# My Extension

When interacting with the map, in many cases it was hard to see where recent quakes happened where quake swarms were common. I decided to add a toggle button to let me choose between showing all quakes and showing only recent quakes.

I didn’t get too far into the process before deciding that I was probably reinventing the wheel: someone must have written a simple widget library for Processing. After searching for possible widget libraries and trying several of them, I decided on PUI by Martin Leopold. The library was small, easy to understand and had enough stuff to support what I wanted to do. At this point I decided to expand my extension to more general filtering: show or hide city markers, land quakes or ocean quakes; show or hide by shallow, intermediate, or deep depth and any combination of those 3 choices; show only recent quakes or all quakes.

I thought about using the Decorator pattern to decorate quakes by depth or by recency. I decided against it since because it felt too heavy for what is a simple enumeration (shallow, intermediate, deep) or boolean (recent or not).

As part of implementing this, I discovered PUI wasn’t really implemented with subclassing in mind. Much of it is package private, so subclassing something meant adding my control to a local copy of its source. I found a few places in PUI where classes really needed to expose information they had and added those. Overall, though, PUI is almost unchanged in my extension.

# Design Description

Depth. New enumeration in its own file. Has values SHALLOW, INTERMEDITATE, and DEEP.

EarthquakeMarker. Now holds a Depth read-only property and sets that property when the marker is constructed. A new calculateDepth() factory method returns the correct Depth value to set the property. colorDetermine() is now a switch statement based on that Depth to return the color to use for the marker. In hindsight, I should have put colorDetermine() into Depth. I changed the quake title to include the depth numeric value. Finally I added an isRecent read-only property along with an isRecent() method to set it based on the age test previously included in draw().

Visibility Control. Because of how UnfoldingMap works, EarthquakeCityMap must contain the visibility control variables. I couldn’t put them in the marker classes, which is what ultimately led me to abandoning the Decorator pattern. Each visibility control is a boolean, show<what> (i.e., showCities, showLandQuakes, showOnlyRecentQuakes). There is a public getter for each of them. PUI makes use of reflection to connect a toggle to a boolean variable, making it straightforward to get the desired control settings from the filter area to the main window. EarthquakeCityMap knows the list of cities and quakes, so it has a new method, setVisibility(), that takes a list of markers and, as long as we are not looking at a single marker in the map, has each marker set their visibility using their setVisibility() method.

CommonMarker now includes an abstract setVisibility() method that all markers must implement. CityMarker’s implementation is straightforward: if EarthquakeCityMap’s getShowCities() is true, then don’t hide the city marker. If getShowCities() is false, then the city marker will hide itself. Earthquake visibility is trickier: a quake must check the correct type filter (land quake or ocean quake), all the depth filters, and finally the recency filter to arrive at the final value. EarthquakeMarker provides a protected helper routine, depthAndRecencyVisibility(), to calculate the shared part of that boolean equation. LandQuakeMarker’s setVisibility() combines EarthquakeCityMarker’s getShowLandQuakes() with depthAndRecencyVisibility() and hides or shows the marker appropriately. OceanQuakeMarker has a similar setVisibility() method, but it refers to EarthquakeCityMarker’s getShowOceanQuakes() instead.

At this point, I could have submitted my extension since I could filter by different combinations of markers, depths, and recency. However, I was not happy with how PUI displays toggles—they look like checkboxes instead of toggle switches or slides, but with the label below the checkbox and in a much smaller font. I wanted the label to appear to the right of the checkbox and be rendered in a larger size. I also wanted the indicator to look more like ones found on today’s cell phones: a sliding button that indicates on or off both by position and by color, so color- and visually-impaired people can see the value. These sliding indicators typically have “On” to the right within a green trough, and “Off” to the left with a gray trough.

SlideIndicator. I created a new widget, SlideIndicator, by subclassing Toggle. Doing so I discovered PUI isn’t really written for subclassing, as most of its class implementations use package private members instead of public or protected members. I ended up copying the source into its own package in my project directory so I could then locate SlideIndicator in the PUI library package. This let me avoid touching the PUI source code for the most part; there were a few methods I had to add for getters to expose a few padding values that shouldn’t have been private.

SlideIndicator manages its own label instead of using the one in Toggle (which is private). The slide trough is double the width of the Toggle checkbox, and there is padding between the slide trough and the label. The widget is fully integrated into PUI and works with PUI event and layout management with no additional changes thanks to subclassing Toggle. Within the slide trough is a circle indicator, drawn in the text color since that is what my phone does. When “On”, the circle is on the right side of the trough (centered at 3/4 of the trough width and ½ the trough height) and the trough color is green. When “Off”, the circle is on the left side of the trough (centered at ¼ of the trough width and ½ the trough height) and the trough color is gray. In both cases the circle diameter is slightly less than the trough height, so the trough color will show completely around the circle.

The SlideIndicator detects a mouse hover over the trough and changes the trough outline color to yellow to acknowledge the hover. Clicking anywhere in the trough will toggle SlideIndicator’s value. I decided not to make the label sensitive to hovering or clicking; if you want to change the value, click the slider not its description.

FilterController. The class that creates the filter area, which is below the key and to the left of the map. Its constructor builds a header label called “Show:” and then 3 SlideIndicators for city markers, land quakes and ocean quakes, then a divider line, then 3 SlideIndicators for shallow quakes, intermediate quakes, and deep quakes, then another divider line, and finally 1 SlideIndicator for choosing to show only recent quakes. Each SlideIndicator is set up to control a different boolean value in EarthquakeCityMap, as previously described. FilterController keeps track of the widest widget it creates and resizes itself and the dividers to that width after all widgets are constructed. It also uses PUI’s color theme support to define the “on”, “off”, text, trough outline, trough outline highlight, and canvas background colors used by all the SlideIndicators. Since all widgets know how to draw themselves, and FilterController itself doesn’t have an on-screen display, there are no other methods needed after the constructor. It doesn’t require even a draw() method, because in PUI the applet actually has the list of all widgets created so it notifies each widget to redraw itself. FilterController itself is not a widget, just a simple class that instantiates 10 widgets as a “group”.

# Time Estimate

|  |  |
| --- | --- |
| 1.75 hr | Initial research into adding a toggle-like control for picking “show only recent” vs. “show all” |
| 2.25 hr | Decide it is silly to reinvent widgets and events, someone must have done already. Search for, study, try and decide on a Widget library meant to use with Processing. Select PUI. Decide to support many different filter options, not just show only recent/show all. |
| 1 hr | Build initial FilterController with stock PUI Toggles, a title Label, and a couple Dividers. |
| 1 hr | Investigate Decorator pattern to use for filtering types of Depth and Recent/Not Recent. Decide this is too heavy. Create Depth enumeration and make all quake markers indicate depth when created. |
| 0.25 hr | Turn recency into a property to make filtering easier. |
| 1.5 hr | Toggles control marker visibility for cities, land and ocean quakes, shallow, intermediate and deep quakes, and show only recent/show all. |
| 0.75 hr | Correct bug that broke ability to focus on 1 marker. |
|  | At this point, could have submitted writeup. However, stock PUI Toggle looks like a check box with the label below it. PUI author punted on allowing label alignment flexibility. |
| 0.5 hr | Create SlideIndicator by subclassing Toggle, get everything working as before. |
| 1.5 hr | Have SlideIndicator manage its own label (instead of trying to control the one buried inside Toggle), centered vertically on the checkbox and to the right of the checkbox. Finalize map, key, and filter area layout. |
| 1.25 hr | Change SlideIndicator to use a “slide trough” instead of a checkbox. Mimic the slide indicator found on phones (with color and visual on/off indications for color blind people): “On” is a circle using the text color in the right side of the trough, which is a green color. “Off” is the same circle but on the left side of the trough with a gray trough color. |
| 1 hr | Make FilterController auto-adjust its width based on the width of the widest widget it is managing. Change the dividers to also span the filter control area. Finalize slider highlight color shown when hovering over the trough. |
| 2.25 hr | Initial version of my extension write-up (this document). |

Total: 15 hours development time