Now it's time to do custom drawing. That is done in subclasses of NSView. But we'll need NSViewController objects to manage those views. Those view controllers will be held on to by a window controller. And that window controller is owned by the AppDelegate.

So with this app you'll not only do custom drawing, but also be practicing a good separation of concerns.

Main Window Controller

- Create a new window controller to manage your app's main window.
- Your window will have a toolbar at the top.
- Cliking a toolbar item will change out the view on the window.
- A window has a contentView. You should add subviews to this view.
- Your AppDelegate should have a strong reference to the

Toolbar

Drag and drop an `NSToolBar` from the library onto the window in your xib. Delete the default items. Drag an 'image toolbar item' from the library onto your toolbar pallette. Set the title and add a cool image. Make sure your item is also in the default toolbar items list or it won't appear with your window.

The toolbar item won't be selectable until it has a target and action. The window controller is a good target. So make some action methods to switch out views based on the selected item.

Swapping views

In response to the toolbar actions you'll swap out subviews of your window's contentView. You can add subviews to a view. Look for a methods on NSView like these: `addSubview` and `replaceSubview:withOtherView:`.

VIEW CONTROLLERS

- Create a view controller to manage each screen.
- View controllers manage a view
- A view controller is like a window controller. It should have a nib, and it's init method should call `initWithNibName:bundle:`. Generally bundle can be nil, because it refers to your app's main bundle. You would pass in a specific bundle if you wanted to load from a framework or if you were developing a framework yourself.

```
- (id)init
{
    self = [super initWithNibName:self.className bundle:nil];
    if (self) {
```

```
        // setup
    }
    return self;
}
```

- When you create a view controller subclass, make sure to check the 'also make a xib' checkbox. Your view controller's `view` property will be assocated with the view in the xib. You can add subviews to this view, just like you would to a window controller's window.
- Add one of your custom views as a subview. Open the library and drag out a blank `NSView` object. Then change the class of the subivew to your `NSView` subclass.

NSView Subclasses

A view knows how to draw itself. Drawing is done in `drawRect`. You give a view commands in draw rect. See the drawing videos on YouTube for more information on how to draw with `NSColor` and `NSBezierPath`. NSView Drawing Video

There are three required views for this assignment:

1. **ShapesView**. You should draw a circle, a rectangle, and a triangle. Give your shapes custom colors. Use at least one pattern image color. `[NSColor colorWithPatternImage: someImage]`. At least one shape should have a border. The view should have a background color.
2. **Graph View**. Given an array of 10 NSNubmer objects representing float values between 0.0 and 1.0, draw a line graph to represent the data. Try to make the graph fill the entire width of the view and respond to resizing.
3. **Art View**. This is your chance to be creative and explore drawing with Objective-C. Have fun with this. Impress us. Make us smile. See below for examples of past implementations.

Layout

- Your views must resize with the window. You may use autolayout. You may use springs and struts. The choice is yours.

Art View Suggestions

- Pac Man
- Music album cover art
- World flags
- Flowers
- Landscapes
- Abstract art using random numbers
- Fireworks complete with animations. -- this student was a graphics master in some other language already.

Resources

I recommend you start by writing up the basic NSView subclasses and playing around with drawing.

These videos will be helpful:

All the drawing logic you'll need is covered in my video on NSView Drawing.

Apple Programming's discussion of NSToolbar. Where he uses an AppController object, you should be using your NSWindowController subclass.

Apple Programming's discussion of NSViewController. Again, his AppController object should be our NSWindowController subclass. While he uses an enum, I have a separate IBAction method to switch out each view controller.

If you're interested, you can check out NSHipster's article about modern programming with NS_ENUM. This is not required. I did not use an enum in my example project.