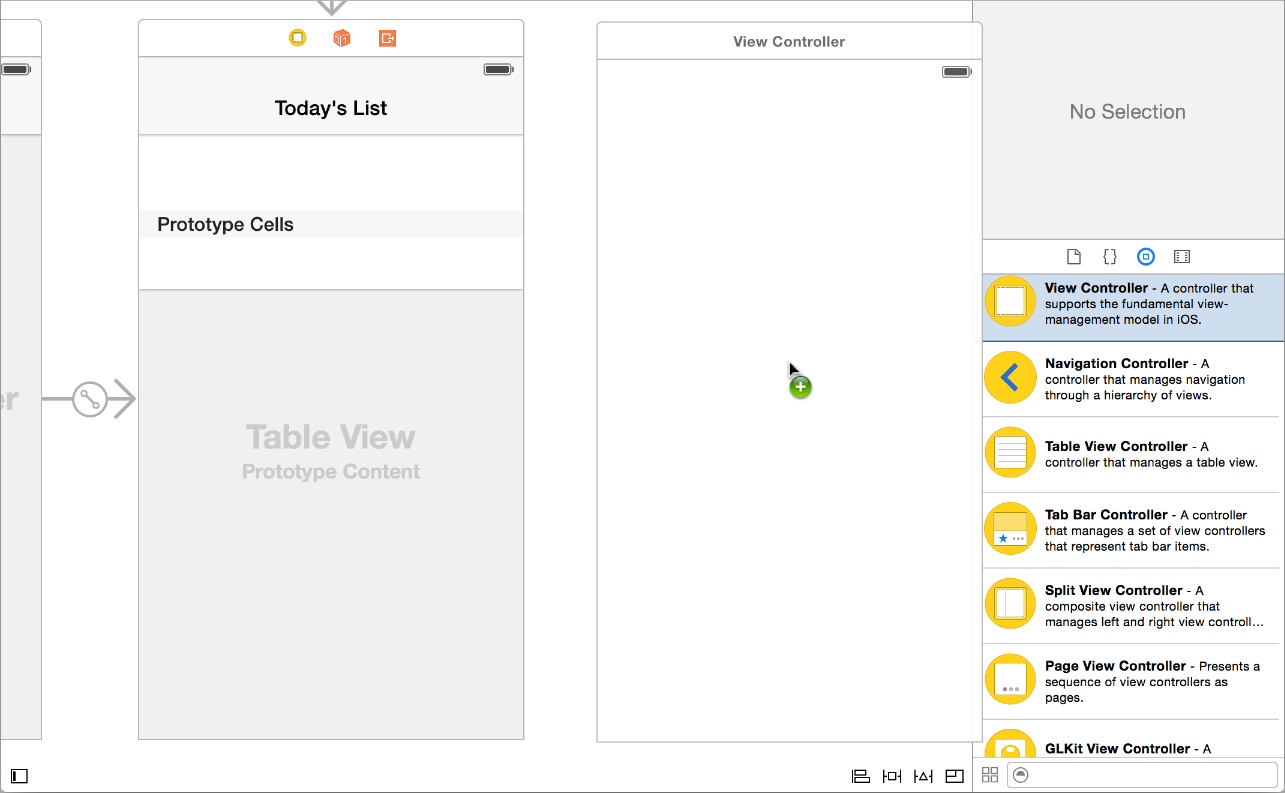






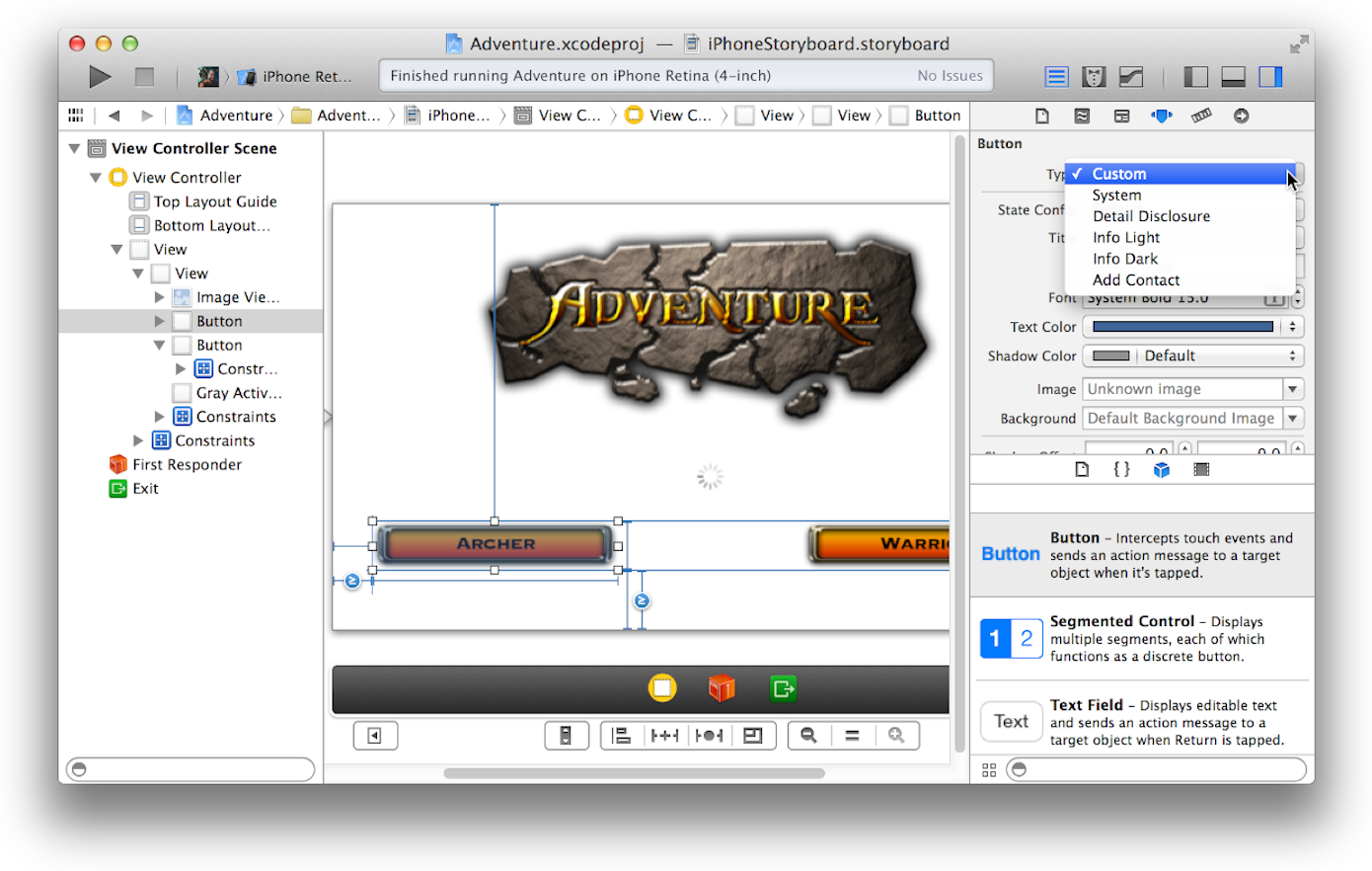
**Adding interface objects:**

To add an object into the user interface we have to open the utilities area for the workspace window by clicking the configuration buttons in tool bar and select the object from the library pane by clicking the object button in the library bar. Click the icon representing the object and the drag it from the library to the outline view in the dock.



**Finding and replacing Strings:**

We can also find and replace the strings as we have to go to the interface builder inspector and these inspectors specify some of the interface objects and appearance and behaviour.



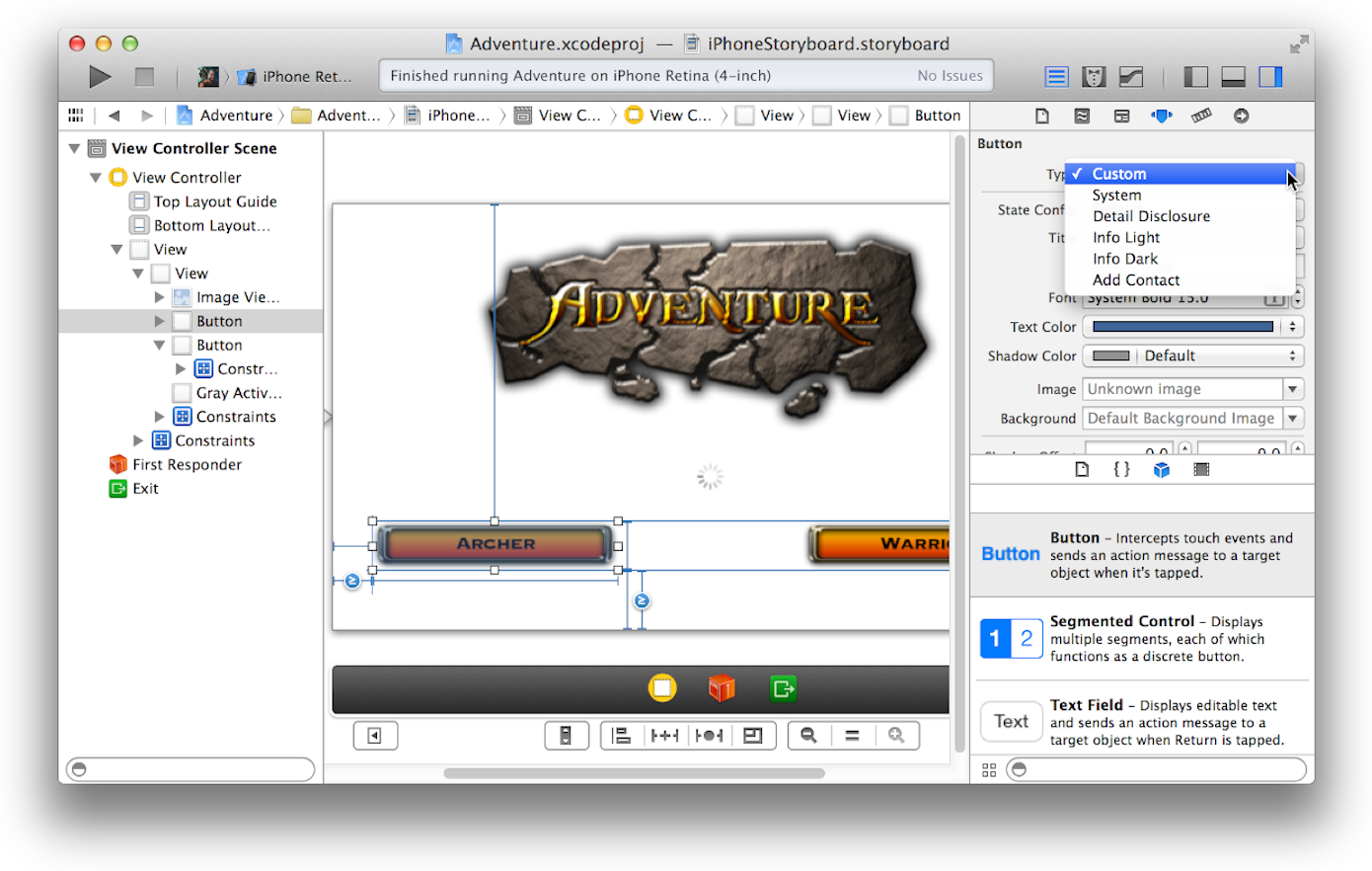
# Designing with Storyboards:

We use a story board typically without writing any code we can add the scenes accordingly. We have to use interface builder to specify your user interface in terms of:

* Scenes
* Segues between scenes
* Controls used to trigger the Segues.

A scene represents on onscreen content area. On the iPhone and iPod touch, a screen generally contains a single scene. A Segue represents the screen from one scene to the next scene.

The below screen shot shows a story board of a master detail-pattern in an iOS project. This story board contains three scenes and two segues. The left screen means the navigation controller which manages user navigation between the master and the detail scenes.

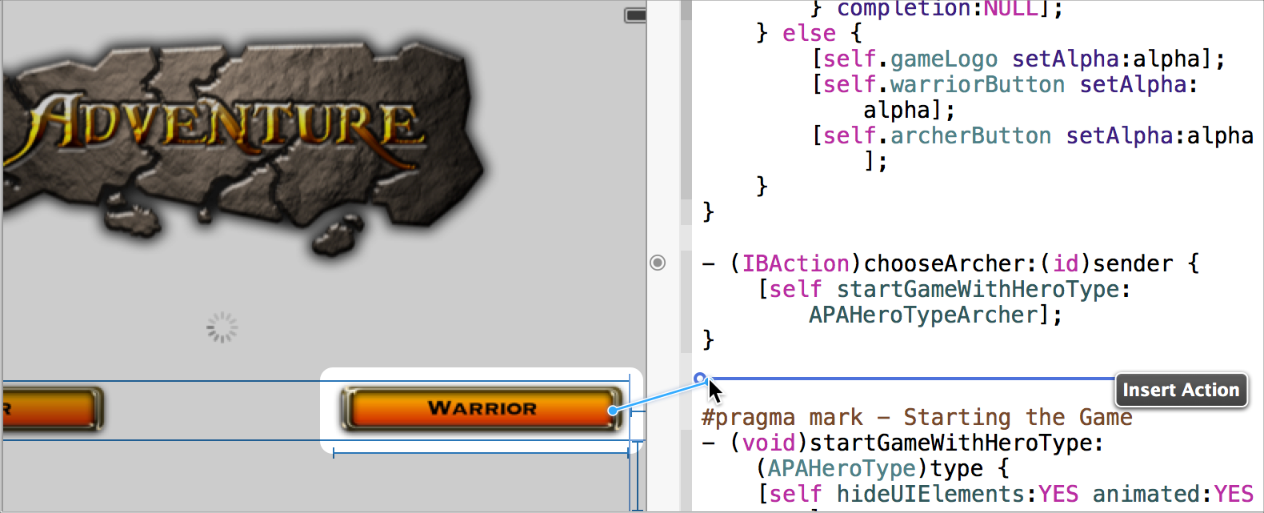


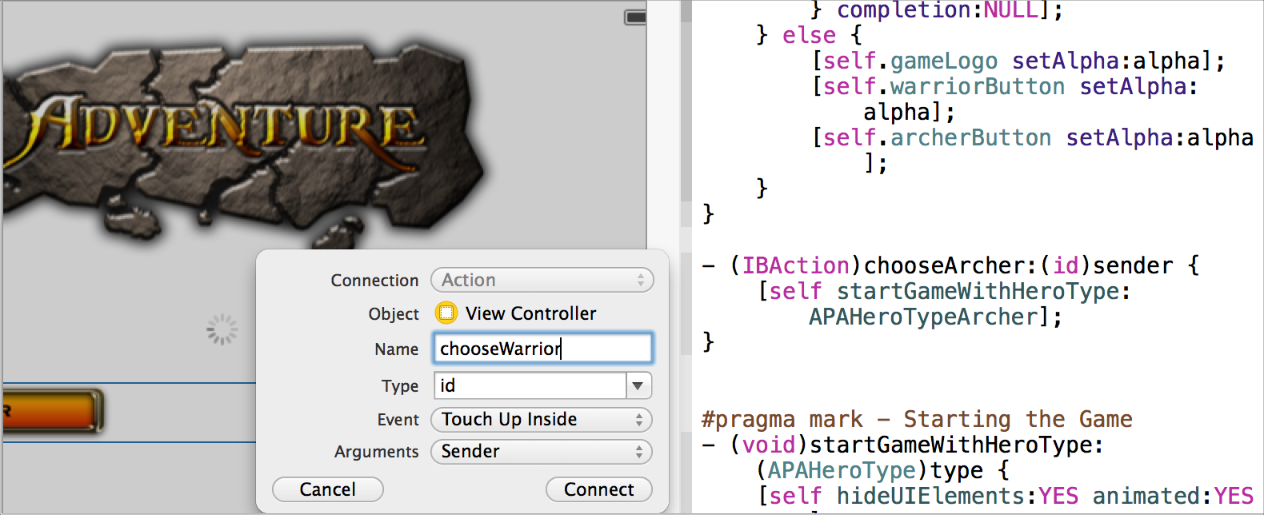
# Connecting the objects to code:

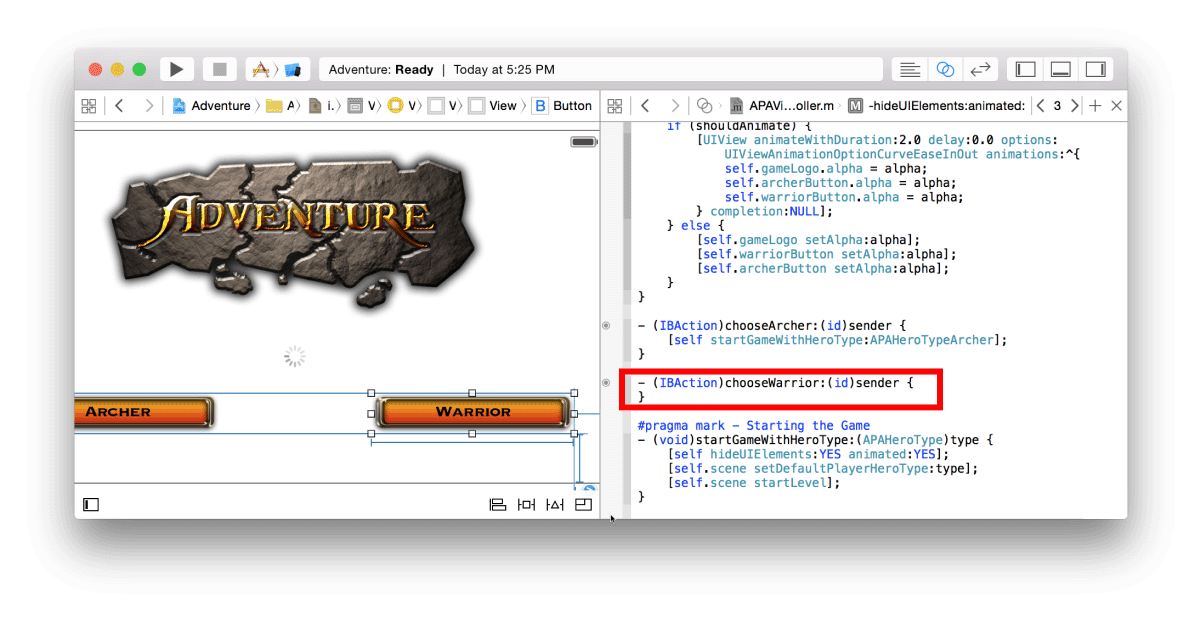
We writes the code for the implementation of behaviour of user interface objects our code is the communication with the user interface objects through action and outlet connections.

## Sending action Messages from a control to your code:

Every time the user activates a control, such as by tapping it, the control should send a message telling its code to take some action. The easiest way to configure a control to send an action message to your code is by dragging Control from the control in Interface Builder to your object's implementation file.







Thank you