The included project contains an interactive graphical object library for processing which supports all features from homeworks 2-4. Additionally, for homework 5, included is a slider widget implementation (homework5.processing.graphicalobject.Slider) as well as an ‘interesting UI’—a cherry blossom generator that allows for tweaking of parameters via sliders. The UI shows off the benefit of integrating with Processing since I was able to use the vast library of beautiful artwork from <http://openprocessing.org> (all written in processing) in my interface.

## Installation

The .zip folder should contain all necessary libraries and project files. Simply unzip the folder and add the project contained in the folder.

## Adding Interactive Components to a Processing Sketch

To add interactive components to a processing sketch, simply

1. When specifying the size of your sketch, add *JAVA2D* as the last argument
2. Create an interactive frame in the setup method of processing
3. In the draw method, draw your interface wherever you’d like

Here is a

**public** **abstract** **class** ExampleSketch **extends** PApplet {

**protected** InteractiveFrame myFrame;

@Override

**public** **void** setup() {

// specify the size of your sketch \*\*IMPORTANT TO ALSO INCLUDE JAVA2D\*\*

size(720, 1080, *JAVA2D*);

// Create your interactive frame, passing in this sketch

myFrame = **new** InteractiveFrame(**this**);

// set up your UI here

}

@Override

**public** **void** draw() {

// Draw your frame

myFrame.draw();

}

}

## Changes from Existing Homework Assignments

1. In order to enable image loading, all image files need to be copied to a folder called ‘data’ in the bin directory of the project.

## How to Run a Test

To make a test, it is recommended to add a main method that launches a PApplet rather than running the test as a PApplet because of a known issue. See public static void main declaration in all tests for an example.

1. Right click on a test file
2. Select Run As…->Java Application

## Known Issues/Unfinished Work

1. TestHomework2 is not completely ported. For a future class this test should be fully implemented according to the pattern in the class already.
2. There is no visual debug console as in the android application, println goes to System.out.
3. When tests are run as a PApplet, sometimes the interface does not initially get drawn. This issue does not come up, however, when a test is run as a java application (i.e. via public static void main)