Part III – Externalizing the panorama assets

There is an issue with distributing source code for your demo's. Besides demonstrating what you want to demonstrate, you give people a look at how you program. But the focus is not how I program, it's demonstrating the concepts of creating interactive panorama's with hotspots that light up. This means I have to find some middle ground in which the code is kept to the bare minimum without hardly any framework classes or complex program structures, while keeping the example readable and extensible.

Therefore I decided to open this post with a little information on application structures. Probably one of the hardest parts of programming is rising to/with the correct level of complexity. You want to be prepared for anything, but not over complicate things (KISS).

So what do you choose for your application structure? You can choose one of the many frameworks, or you can roll with your own. And after that, do you apply that framework to any application you write?

Personally I love to use a setup which allows me to refactor fast, so I can keep the application as simple as possible and then go more complex when needed. This might not always be preferable when working with larger teams, but for these tutorials, for me it is a nice way to work.

For example:
super simple concept demonstration => some quick & dirty code on a timeline
simple concept demonstration => single class demo without any clear division of responsibilities
concept demonstration => couple classes, mainapplication and

subparts start to take shape
more formal simple single view application => mainapplication, mainmodel, mainview (with possibly other names like we have in this example)
large scale multiview application => mainapplication, startup tasks, mainmodel, mainview, submodels, subviews etc.

I could write a book about this but I won't anytime soon ;).

In the more simpler variants, models might initialize themselves and they might use simple callbacks (like in this example). In more complex variants, models might be deserialized from xml, but queueloaders and services might take care of loading other required data, and everything might communicate through events instead of callbacks, maintaining some sense of progress and reporting errors when necessary.

If the application gets really complex, I usually employ a main application model, through which all the application parts are accessible. For people familiar with PureMVC's ApplicationFacade, it shares similarities (although the way I build stuff everything is strongly type-checked and I almost never employ singletons. What I am doing in these examples with arrays without accessor functions is something that would be a nono in large applications). If the application gets even more complex, the application gets divided into sub applications / modules / application cores etc and the process repeats itself.

All in all there is a pretty seamless way to scale up when required, and when you know ahead of time that you will be building a monstrosity there is always the option of scaling up from the very beginning. The main thing is that the complexity of the architecture is usually matched by the complexity of what you want to achieve (warranted by).

So back to these tutorials, I'm trying to find a middle ground between demonstrating the principles, keeping a decent

structure which can be extended, not over complicating things, not adding a bunch of library classes to bulk load content etc etc. And some days I succeed better at that than other days :).

Just remember the focus is on concepts, not on application structure nor holy code conventions wars. And that most of this is written after 9:30 PM when my gremlins have gone to bed…

All that said, since we will be adding hotspots and light up features in the coming examples, this example prepares the way to add those things later. I've scaled up the example's complexity a bit and replaced the DistortedPlane class by a less general CubicPanoramaPlane class.

All images are now loaded externally, and the example can no longer be run from the Flash IDE since there is no longer a fla required (although it is very simple to create one yourself). Instead you should download the latest FlashDevelop version.

As mentioned before, there is no errorhandling, no progress display and probably lots of other stuff missing.
Feel free to add it, but it is not part of what I am trying to demonstrate here:

(Please give the panorama a moment to load)
Download the source here: 3d panorama v0.3 (628)