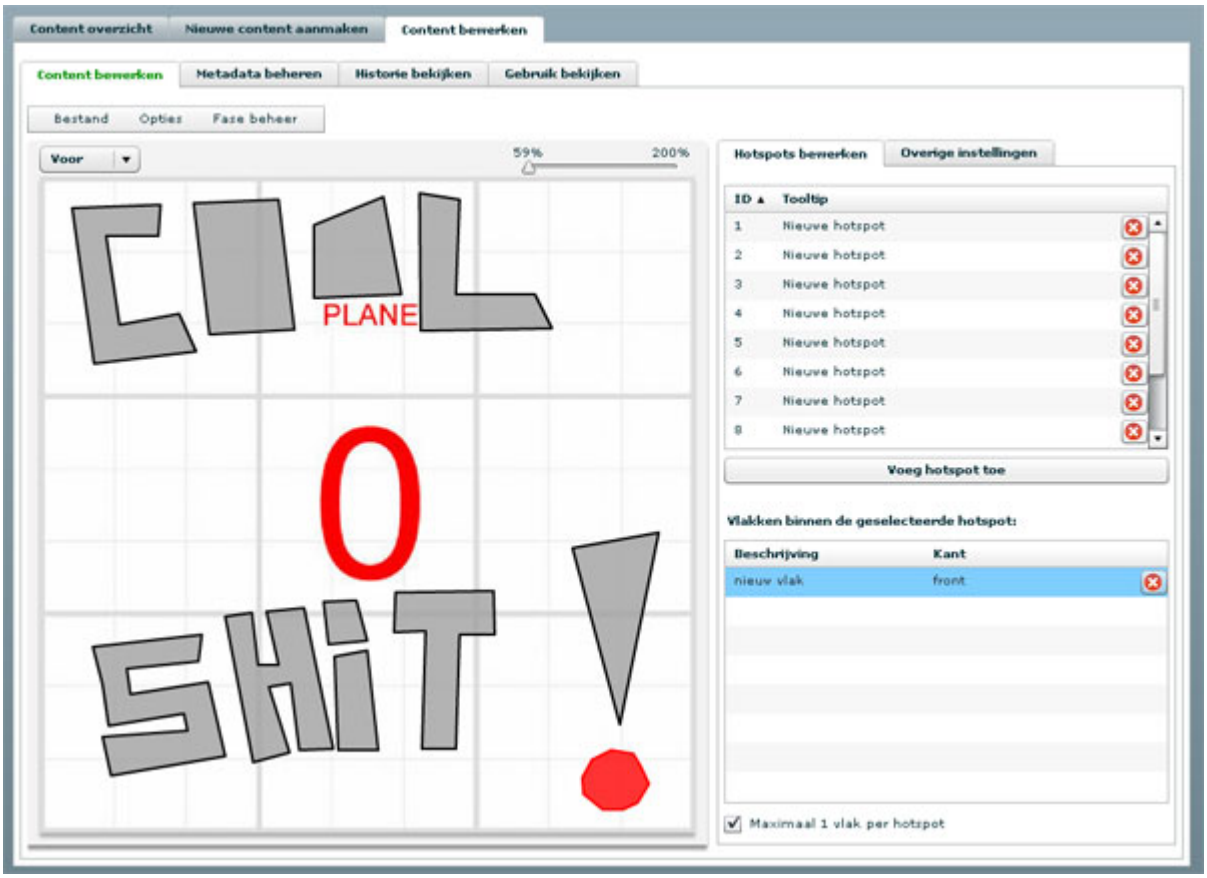
Part VI – Panorama hotspot interaction

How you define your hotspots is up to you. For example you could write a tool which allows you to edit and generate a CubicPanorama. For a project for the dutch police academy I'm currently working on at TriMM, I am writing such an editor in Flex. Here is a screenshot of the work in progress:



Basically it allows you to select a cube side, import a picture for it and draw hotspots on it. The complete definition/model for the cube is then saved to xml, the polygon information with it.

In this demo I'm simply generating rectangular hotspots on the fly randomly as you can see when you reload the page. No matter whether you are using irregular shapes through xml or random rectangles the principle stays the same.

Download the sources here: 3d Panorama v0.6 (741 downloads)

One important note though before I continue, **the hotspots do not have to be visible and not rectangular either**. The only reason they are here, is to keep it simple and to provide some visual clue of where you should point and click. In addition whether you show a handcursor or not is all up to your implementation.

But like I said the principle stays the same. In the last post we showed how to derive an x,y coordinate within a plane's texture. But we do not have to use the plane's texture, we can use any x,y lookup system. For example an array as a look up table. But wait, there is a better way, we simply use another bitmap as a lookup table.

So the basic idea is: for each cubeside, create another bitmap called lookupbitmap with the same size as the texture, and draw the hotspots for each plane onto the lookupbitmap with the hotspot's ids as colorvalue.

From an architectural standpoint there are again multiple approaches of course. I introduced the CubicPanoramaModel a couple of examples ago, so if you do define your hotspots in xml, that would be a good place to store them. I decided to leave that as an exercise to the reader (you) however, and keep it simple.
We could implement a bitmap manager, material manager, hotspot manager or whatever. I decided to implement this feature in a subclass of a plane's material and call it InteractiveMaterial. In this case, since it's a bit hacked together and unoptimized, it's called MockupInteractiveMaterial, since it is just there to demonstrate the principles.

The MockupInteractiveMaterial generates 5 random hotspots with an id of 1-5 for plane 0, 11-15 for plane 1, 21-25 for plane 2 etc. It paints these hotspots into it's lookup table and onto

it's own material so you have a visual clue of where to point your mouse. You might even want to be able to toggle this hotspot visibility in which case you would have to retain a copy of the original material, before you ruin it with random rectangles.

There are a couple of things to keep in mind though:

- you will have to deal with overlapping hotspots. In this example I simply drew the hotspots into the lookup table using the lighten blend mode, which will give the hotspot with the highest id precedence.
- by default flash will draw everything anti-aliased, which cannot be switched off except by setting quality to low. Drawing a hotspot with an id of 10 will result in some pixels around the edges of the hotspot having id 9 which ruins the idea. The 'trick' is to draw your hotspot into an inbetween bitmap and threshold that bitmap on the id/pixel value that you want.

Checking which hotspot is under the mouse is now as simple as extending the interactive material with the following method: