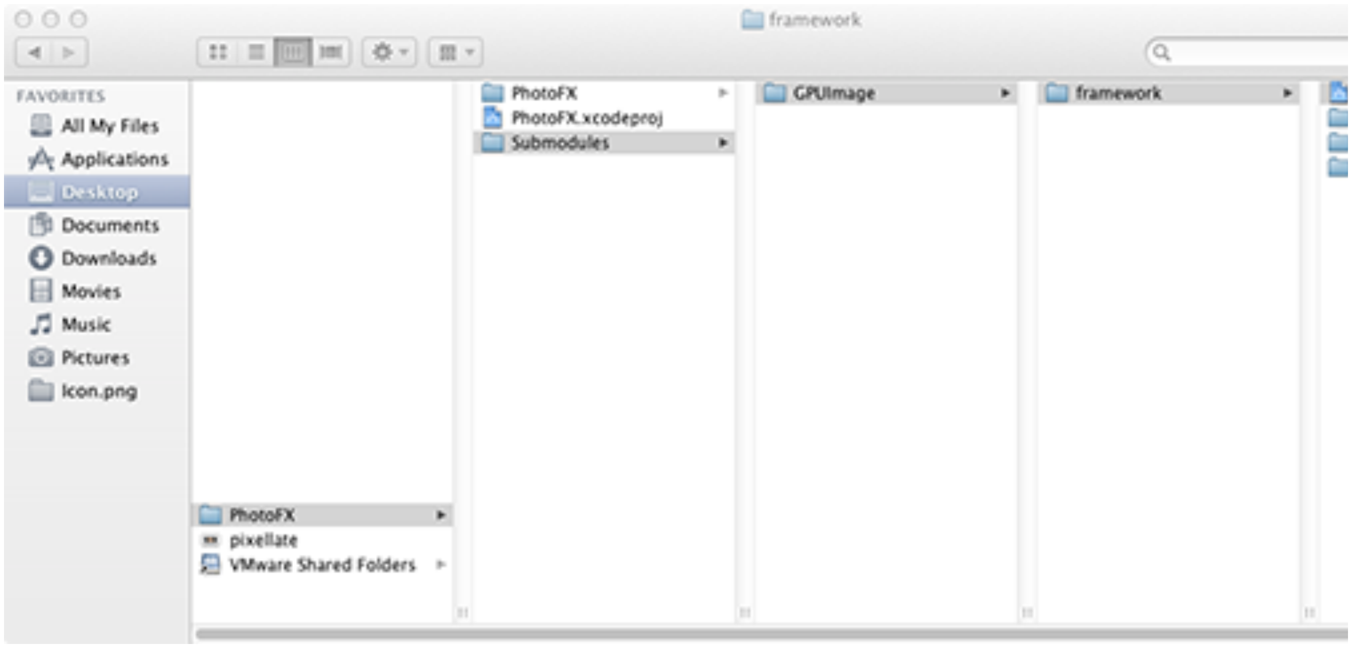
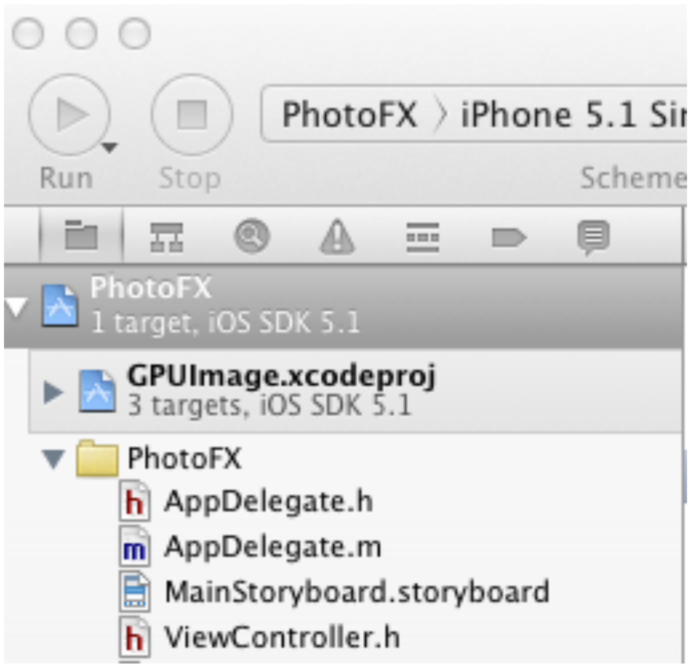
Add GPUImage to Your Project

Adding GPUImage to your project is a bit trickier than you might expect, but by following this step it should only take a few minutes for you to be up and running.

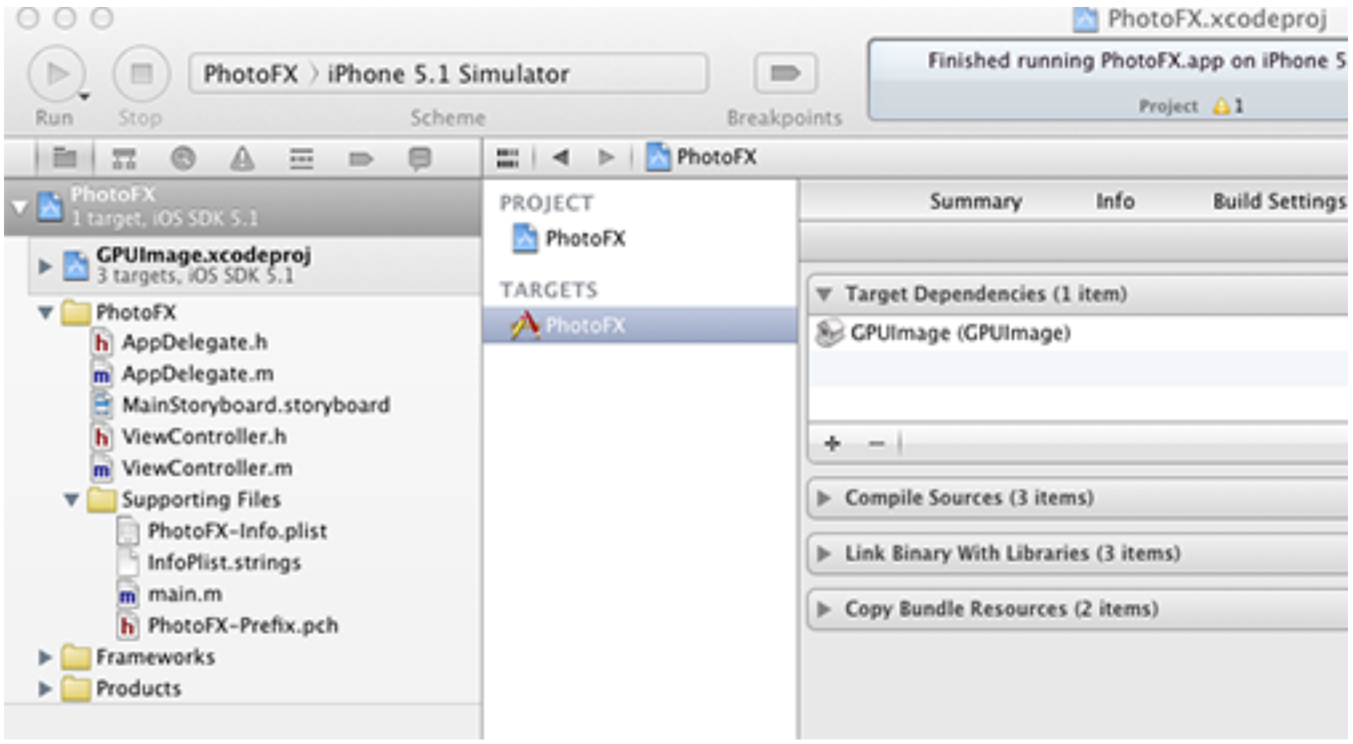
First, you need to download a copy of GPUImage from the official project GitHub. Unarchive the downloaded file and open the "framework" folder. These are the essential files needed to import GPUImage in your project. Rather than copying all of these into your project directly, use Finder to go to the location you saved your Xcode project. Create a new folder called "Submodules" with a child folder called "GPUImage". Now copy the "framework" folder downloaded from GitHub and paste it into the "GPUImage" folder. Next, open the "framework" folder and select the GPUImage.xcodeproj file. Your screen should look something like this:



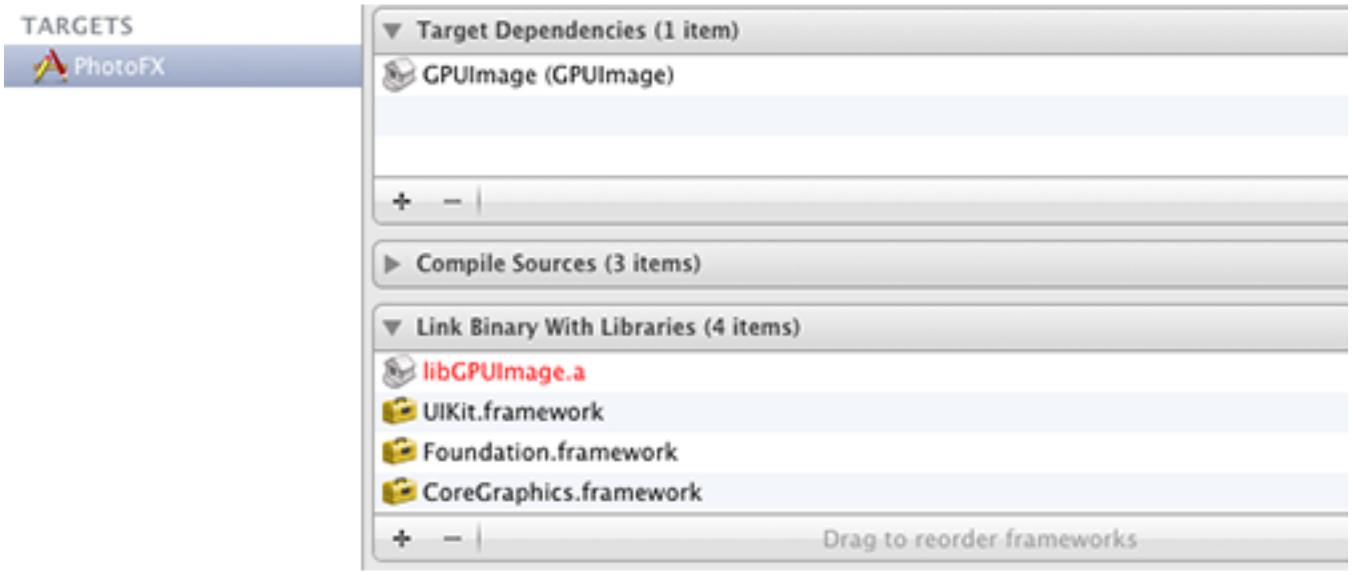
Now drag the GPUImage.xcodeproj file into the Xcode Project Navigator. If you did this successfully, you should see something like the following:



With the project added successfully, you'll need to add GPUImage as a dependency in your app's build settings. Select "[Your Project Name]" from the project navigator, select the "[Your Project Name]" target, and then go to the "Build Phases" tab. Expand the "Target Dependencies" drop down and then click the "+" icon. Select "GPUImage" from the list that appears. Take a look at the following image to get a feel for how this is done:



Now you need to drag the libGPUImage.a file (found within Xcode's Project Navigator at **GPUImage.xcodeproj > Products**) to the "Link Binary with Libraries" drop down. With this completed, you should see something like the following:


TARGETS
PhotoFX
Target Dependencies (1 item)
Compile Sources (3 items)
Link Binary With Libraries (4 items)
+ −
Drag to reorder frameworks

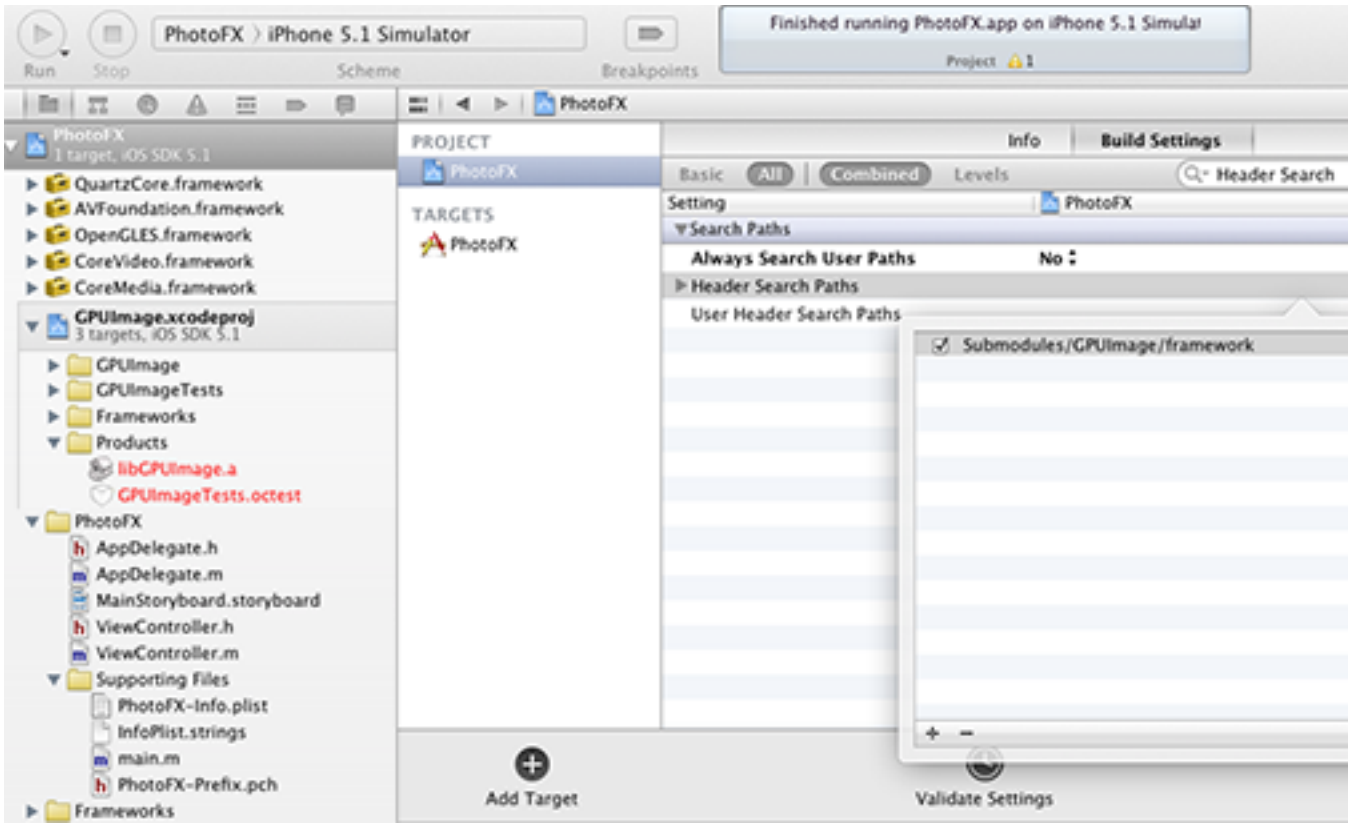
GPUImage (GPUImage)

libGPUImage.a
UIKit.framework
Foundation.framework
CoreGraphics.framework

While you're focused on the "Link Binary With Libraries" dropdown, go ahead and add the following required frameworks by clicking the "+" button in the bottom left corner:

- CoreMedia
- CoreVideo
- OpenGLES
- AVFoundation
- QuartzCore

Almost done! The next step is to select the "[Your Project Name]" project and go to "Build Settings". Search for "Header Search Paths" (you may need to select the "All" button instead of "Basic" in order for this option to appear), and then double-click to add `Submodules/GPUImage/framework` in the popup dialog that will appear. Click the checkbox next to the entry in order to indicate that this path should be searched recursively. If you did this correctly, you should be looking at something like the following:



Always Search User Paths: No
User Header Search Paths: ☑ Submodules/GPUImage/framework

The final step is to return to **ViewController.m** and add the following line at the top:

```
#import "ViewController.h"
#import "GPUImage.h"
```

You should now be able to compile and run the project without issue. **Having trouble? Email me c.konkol@rockvalleycollege.edu**

Step 8: Display a List of Filters

GPUImage comes with an impressive number of filters for use within your applications. For this tutorial, I've selected the following sample to get our feet wet:

- GPUImageGrayscaleFilter
- GPUImageSepiaFilter
- GPUImageSketchFilter
- GPUImagePixellateFilter
- GPUImageColorInvertFilter
- GPUImageToonFilter
- GPUImagePinchDistortionFilter

To create your own list or simply see all the filters GPUImage has to offer, check out the official documentation on GitHub.

In order to respond to the filter selection, we'll have to implement the `UIActionSheetDelegate`. Go to **ViewController.h** and declare that the class will conform to this delegate as follows:

```
    GPUImageFilter *selectedFilter;

    switch (buttonIndex) {
        case 0:
            selectedFilter = [[GPUImageGrayscaleFilter alloc] init];
            break;
        case 1:
            selectedFilter = [[GPUImageSepiaFilter alloc] init];
            break;
        case 2:
            selectedFilter = [[GPUImageSketchFilter alloc] init];
            break;
        case 3:
            selectedFilter = [[GPUImagePixellateFilter alloc] init];
            break;
        case 4:
            selectedFilter = [[GPUImageColorInvertFilter alloc] init];
            break;
        case 5:
            selectedFilter = [[GPUImageToonFilter alloc] init];
            break;
        case 6:
            selectedFilter = [[GPUImagePinchDistortionFilter alloc]
init];
            break;
        case 7:
            selectedFilter = [[GPUImageFilter alloc] init];
            break;
        default:
            break;
    }

    UIImage *filteredImage = [selectedFilter
imageByFilteringImage:originalImage];
    [self.selectedImageView setImage:filteredImage];
}
```

Bam! Your app should now work as desired. As you can see from the above, applying filters to an existing image with GPUImage couldn't be simpler. You simply need to instantiate a `GPUImageFilter` and then call the `imageByFilteringImage:originalImage` method.