

PTViewer Documentation

Updated for Version 1.8

December 6, 2000

Helmut Dersch

Technical University Furtwangen

der@fh-furtwangen.de

PTViewer is an open and free Panorama viewer. It is written in the Java language (v.1.0.4) and should run in any java-enabled browser. From Version 1.4, there is an additional Java-application which runs without any browser on Java 1.1 installations.

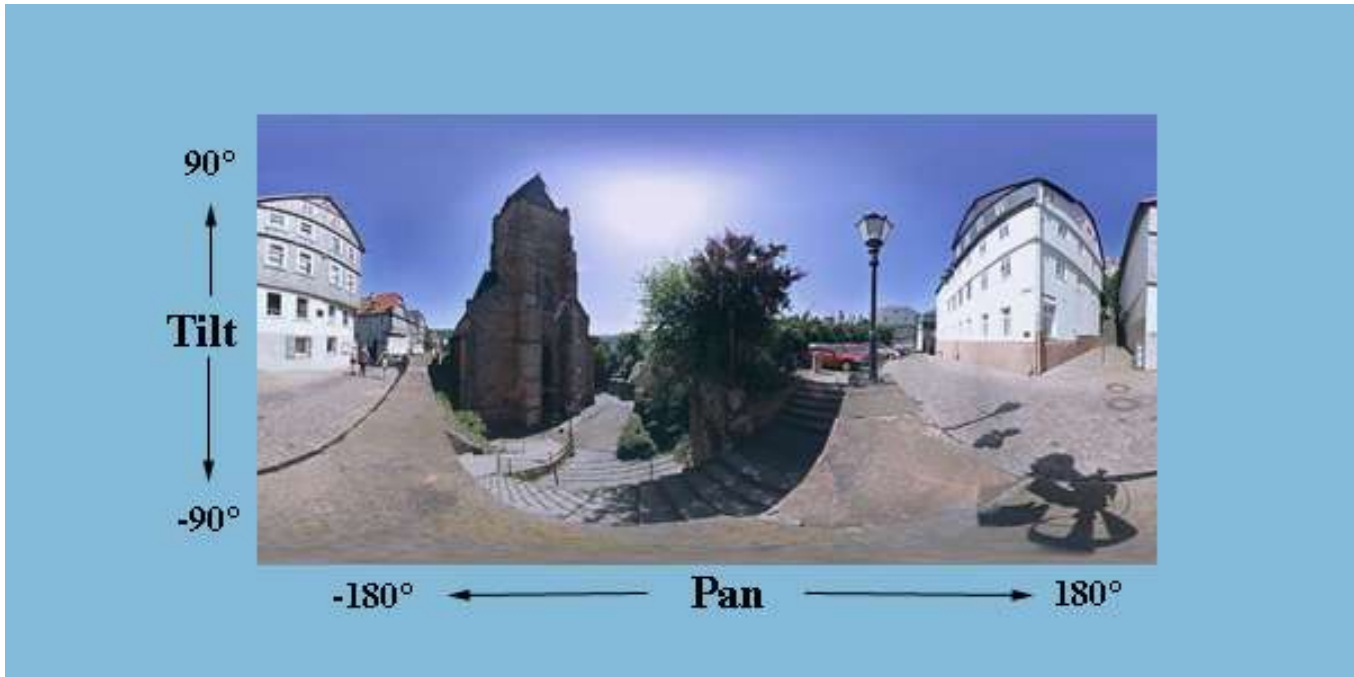
Its main features are:

- Spherical and Cylindrical Panorama Playback: Horizontal field of view: 360°, Vertical field of view 0-180°.
- Panning, tilting and zooming. Full navigation using either mouse or keyboard.
- High Quality bilinear rendering as found in better plug-in viewers, and unlike many other java viewers.
- Image and window size only limited by system memory.
- Link any document to any point in the image using hotspots.
- Configurable Controls in Appletwindow.
- Complete VR-Tours in Browsers and Appletviewer.
- Scriptable via html/javascript and internal scripting system
- Tiny file size (~20kByte) for fast download.
- Package all files into one self-displaying tour using the Jar-utility.
- Display 3D-objects, Panorama Movies and animations using auxiliary helper applets included in the distribution.
- Supports fast downloading low resolution preview panoramas and high resolution zoomable features (ROI).
- Optional protection of panoramas and images by encryption.

Download the latest version of PTViewer from [this site](#)

Images

PTViewer uses equirectangular panoramic images, which must cover 360° horizontally. Vertical coverage may be any value between 0° and 180° . In the equirectangular projection, both horizontal and vertical axis are proportional to the viewing angle. Therefore, a full spherical panorama has an aspect ratio of exactly 2:1. An example, which is taken from the Virtual tour of Marburg from this site is shown below:



The "Pan", "Tilt" and "FOV" values which you can set to override the defaults, are easily deduced from the figure above. JPEG and GIF images are supported. Any size for the source panorama can be used, however, memory requirements for image decompression may be quite large for some java engines. Suitable images can be created using various commercial packages (LivePicture, iMove) or the free PanoTools software from this site.

Cylindrical images as used in QTVR or other viewers have to be converted to the equirectangular projection, which can easily be accomplished using the free Panorama Tools from [this site](#). Use the following procedure:

Open the cylindrical image in Photoshop (or any other Graphics program with the Panorama Tools plug-in installed). The example below is such a cylindrical rendering of the scene above as might have been used for QTVR. It is converted to PSphere/equirectangular format using the remap tool with the shown settings. Notice that the image gets smaller without losing information. The savings are quite considerable in large vertical field of view images.

Cylindrical
for QTVR



Panorama Tools Remap: From QTVR; To PSphere; HFOV 360°

↓
Equirectangular
for PTVViewer



Partial panoramas with pan-range smaller than 360° can also be displayed. They also have to be converted to the equirectangular format and inserted into a full spherical image. This can be performed using the adjust function of Panorama Tools similar to the conversion above. The following example shows the conversion for a 135° cylindrical image typical for swing-lens cameras (Noblex, Widelux, Horizon). Rectilinear and fisheye images have to be converted similar. For the resultimage to work you have to set proper tilt and pan limits. The use of full 360° images as source looks like a waste of bandwidth, but it is not. The large black parts in the image are very efficiently compressed in the JPEG format used by PTVViewer.

Cylindrical
135° FOV



Panorama Tools -> Adjust (insert/ use options)
Options: Image: Panorama, HFOV=135
Panorama: PSphere, HFOV=360,
width=2000, height=500

↓
Equirectangular
for PTViewer



Image format may be JPEG (recommended for photographs) or GIF. To protect images against theft PTViewer images may be encrypted using the [PTCrypt](#) utility. There are various encryption options available, consult the PTCrypt documentation for details. Encrypted images are recognized by their extension (jpa, jpb, jpc,..) and can not be read by standard graphics programs. Using the stronger encryption features, they can't even be read by PTViewer if they are moved or copied from their original location.

Controls

PTViewer is controlled by the html-file which sets the applet parameters. As a minimum, the size of the viewer window and the name of the image file have to be set. This section of the html file may then look like:

```
<APPLET archive=ptviewer.jar code=ptviewer.class width=320  
height=200>
```

```
<PARAM name=file value="pano.JPG">
```

```
</APPLET>
```

The java code is contained in the file *ptviewer.jar*, which has to be placed in the same directory as the html-file, or you may provide a different path in the *archive* tag. Some viewers can not

read jar files, and you have to provide the uncompressed class file instead.

PTViewer may be used with these minimum controls. The file SimplePano.html in the distribution demonstrates this minimum functionality which is sufficient to view and navigate the panoramic images.

PTViewer supports many more commands, which are all set by adding more PARAM-tags inside the APPLET tag. This is a list of options:

- file - The filename of the panoramic image. Alternatively, a panorama from a list specified using the pano0/1/2 tag can be loaded using the filename 'ptviewer:Number' . Example: 'ptviewer:3' loads the panorama with list number 3.
- pwidth, pheight - Width and height of Panoramic image in pixel. By default, these parameter equal the width and height of the image specified in 'file'. Required only if regions of interest (ROI) are inserted later. See chapter below.
- roi0/1/2 - List of high resolution images to be inserted into the panorama as zoomable feature. For syntax and details see separate chapter below
- tilt - initial tilt angle (-90...90, default 0)
- pan - initial pan angle (-180...180, default 0)
- panmax - maximum pan angle (0...180, default none)
- panmin - minimum pan angle (0..-180, default none)
- fov - initial horizontal field of view (12...165, default 70)
- fovmin - minimum field of view (default 12)
- fovmax - maximum field of view (default 165)
- tiltmin - minimum tilt angle (-90 to 0, default -90 for spherical, or -vertical field of view for cylindrical panos)
- tiltmax - maximum tilt angle (90 to 0, default 90 for spherical, or vertical field of view for cylindrical panos)
- wait - Name of image (gif or jpeg) to be displayed during download. Specify path relative to html-document. This image is displayed centered in the applet window
- waittime - Minimum time to display the wait image, in Milliseconds (default 0). This is useful if an animated gif-image is used, and the animation should finish before display of the panorama.
- auto - autorotation angle (-360...360, default 0) pan angle is incremented by that amount for

each frame. Specify degrees, fractional values allowed.

- hotspot0,1,2,... - Alternative way to set hotspots, see next chapter
- shotspot0,1,2,...- Description of static hotspots, see next chapter
- maxarray - the maximum size for linear arrays on this machine (default 524288 for Netscape). See the notes about large images.
- frame - An image to be displayed in front of the panorama window. This should be a gif or jpeg image. It will be inserted into the applet window aligned to the lower right edge. See the control buttons in *Controls.html* for an example.
- mousehs - Name of a user supplied javascript function which gets called everytime the mouse enters or leaves a hotspot. See example and notes on scripting below.
- getview - Name of a user supplied javascript function which gets called everytime the view (pan, tilt or field of view) changes. See example and notes on scripting below.
- bar_x - the x-coordinate of the upper left point of the progress bar (default width/4)
- bar_y - the y-coordinate of the upper left point of the progress bar (default height*3/4)
- bar_width - the width of the progress bar (default width/2).
- bar_height - the height of the progress bar (default 10 pixels).
- barcolor - the color of the progress bar (default dark gray).
- view_width - the width of the panorama viewer in pixels. Defaults to width of applet window.
- view_height - the height of the panorama viewer in pixels. Defaults to height of applet window.
- view_x - x-coordinate of upper left corner of the panorama viewer. (default 0).
- view_y - y-coordinate of upper left corner of the panorama viewer. (default 0).
- pano0,1,2,3 -List of panorama images for built-in controls. See next chapters for details.
- bgcolor - Hexadecimal integer specifying color of background
- hsimage - Name of hotspotimage describing masks for all hotspots. This image must have identical dimensions as the panoramic image. See chapter on hotspots below.
- sound0, sound1, sound2,... - Names of sound files to be used by the applet.
- quality - determines which of the two built-in pixel-interpolators are used for rendering the

images: either the nearest-neighbor (nn, fast, low image quality) or the bilinear (bi, slow, high image quality) interpolator. The available options are 0 - always nn; 1 - nn for panning and autopanning, bi for stills; 2 - nn for panning, bi for stills and autopanning; 3 - always bi. (default 1).

- **inits** - Initialisation command which is executed after the panorama is loaded. This may be a ptviewer or javascript command, or a URL. The same syntax as in 'u' tags for hotspots should be used.
- **applet0, applet1,...** - Applets to be loaded into ptviewer. Use this feature to load Panorama Movies, Objects or other helper applets.
- **preload** - Comma separated list of image files to be downloaded in the background. This is useful for VR-tours with many panoramic images. These are instantly available, without any download delay.
- **cache** - Set to 'false' if you want to disable PTViewer's internal RAM image cache (default: true). This may be required for very large tours which don't fit into RAM.
- **cursor** - Set to **CROSSHAIR** or **MOVE** to change the default arrow cursor. This change takes place only when the cursor is inside the applet window.
- **loadAllRoi** - set to "false" to disable download of all ROI-images at initialization time (default: true).

A list of panoramas can be defined using the **pano0/1/2/3..** tag. This is required for use with the **newPanoFromList()** function. Using this feature PTViewer can be scripted without the need for a scriptable java-implementation. A panorama description consists of parameter tags similar to parameter tags in html pages. Each parameter tag is placed between braces. It consists of an identifier, followed by the equal sign ('=') followed by the value. The parameters have identical names as the above described tags, and are available except the view-related tags. An example pano tag:

```
<PARAM name=pano0 value=" {file=images/pano.jpg}
    {pan=-45}
    {tilt=-50}
    {fov=80}
    {fovmax=120}
    {fovmin=30}
    {auto=0.5}
    {hotspot0=x1141 y207 cff0000 n'Marburger Schloss'
u'Controls.html' }
    {hotspot1=x311 y265 c00ff00 n'Pfarrkirche'
u'Controls.html' } ">
```

Please note that no quotes are used to enclose the parameters to the right side of the equations.

Also, no spaces should be used on the left sides, since the simple parser used to read these parameters might get confused.

Download feedback for the panoramic image is provided in two different formats depending on whether a 'wait'-image has been set. If not, then download feedback is provided by a message "Loading Image....X% complete" with X being the proper percentage. If a waitimage is specified, a progress bar is plotted inside the image. Size, location and color of this bar can be specified using the 'bar_..' tags. Download feedback indicates 80% when the image file has been loaded, and 100% after the image has been decompressed.

Hotspots

Hotspots may be used to link other documents to specific points in the panoramic image. Hotspots are set using the 'hotspot' parameter in the APPLET tag. The active region of the hotspot may be specified in 4 different ways:

1. Default (no specifications). A square region 24x24 pixels (in the viewer!) surrounding the x/y coordinate of the hotspot.
2. Rectangular (specify a/b). A rectangular region (in the panoramic image!) limited by point x/y to the left and a/b to the right .
3. Arbitrary shape (specify mask image m for each hotspot) The white regions of the mask image (in the panoramic image).
4. Arbitrary shape (specify grayscale hotspot image himage, one for all hotspots, same size as panorama). The gray value of a pixel (0-254) determines the hotspot number it belongs to. 255 (white) means 'no hotspot'. This last method can not be mixed with the other three cases.

Parameters in hotspot description lines and in other multi-parameter tags (static hotspots, ROI, etc) consist of an identifier, which is a single character, and its value. Parameters are separated by spaces, or, if they contain spaces, may be quoted by single quotes (') or any other letter, which is defined as quoting character by preceding it with the letter \$. Example

```
<PARAM name=hotspot0 value="x1141 y207 u'pano.html' n$+Marburger 'Schloss'+ ">
```

Each hotspot description consists of 4 mandatory parameters:

- *xnumber* or *Xnumber*- the x - coordinate of the hotspotlocation. 0 is left. x - absolute pixelcoordinate (0...width), X - relative pixelcoordinate (0...100), fractional values allowed.
- *ynumber* or *Ynumber*- the y - coordinate of the hotspotlocation. 0 is top. y - absolute pixelcoordinate (0...height), Y - relative pixelcoordinate (0...100), fractional values allowed.
- *n'name'* - the name of the hotspot displayed in the status line if the mouse moves over it. Must be enclosed with single quotes (').

- *u'name'* - the URL of the linked document relative to the html-document. Must be enclosed with single quotes ('). Instead of an URL, any of the ptviewer commands can be specified using the prefix "ptviewer:".

The following are optional parameters:

- *chexnumber* - the RGB-color of the default hotspot marker. Specify as hexadecimal number RRGGBB. Default is red.
- *i'name'* - name of image to be displayed when hotspot is activated. Must be gif or jpeg, and will be shown centered at hotspot location. Instead of an image, any of the ptviewer commands can be specified. This command will be executed when the mouse enters the hotspot region. Example:

```
<PARAM name=hotspot6 value=" x776 y124 n'Lamp' i'ptviewer:PlaySound(0)' u'Controls.html' ">
```

 This hotspot plays the sound numbered 0 when the mouse enters. Hotspots executing commands are always of type popup and don't react to the showHS() command. If the 'e' parameter is specified, then 'i' specifies a text window, see below.
- *t'name'* - name of target in html-document specified with option u.
- *m'name'* - name of an image to be used as mask for shaping hotspots. Normal hotspots have square format and are 20x20 pixels large. To set other shapes, create a gif-image which is white at locations belonging to the hotspot. This image will be referenced with the upper left point being the x/y coordinate specified above. Hotspot markers appear centered in the image. Please see the streetlight example in the distribution.
- *p* - Specifying p (without parameter) makes this hotspots image pop up when the mouse moves over it, see the streetlight example.
- *q* - Specifying q (without parameter) makes this hotspot always visible. This may be used as floating logo.
- *w* - Specifying w (for 'warped', without parameter) causes the specified image (see i-tag) to be inserted into the panorama image rather than into the viewer window. As a result, the image is blended and fixed to the background. See the roof of the building to the left in the example "Controls.html" which turns green when showing the hotspots. This feature works for permanent, popup and normal hotspots. The image will be inserted centered at the x/y coordinate. Please note that x/y = 0/0 refers to top left, while some graphics programs use the convention (1/1) for this point. The image must fit into the panoramic image and may not extend across the back seam. Hotspots using warped images are by default rectangular, ie the area of the image is active region of the hotspot, unless a separate mask image is specified.
- *e* - Specifying e (for *text*, without parameter) causes the specified image parameter (i-tag) to be interpreted as text message. This message is displayed in a rectangular window, with one corner being the hotspot. This corner is marked by a dot. The text window is positioned to fit

into the viewer. Line breaks are marked by the character '|'. The c-parameter (see above) is used to specify the color of the text. Example:

```
<PARAM name=hotspot0 value=" x10 y20 n'myname' i'This is a | Textmessage' e ">
```

- *a number* or *A number* - the x-coordinate of the right edge of a rectangular hotspot. a - absolute pixelcoordinate (0...width), A - relative pixelcoordinate (0...100), fractional values allowed. If a or A is set, the active hotspot region is a rectangle with left coordinate x and right coordinate a. The hotspot marker image is shown centered. Please note, that a rectangle in the panoramic image is warped in the viewer window. If you need exact shapes, then use the m-tag for a masked hotspot.
- *b number* or *B number* - the y-coordinate of the right edge of a rectangular hotspot. b - absolute pixelcoordinate (0...height), B - relative pixelcoordinate (0...100), fractional values allowed. If b or B is set, the active hotspot region is a rectangle with left coordinate x and right coordinate a. One of b or y should be the top, the other coordinate is the bottom of the rectangle. The hotspot marker image is shown centered. Please note, that a rectangle in the panoramic image is warped in the viewer window. If you need exact shapes, then use the m-tag for a masked hotspot.

Options may be specified in any order.

These are examples for hotspot tags:

```
<PARAM name=hotspot0 value=" x1141 y207 cff0000 n'Marburger Schloss'
u'pano.html' ">
<PARAM name=hotspot1 value=" x311 y265 c00ff00 n'Pfarrkirche'
u'http://www.fh-furtwangen.de/~dersch' " >
<PARAM name=hotspot2 value=" x386 y506 c0000ff n'Stairs'
u'pano.html' ">
<PARAM name=hotspot3 value=" x209 y82 i'images/hs.gif' n'Sky'
u'pano.html' ">
<PARAM name=hotspot4 value=" x393 y386 c00f0f0 n'Entrance'
u'javascript:test()' ">
<PARAM name=hotspot5 value=" X50 Y50 cf0f0f0 n'Center'
u'panos.jpg' ">
<PARAM name=hotspot6 value=" x776 y124 cf0f0f0 n'Lamp' m'images/lmask.gif'
p i'images/popup.gif' u'pano.html' ">
```

Hotspots may be stacked. If one hotspot has identical x and y coordinates as another hotspot prior in the list, it is assumed to be identical. The mask of the prior hotspot will then be reused. If this hotspot gets activated by clicking the mouse, or moving the mouse over it, the appropriate actions of both hotspots are executed. Arbitrary many hotspots may be stacked at any position. The file 'Controls.html' of the distribution contains this example for a stacked hotspot:

```
<PARAM name=hotspot6 value=" x776 y124 n'Lamp' m'images/lmask.gif'
i'ptviewer:PlaySound(0)' u'Controls.html' ">
<PARAM name=hotspot8 value=" x776 y124 p i'images/popup.gif' ">
```

Static Hotspots

From Version 1.1. PTVIEWER supports static hotspots. These are static locations in the applet window which can be used like clickable image maps in html. The syntax to set these static hotspots is similar to the panorama hotspots. Note that all coordinates are absolute pixelcoordinates in the applet window. Static hotspots are set using the '*shotspotNumber*' parameter in the APPLET tag. Each static hotspot description must consists of 5 mandatory parameters:

- *xnumber* - the left x - coordinate of the static hotspotlocation.
- *ynumber* -the top or bottom y - coordinate of the static hotspotlocation.
- *anumber* - the right x - coordinate of the static hotspotlocation.
- *bnumber* -the bottom or top y - coordinate of the static hotspotlocation.
- *u'name'* - the URL of the linked document relative to the html-document. Must be enclosed with single quotes ('). Instead of an URL, any of the ptviewer commands can be specified using the prefix "ptviewer:".

The following are optional parameters:

- *t'name'* - name of target in html-document specified with option u.
- *i'name'* - name of image to be displayed when hotspot is activated. Must be gif or jpeg, and will be shown aligned so that top left coincides with x/y. Please note that the image is never displayed unless either of the p or q parameter is specified, or the DrawSHSImage() scripting command is executed. Instead of an image, any of the ptviewer commands can be specified. This command will be executed when the mouse enters the hotspot region. Hotspots of the command type are automatically of type popup.
- p- Specifying p (without parameter) makes this shotspots image pop up when the mouse moves over it.
- q- Specifying q (without parameter) makes this shotspots always visible.

Here's an example for static hotspots:

```
<PARAM name=shotspot0 value=" x236 y186 a250 b200
u'ptviewer:startAutoPan(0.5,0,1)' ">
<PARAM name=shotspot1 value=" x250 y186 a264 b200
u'ptviewer:stopAutoPan()' ">
<PARAM name=shotspot2 value=" x264 y186 a278 b200
u'ptviewer:startAutoPan(0,0,0.97)' ">
```

This example shows the use of the ptviewer: controls which can be used like URLs and javascript: functions. See the next chapter for details.

Static Hotspots may be stacked like normal hotspots. If one hotspot has identical x,y, a and b coordinates as another hotspot prior in the list, it is assumed to be identical. If this hotspot gets activated by clicking the mouse, or moving the mouse over it, the appropriate actions of both hotspots are executed. Arbitrary many hotspots may be stacked at any position.

Preview Panoramas, Regions of Interest, Zoomable Features

The panoramic image used in PTVIEWER may be dynamically assembled during viewing. This is accomplished by providing 'roi' tags. This tag consists of three parameters with a similar syntax as the other list tags:

- *i'name'* - name of image to be inserted (required).
- *x'number'* - horizontal pixel coordinate of insertion point (upper left point of image).
Default: 0
- *y'number'* - vertical pixel coordinate of insertion point (upper left point of image). Default: 0.

Example

```
<PARAM name=roi0 value=" i'myimage.jpg' x50 y80 ">
```

The image 'myimage.jpg' is inserted at coordinates 50/80 into the panoramic image. The image must fit into the panorama without extending to the right or bottom.

If the panoramic image used as source in PTVIEWER should have higher resolution than the image supplied in the 'file' tag, then additional parameters *pwidth* and *pheight* have to be supplied, which specify width and height. The file-image is internally scaled up to pwidth/pheight, and then the full resolution roi-images can be inserted. The following code preloads a small image (600 * 300 pixels) and later inserts a full size (1800 * 900 pixels) version. The preload is visible almost instantly:

```
<PARAM name=file value="SmallPano.jpg">
<PARAM name=pwidth value="1800">
```

```
<PARAM name=pheight value="900">
```

```
<PARAM name=roi0 value=" i'LargePano.jpg' ">
```

Preparing the scaled image "SmallPano.jpg" may be tricky: Most Graphicsprograms slightly shift images upon scaling by some fractions of a pixel size vertically and horizontally. This shows up during progressive download as annoying movement of parts of the image. Therefore, the scaling-algorithm has to be the exact inverse of that used by PTViewer, which scales relative to the center of the image. This is also the scaling algorithm used in the 'Correct'-tool of the [Panorama Tools](#) plug-in software, which is recommended for this task.

By default, all ROI images are loaded and inserted at initialization time. This can be disabled by setting the parameter loadAllRoi to "false". It is then possible to load individual roi images using the command ptviewer:loadROI(*int number*) (or the corresponding javascript version). This command can be linked to any hotspot or triggered by javascript, see the chapter on scripting below.

If no "file" parameter is set, but nevertheless the panorama width ("pwidth"), then ptviewer displays a grid, with horizontal and vertical lines spaced by 10°. This grid can then be filled progressively by ROI images to create a progressive download.

Sound and Other Media

Sound files of type au, aiff, wav and midi are supported in PTViewer 1.2. They should be specified in a list using the parameters sound0, sound1, sound2, etc. Specify the name of the sound file. They can then be played back using the ptviewer command 'PlaySound(n)' with n being the number of the sound in the list. This command can be linked to any normal and static hotspot, or linked to the popup function, or called from javascript.

Many more media formats are available if [JMF](#) is installed on the host computer. These include avi-video, linear quicktime video, flash 2, mpeg1/2/3, many soundformats etc. From version 1.7.2 of PTViewer, JMF media are loaded via the optional [PTMedia extension](#).

Configurable Controls

From Version 1.1 PTViewer can be configured to control almost any internal function from within the applet. No html-browser is required and complete VR tours can be displayed using just the appletviewer. The basis for this is an internal scripting system which works like a subset of javascript. Many PTViewer-functions can be called like URLs using the prefix "ptviewer:". They can be used as u-tag in both static and normal hotspots, see the examples for static hotspots above. This is a list of accessible functions which is similar to the corresponding list of javascript functions:

- ZoomIn() Zoom in by 3%
- ZoomOut() Zoom out by 3%
- panLeft() Pan 5° to the left
- panRight() Pan 5° to the right
- panUp() Pan 5° up
- panDown() Pan 5° down
- gotoView(*panangle, tiltangle, field-of-view*) jump to location specified by these three angles (degrees, fractional values allowed)
- showHS(); show Hotspots, displays markers or images at hotspot locations
- hideHS(); hide Hotspots
- toggleHS(); hides Hotspots if currently displayed, or displays them if currently hidden
- startAutoPan(*pan_inc, tilt_inc, zoom*) initiate autopanning. Each successive view's panangle is incremented by pan_inc, its tiltangle is incremented by tilt_inc, and its field of view is multiplied by zoom.
- stopAutoPan() Stops autopanning.
- newPanoFromList(*num*); Load Panorama numbered num into the applet. This panorama must have been previously defined using the *panonum* tag.
- newPanoFromList(*num, panangle, tiltangle, field-of-view*) Same as above, but set new view parameters.
- moveFromTo(*start-panangle, stop-panangle, start-tilt, stop-tilt, start-field-of-view, stop-field-of-view, number -of-frames*) Moves from start-location to stop-location using a specified number of frames
- moveTo(*top-panangle, stop-tilt, stop-field-of-view, number -of-frames*) Moves from current location to stop-location using a specified number of frames
- PlaySound(*num*); Play back sound No num.
- DrawSHSImage(*num*); Draw static hotspot image.
- HideSHSImage(*num*); Hide static hotspot image.
- ToggleSHSImage(*num*); Toggle visibility of static hotspot image.
- DrawHSImage(*num*); Draw hotspot image

- HideHSImage(*num*); Hide hotspot image.
- ToggleHSImage(*num*); Toggle visibility of hotspot image.
- ShowCompass(); Show Compass Image
- HideCompass(); Hide Compass Image
- waitWhilePanning(); waits until autopanning, moveTo() or moveFromTo() has finished.
- startApplet(*num*); initialises and starts applet *num* from list of applets
- stopApplet(*num*); stops and destroys applet *num* from list of applets
- loadROI(*num*); load and insert ROI image *num* from list of ROI images.

Some functions require arguments, eg. startAutoPan(). These arguments should be supplied as constants. Allowed is startAutoPan(2,3,1.2), not allowed is startAutoPan(1+0.5, tilt, zoom*1.05).

Scripting

PTViewer can be scripted by the html document. Several functions are available to control navigation, zooming, panning and jumping to specific locations. The syntax follows the usual javascript rules. All ptviewer commands from the list above are available as javascript commands also. In addition, the following commands are available only in javascript:

- isVisibleHS() returns true if hotspots are visible, else false
- pan() returns current pan angle (double)
- tilt() returns current tilt angle (double)
- fov() return curent field-of-view (double)
- getAutoPan() returns true if autopanning, else false.
- get_x() returns the mouse coordinate of the last click (mouse-up event)
- get_y() returns the mouse coordinate of the last click (mouse-up event)
- getPanoNumber() returns the list number of the current panorama
- getAppletInfo() returns a string with the copyright message and version

See the html file ditributed with the java viewer for an example. It uses an image map created by [Andrei Bodrov](#) and links some of the controls to specific buttons. You may also link to standard javascript buttons in your document as used for the 'gotoHS' links.

For a more descriptive list of all publicly accessible parameters and functions in ptviewer, see the

Loading New Panoramas from Running Applets

New Panoramas can be loaded using the `newPanoFromList()` function. This function requires the previous definition of a list of panoramas.

`newPanoFromList()` can be used from internal controls using the URL-tag '`ptviewer:newPanoFromList(num)`' in hotspots (normal and static hotspots) or using javascript from the browser. All parameters are reset with the exception of the view-parameters (width/height/offset), the wait image, the frame image, all static hotspots and sounds. Static hotspots, frame, wait image and sounds may be newly set in the panorama description, in which case the old values are discarded. See the example `Controls.html` of the distribution.

Calling Javascript functions from PTViewer

PTViewer can call user supplied javascript functions. To be able to call javascript from the applet, the parameter *mayscript* must be set. This can be done in the applet tag:

```
<APPLET name="ptviewer" archive=ptviewer.jar code=ptviewer.class width=320
height=200 mayscript=true>
```

There are three possible ways to call javascript from PTViewer:

- Linking to hotspots using the `u`-tag in the hotspot description, see examples above.
- Specifying the `mousehs` parameter. This should be a javascript function, which will be called with the number of the currently active hotspot, or -1 if there is none active. The syntax should be `mousehs(int number)`. See the example in the distribution. Specify only the name of the function without parameters, parenthesis, or leading javascript: name.
- Specifying the `getview` parameter. This javascript function will be called whenever the view (pan, tilt, field of view) has changed. It can be used to create a compass, map, directional sound etc. It will be called with the current view-parameters using the syntax `getview(double pan, double tilt, double field-of-view)`. For speed reasons it is disabled during panning and autopanning. Specify only the name of the function without parameters, parenthesis, or leading javascript: name.

My own tests have shown that interaction of javascript with java may be quite unstable and does not work on all browsers, eg not on the combinations Mac/IE. It is recommended to check for a safe combination by query of the javascript navigator object, and provide an alternative for unsafe systems.

Large Images

Some browsers have limits on the maximum size of linear arrays, eg 4MByte on Netscape, even if more system memory is available. This limits the maximum size of images some java viewers can handle, eg LivePicture's IVR-viewer can't read spherical images of more than 1500 pixels width. PTVViewer can read much larger images by internally splitting the image into chunks of smaller pieces. The size of these chunks can be set via 'maxarray' parameter tag. The default value is #80000 hexadecimal, which corresponds to roughly 500000pixels, ie 2Mbyte (4byte = 1pixel). Unfortunately, this splitting causes other Browsers (Internet Explorer) to download images slower than normal. For best performance one can set the maxarray parameter depending on the browser, see the javascript code in the example above. However, in most cases this is not needed and the default value works for any browser.

Packaging files using the Jar utility

All files belonging to a VR-scene (images, sounds, textfiles) can be packaged into one single archive using the Jar archiving tools. This is a convenient way to store, mail and present virtual tours. Only the html-file has to be kept separate. Double-clicking this html-file launches the browser, which then automatically extracts and displays the content of the file. See the separate [documentation](#) about packaging tours for details.

Authoring

From version 0.8, PTVViewer has some additional helper commands to aid authoring of documents:

- Pressing 'v' displays the current pan, tilt and field-of-view angles. This is useful for setting initial views, or specifying parameters for the gotoView() command.
- Pressing 'h' and clicking the mouse displays the current X/Y coordinates in the panoramic image. These are useful for specifying hotspot parameters. The values are relative coordinates (0...100) and have to be used with capitol X and Y in the html-document.
- Pressing 'i' displays a copyright message and version number in the status bar.
- Pressing 'u' displays the URL to the HTML document.
- Pressing 'p' displays the path to the HTML document.

PTViewer Java Application

This application uses the Java-applet 'PTViewer' and a custom AppletStub to implement a standalone cross platform spherical panorama viewer. Using the internal scripting capability of PTVViewer, full featured VR-tours can be displayed. The same html-file used for display in a

browser can be reused for the application. Using the Java archiving tool 'Jar', it is possible to pack tours into one self-displaying file. See the separate [documentation](#) describing this feature.

Using the Application:

After double-clicking the application, a file dialog is presented. Choose any panoramic image in jpeg format, eg the file pano.jpg in the distribution. Images are recognized by the extension jpg or jpeg. Alternatively, you may specify a html-file, eg the file example.html. PTVIEWER extracts the <applet>-tag and all parameters relevant for display, and shows exactly the same tour as can be seen in a browser, with the following restrictions:

- In contrast to applets, applications can't handle sounds, at least on pre-java 1.2 installations.
- Only internal scripting commands (ptviewer:...) can be used. Do not use javascript functions. However, calling other html-files in the 'u'-tag of hotspots works, and displays a new window showing the corresponding page (only if it also contains a ptviewer-tag).

The application allows the user to resize the viewer window. If the viewer-size has not been set (parameter view_width and view_height), the applet will also be resized according to the window. If the viewer-size has been set (most likely to fit the applet into a frame) then the applet does not react to window size changes. Do not resize the window while the applet loads the image.

PTVIEWER does not display a file dialog if a html-file with name 'default.html' is present in the applications directory. PTVIEWER then loads and processes this file. This can be used for self-displaying tours. PTVIEWER may also be called by other java applications. The main() method of the ptvjapp class should be called with the filename of an image or html-file as argument.

Extensions: PTOBJECT, PTMOVIES, PTFLAT, PTCOMPASS, PTMEDIA

Using the [API-documentation](#), it is possible to extend PTVIEWERS capabilities using additional applets. Five such applets are included in the distribution: PTOBJECT, PTMOVIES, PTFLAT, PTCOMPASS, PTMEDIA. See this [documentation](#) about the use of these applets, and how to build your own extensions.

License

PTVIEWER is distributed under GNU license, see the file 'Copying' in the distribution. Sources are not available yet, but will be published with a later version. You may distribute the PTVIEWER applet freely, but you have to make sure that recipients have this same right, see the license for details. Practically, this means that you have to provide at least a link to the download page of PTVIEWER.

Copyright ©2000 Helmut Dersch
der@fh-furtwangen.de