KrPano Beginner Tutorial

Einleitung

Der Einstieg in krpano ist aufgrund der (noch) nicht vorhandenen GUI für viele Einsteiger nicht einfach und ob der vielen Möglichkeiten sehr verwirrend. Dieses Tutorial soll helfen, die Verwirrung etwas zu mildern, um den Einstieg in krpano zu finden. Der Anfänger braucht meiner Meinung nach zunächst schnelle Erfolgserlebnisse um nicht die Lust am Experimentieren zu verlieren. Da ich im Internet kein (deutsches) Tutorial zu krpano gefunden habe, will ich versuchen selbst eines zu erstellen. Wenn diese Beschreibung dazu beiträgt, dass krpano als nicht mehr so kryptisch oder kompliziert angesehen wird, dann habe ich mein Ziel erreicht. Diejenigen Leser, die eine allumfassende Beschreibung oder eine vollständige Dokumentation suchen, sind hier leider falsch. Aber legen wir doch erstmal los

Voraussetzungen

Dieses Tutorial geht davon aus, dass ein eigenes "equirectangulares" Panorama vorhanden ist. Falls noch kein eigenes Panorama vorhanden ist, stelle ich hier mal eines zur Verfügung (Achtung: ca.11 MB). Die Auflösung ist relativ egal, hier geht es ja erstmal um den Einstieg. Des Weiteren sollte neben dem krpano-Player auch die krpanotools vorhanden sein (gibt's beides als Demo auf der Seite krpano.com). Für dieses Tutorial verwende ich die Versionen 1.0.8 Beta 8 (Player und Tools) und zusätzlich die Examples aus der Version 1.0.7 (Player).

Die krpanotools sind für die Darstellung des interaktiven Panoramas nicht erforderlich, aber sie erleichtern den Einstieg erheblich. Wenn der geneigte Leser beabsichtigt, sich eine Lizenz für krpano zu kaufen (die ist notwendig, wenn die erstellten Panoramen veröffentlicht werden sollen), empfehle ich die Tools gleich mit zu kaufen. Die Tools bieten noch einige andere Annehmlichkeiten, die ich später noch beschreiben werde. Und um Spekulationen gleich zu unterbinden: nein, ich erhalte für diese Werbung keine Provision o. ä. (Anfragen bzgl. meiner Bankverbindung zum Zwecke des finanziellen Sponsoring beantworte ich aber jederzeit gerne...)

Das schnelle Erfolgserlebnis

Im Verzeichnis der krpanotools gibt es eine Datei, die sich "SPHERE to CUBE MULTIRES droplet.bat" nennt. Von dieser Datei erstellen wir zunächst eine Verknüpfung auf dem Desktop, da wir die zukünftig häufiger brauchen. Als nächstes erstellen wir einen schnell zu findenden Ordner, z.B. c:\panotest (Name ist wurscht). In diesen Ordner kopieren wir ein beliebiges equirectangulares Panorama unserer Wahl. Und nun beginnt die Zauberei: wir ziehen das kopierte Panorama auf die Verknüpfung "SPHERE to CUBE MULTIRES droplet.bat". Es öffnet sich eine DOS-Box und irgendwas passiert da mit dem Pano. Anschließend drücken wir eine Taste und die DOS-Box hat ihre Schuldigkeit getan und verschwindet wieder. Das Ergebnis: In dem vorher erstellten Ordner existiert nun eine xml- und eine html-Datei mit dem Namen des Panoramas. Außerdem noch ein Ordner, ebenfalls mit dem Namen der Panorama-Datei und zusätzlicher Endung .tiles und die beiden Ordner "swfobject" und "skin". Wenn wir nun die html-Datei per Doppelklick öffnen - Vodoo - erscheint das fertige interaktive Panorama. Verstanden haben wir jetzt zwar noch nix, aber immerhin ein fertiges

Panorama. Damit das mit dem nix verstehen nicht so bleibt, der Versuch einer Erläuterung (ohne Anspruch auf Vollständigkeit):

Die "SPHERE to CUBE MULTIRES droplet.bat" (der Einfachheit halber ab sofort nur noch "Droplet" genannt) sorgt dafür, dass aus der ursprünglichen equirectangularen jpg-Datei ("Sphere") ein "Cube" (Würfel) erzeugt wird. Außerdem wird der Cube in viele einzelne Stücke (Tiles) mit unterschiedlichen Auflösungen transformiert. Das hat den Riesenvorteil, dass selbst gigantische Panoramen in annehmbarer Zeit dargestellt werden können, da beim Betrachten nur die Tiles der jeweiligen Zoom-Stufe geladen werden. Um die Zuordnung, wann welches Tile geladen wird, brauchen wir uns nicht kümmern. Ich sach ja: krpano is cool.

Des Weiteren wurde eine html-Datei erstellt, die nix anderes macht, als den eigentlichen Player (krpano.swf) zu laden und als Parameter bzw. Variable den Namen der xml-Datei übergibt. Eigentlich ist die html-Datei jetzt schon fertig zur Veröffentlichung auf dem Web-Server, aber um die gelegentlich vorbei schauenden Suchmaschinen zu streicheln, sollten noch kleinere Optimierungen hinsichtlich von Meta-Tags, Seitentitel usw. vorgenommen werden. Das ist aber wieder ein anderes Thema und Tips und Tricks dazu sind im Internet reichlich zu finden. Suche zum Thema SEO (Search Engine Optimization oder so ähnlich)...

Da wir uns ja mit dem krpano-Player beschäftigen, interessiert uns besonders die xml-Datei. Mit dieser Datei wird der Player gesteuert. Bevor wir uns die Datei näher ansehen, hier schonmal der erste wirklich nützliche Tip für die Zukunft: Grundlage für die xml-Datei ist die "standard_skin.xml" im Verzeichnis "krpanotools-1.0.8-beta8-win\kmakemultires.templates\xml". Wenn wir später Dinge an der xml-Datei ändern, die in allen unseren zukünftigen Panoramen erscheinen sollen, wie z.B. ein eigenes Logo oder nützliche Funktionen oder ähnliches, dann tragen wir die Änderungen in genau diese Datei, das "Template" ein. Für's erste lassen wir die aber mal so wie sie ist und experimentieren mit der erstellten xml-Datei in unserem Testordner.

Werfen wir also mal einen Blick in diese Datei und schauen, was da so drin steht:

```
  <!-- set initial view -->
  <view hlookat="0" vlookat="0" fov="90" maxpixelzoom="1.00"
fovmax="150" />
```

"set initial view", also hier geben wir an, mit welchem Blickwinkel der Betrachter startet.
"hlookat" = horizontal look at
"vlookat" = vertikal look at
"fov"= field of view
"maxpixelzoom" = maximale Vergrößerung
"fovmax" = field of view maximum

Und bevor ich mir nun einen Wolf erkläre, erste Übung: Werte verändern, Datei speichern, Pano anschauen und sehen wie sich das Start-Bild verändert.

Als nächstes folgen diverse Abschnitte um die Navigationsbuttons darzustellen, die wir zunächst mal ignorieren.

```
  <!-- events - fullscreen button change, set cursor on start -->
  <!-- actions to change the mouse cursor -->
  <!-- button showtext() style -->
```

```
  <!-- some default buttons (zooming, direction, hotspots on/off,
fullscreen) -->
```

Am Ende der Datei finden wir unser Pano wieder:

```
  <!-- the preview and pano image -->

  <preview type="CUBESTRIP" url="testpano.tiles/preview.jpg" />

  <image type="CUBE" multires="true" tilesize="480">
    <level tiledimagewidth="3840" tiledimageheight="3840">
      <left  url="testpano.tiles/l3_l_%0v_%0h.jpg" />
      <front url="testpano.tiles/l3_f_%0v_%0h.jpg" />
      <right url="testpano.tiles/l3_r_%0v_%0h.jpg" />
      <back  url="testpano.tiles/l3_b_%0v_%0h.jpg" />
      <up    url="testpano.tiles/l3_u_%0v_%0h.jpg" />
      <down  url="testpano.tiles/l3_d_%0v_%0h.jpg" />
    </level>
    <level tiledimagewidth="1920" tiledimageheight="1920">
      <left  url="testpano.tiles/l2_l_%0v_%0h.jpg" />
      <front url="testpano.tiles/l2_f_%0v_%0h.jpg" />
      <right url="testpano.tiles/l2_r_%0v_%0h.jpg" />
      <back  url="testpano.tiles/l2_b_%0v_%0h.jpg" />
      <up    url="testpano.tiles/l2_u_%0v_%0h.jpg" />
      <down  url="testpano.tiles/l2_d_%0v_%0h.jpg" />
    </level>
    <level tiledimagewidth="960" tiledimageheight="960">
      <left  url="testpano.tiles/l1_l_%0v_%0h.jpg" />
      <front url="testpano.tiles/l1_f_%0v_%0h.jpg" />
      <right url="testpano.tiles/l1_r_%0v_%0h.jpg" />
      <back  url="testpano.tiles/l1_b_%0v_%0h.jpg" />
      <up    url="testpano.tiles/l1_u_%0v_%0h.jpg" />
      <down  url="testpano.tiles/l1_d_%0v_%0h.jpg" />
    </level>
  </image>
```

Das war schon Alles.

Das noch schnellere Erfolgserlebnis

Die zweite Möglichkeit zu einem noch schnellerem Erfolg zu gelangen besteht darin, sich auch die Version 1.0.7 des krpano-Player herunter zu laden und aus den dort mitgelieferten Examples den Ordner "Buttons" zu wählen. In die entsprechende xml-Datei geschaut stellen wir fest, dass hier alles sehr übersichtlich ist. Es gibt zwei Einträge, bei denen jpg-Dateien eingetragen sind:

```
  <preview url="testpano.jpg" />

  <image type="SPHERE">
```

```
  <sphere url="testpano.jpg" />
</image>
```

Ersetzen wir hier "testpano" durch den Namen unseres equirectangularen Bildes, kommen wir noch schneller zum Ziel. Namen ändern, Datei speichern, html-Datei aufrufen - fertig.

Der Umstand, dass ich zuerst die Variante mit dem Droplet beschrieben habe, hat neben der Tatsache, dass ich diese Variante für besser halte noch einen ganz bestimmten Grund, den ich später beschreibe, wenn es darum geht, die Dateien auf dem Webserver zu platzieren.

Eigene Forschung

Am besten im krpano-Ordner "Examples" die einzelnen Beispiele mal anschauen. Jedes Verzeichnis enthält ein separates Beispiel für eine Funktion, wobei der Verzeichnisname die entsprechende Option beschreibt.

Panorama auf Webserver laden

Um unser Panorama nun auch im Internet zu präsentieren, muss eine Reihe von Dateien und Verzeichnissen auf den Webserver geladen werden:
Auf jeden Fall die krpano.swf und die dazugehörige Lizenz-Datei (beide im gleichen Verzeichnis), den Ordner "swfobject" mit inhalt, den Ordner "skin" und optional den Ordner "plugins". Wie die Ordnerstruktur auf dem Webserver letztendlich aussieht, ist eigentlich egal. Hauptsache die Verweise in den xml- und html-Dateien laufen nicht ins Nirwana. Um die Lizenz-Datei und die krpano.swf nicht mehrfach auf dem Webserver speichern zu müssen, empfielt es sich, die Panoramen grundsätzlich in Unterordner zu platzieren und die URL für den Verweis auf die krpano.swf in der jeweiligen html-Datei dann entsprechend zu gestalten ("../krpano.swf"). Gleiches gilt für die Ordner swfobject, skin und plugins.
Je nachdem, wie das Panoraman dargestellt werden soll, muss noch das equirectangulare Panorama (Variante 2, noch schnelleres Erfolgserlebnis) oder der entsprechende *.tiles-Ordner (Variante 1, Multiresolution-Panorama) auf den Webserver übertragen werden.

Der Grund, warum ich die Variante mit den Multires-Droplet für besser halte ist ganz einfach: erstens gibt es auch ein Template für die html-Datei und zweitens wird uns so für jedes Panorama eine eigene html-Datei erzeugt. Im Template für die html-Datei (Verzeichnis krpanotools-1.0.8-beta8-win\kmakemultires.templates\html) machen wir einmal die Anpassungen für die Verweise auf die krpano.swf und den anderen Kram, auf den noch verwiesen wird und brauchen uns später diesbezüglich um nix mehr kümmern. Auch bietet es sich an, grundlegende Meta-Tags in die Vorlage zu integrieren um das spätere Feintuning zu erleichtern.

Nützliches (Gut zu wissen)

Options Plugin

Für die ersten eigenen Experimente kann es sinnvoll sein, zwei weitere Plugins einzubinden. Plugins sind kleine (oder auch große) Programmteile, die zum Panorama "dazugeladen" werden können. Klingt

wieder kompliziert, wenn man's erklären (oder gar verstehen) soll, ist aber in der Praxis ganz simpel. Dazu fügen wir in unsere xml-Datei einfach mal die folgende Zeile ein:

```
<plugin name="options" url="./plugins/options.swf" />
```

Der Ort in der xml-Datei, an dem diese Zeile eingefügt wird, ist relativ wurscht. Nur sollte man darauf achten, dass kein "Block" zerschnitten wird. Mit Block meine ich einen Bereich zwischen "<" und "/>". Und wer jetzt aufgepasst hat, wird bemerkt haben, dass die neu eingefügte Zeile ja auch so ein "Block" ist. Man kann so einen Block vielleicht auch als "Anweisung" oder wie auch immer bezeichnen, jedenfalls kann man diese Zeile sinngemäß interpretieren: (Gespräch zwischen Programmierer und krpano)

<	Achtung, jetzt kommt eine Anweisung!
plugin	lade einen Programmteil
name	diesen Programmteil nennen wir zukünftig:
"options"	wenn ich also zukünftig "options" sage, meine ich diesen Programmteil
url	das/den Programmteil findest du:
"./plugins/options.swf"	im Ordner plugins und die Datei heisst (zufällig auch) options.swf
/>	Anweisung ende - weitermachen! (Vor dem Schrägstrich Leerzeichen nicht vergessen!!!)

Und was macht diese Plugin nun? Wenn die Zeile eingefügt wurde und wir uns das Panorama ansehen, stellen wir fest, das rechts oben ein kleiner Button sichtbar ist. Was macht man mit Buttons? Richtig: anklicken. Es erscheint eine Box mit vielen Parametern. Diese Parameter können jetzt verändert werden, teils durch anklicken, teils durch verschieben der Regler (dunkelgraue Kästchen). Bei einigen Parametern ist die Auswirkung sofort sichtbar, bei einigen nicht. Keine Angst, hier kann hemmungslos gespielt werden, dem Panorama passiert nix.

Editor Plugin

Das zweite sinnvolle Plugin ist der Editor (der in der nächsten krpano-Version noch erheblich erweitert / verbessert werden soll). Die Kommandozeile zur Anzeige des Editors lautet:

```
<plugin name="editor"  url="../plugins/editor.swf" align="bottom"
x="10" y="10" />
```

Der Aufruf ist ähnlich wie beim options-Plugin. Zusätzlich sind die Parameter align (Ausrichtung), eine x- und eine y-Koordinate hinzugekommen, die dafür sorgen, dass der Editor-Button unten mittig angezeigt wird.
Die Nützlichkeit des Editors erschließt sich nicht unbedingt auf dem ersten Blick, zumindest erging es mir so. Von daher wieder ein praktisches Beispiel: Wir wollen den Anfangsblickwinkel komfortabel verändern. Dazu öffnen wir das Pano und wählen einen Blickwinkel, wie er uns gefällt. Wie bekommen wir den jetzt gesichert? Ganz einfach: Editor-Button anklicken, xml-Button anklicken und es erscheint die xml-Datei mit den aktuellen Einstellungen. Nun haben wir die Möglichkeit, einzelne Abschnitte, oder die gesamte Datei in die Zwischenablage zu kopieren. Um die gesamte Datei zu kopieren, nutzen

wir den Button "copy to clipboard". Es besteht aber auch die Möglichkeit, einzelne Abschnitte zu kopieren. Dazu markieren wir jetzt den Abschnitt mit dem Namen "view", in dem wir mit gedrückter Maustaste über den Abschnitt streichen. Zum kopieren drücken wir <strg>+<c>, Achtung Stolperfalle: <strg>+<Einf> funktioniert nicht! Nun können wir den in der Zwischenablage enthaltenen Anweisungsblock in unsere xml-Datei kopieren und das Panorama startet zukünftig genau dort, wo wir das festgelegt haben.

```
<view hlookat       ="-19.134613"
   vlookat       ="1.629859"
   camroll       ="0.000000"
   fov           ="22.281591"
   fovmin        ="22.281591"
   fovmax        ="150.000000"
   maxpixelzoom  ="1.000000"
   limitfov      ="true"
   fisheye       ="0.00"
   fisheyefovlink="0.50"
   stereographic ="false"
   architectural ="0.0"
   architecturalonlymiddle="false"
   limitview     ="auto"
   hlookatmin    ="NaN"
   hlookatmax    ="NaN"
   vlookatmin    ="NaN"
   vlookatmax    ="NaN"
   />
```

Das Option- und das Editor-Plugin ist nützlich für die Erstellung eines Panoramas, im veröffentlichten Panorama auf dem Webserver möchten wir die beiden aber nicht unbedingt zeigen. Für dieses Problem habe ich zwei Möglichkeiten anzubieten:

Zeilen auskommentieren

Die Syntax, um Zeilen oder Abschnitte in der xml-Datei auszukommentieren ist recht simpel:

<!--	Kommentar start
/>	Kommentar ende (Leerzeichen vor dem Schrägstrich)
-->	Kommentar ende (siehe Anmerkung)

Um also z.B. das Editor-Plugin "abzuschalten", genügt es am Anfang der Zeile !-- einzufügen:

```
<!-- plugin name="editor"  url="../plugins/editor.swf" align="bottom"
x="10" y="10" />
```

Anmerkung:
*Klaus, (Entwickler von krpano) hat mich darauf hingewiesen, das ein XML Kommentar eigentlich mit "**-->**" endet. Das ist natürlich richtig. Die Tatsache, dass die von mir beschriebene Variante auch funktioniert, liegt am sehr toleranten XML Parser des Flashplayers. Falls es also in einer späteren Version des Flashplayers nicht mehr funktioniert, dann wissen wir wenigstens warum. Mit "-->" am Ende sind wir auf jeden Fall auf der sicheren Seite, da die XML-Konventionen das so*

vorsehen. Dank an Klaus für den Hinweis!

Eine zweite, wesentlich elegantere Methode (die ich selber nutze) ist das

Einbinden externer xml-Dateien

Damit können wir gleich mehrere Vorteile vereinen. Erstens können wir die xml-Dateien für das einzelne Panorama übersichtlich halten, in dem wir Teile der xml-Dateien (wie z.B. die Navigations-Buttons u.a.) in eine separate xml-Datei auslagern, zweitens platziere ich in dieser Datei Dinge, die für alle meine Panoramen verfügbar sein sollen (wie z.B. Kontextmenu, Buttons und Logo) und drittens kann die separate xml-Datei auf dem Webserver anders aussehen als die lokale. So habe ich z.B. die beiden eben beschriebenen Plugins lokal immer verfügbar und wenn ich meine Panoramen veröffentliche, eben nicht, da die Plugins in der "veröffentlichten externen" Datei eben nicht enthalten oder auskommentiert sind. Und ich brauche mich bei der Veröffentlichung neuer Panoramen nichteinmal mehr darum kümmern. Meine "externe" Datei habe ich "allpanos.xml" genannt und mit der Zeile

```
<include url="../allpanos.xml" />
```

in das bereits erwähnte Template ("Das schnelle Erfolgserlebnis") eingebunden.

Bevor ich nun verrate, was noch alles in meiner externen Datei steht, hier noch ein Tip für das Template:

Logkey abschalten

Wird beim betrachten eies Panoramas die Taste "o" (nicht null) gedrückt, erscheint im Normalfall im unteren Rand ein Fenster, in dem u.a. der Name, auf den der krpano-Player registriert ist. Wenn das nicht gewünscht ist, kann in die Startzeile der xml-Datei der Eintrag

```
logkey="false"
```

eingefügt werden, um das Fenster zu verhindern. Die Vollständige Zeile sieht dan so aus:

```
<krpano version="1.0.8" logkey="false">
```

Kontextmenü

In meiner externen Datei habe ich u.a. das Kontextmenu eingebunden, dass beim Klick mit der rechten Maustaste im Panorama erscheint. Mit diesem Kontextmenu gebe ich dem Betrachter die Möglichkeit, zwischen verschiedenen Projektionsarten zu wählen. Auch hier ist die Praxis wieder einfacher als die Theorie. Also starten wir wieder mit dem einfachen Teil.
Dazu wählen wir aus den Examples der Version 1.0.7 das Beispiel Kontextmenu. Der wesentliche Teil steht gleich am Anfang der Datei:

```
  <!-- right-click context menu to change the viewing settings -->

  <contextmenu>
```

```
    <item caption="KRPANO"     />
    <item caption="FULLSCREEN" />
    <item caption="normal view"
onclick="action(rectview);"          separator="true" />
    <item caption="fisheye view"
onclick="action(fisheyeview);"       />
    <item caption="architectural view"
onclick="action(architectural);"     />
    <item caption="stereographic view"
onclick="action(stereofisheyeview);" />
    <item caption="little planet view"
onclick="action(littleplanet);"      />
  </contextmenu>
```

Damit sind erstmal die Menü-Einträge vorhanden. Die Zeilen beginnen jeweils mit "item" (sinngemäß sowas wie "Eintrag" oder "Teil") und daneben wird die "caption" angegeben. Das ist genau der Text, der als Menüzeile später im Menü erscheint. Als nächstes folgt die onclick-Definition, die nichts weiter bedeutet als "wenn diese Menuzeile angeklickt wird, dann führe die bezeichnete Aktion mit dem Namen xxx aus". Eine Aktion ist nichts anderes als eine Reihe von Anweisungen, die uns im verwendeten Beispiel auch schon auf dem Silbertablett serviert werden:

```
  <!-- actions for view changing -->
  <action name="rectview">
    tween(view.fovmax,       155.0, distance(179, 0.25),
easeoutquad);
    tween(view.architectural,  0.0, distance(1.0, 0.25),
easeoutquad);
    tween(view.fisheye,        0.0, distance(1.0, 0.25), easeoutquad,
set(view.stereographic,false); );
  </action>

  <action name="fisheyeview">
    tween(view.architectural, 0.0, distance(1.0, 0.25), easeoutquad);
    tween(view.fisheye,       0.0 ,distance(1.0, 0.20), easeoutquad,
       set(view.stereographic,false);
       set(view.fovmax,179);
       tween(view.fisheye, 0.35, distance(1.0, 1.25));
      );
  </action>

  <action name="stereofisheyeview">
    tween(view.architectural, 0.0, distance(1.0, 0.25), easeoutquad);
    tween(view.fisheye,       0.0 ,distance(1.0, 0.10), easeoutquad,
       set(view.stereographic,true);
       tween(view.fisheye, 1.0, distance(1.0, 1.25));
       tween(view.fovmax,  150, distance(179, 1.25));
      );
  </action>
```

```
  <action name="littleplanet">
    tween(view.architectural, 0.0, distance(1.0, 0.25), easeoutquad);
    tween(view.fisheye,       0.0 ,distance(1.0, 0.10), easeoutquad,
        set(view.stereographic,true);
        tween(view.fisheye, 1.0, distance(1.0, 0.75));
        tween(view.fov,     130, distance(179, 0.75), easeoutquad,
set(view.fovmax,150););
        tween(view.vlookat,  90, distance(179, 0.75), easeoutquad);
      );
  </action>

  <action name="architectural">
    tween(view.fovmax,       155.0, distance(179, 0.25),
easeoutquad);
    tween(view.architectural,  1.0, distance(1.0, 0.25),
easeoutquad);
    tween(view.fisheye,        0.0, distance(1.0, 0.25), easeoutquad,
set(view.stereographic,false); );
  </action>
```

Die Actions auch noch in die externe Datei eingefügt und schon ist das Thema Kontextmenu für's erste abgefrühstückt. Und da behaupte noch jemand, krpano sei kryptisch oder kompliziert. Die eigentliche Programmierarbeit ist doch schon erledigt, wir müssen uns nur die gewünschten Teile aus den Examples zusammenkopieren. Wer jetzt noch genauer wissen möchte was da passiert, der sei auf die Dokumentation der krpano-Seite verwiesen.

Buttons

Für ein "vollwertiges" Panorama fehlen jetzt nur noch die Buttons für die Navigation. Als schnelle Lösung bedienen wir uns des Examples aus der Version 1.0.8. In dem entsprechenden Verzeichnis gibt es eine Datei die sich "buttons-include.xml" nennt. Den Inhalt der Datei kopieren wir in unsere externe Datei. Die im Verzeichnis enthaltenen *.jpg und *.png-Dateien benötigen wir ebenfalls, sollten aber auch schon im Ordner "skin" vorhanden sein.
In der Datei "buttons.jpg" sind die Grafiken für die Buttons zusammen gefasst. Jeder einzelne Button ist 40 x 40 Pixel groß. Die obere Reihe sind die Standard-Buttons, die zweite Reihe enthält die Buttons, die angezeigt werden, wenn die Maus über einen Button steht und die untere Reihe enthält die Buttons, die beim Klicken auf den Button angezeigt werden. Wer eigene Buttons erstellen möchte und wem die Zusammenfassung in einer Datei dabei nicht gefällt, der sollte sich das Button-Beispiel der Version 1.0.7 ansehen, dort sind die jeweiligen Buttons in separaten Dateien abgelegt.

Damit dürfte erstmal genügend Information zum experimentieren vorhanden sein. Ich hoffe der Einstieg in krpano fällt mit dem vorliegenden Tutorial etwas leichter und weckt die Neugierde, weitere Möglichkeiten mit krpano zu erforschen. Material für weitere Tutorials ist auf jeden Fall noch reichlich

vorhanden...

Zum zweiten Teil geht's hier lang.

Sollte ich mich irgendwo irgendwie vollkommen unverständlich ausgedrückt haben, oder sogar falsche Fehler übersehen haben, wäre ich für Rückmeldung (Kontaktformular) dankbar.

Weitergehende Fragen empfehle ich im Forum (krpano.com oder www.panorama-forum.net) zu stellen. Dort gibt es meistens hilfreiche Antworten.

Viel Spaß beim ausprobieren!

2.Teil Einsteiger-Turorial für krpano

Eigenes Logo einbinden

Im 2. Teil des krpano-Tuorials geht es weiter mit "Gut zu wissen". Als nächstes wollen wir unser eigenes Logo einbinden. Wer sich schonmal meine Panoraman auf dieser Seite angesehen hat, wird vielleicht festgestellt haben, dass ich immer 2 Logos eingebunden habe: Das erste oben links, das nach einer Weile im Nichts verschwindet und ein zweites, kleineres Logo im Nadir, das permanent vorhanden ist. Gemeinsam ist beiden, dass sie zu meiner Homepage verlinkt sind. Bereits im ersten Teil habe ich meine "externe" xml-Datei beschrieben, die ich allpanos.xml genannt habe und die über das Template in alle meine Panoramen eingebunden wird. Auch die Logos werden in dieser Datei platziert, so dass nicht nur alle zukünftigen Panoramen automatisch mit Logo versehen sind, sondern auch alle bisherigen, in die die externe Datei eingebunden wurde. Beginnen wir mit dem großen Logo oben links (oder "lefttop", wie der moderne Angelsachse sagt):

Grundlage für meine Logos ist eine Bilddatei (*.png), die ich mit einem Bildbearbeitungsprogramm erstellt habe. Das png-Format deshalb, weil jpg keine Transparenz darstellen kann. Grundsätzlich ist aber jedes gängige Grafikformat geeignet, dass von krpano unterstützt wird.

Die eigentliche Einbindung des Logos ist (wieder mal) absolut simpel:

```
<plugin name="logo"
  url="xxxxxx_logo.png"
  align="lefttop"
  x="10"
  y="10"
  />
```

Somit ist die Grafik schonmal in allen Panoramen zu sehen. Die Parameter sind fast schon selbsterklärend: url = Pfad zur Grafik, align = Ausrichtung, x und y-Koordinaten. Wer's mag, kann noch ein Event hinzufügen, nämlich `onclick = " "`. Zwischen die beiden Gänsefüßchen noch eine auszuführende Aktion einfügen und fertig. Bei meinen Panos gelangt man z.B. auf die Startseite meiner Website. Vollständig sieht das dann so aus:

```
<plugin name="logo"
  url="xxxxxx_logo.png"
  align="lefttop"
  x="10"
  y="10"
  onclick="openurl(http://www.irgendwohin.de,_self)"
  />
```

Hinter dem URL kann noch angegeben werden, wie die Zielseite geöffnet werden soll. Wer sich etwas mit html auskennt, dem kommen die Möglichkeiten bekannt vor. _self = im gleichen Fenster öffnen, _blank = im neuen Fenster öffnen. Achtung Stolperfalle: die openurl-Aktion funktioniert nur, wenn die Dateien im Netz liegen! Lokal beim testen nicht wundern, wenn der URL nicht aufgerufen wird.

Da dieses Logo ziemlich groß ist, wollen wir den Betrachter nicht permanent damit belästigen. Also lassen wir das Logo nach einer bestimmten Zeit einfach verschwinden. Die Action dafür sieht so aus:

```
<action name="ausblenden">

tween(plugin[logo].alpha,0,7,easeInExpo,set(plugin[logo].visible,fals
e));
  </action>
```

Die genaue Beschreibung der Syntax ist auf der krpano-Seite zu finden (nach "tween" suchen), daher verzichte ich hier auf eine genauere Beschreibung. Wir wollen uns ja schließlich nicht mit Fachwissen quälen, sondern Ergebnisse sehen ;)) Grundsätzlich kann man sich "tween" als eine Eieruhr vorstellen, der wir eine Variable nennen (`plugin[logo].alpha`), den Zielwert (0, also vollkommen ausgeblendet) und die Zeit in Sekunden (hier 7) bis die Aktion abgeschlossen sein soll. Zu allem Überfluss (welch ein Luxus) können wir auch noch eine weitere Aktion angeben, die ausgeführt werden soll, wenn die eigentliche Aktion fertig ist. In unserem Fall setzen wir die Eigenschaft "visible" auf "false", also sichtbar = falsch. Jetzt kann man natürlich zu recht fragen, warum wir die Sichtbarkeit abschalten, wenn die Grafik doch schon ausgeblendet ist. Ganz einfach: damit der Link nicht mehr funktioniert. Die Tatsache, dass etwas ausgeblendet ist, bedeutet ja nicht, dass es nicht mehr da ist. Fragt mich jetzt aber bitte nicht nach dem Unterschied zwischen ausgeblendet und unsichtbar. Das ist höhere Programmiererlogik, der wir uns im Tutorial Nummer 628 widmen werden... ;))

Wo wird die action nun aufgerufen? Mit der Deklaration alleine ist es ja noch nicht getan. Das Logo ist bei jedem Aufruf eines Panoramas sichtbar. Also rufen wir die action doch einfach auf, sobald das Pano startet. Dazu platzieren wir den Aufruf in die Kopfzeile der Panorama-xml:

```
<krpano version="1.0.8" onstart="action(ausblenden);">
```

Wer sich noch an den ersten Teil des Tutorials erinnert, dem ist klar, dass natürlich die Kopfzeile des Templates entsprechend angepasst wird, damit das für zukünftige Panoramen automatisch enthalten ist. Da die externe (allpanos.xml) ja in andere xml-Dateien eingebunden wird, hat diese selbst keine Kopfzeile.

Nadirlogo

Das Nadirlogo können wir im Prinzip genau so einbinden, wie das erste. Es gibt nur ein grundlegenes

Problem: das Nadirlogo am Bildschirm mit x- und y-Koordinaten ausrichten macht wenig Sinn, denn es soll ja nur im Bodenbild zu sehen sein. Aber das Problem ist natürlich kein Problem, hier die Lösung:

```
<hotspot name="nadirlogo"
  url="logo_kl.png"
  ath="0"
  atv="90"
  distorted="true"
  scale="1.0"
  rotate="0.0"
  rotatewithview="true"
  onclick="openurl(http://www.irgendwohin.de,_self)"
  />
```

Die beiden Variablen ath und atv kann man sich vielleicht als @horizontal und @vertikal merken, in unserem Beispiel liegt das Logo also horizontal bei 0° und vertikal bei 90° (-90° wäre im Zenit). Der Parameter "rotatewithview" sorgt dafür, dass sich das Logo horizontal mit dem Betrachtungswinkel dreht. Wer das nicht mag, der setzt rotatewithview einfach auf "false".

Wer jetzt etwas Fantasie entwickelt, wird feststellen, dass wir nun im Prinzip schon das grundlegene Rüstzeug für eine Tour beisammen haben. Statt unserem Logo im Nadir können wir natürlich beliebige Grafiken einbinden, wie z.B. Pfeile oder was auch immer für einen Hotspot geeignet ist. Die Platzierung innerhalb "der Kugel" ist nun dank der ath- und atv-Parameter überall möglich. Als URL tragen wir dann einfach das nächste Panorama ein und schon klappt's. (Die Stolperfalle nicht vergessen: funktioniert lokal nicht!) Für die Platzierung der Hotspots ist wieder das Editor-Plugin sehr von Nutzen: Hotspot irgendwo einfügen, Panorama aufrufen, Klick auf den Editor-Button, Klick auf "hotspots" und schon ist der Hotspot bequem mit der Maus zu positionieren. Die Hinweise im Panorama oben links beachten! Hat der Hotspot seine Position gefunden, Klick auf "back", Klick auf "xml" und im Editor wie bereits im ersten Teil beschrieben, den Block für den Hotspot kopieren und in die eigene xml einfügen.

Für eine rudimentäre Verbindung mehrerer Panoramen mag das zunächst ausreichen. Aber gerade im Hinblick auf die beschriebene Stolperfalle ist die Erstellung umfangreicherer Touren natürlich nur Sinnvoll, wenn auch lokal getestet werden kann. Das wie verrate ich dann bei der nächsten Fortsetzung.

Fortsetzung folgt...
