Documentation for HAniS - the HTML5 Image AnimationS webapp

First release: November, 2014
Page updated: November, 2015

Current release version 2.9 (November 25, 2015)

Welcome to the homepage for the HTML5 Animations webapp (HAniS for short). This is a re-casting of the FLAniS Flash applet (which, of course was a re-casting of the original Java applet, AniS), done in HTML5. Why? Modern mobile devices no longer support Java or Flash. In addition, this version is coded in JavaScript (no relation to "Java") which has been standardized by an independent organization, rather than a company.

If you would like to test your browser's HTML5 capabilties, please visit HTML5 Test

What does HAniS do? Like it predecessors, it is a tool you can employ on your web pages that provides the ability to animate a sequence of individual images. It also lets you use overlays and provides many options for creating "hotspots", probing data, and the like. This version is coded entirely in JavaScript and uses the HTML5 standards so is usable on multiple platforms with modern browsers.

The HAniS Model is the same as it predecessors: once the program starts, it reads *configuration* information that persists throughout the run/session (like the controls and layout). If your content (image filenames, for example) is changing with time, and you want the user to be able to "refresh" the content, then you put the dynamic information into a *file_of_filenames* which is re-read when a "refresh" is done, and usually contains the updated image filenames (and perhaps other, dynamic content). There are a few different ways of specifying the *configuration* information, but the *file_of_filenames* is a text file that must reside on the server with your image files.

This document is the detailed information. If you want to take a quick look at the examples, please click here!. If you have questions or comments, please let us know! If you find errors or things that are not clear in this document, we would also appreciate hearing from you!

Quick Links in this Document

- **Examples!!**
  - Basic animation and control layout
  - Using in-line parameters instead of config file
  - Using a JavaScript object to define the config parameters New!
  - Two animations on one page New!
  - Adding fancy controls and other things
  - Overlays and file_of_filenames
  - Animating and zooming with overlays, plus capturing the scene
  - Using in-line parameters: Animating with overlays
  - Adding frame labels and toggle buttons using bottom_controls
  - Adding more customization
  - Using hotspots to switch regions and more
  - Changing the display window size New!
  - The Extrapolator Tool
  - The Data Probe
  - Using high-resolution basemaps
  - Hotzones and Hoverzones
  - Hotzones and Hoverzones in a re-scaled window New!
  - Colorizing (enhancing) small 'base' images
  - Colorizing (enhancing) large, real-time images
  - Colorizing (enhancing) and probing with an overlay
- HTML Format and Controls parameter
- The Configuration File
- Other parameters
- Image file names
- Using Overlays
- Update details
- Join the HAniS mailing list to get update info
- Copyright Notice
- **Special note on testing your animations locally**

To run HAniS, you need these files:

1. an **HTML file** that contains the size (width and height) specification, and names the *configuration file*
2. the **configuration file** that contains the options ("parameters")
   - You may "in-line" these parameters as an alternative - see this example.
   - Or ou may now define these parameters in a JavaScript object - see this example.
3. the **minimized JavaScript** (`hanis_min.js`) file.
4. Optionally, you may also have a "file_of_filenames" and perhaps some other supporting files to enable certain features (like the ("probe"), as detailed below.

Downloading: You may pick up this ZIP archive file which contains the hanis_min.js file mentioned above, as well as the easy-to-read source code.

Here is a **skeleton HTML file**. This *skeleton* is to illustrate a) how to embed the HAniS display in your HTML, and b) how to invoke it.

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=Edge"/>
  <title>My Animation</title>
  <script type="text/javascript" src="./hanis_min.js"> </script>
  </head>

  <body style="width:800px;" onload="HAniS.setup('config.txt','handiv')">

  <div id="handiv" style="width:800px;background-color:#808080;">
  </div>

  <hr>
  <font size=-1>This webapp is Copyright© 2014, 2015 by Tom Whittaker
```

```
</body>
</html>
```

You may use the above as a *template*, since very few changes will be needed. If you copy this, then you must:

- set the `width` shown above in the <body> tag to meet your needs -- this is the values for the entire "panel" -- including your images and controls. The `height` is computed automatically.
- The `HAniS.setup()` call gets it running. There are two parameters (NOTE: use single quote marks for these, since the *onload*= already uses double-quotes):
  1. The name of the configuration file (`'config.txt'` is this example)
  2. The *id* of the `<div>` element to be used for the image window (`'handiv'` in this example).
- Set the path to the `hanis_min.js` code (shown as `./` above).

Significant differences between FlAniS and HAniS

- HAniS may be used on computers, tablets, phones, etc., as it is coded in HTML5
- The "file_of_filenames" text file may have a "fof_extension" parameter, which names a file that may contain dynamic content for hotzones and hotspots.
- Styling of the control widgets (buttons, etc) is now done using CSS-style specifications which provide more tailoring
- There is no "fader" option in HAniS (but see this HTML5 implementation if you want to use fading and/or wiping!)
- The "probe" control has been implemented using color lookup tables similar to the "enhancement" tables in AniS/FlAniS.
- The "hotspot" concept has been augmented by creating "hotzones" (defined by overlays) and "hoverzones" (defined by polygons). In all cases, the 3 main "action" options (popup, link, fof) are available.
- The "extrapolation" feature was recoded to be more "user-friendly" for linear extrapolation.
- The "speed control" has been changed from a slider to two buttons (which, of course, you can set the labels on). This was done because it was found that sliders are difficult to use on touch-screen devices.
- Only one instance of HAniS may be used on a "page"; however, see this example for a way to have more than one animation on a page using the HTML <iframe> tag.
- The "static" and "caching" approaches that evolved in FlAniS has been replaced. We now rely on the browser to cache images that do not change. Non-static overlay images are designated using the same "overlay_static" parameter. The "background" (or "base") image may be tagged as non-static using the "background_static" parameter.
- In order to allow for using higher resolution GIS-type data (terrain, roads, cities, etc), the idea of a "high_res_basemap" and "high_res_overlay" have been added, where you can specify an "alternate" image to be used when the zoom level passes a specified amount. These high_res images must have dimensions proportional to the original images.

The configuration file

This is a text file you create that contains the parameters and their values. We have tried to keep the same *keyword* values in this version. The general format is identical to previous versions:

```
keyword = value, value, value
```

Note that each parameter and its arguments MUST be on one line of text in the file! There are no "continuation" lines or marks available!

Here is an example of a very simple configuration file:

```
filenames=img0.png, img1.png, img2.png, img3.png
controls = startstop,step,zoom
```

What follows is an example of a more complex configuration file that includes styling of the control widgets and uses overlays and the file_of_filenames, etc.:

```
#This is a comment
debug=true
dwell = 500
auto_refresh = 1
file_of_filenames = fof.txt
controls = startstop,step,refresh,overlay
controls_style = padding:5px;background-color:green;
controls_tooltip = Start/Stop the animation, Step one frame, Refresh images
buttons_style=padding:5px;background:linear-gradient(white,blue);vertical-align:middle;margin-left:10px;font-family:arial;font-size:18px;padding:5px;border-radius:10px;

startstop_labels = Animate, Stop Them, 120
startstop_style=left:10px;background:linear-gradient(blue, white);
refresh_label=Refresh
step_style=width:50px;

overlay_labels=Reflectivity, Visible Satellite, Watches
overlay_transparent_amount = 100, 100, 40
overlay_labels_style=font-family:arial;color:white;font-size:20px;padding:2px;background-color:green;
overlay_tooltip = First overlay, Second overlay, Watch overlay

bottom_controls = speed,zoom, toggle
bottom_controls_style = padding:5px;background-color:pink;
bottom_controls_tooltip = Decrease or increase animation rate, click to zoom, Click on frame square to remove from animation; click again to add it back

speed_style=width:40px;
zoom_style=width:120px;
toggle_size = 20,10,20
```

Names and Definitions of Controls and Parameters

The rest of this document describes the parameter names and values needed to drive the HAniS webapp. We begin with the controls and then describe the specification of the filenames of the images.

Controls

This **`controls`** parameter will position the controls above the image window. If you would like some (or all) of the controls below the image window, use the `bottom_controls` tag, as in:

Note: You must not specify the same control in both the `controls` and the `bottom_controls` parameters!

If you want to have your controls appear on more than one horizontal line, preceed the first one on the new line with a /. For example:
`controls=startstop, step, speed, /framelabel, looprock, refresh, overlay`
would make two lines of controls, with the "framelabel" being the first one on the second line. You may have more than one split. You may also use this on the *bottom_controls*.

Here are the details about each control:

- **startstop** is a button for starting and stopping the looping. If it is not specified, the loop will run forever (if you have otherwise set the `start_looping` parameter to "true"). You may use the "rate" parameter to set the initial looping speed.
- **looprock** is a button that is used to control whether a movie loop or a rock back-and-forth mode is used. The default is loop, but may be specified as rocking using the "rocking" parameter.
- **step** creates two little buttons useful for single stepping, forward and backward. The left- and right-arrow keys on the keyboard may also be used for this purpose.

- **firstlast** creates two buttons that allow the user to immediately jump to the first or last frames in the sequence.
- **speed** two buttons that are used to decrease or increase the animation rate. If not used, a default rate of 3 frames per second is used (unless overridden by the 'rate' parameter).
- **refresh** is a button that will attempt to force a reload of the image files. This is useful for realtime data applications, where the contents of the image files may be changing. **NOTE:** *If you are using the "file_of_filenames" method of naming the image files, it is very important not to change the total number of images if you change the contents of the "file_of_filenames".* Also, see the `quiet_reload` parameter, below.
- **autorefresh** is a button that will turn the auto refresh function on and off. If you have the ability to zoom or enhnace images, along with an auto refresh, you might want to include this button (when an auto refresh cycle happens, all zooming and enhancements are cancelled). Normally the auto-refresh is enabled at the start. If you want autorefresh to be initially turned off then append the characters `/off`...for example: `autorefresh/off`.
- **toggle** provides a way for the user to toggle images on and off. If you specify this control, a series of little colored boxed will be displayed on a separate line. As each frame is looped through, the box corresponding to the image will change color. If the user (left button) clicks on a box, she will disable it (clicking again will re-enable it). If the user Shift+clicks on a box, she will cause the program to show that frame. Note: you may set the width, height, and spacing of these boxes -- see the *toggle_size* parameter below. Normally, a text line of help information appears below the toggle boxes; if you want to supress this, append the characters `/nohelp`....for example: `toggle/nohelp`. Finally, this is the only control that has a default "tool-tip" to explain the clicking functions.
- **zoom** will provide a pixel-duplication zooming ability. When the 'zoom' button is clicked, the cursor will change shape. When the user clicks with the left button that on the image, it will be zoomed one factor (x2, x3, x4...) at the cursor location. After the first zoom, the button will be relabelled "un-zoom". The user may then 'roam' the image around by *dragging* the mouse. More left-clicks will zoom in farther; ctrl+click (command + click on Macs) will zoom out one step per click. If the "un-zoom" button is clicked, the image will be restored to the original.

  See the "active_zoom" parameter (below) to enable zooming without this explicit control button. click to zoom in; ctrl+click to zoom out (Command+click on Macs).

  See the "keep_zoom" parameter (below) to make the zoomed state persist after a "refresh" is done.

- **extrap** will enable the extrapolation feature. To use this, you must also supply information about the times of each of the frames. See the parameters "times" and "extrap_times_template" for details.
- **overlay** provides one or more checkboxes that allows for images to be overlaid on the base image. Each box will be labelled according to the 'overlay_labels' parameter. You may also use "radio buttons" instead of checkboxes for some or all of these, using the 'overlay_radio' parameter. See the "overlay_labels_spacer" parameter below for adding some space around the checkboxes.
- **framelabel** provides a text field into which a label will be put for each image in a sequence of images. The text may be specified using the *frame_label* PARAM or from the *file_of_filenames* file (see below).
- **setframe** creates a slider that allows the user to "slide" between images. Normally, this control replaces other animation controls (like 'startstup'). The "setframe_label" defines the textual label for the control (see "setframe_label").

  NOTE! Presently, Internet Explorer does not support the "slider", so if you have users with IE, you should not use the *setframe* control! You will have to let them click *step* controls.

- **distance** provides an on-screen distance measurement tool consisting of a line the user draws by draggin the mouse/pointer, and a numeric value of the length of the line. Normally, you should provide a "map_scale" parameter to define the size of each pixel.
- **probe** allows the user to probe data values for overlays. Unlike AniS and FlAniS, this is done using the colorized images and does not require a separate, gray-scale image. However you must provide a look-up table (see "probe_table").
- **enhance** provides a "drop-down menu" (also known as a "select" widget) with a list of enhancement table names taken from the required *enhancement_filename* file. This control provides the means to colorize gray-scale images, and may be applied to either the "base" or one "overlay" (see "overlay_enhance") image set.
- **show** will cause the current "scene" to be captured and displayed in a separate tab or window. This will only capture the image part of a scene -- no line drawing, pop-ups, etc.

Parameters that may also be used (details for those parameters related to filenames and overlays appear after this section):

Tool-tips

Every *control* may have an optional "tool-tip" associated with it. The format is:

```
controls_tooltip = value for the tip of first control, tip for the 2nd one, etc.
```

The order of the tooltips must be identical to the order of the controls names in the *controls* parameter.

```
bottom_controls_tooltip = Click to step one frame forward/backward, Click to go to the
first or last frame,.....
```

specifies a "tooltip" for each of the `bottom_controls`. If you use the "toggle" control, there must be a place-holder "tooltip" in the list.

```
overlay_tooltip = Enable topographic display, Enable radar reflectivity display, Show the roads, Show the watches and warnings,....
```

specifies a "tooltip" for each of the `overlays`. If you use the "hidden" option on an overaly, there must be a place-holder "tooltip" in the list.

Style

Styling of the "control" widgets is generally done with CSS strings. Most on-screen styling (for example, the display of the "probe") follows the format used in FlAniS..with the exception of the "font".

Note:

- *colors* are in the form: `0xRRGGBB` or `#RRGGBB` or `#RGB` where each letter (R=red, G=green, B=blue) is a hexadecimal digit.
- *font* is a CSS-style string (e.g., `12px arial`). The size must be first!
- *shadow* specs are optional and refer to the "drop shadow" effect
- Just a few examples of potentially useful CSS strings (also, look at the *configuration file* for each example):
  - `height:20px;width:100px;`
  - `font:20px arial;`
  - `font-size:20px;`
  - `font-family:arial;`
  - `padding:5px;`
  - `background:linear-gradient(white,green);`
  - `border-radius:10px;`
  - `margin-left:10px`
  - `background-color:green;`
  - `background:URL("iconimage.png");`

Styling of the widgets (buttons, etc) can be accomplished in two ways:

1. Using the generalized `controls_style` and/or `bottom_controls_style` parameters. These are CSS strings, mostly useful for setting background colors, etc..
2. Using the `buttons_style` parameter to set styles for all control buttons. This is a CSS string.
3. Using a `_style` parameter for an individual control (e.g., `startstop_style=`). These are also CSS strings (e.g., width:100px;background-color:orange;)
4. Most on-screen displays also have their own styles:
   - `distance_display_style` the style for the "distance" control screen display:
     `distance_display_style = background, foreground, font, decimadigits [,linecolor]`
     or

distance_display_style = background, foreground, font, decimadigits [,linecolor ,shadowColor, shadowBlur, shadowX, shadowY]
- probe_display_style for the "probe" tool:
  probe_display_style = background, foreground, font [, shadowcolor, shadowblur, shadowX, shadowY]
- times_label_style for the display of the "times" when using the "extrap" control:
  times_label_style = color, utcOffset, zoneLabel, background, font [, AM/PM(='true']
- tipbox_display_style for all tooltip on-screen displays:
  tipbox_display_style = background, foreground, font [,shadowColor, shadowBlur, shadowX, shadowY]
- popup_style is the styling for popup windows when using hotspots, hotzones or hoverzones. It is a CSS string (e.g., background-color:pink;).
- popup_window_size=800,300 sets the *width* and *height* of the parent window.
- setframe_style defines the style of the *setframe* control slider
- setframe_label_style sets the style for the textual part of the *setframe* control display. This is a CSS string.
- framelabel_style specifies the CSS styling for the "framelabel" display -- usually the height and width of the box.
- checkbox_style is a CSS string for the "checkbox" used for the "overlay controls" -- this is useful only to set the size of the actual checkbox (e.g., height:20px;width:20px;) and may not be supported by all browsers!

5. divcan_style for the element which contains the image window
6. imagecan_style for the *canvas* element which contains the images and supplemental graphics (lines, messages, etc). **Use this with some caution!**

The styling specification is done using CSS. Numerous examples are provided in the Examples section.

Miscellaneous parameters

debug = true

Enables a "debug pop-up window" that will contain information about errors encountered, and other information that might be helpful in setting up your animations. The window may be moved around by dragging on the "title bar". "Pop-Ups" must be enabled in the browser. The default value for this parameter is "false".

use_progress_bar = true

By default, when images are being loaded from the server, a "Progress Bar" is displayed on the screen that shows the progress of loading the images. Set this to "false" to disable this.

window_size=800,400

Specifies that the display "window" for the images will be 800x400. The actual images will be re-scaled to this size.

enable_smoothing = t

Setting this value to "t" (true) will cause images to be rendered with "smooth" edges using whatever built-in algorithm is available for the browser. The default is "f" (false), so images will not be smoothed.

dwell = 500

specifies that the nominal, default dwell rate for each frame is 500ms. You may also specify the minimu, maximum and step for the *speed* buttons. For example:
dwell = 500, 20, 2000, 50
indicates the minimum is 20ms, maximum is 2s, and the step-per-click is 50ms.

start_looping = false

specifies that animation will not automatically start as the images are being loaded from the server. Without this option, the animation will start when the first image is received.

active_zoom = true

specifies that the zoom ability will always be active; this cannot be used in conjunction with the "zoom control". If you specify this parameter, user left-clicks on the image will zoom at that point, right-clicks will zoom out. While zoomed, the image may be roams by dragging the mouse around.

keep_zoom = true

specifies that the zoom level and position will be restored after a "refresh" is done. Without this parameter, the zoom is reset to the "un-zoomed" view.

zoom_scale = 3.4

sets the zoom scale factor to 3.4. The default is 1.0. This is the value that determines the zoom "step" for each mouse click. Note that a value of 10.0 approximates what the old AniS did.

maximum_zoom = 1.0

sets the maximum zoom factor to 1.0. This is useful if you want to prevent the user from zooming in "too far" (usually resulting in a highly pixelated display).

overlay_zoom = y,y,y,n

specifies whether an overlay will be zoomed along with other images. If the value is "y", then the overlay will be zoomed. This is the default.

If the value is "n" then the overlay will NOT be zoomed. This is useful for overlays which are legends, since the legend information will remain on the frame during zoom and roam.

high_res_zoom = 2.0, 2.5

specifies that high resolution images will be used when the user zooms higher than 1.8x and 2.5x. Note: in order to maintain the locations, the high resolution images must be exactly these ratios larger than the original images. For example, the "2.0" means that if the original image is 500x500, the high resolution one must be 1000x1000.

high_res_basemap = bgres1.png, bgres2.png

specifies that two higher resolution images are available for the "background" image. These will be used when the user zooms higher than the values specified in the "high_res_zoom" parameter. Note: this can only be specified in the file_of_filenames, and must be the same on each "frame" of the animation (for example, geographically-related images like map boundaries or roads).

high_res_overlay = 3,ovres1.png, ovres2.png

specifies that two higher resolution images are available for overlay #3 (the third one in the overlay list), and will be used when the user zooms to a higher value than specified in the "high_res_zoom" parameter. Note: this can only be specified in the file_of_filenames and must be the same on each frame of the animation (for example, roads or city names).

map_scale = 347.32

specifies the conversion from pixels to "physical units"

distance_unit = km

specifies the string to be appended to the "distance readout" when the *distance* control is used.

auto_refresh = 16

specifies that the base images will be automatically reloaded from the source every 16 minutes. If you want to be able to enable/disable automatically refreshing images, use the "autorefresh" control, described above.

hotspot = 10, 20, 50, 50, nopan/4, fof, mynewfof.txt, Click here for a new view

```
hotspot = 30, 500, 50, 50, nopan, popup, this is the pop-up message
hotspot = 110, 0, icon, lnkimg.gif, pan, link, http://www.ssec.wisc.edu/hanis
```

specifies three "hotspots" on the image display. This may only be used within a "file_of_filenames"!

The parameters are specified in one of two formats:

```
hotspot = x, y, width, height, nopan, action, value [,tooltip]
hotspot = x, y, icon, filename, nopan, action, value [,tooltip]
```

- **x,y** are the coordinates of the upper left corner of the rectangual hotspot, and the center of an icon hotspot.
- **width, height** are the width and height of a rectangle that is sensed when the user clicks on the screen
- **pan** says the hotspot will pan around when the user zooms/pans. **nopan** means the hotspot is fixed. **NOTE: *icon* hotspots will not expand as the scene is zoomed; the icon will always be presented in its original size. Non-icon (rectangular) will zoom to a larger size and the scene is zoomed.** However, see "window_size" which may cause the icon rendering to be changed.

  Also, if you append "/4" this means this is only active when overlay #4 is enabled; otherwise, it is always active.

- **action** may be:
  - "fof" (wihtout quotes) to load a new file_of_filenames)
  - "link" (without quotes) to load a new URL.
  - "popup" (without quotes) to pop-up a text window with the text given

    Note that the default popup message box location is near the mouse-click location, and the box will following any panning the user does. To change this, use this form of the "popup" action key:
    ```
    popup[x/y/width/height]
    ```
    where "x,y" are the screen coordinates (regardless of pan/zoom!) and "w,h" are the width and height of the window. For example:
    ```
    hotspot = 100, 150, 10, 10, nopan, popup[30/50/100/150], Message text, Tool-tip text
    ```
    if x or y are a signed value (e.g., +10), then this will be an offset from the location of the mouse click rather than an absolute location.

- **value** depends on the "action" chosen....it will either be:
  - a new file_of_filenames to be loaded
  - a URL to pass back to the HTML
  - a message to display in a text box. You may use HTML tags within the text -- just no commas!
- optionally, a "tooltip" may be specified and will be displayed when the user hovers the mouse pointer over the hotspot.

NOTE -- sensing a hotspot click action takes priority over zooming. If the user puts the mouse pointer on a hotspot and clicks, only the hotspot action will happen.

NOTE -- see "file_of_filenames", below, for additional information.

```
hotzone = 6, 0xff00ff, fof, mystuff/fof3.txt
hotzone = 6, 0xffff00, link, https://weather.gov
hotzone = 6, 0x00ffff, popup, This is the popup message...
```

specifies that overlay #6 should be treated as a image with "hot zones". Usually, this image would be mostly transparent, with the "hot zones" having an opaque color. The colors specified above should exactlymatch the color of the pixels in the "hot zone". If you want semi-transparency, you must use the `overlay_transparent_amount` parameter (described below).

The action (fof, link, popup) are identical to the "hot spots", noted above.

NOTE: this can only be used in a *file_of_filenames*!!

```
hoverzone = 0xfe00fe, fof, temp/fof.txt, Click me, (100, 200, 200, 200, 200,100, 100,100, 100,200)
```

The polygon defined by the "x,y" coordinate pairs in the last parameter (inside the parentheses) using the color "#fe00fe" will appear whenever the user moves the pointer over the polygon region, and zooming is not enabled. If the user clicks in the region, a new *file_of_filenames* will be loaded from the file `temp/fof.txt`. Other "actions" are "link" and "popup" as defined above.

NOTE: this can only be used in a *file_of_filenames*!!

```
overlay_allow_hoverzones = y,y,n,y,n
```

This parameter is used to disable the display of hoverzones when the designated overlays are enabled. The parameter values correspond to the "overlay number", and must include place-holders for any hidden overlays. The default for all overlays is "y" (meaning that hoverzones will always be displayed and will take precedence over any hotzone defined by an overlay.

```
pause = 2000
```

pause on the last frame of a loop the additional number of milliseconds given (2 seconds in this example). The default is zero.

```
show_prompt = Right-click to save or copy the image
 (to save in some browsers, you may need to first 'Open in New Tab')
```

Will show the text (using HTML markup) at the top of the image displayed when the *show* control is used, instead of the default. Presently, some browsers will require an extra step, as stated above, if the user wants to "Save As" the image. (Note the text above is the default....)

```
enhance_filename = myenhtables.txt
```

The named file contains one or more "enhancement tables" that define how to colorize gray scale images. Click here for more details. This parameter may also be named "enhance_table". When the *enhance* control is used, this parameter is **required**.

```
overlay_enhance = 3
```

The overlay number 3 (the first overlay is #1) will be used with the *enhance* tool to provide colorized views of this gray-scale image. If this parameter is not specified, then the enhancements will be applied to the background ("base") images instead. See "enhance_filename".

```
overlay_transparent_amount = 100, 40, 20
```

The non-transparent pixels in overlays should be set to the opacity percentage indicated. In this case, the first overlay (see overlay_labels) is 100 opaque, the second is 40% and the third is 20%. 0 = totally transparent (you will not see anything! 100 = totally opaque (what would normally happen).

```
overlay_preserve=0,0,639,49, 610,0,639,479
```

specifies that when images are zoomed, the two rectangular areas specified (top 50 lines, right-most 30 pixels) should be preserved from the original overlay image. This is useful for showing labels and logos that are part of the original images. There may be one or more such rectangles; the number of values must be a multiple of 4.

The values of the coordinates are ordered: x_upper_left,y_upper_left, x_lower_right, y_lower_right. The (0,0) point for images is in the upper_left corner. The 'x' coordinate is horizontal, the 'y'

coordinate is vertical.

You must also provide a list of which overlays this overlay_preserve should be applied to (see below), or no preserve will be done.

You may preserve regions in the **background image** by using the parameter: `image_preserve=0,0,639,49, 610,0,639,479`

`overlay_preserve_list=false, false, true, false, true, false`

Specifies which overlays the "overlay_preserve" regions should be applied to. You may use "t" and "f" for the values as well as "true" and "false".

`hide_bottom = 24`

The bottom 24 lines of a zoomed image will be hidden when the image is zoomed/roamed. This is usually used with the *image_preserver* parameter to prevent annotation lines on the bottom of an image from appearing in the zoomed and roamed display, when these annotation lines have been "preserved".

`hide_top = 31`

The top 31 lines of a zoomed image will be hidden when the image is zoomed/roamed. This is usually used with the *image_preserver* parameter to prevent annotation lines on the top of an image from appearing in the zoomed and roamed display, when these annotation lines have been "preserved".

`overlay_spacer = 10, 20, 1, 19`

Add an extra 10 pixels to the spacing to the left of the first overlay checkbox, 20 pixels between the 1st and 2nd, just one pixel between the 2nd and 3rd, and 19 pixels between the 3rd and 4th. Values given are treated as a CSS style "margin-left:".

Styling labels

Each of the following parameters allows you to specify alternate labels for many of the *controls*. For those controls having more than one label, you may also specify a 3rd parameter which is the width (in pixels) of the button -- this can be helpful to avoid any automatic resizing when toggling button states.

For the "...labels_colors" parameters, the first three values define the colors for the "active" state of the button and are the a) text, b) top portion of the button, and c) bottom portion of the button (the colors are automatically scaled between these). The optional second set of 3 colors define the "in-active" state colors; the default is off-white.

`times = 1200,1300,1400,1500`

Specify the times (hours and minutes, HHMM format) for each of the images for use only with the *extrap* feature. There must be one item for each frame.

`extrap_times_template=(\d\d\d\d)\d\d\.jpg`

Specifies that the 'times' required for the *extrap* feature should be picked up from the filenames specified in the *file_of_filenames*. There must be one specification of:
`(\d\d\d\d)`
in the template, as this matches 4 digits and the parentheses indicate the value is the time (HHMM format). In this case, the pattern will match the first occurance on a line in the f_o_f that has 6 digits, followed by ".jpg". So if a line looks like this:
`rad030000.jpg`
then the time for this image would be picked up as "0300".

If you imbed the times in the file_of_filenames using the "FlAniS method" of enclosing the time in curly brackets (like: `backimage.png {1330} overlay=....`), then you must now use this:
`extrap_times_template={(\d\d\d\d)}`

`times_label_style = 0xffff00, -5, CDT, 0x222, 14px arial, true`

Provide for altering the values displayed on the "extrap" display. The parameters are:
`color, UTC offset, timezone label, background_color, font, AM/PM (=true)` The optional "UTC offset" may be a decimal value (e.g., -5.5), and the "timezone label" is optional. Both foreground and background colors are in the form: 0xRRGGBB (RR=red, GG=green, BB=blue), although you may also use color names (e.g., red, green, etc).. Finally, the 'true' indicates that times should be labeled with "AM" or "PM" (12-hour clock), instead of a 24-hour format.

`extrapPrompts = Click on target's initial position,Click on target's final position,Move pointer around or click here to select target`

The *extrap* function has three states. This parameter allows you to change the prompting text for these. You must supply all 3 prompts.

`to_from_lock = false`

Specifies that the extrapolation feature will show times along a line from the pointer position. A value of *true* would reverse this.

`probe_table = probevalues.txt`

Names the text file that contains the look-up values for the probe. See the example for details on the format of this file.

`probe_undefined = Missing, 2`

Specifies that the displayed value for pixels whose RGB values cannot be found in the *table* will say "Missing". The optional value of "2" says that if a match cannot be made in the table, see if any table entries have an RGB value within 2 counts of the pixel value; if so, show that value.

`overlay_probe_table = 2, 4, 0, 3`

Specifies that the first overlay will use Table #1, the 2nd overlay will use Table #4, the 3rd overlay is not to be probed, and the 4th overlay will use Table #3.

`overlay_order = 2, 3, 4, 1`

Defines the "stacking order" for overlays in the display. Without this parameter, overlays are stacked in the order given in the *overlay_labels* parameter, with the first overlay being on the bottom and the last one at the top. By using the *overlay_order* parameter, you change the ordering. Here, the 4th overlay will be on the bottom and the 3rd will be on top. The values are the stacking order (1 at the bottom). The list is in order of the overlays (so the first item on the list is the stacking order for the first overlay).

`overlay_clear = n, z, s`

Specifies that when a new *file_of_filenames* is loaded, whether the visible state of the overlay shall be set to "off". Values are: "n" = never, "s" = if a hotspot, 'z' if a hotzone or hoverzone.

`overlay_labels_color = white, red, white, white, green, #f30`

specifies the color of the text for the *overlay_labels* on the checkboxes. There must be one specification for each declared overlay, even if one is "hidden" or "always". The colors may be a name, or the form "#RRGGBB" or "#RGB".

`probe_display_style = 0x101010, 0xf0f0f0, 14, 1, mm/h, Rate=`

Define the color of the background (0x101010), and the foreground (0xf0f0f0) for the probe readout. The font size should be 14. The last 3 parameters are optional since they may be defined in the associated probe table: number of decimal places to show (2), the units (mm/h), and a label prefix (Rate=). As shown, the resulting probe readout would be of the form: Rate=23.58 mm/h

`setframe_label = Image Frame Number *`

set the template to use for labelling the "setframe" control slider/scrollbar. The given string of characters must contain an asterisk ("*") character; the actual frame number will replace this when displayed.

frame_labels = label for 1, label for 2, ...

Specify the labels to use for the framelable control. There must be one label for each "frame".

image_base = http://www.my.host/mydir/

this directs that the base URL of the image files (and the file_of_filenames, if used) should be the value given. The default URL is the directory containing the HTML. Note that the hostname in the specified URL must be the same as the hostname where the webapp is read from.

Note that if this is a directory reference, it **MUST** end with a slash character (that is: /).

toggle_size = 5,10,3

this sets the toggle control's little boxes width to 5, the height to 10, and the spacing between them to 3. This can be very helpful if you have a large number of frames and don't want to make your webapp width too large just to accomodate these boxes.

toggle_colors = blue, red, orange

this sets the colors of the 3 states for toggle control's little boxes. The defaults are blue, red, orange...as shown, for "on", "off", "frame showing".

Setting non-default button and widget labels

Most also (optionally) end with the value of the "width" of the control button:

```
startstop_labels = Start, Stop, 20
startstop_style=font-weight:bold;position:absolute;top:20px;left:100px;background:linear-gradient(blue, white);

looprock_labels = Loop, Rock, 20
looprock_style=background-color:orange;

autorefresh_labels = Disable Auto Refresh, Enable Auto Refresh, 30
refresh_label = Refresh

zoom_labels = Zoom, Un-zoom, 30
zoom_style=background-color:green;

location_labels = Show Location, Hide Location, 100
location_style=background-color:green;

extrap_labels = Extrap, Normal, 40
extrap_labels__style=background-color:green;

probe_label = Readout, 25
looprock_style=background-color:green;

step_labels = Backward one, Forward one
step_style = height:25px;width:25px;

firstlast_labels = First, Last, 30
firstlast_style=background:linear-gradient(red,blue);

speed_labels = Slower, Faster
speed_style=height:30px;width:50px

show_labels = Capture Image, Capture Image
show_style=background-color:orange;
```

Image file names

All the image files can be in GIF, JPEG or PNG formats. The files identified using the parameter names described in this section should fill the desired window. If used, *overlays* should be the same size and have the same geometry as these "background" images.

For the background images, the image files may be specified in one of three, mutually exclusive, ways.

1. Using a "root" name, to which successive numbers are appended to create the actual filenames:

   ```
   basename = file
   num_frames = 4
   base_starting_number = 0
   ```

   In this case, the root name is *file* and since there are 4 images specified with the numbering starting at 0, the actual filenames must be: *file0, file1, file2, and file3*. Please note that the default *base_starting_number* is zero, and so it would not have to specified in this case...

   Also, please see the **Note** below about this form and *wildcards*

2. Using the filenames themselvels:

   ```
   filenames = file0, file1, file2, file3
   ```

3. Using a file which contains a list of the image filenames (one per record)

   ```
   file_of_filenames = file-containing-filenames
   ```

   where the file "file-containing-filenames" contains lines of text that are in the form:

   ```
   file0
   file1
   file2
   file3
   ```

   Lines beginning with # are ignored, so you may put comments into your file_of_filenames.

   NOTE: If you want to specify an optional *frame label* (see the *controls* section above), you may put the value with quote marks after the filenames. (The label may, optionally, contain a date-time to be reformatted for the user's timezone -- see "frame_label", above.) For example:

   ```
    file0 "label one"
    file1 "label two"
   ```

   NOTE: if you are using overlays, then you may put them into the file_of_filenames as well. Please see the section, below, on overlays.

NOTE: if you would like to re-define hotspots in the file_of_filenames, use any of the forms specified in the definition of the parameter "hotspot" (above). Some examples:

```
hotspot=x,y,w,h,pan,fof,mpx_ncr20090405.txt, Click here to load new data
hotspot=x,y,icon,himage3.png,pan,popup,This is the popup text and may be  in bold
hotspot=x,y,icon,hotimage1.gif,nopan,link,http://www.ssec.wisc.edu,Click here to open a link to the SSEC homepage
....
```

You may also re-define the "map_scale" parameter within the file_of_filenames.

NOTE: For the first form (specifying a "basename"), you may also use a *wildcard* format. There are two forms of this, one using a single ***** (to substitute numbers 0,1,2...,10,11,.. at that point) and the second using one or more **?** (to substitute the numbers with leading zeroes for all the ? marks). For example, if you say:

```
basename = file*.gif
```

the actual filenames must be: *file0.gif, file1.gif, file2.gif, and file3.gif*

You may also use the form:

```
basename = file????.gif
```

the actual filenames in this case must be: *file0000.gif, file0001.gif, file0002.gif, and file0003.gif*

Note that in both these cases,

- you must use the `num_frames` parameter to defined the total number of frames
- you may also use the *"base_starting_number"* parameter to modify the starting value (which otherwise defaults to zero).

NOTE: For background images that are changing on the server, but have the same filename, you may specify the parameter

```
background_static=n
```

to force the image(s) to be reloaded when a refresh is done. Otherwise, whatever caching may be used by the browser will likely be employed.

Overlays

You may also use overlays -- one or more images that are (optionally) displayed on top of the base image, usually in such a way that part of the base image shows through. For example, a map or plotted data may be an overlay. In order to let some of the base (background) image show through, you must designate one color level in the overlay images as "transparent" and then specify that value using the *transparency* parameter (above). You may also specify the opacity/transparency of the non-transparent portions of overlay images -- see the *overlay_transparent_amount* parameter (above).

You may also specify the color of the labels for each overlay using the *overlay_labels_colors* parameter (above).

In addition, you need to specify the *overlay* control, and provide a few more parameters:

- **Define the labels** for the checkboxes and/or radio buttons:

  ```
  overlay_labels = label1, label2, label3...
  ```

  This specifies the labels that will be used on the controls for each overlay. The ordering of the values should be the same as the filenames (below). You'll want to keep these labels brief!

- **Specify the initial state** of each overlay. By default, all overlays are initially "off". If you want to force one to be on, append */on* to the label. From the previous example:

  ```
  overlay_labels = label1, label2/on, label3...
  ```

  would force the second overlay to be initially turned on.

- An overlay with **no control button**. If the overlay is to be "on" all the time (and thus, no checkbox or radio button will appear for it), append */always* to the label. For example:

  ```
  overlay_labels = label1, label2/always, label3...
  ```

- **Hidden overlays**. If the overlay is to be linked (see "overlay_links", below) and should have no checkbox or radio button for it), use */hidden*. This is only useful when `linked` to an "owner" and will follow its state. (the "hidden" item should have the *negative* link value, and thus is the "target"). (Please note that if you are specifying the colors for the overlay labels (see "overlay_labels_colors") you must specify a color even for a hidden overlay!

- Using **"radio buttons"**. If you want to treat some or all of the overlays as 'radio buttons' (that is, only one of them may be showing at once), you may use the *overlay_radio* parameter. For example:

  ```
  overlay_radio = true,false,true,true,...
  ```

  This would specify that overlays 1, 3, and 4 would be 'radio buttons' and only one could appear (be "ON") at once. Overlay #2, however, may be toggled on and off indepdently. If you want to use radio buttons for all of your overlays, you may simply say `<param name="overlay_radio" value="true">`

  If you have more than one "group" of buttons, then you may logically group them using a numeric index in this form:

  ```
  overlay_radio = true/1, false, true/2, false/3, false/3, true/1, true/2
  ```

  So the first and 6th buttons are together, and the 3rd and 7th are in a separate group of "radio buttons". The 4th and 5th are "checkboxes" that are also grouped, so only one (or none) will be "on" at once.

- **Layout** of checkboxes and radio buttons. Normally, the checkboxes and radio buttons will be laid out in a single row; unfortunately, the layout manager will simply not show any that would appear beyond the WIDTH. You may specify that the checkboxes and/or radio buttons appear on two rows in the GUI. If you preceed the label name with a "/" then this will start a second row. For example:

  ```
  overlay_labels = label1, label2/on, label3, /label4, label5/on...
  ```

  specifies that the checkboxes for #1, #2 and #3 will appear on one row, while the checkboxes for #4, #5, etc., will appear on a second row. Using this option will increase the `height` required for the webapp!

  You may force the checkboxes to be on the same line ("div") as the other widgets using this parameter:

  ```
  overlay_nonewdiv=t
  ```

  meaning "no new div".

- **Linking overlays together**. It is also possible to link overlays together so that when the user clicks one on, more than one is activated. For example:

```
overlay_links  = 0,1,0,-1,-2,0,2
```

The values of zero ("0") indicate no linking. Positive values are the "owner" and the corresponding negative value is the "target". When an "owner" is turned on, the corresponding "target" is also turned on. When the "owner" is turned off, the "target" will not change, unless it was a "/hidden" overlay (see above). Each "owner" may have more than one "target", and vice-versa.

- **Static, unchanging overlay images.** If you have some overlays that are "static" and do not need to be reloaded from the server when the user clicks "Refresh", you can used the *overlay_static* parameter to designate which one(s) are static (and which ones are not static, and should be reloaded).

```
overlay_static = y,y,n,n,y
```

Indicates the first, second and fifth overlays do not change and need not be reloaded when a "refresh" is done. The third and fourth will be reloaded when the user so requests (or the automatic timer goes off). You may also use the *overlay_caching* parameter -- it has the same meaning.

- **To zoom, or not to zoom.** It is often times useful for some overlays not to be zoomed with the rest of the image(s) -- for example, legends. To keep a legend overlay on the image while the rest are zoomed, use the *overlay_zoom* parameter.

```
overlay_zoom  =y,y,y,n
```

Indicates that the first 3 overlays will be zoomed along with the base or background image. The 4th overlay, however, will not be zoomed along with them.

- **More parameters for overlays!!**. See information above about the `overaly_transparent_amount` and also `overaly_order`, above.

Overlay file names

Note: If you are using the *file_of_filenames* for the image files, you must use that method for specifying the overlay filenames as well! (See below...)

1. If you are not using the *file_of_filenames* for the background image file(s), you may use the *overlay_filenames* parameter. The grouping of names -- commas separate the overlays and ampersands separate frames for each overlay.

```
overlay_filenames = ofileA0 & ofileA1 & ofileA2 & ofileA3, ofileB0 & ofileB1 & ofileB2 & ofileB3, ofileC0 & ofileC1 & ofileC2 & ofileC3
```

where the digits refer to a "frame number" and the letters (A,B,C) refer to an "overlay number".

2. You may also use a text file (the "file_of_filenames" form) to specify all the filenames. (As mentioned, this form is required if you are also using the file_of_filenames for the background/base images). To use overlays with this form, the filenames for all the overlays for one frame appear on one line.

```
file0 overlay=ofileA0, ofileB0, ofileC0
file1 overlay=ofileA1, ofileB1, ofileC1
file2 overlay=ofileA2, ofileB2, ofileC2
file3 overlay=ofileA3,ofileB3, ofileC3
```

NOTE: You MUST supply all overlay file names for each frame, even if the filename is repeated!!

NOTE: If you want to specify an optional *frame label* (see the *controls* section above), you may put the value with quote marks between the filenames and the keyword that follows. For example:

```
file0 "label one" overlay=ofileA0, ofileB0, ofileC0
file1 "label two" overlay=ofileA1, ofileB1, ofileC1
```

Software Updates

Since the first release version, here is a summary of changes:

- Version 2.9
  - Fixed issue of disabling the first or last frames when in "rock" mode that caused a hang.
- Version 2.8
  - Added the *show* control to allow the image portion of the scene to be captured and displayed in a separate tab or window. It can then be saved, copied, etc. Any "drawing" (lines, popups, etc) on the scene will not be captured. See example 4.
  - Added a new parameter, *show_prompt* to allow you to specify an alternative to the default prompt message that is displayed with the 'show' image.
- Version 2.7
  - Added the *start_looping* option to allow for the animation to not start as the images are loaded. Also, try to start initially on the first frame (previously, the 2nd frame was the first one shown).
  - Updated the documentation about "tooltips" for the controls to make it clear there must be a tooltip for each and every control (including "overlay" and "framelabel"). Example 6 now illustrates this.
  - Changed the code that handles parameter values of "true/false" and "yes/no" to use just the first letter of the value regardless of case.
- Version 2.6
  - Added the ability to define a "group" for the checkboxes for overlays, using the *overlay_radio* parameter. This causes group(s) of checkboxes to behave like "radio buttons", except that you can de-select all of them. (Radio buttons always have one selected.). Example:
    `overlay_radio = true/1, false, false/2, true/1, false, false/2 ,false/2`
  - Added new parameter in the config file, *image_only_base* that defines the "prefix" for the path to only the images. The "image_base" parameter, when used in the config file, is used for both the image filenames and the name of the "file_of_filenames".
  - Changed the code so that the documentation of the external call `HAniS.newFOF(path)` agrees. The "path" is never prefixed with the "image_base" parameter value.
  - Added new parameter: *toggle_colors* to allow for the colors of the toggle boxes to be changed.
  - Allow for a JavaScript object to be used to define the parameters in the HAniS.setup(), in addition to either a "config file" or an in-line string of characters. For example:

```
var myconfig = {
  dwell:"500",
  auto_refresh : "1",
  file_of_filenames : "fof11.txt",
  controls : "startstop,step,refresh,overlay",
  controls_style : "padding:5px;background-color:green;",
  controls_tooltip : "Start/Stop the animation, Step one frame, Refresh images"
};
```

  - Fixed an issue with framelabels not responding to a "firstlast" control properly.
  - Fixed (in 2.51) an issue with not being able to click on the "toggle boxes" when using the "window" parameter, introduced in verison 2.5.
- Version 2.5
  - The *window_size* parameter has been added. This allows you to specify the dimensions of the image display window, regardless of the dimensions of the images. The image display will be scaled to the *window_size*.
  - The *pan* option has been implemented for *hotspots*. This includes the icon and non-icon hotspots,
  - A new parameter, *enable_smoothing* has been added. The values are either "t" (true) or "f" (false, which is the default). When enabled, this will smooth out the edges of images, especially noticeable when zoomed.
  - A new parameter, *reverse_order* that stipulated the image names generated by using the "wildcard" form of the "basename"/"base_starting_number"/"num_frames" should be

ordered in descending rather than ascending order.
    - A new parameter, *imagecan_style* that allows you to set some CSS styling on the canvas element that is used to render the images.
    - A new parameter, *probe_undefined* allows you to specify the text to show when the probed value is not in the associated enhancement table. The format of this parameter is:
      `probe_undefined = text [,cutoff]`

      If the optional "cutoff" value is given, then as a last resort, HAniS will search the enhancement table to find an entry with an RGB value within the cutoff value of the probed pixel.

    - In addition, one bug was fixed: if you used *image_base* along with a *file_of_filenames* and the *Refresh* control, the first time a refresh is done, the path to the *file_of_filenames* was getting corrupted.
- Version 2.4
    - Changed the logic for the "Start/Stop" button so that when a user tries to stop the animation during loading, the button labels will reflect the state correctly.
    - Fixed a typo in the parsing of the
      `dwell = def, min, max, step`
      parameter which caused the min to get set equal to the max.
    - Added a new example showing how to put more than one animation on a page.
    - Updated the documentation (this page) to clarify (I hope) the meaning of the values for "static=" settings.
- Version 2.3
    - Fixed an issue that caused an internal error when a *hotzone* pointed to a "hidden" *overlay*, that had been linked to a visible (active) overlay checkbox.
- Version 2.2
    - Added the *overlay_nonewdiv* parameter to force the overlay checkboxes into the same line ("div") as the controls.
    - Reworked the logic for the vertical centering of the overlay checkboxes and associated labels to be more consistent.
- Version 2.1
    - Finished implementation of allowing for control widgets to be split between 2 or more lines, using the "/" prefix on the control name..
    - Allow *bottom_controls* to also be split onto more than one line.
- Version 2.0
    - Fix one issue with enhancements not resetting when data reloaded
    - Integrate *pinch-zoom* for touch devices. This two-fingered action will zoom the entire page, not the HAniS image (which is zoomed and roamed using the *zoom control*). During the 'pinch', users may 'roam' the whole page by moving both fingers together. A 'single finger motion' will be interpreted by HAniS and not the browser.
- Version 1.91
    - Fix issue with "probe_display_style" not picking up the "font" correctly.
    - Allow the "/on" and "/always" options on the *overlay_labels* to be used with the *enhance* control.
- Version 1.9
    - Added ---- parameter to define the minimum, maximum and increment of speed changes for the *speed* control.
    - Implemented the *enhance* control to allow for gray-scale images to be "colorized" based on a pre-defined table
    - Added the *enhance_table* (which is also known as the *enhance_filename*) that specifies the filename of the enhancement table(s)
    - Added the *enhance_overlay* parameter that specifies which overlay will be used for enhancements. If this parameter is not specified along with the *enhance* control, then the "background" (or "base") images will be enhanced.
    - Enabled the *probe* to be used with the enhancement function for one overlay.
- Version 1.8
    - Changed the reading of the *config* file to use the anti-caching strategy that is also used for the *file_of_filenames* and *image* files (when the "_static" option is set).
    - Added the *hide_top* parameter to complement the existing "hide_bottom" parameter.
- Version 1.7
    - Enabled the *overlay_filenames* parameter which allows overlay filenames to be specified without the use of the "file_of_filenames".
    - Added the *background_static* parameter to force background ("base") images to be treated as non-static. This is particularly useful when not using a dynamic "file_of_filenames" parameter.
    - Refactored a bit of code dealing with the "non-static" images to make it more efficient.
    - Fixed the *autorefresh* control -- it was not enabling the toggling of the labels and not toggling the auto refresh state correctly. **Note:** the FlAniS "autotoggle" control name has been removed -- you must use "autorefresh" for this control in HAniS.
- Version 1.6
    - Allow for more than one "preserve zone" in the background image using the "image_preserve" parameter
    - Fix CSS default issue on "z-order" of the image and drawing frames.
- Version 1.5
    - Added the ability to use "in-line" parameters instead of a config file
- Version 1.4
    - Added the "popup_window_size" parameter that allows setting of the width and height of the parent window used for popup text display.
- Version 1.3
    - Corrected a problem with the coordinate of non-icon hotspots
    - Do not allow hotspots located in the upper-right corner of the image display to be active when the *extrap* mode is active.
- Version 1.2
    - Added the "overlay_allow_hoverzones" parameter to allow overlays to switch off the use of hoverzones
    - Reworked the logic for the "overlay_order" and clarified the documentation.
- Version 1.1
    - Corrected documentation for "times_label_style" parameter.
    - Added documentation for "extrap_prompts" parameter.
    - When "extrap" feature is enabled, disable "hotzones" to avoid "clicking conflicts"
    - Display "12" for the hour when the "AM/PM" mode is set and the hour is "00".
- Version 1.0
    - Initial release

Join the HAniS mailing list

If you would like to get notified of updates to HAniS, send an message using this form. and we'll get you signed up. Note this is the same list that is used for AniS and FlAniS as well -- it's just one big, happy family...

Copyright Notice

hold us harmless from any consequences related to the use of this software.

That's it.
