Technical Documents

The free web system "Simplicity" as implemented on IAC.es

By
Lars Holm Nielsen, Holm Nielsen IT
Lars Lindberg Christensen, ESA/ESO/ST-ECF

http://www.spacetelescope.org/projects/web

Last modified: 1 September 2006

“Simplicity” on IAC.es

Table of Contents

1 Introduction

Simplicity is a web system originally designed for the production and maintenance of popular scientific websites that contain large quantities of data (images, videos, posters, wallpapers and the like) and metadata (image captions, descriptions, sizes, priorities etc.). However the system may have a much broader application. The system is capable of displaying dynamic indices of images/videos for different kinds of search queries (eg, it is possible to search for wallpapers, different categories etc.). Each index shows a number of images/videos that each link to a static page with information about the given image/video and different display sizes and formats .

2 Prerequisites/Readers' guide

Some preliminary advice and information:

- A basic working knowledge of Microsoft Excel and websites is desirable to make the fullest use of the user guide documentation.
- A good knowledge of web programming with Perl and UNIX is desirable to make the fullest use of the technical documentation.
- Names of parameters etc. are typed in **Bold**, pieces of code with `Courier New` and commands or file paths with *Italics*.

3 User Guide

This chapter targets content authors who maintain image and video archives. Further information about the customisation of image and video archives is located in the technical documentation (See Section 4). Knowledge of the features of image and video archives is a prerequisite.

3.1 Overview

To publish new images and videos to the archives, follow the steps outlined below:

1. Prepare image/video files in different formats from one master file.
2. Edit the metadata for the new image/video.
3. Publish changes to the webserver and trigger generation of static pages.

The image files are prepared by a fully automated process that consists of using a Photoshop CS2 script to generate the different display formats and put the files in the right directories. In contrast, video files are prepared using a manual process. Each image and video must be accompanied by some metadata such as a title and category that describes the image. The metadata is stored in CSV-files that can be edited with Microsoft Excel or a similar program.

When the image/video files have been prepared and the metadata entered into the CSV-files, all the new files must be uploaded to the webserver and a static page that shows the image/video and any possible downloadable formats must be generated. The static page generation is either carried out automatically within 5 minutes or can be triggered immediately via an administration interface.

3.1.1 Directory structure

The only prerequisite for using the Simplicity Web System is knowledge of the basic directory structure of the system.

- */bin/images.pl* and */bin/videos.pl* generate the dynamic indices.
- */images/html* and */videos/html* contain all the static pages for both Spanish and English and are generated automatically.
- */images* and */videos* contain folders such as *thumbs*, *original*, *180px*, *320px*. These are all folders for the different display sizes and video formats respectively and are where the prepared files need to be

placed.

- */simplicity/csvfiles* contains the CSV-files for both the image and video archive and may be edited using Microsoft Excel.

3.1.2 Naming convention

All files related to a single image/video in the image/video archive are named according to a special naming scheme. A single image/video entered into the CSV-file has a unique id. This unique id must be used to name all the related files. For example, if an image has the id "heic001", then the different display sizes of the image are given names of the form *heic001.jpg* or *heic001.tif*. This naming scheme applies to both the image and the video archives.

3.2 Preparing Files

The first step in adding an image to the image archive is to generate the different display formats (eg, Screensize JPEG , Wallpaper 1024x768, Zoomable etc.). This is done using a Photoshop CS2 script and a Photoshop Droplet. Once the initial set up is complete, drag-and-drop the original image(s) onto the droplet. This will open Photoshop, make the different display formats and place them in the right folders. A zoomable image can also be generated in a simple drag-and-drop operation.

NOTE: The original image/video file must be named by a unique lowercase id without spaces (see 3.1.2 above). The same id must be entered in the metadata later on.

3.2.1 Installation of Photoshop Scripts

To make a droplet, first install the Photoshop scripts. The following files[1] should all be placed in *<PHOTOSHOP ROOT>/Preset/Scripts*:

- *Images.jsx*
- *Videos.jsx*
- *spacetelescope.jsinc*

Then edit both *Images.jsx* and *Videos.jsx* with a text editor such as Notepad to ensure that they save the new display sizes in the correct folder. The first few

[1] Located on the server in *V:\simplicity\installation\clients*.

lines of the script in *Images.jsx* look like this:

```
var store = "M:/llc/heicweb/images/";

#include "spacetelescopemethods.jsinc";

// Apply the script to the active document
run( app.activeDocument );
```

The first line defines where the image folders are located. Change it to the right path, so, if the images folder is located on *V:\simplicity\images* the line should be changed to:

```
var store = "V:/simplicity/images/";
```

Note that slash is used instead of backslash and note the trailing slash.
There must be a folder for each display size in *V:\simplicity\images*. Namely: *original, large, screen, medium, thumbs, mini, wallpaper1, wallpaper2, wallpaper3, wallpaperthumbs, zoom* and *hofthumbs*.

3.2.2 Making a Droplet

Once the Photoshop scripts have been installed, create a droplet. To do this, first create a new action:

1. Open any image.
2. Open the action window.
3. Press the *Create new action* button and name the action.
4. Press *Record*.
5. Now go to the menu *File* > *Scripts* > *Images* (the image formats will now be created for the opened image).
6. Close the image without saving when the script is done.
7. Press *Stop Recording* in the action window.

Next create the droplet.

1. Go to the menu *File* > *Automate* > *Create Droplet…*
2. Choose a place to save the droplet.
3. Choose the newly created action (changing the *Set* to see the action if necessary)
4. Press *OK* to complete the droplet.

Now drop one or more images onto the droplet to create all the different display formats. The droplet will automatically place the different display sizes

into the correct folders in, for example: *V:\simplicity\images*.

3.2.3 Zoomifyer

Zoomable images cannot be made with Photoshop, but a tool called Zoomifyer EZ can be downloaded from:

- Windows: http://www.zoomify.com/downloads/zoomifyerEZ.zip
- Mac OS X: http://www.zoomify.com/downloads/zoomifyerEZ.sit

Drag and drop your image onto *ZoomifyerEZ.exe*. This will create a folder with same name as your image file in the same directory as the image file. Move this folder to *V:\simplicity\images\zoom* to complete the process.

3.2.4 Videos

The different video formats must be created manually. The following formats should be made:

- **Small QT:** 180 x 144 pixels, QuickTime Sorensen-3 encoded (needs QuickTime 4 or 5).
- **Small MPEG:** 180 x 144 pixels, MPEG-1 encoded.
- **Medium QT**: 360 x 288 pixels, QuickTime Sorensen-3 encoded (needs QuickTime 4 or 5).
- **Medium MPEG:** 360 x 288 pixels, MPEG-1 encoded.
- **H.264:** Full PAL format (720 x 576 pixels) MPEG-4 encoded. Can be played with Apple QuickTime 7.
- **Broadcast:** 720 x 576 pixels, Uncompressed QuickTime or AVI, Broadcast quality (CCIR 601 PAL). Zipped with WinZIP. This format can be imported by any broadcast video editing system and edited.

Each video file must be named by a unique id as for the images and placed in the proper folder as shown in the following example:

Format	Filename	Folder
Small QT	*id1.mov*	*videos/180px/*
Small MPEG	*id1.mpeg*	*videos/180px/*
Medium QT	*id1.mov*	*videos/320px/*
Medium MPEG	*id1.mpeg*	*videos/320px/*
H.264	*id1.mp4*	*videos/h264/*
Broadcast	*id1.zip*	*videos/broadcast/*

The video formats can be made "pipeline-fashion", for instance with Cleaner XL

from Discreet. It is possible to use different input formats, and a huge number of output formats, sizes, effects etc. can be chosen. It is possible to make a batch list of files to process overnight. The program is very fast, and has excellent preview functions.

3.3 Editing Metadata

The metadata for the images and videos, such as **id**, **title**, **caption** etc. are stored in Microsoft Excel compatible comma-separated ***.csv***-files (hereafter called CSV-files). The CSV-files are located in *V:\simplicity\simplicity\csvfiles:*

1. *iacimagedata.csv*
2. *iacvideodata.csv*

CSV-files should be edited with Microsoft Excel (or similar), and saved in CSV-format. There is one important issue. As commas often appear in normal text it is not possible to use commas as list separators and so the pipe character "|" is used.

In order to use "|" instead of commas, set the *List Separator* to "|" in *Control Panel > Regional Settings (Choosing Advanced button)*. This refers to Windows only.

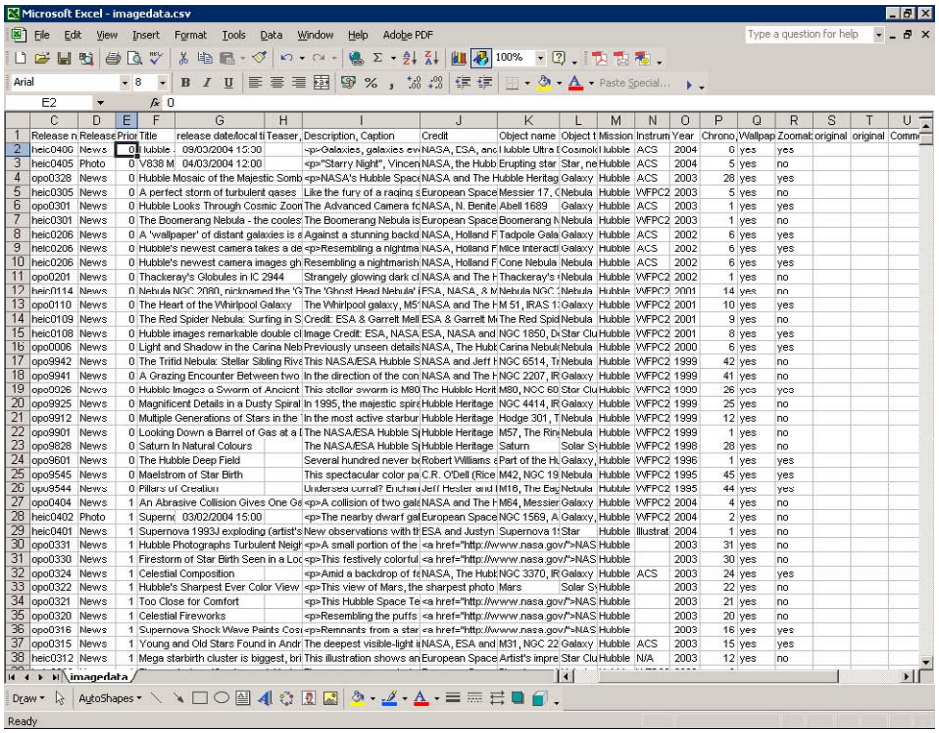

Figure 1 A screenshot of how the metadata look in Microsoft Excel (here from spacetelescope.org).

3.3.1 Metadata Fields

The *iacimagearchive.csv* consists of the following fields for each record:

Field	Description
Id	Unique identification of image. Image files should be named according to this id.
Priority	A number between 0 (best) and 5 (worst) describing the quality of the image.
Release date/local time.	A date and time in format YYYY/MM/DD HH:MM indicating when to release an entry. Leave empty to publish immediately

	(normal case).
Title *(es/en)*	Headline for image.
Teaser *(es/en)*	Short introduction emphasised in **bold** for example.
Description *(es/en)*	Description of image content displayed just after the teaser in normal text.
Credit *(es/en)*	Copyright and name of author.
Object Name *(es/en)*	Name of the object(s) in the image. If an image depicts several objects, then their names are separated by commas. Ideally naming should be Simbad compliant.
Object Type	The object type defines the category to which the image belongs. The field MUST either be left empty or be one of the following names: • Solar System • Nebulae • Stars • Star Clusters • Galaxies • Telescopes • Miscellaneous • Illustrations Note: Take care to type the object type exactly as above.
Object Subtype	Some of the object types may also have a sub-type. The field must be left empty or one of the following subtypes for a given object type: • Solar System: Planets, Small Bodies, Sun • Star Clusters: Globular Clusters, Open Clusters • Galaxies: Spiral, Elliptical, Irregular

	• Telescopes: Space, ORM, OT, Others
Instrument	The instrument must be left empty or be one of the following instruments: • VTT • THEMIS • GREGOR • Carlos Sánchez • MONS • IAC-80 • SST • DOT • INT • NOT • TNG • Liverpool • GTC
Wallpaper	"yes" or "no" Yes if the image should have a wallpaper, no if not.
Zoomable	"yes" or "no" Yes if the image has a zoomable image.
Original width	The width of the original image from which the other display sizes have been made.
Original height	The height of the original from which the other display sizes have been made.

Metadata fields for *iacimagedata.csv – es/en* means that there are two fields. One field in Spanish and one field in English.

The *iacvideoarchive.csv* consists of the following fields for each record:

Field	**Description**
Id	Same as for image archive.
Priority	Same as for image archive.
Release date/local	Same as for image archive.

time.	
Title *(es/en)*	Same as for image archive.
Teaser *(es/en)*	Same as for image archive.
Description *(es/en)*	Same as for image archive.
Credit *(es/en)*	Same as for image archive.
Duration	A number defining the duration of the video in seconds.
Category	Defines the category of the video. The field MUST either be left empty or be one of the following names: • Instruments • Educational • Astronomical Videos • Astronomical Animations Note: Take care to type the object type exactly as above.

Metadata fields for *iacvideodata.csv – es/en* means that there are two fields. One field in Spanish and one field in English.

3.3.2 Editing Metadata with Microsoft Excel

The metadata files are best edited with Microsoft Excel as this program works well with columns and rows of information. It has built in functions such as *spell-checking, "fill-Down"* and others that make maintenance work for the website easier.

To enter the metadata in the CSV-file, open the CSV-file with Microsoft Excel. If the list separator is set up correctly each field should be shown in its own column. Furthermore both CSV-files start with a line defining the metadata fields for an image or video (this line must not be deleted). To add metadata for a new image or video insert a new row anywhere but the first line.

When editing the fields it is easy to make "silly mistakes" and some care should be taken that the data are consistent. One common mistake is to include a space character in the object type. This is a mistake that is difficult to find, and the only trace is that the Perl script will return nothing when queried.

Text formatting for fields such as **teaser**, **description** and **credit** can be done with simple html codes. The following is a small selection of codes that can be used:

- `<p>sometext</p><p>sometext</p>`: Will display two paragraphs, separated with one line of space.
- `<strong>boldtext</strong>`: Text will be printed in bold.
- `<i>italicstext</i>`: Text will be printed in italics.
- `<a href="http://www.link.es">Linktext</a>`: Will create a link with the text Linktext referring to http://www.link.es.

When closing the CSV-file in Excel be sure to save the file in CSV-format – rather than the standard format preferred by Excel.

3.3.3 Priorities and Sorting

Perl scripts return output from the metadata files in the order that they retrieve them, so it makes sense to include a **Priority** field in the metadata, and to sort on this before saving the CSV-file. It is very important to do this sorting. An Excel macro can be created to do this.

To run an existing sort macro, go to the menu item *"Tools > Macro > Macros…"* and choose the appropriate macro.

3.3.4 Creating a Macro

As an example we will create a macro that sorts the content of a CSV-file. First make sure the CSV-file is open.

1. Go to a randomly chosen cell.

2. Choose *Tools* > *Macro* > *Record New Macro*... from the menu (a new window opens).

3. Name the macro (without white space), and choose the option *Personal Macro Workbook* under *Store macro in:*. You can choose to make a shortcut for the macro.

4. Press *OK* (notice that a little window with a stop-button appears).

5. Perform a normal sort of the data: Press *Ctrl-A* to select all cells, choose *Data*

> *Sort…* from the menu, choose the columns to sort by (and whether the sort is ascending or descending), and press OK in the sort window.

6. Press the *Stop* button in the little window that opened in stage 4.

The macro has now been created. NOTE: Save the changes in *Personal Macro Workbook* when Excel closes.

See Section 3.3.3 on how to run the macro.

3.4 Publish changes

The final step in adding an image/video to the archives is to upload the files to the webserver and trigger the generation of static pages.

3.4.1 Uploading Files

All kinds of file transfer programs can be used to upload files, but Dreamweaver, for example, has an option to synchronise a local directory with a remote directory that makes the publishing process easier. The files should be uploaded to the following server:

Hostname:	*pimienta.ll.iac.es*
Protocol:	SFTP (port 22)
Username:	Your username
Password:	Your password
Local directory:	*V:\simplicity*
Remote directory:	*/net/ttwww/si/www/catimage*

3.4.2 Generating Static Pages

Once the files have been uploaded a static page must be generated for each line added in the CSV-file. If nothing is done the pages will be generated within five minutes. However, to check everything is in order, page generation can be triggered from the administration interface located at the address http://ttwww.ll.iac.es/catimage/images/archive/admin/ or http://ttwww.ll.iac.es/catimage/videos/archive/admin/. Both addresses link to the same administration interface.

The administration interface provides a convenient way of administering the static html pages of the archives. Below is a screenshot of the administration

interface.

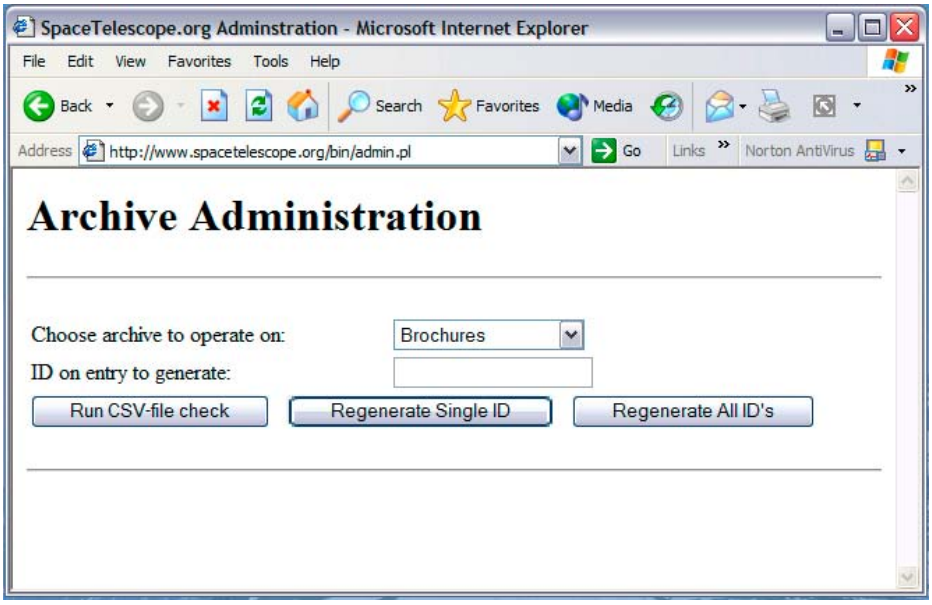

Figure 2: Screenshot of the administration interface.

The administration interface has three actions:

1. *Run CSV-file check*.
2. *Regenerate Single ID*.
3. *Regenerate All IDs*.

For all three actions, the archive to be operated on is chosen from the drop-down box.

3.4.3 Run CSV-file check

Use this action, for example, to run the generator-script immediately for a new CSV-file and rather than waiting five minutes for the static pages to generate automatically. It will check if any pages need to be updated based on the changes in the CSV-file. The output from the maintenance script will be shown below the buttons and will look something like this (one image has been added/updated in the image archive in this case):

```
File changed: /si/www/catimage/simplicity/ /csvfiles/iacimagedata.csv...
Starting check for new/updated/deleted ids...
No changes for infrared...
Generating HTML-page for hubble...
Generating zoomable page for hubble...
No changes for hubble10years_eng...
No changes for hubble10years_french...
No changes for hubble10years_german...
No changes for ngst...
No changes for heic...
No changes for archive2...
No changes for stecf...
No changes for factsheet...
```

3.4.4 Regenerate Static Pages

If some files have not been uploaded when a static page is generated, the given display sizes will not be shown on the page. However if you press *Run CSV-file Check* it will not generate a new page since the metadata has not changed. It is therefore necessary to force the generation of the static page, which can be done with the action *Regenerate Single ID*.

Enter the id of the image/video you want to generate and press *Regenerate Single ID*. All the static pages in an archive can be regenerated, if desirable, by using *Regenerate all IDs*.

4 Technical Documentation

This chapter targets those responsible for the technical maintenance of the image and video archive and assumes a good technical knowledge of Apache, Perl and UNIX system administration. The documentation gives an overview of the system, while more detailed documentation can be found in the source code. If you are not familiar with Apache or Perl the following links provide good structured documentation for both.

Perl:
http://perldoc.perl.org

HTML::Template:
http://html-template.sourceforge.net/

Locale::Maketext:
http://perldoc.perl.org/Locale/Maketext.html

Locale::Maketext::Lexicon:
http://search.cpan.org/~autrijus/Locale-Maketext-Lexicon-0.62/lib/Locale/Maketext/Lexicon.pm
http://search.cpan.org/src/AUTRIJUS/Locale-Maketext-Lexicon-0.62/docs/webl10n.html

Apache:
http://httpd.apache.org
http://httpd.apache.org/docs/2.2/mod/mod_rewrite.html
http://httpd.apache.org/docs/2.2/mod/mod_auth_basic.html

The next chapter will focus on how to customise the system in a minor way, anticipating future requirements over the next couple of years and requires a good knowledge of the system to understand these sections.

4.1 Overview

Generally the system consists of index pages that display search queries generated dynamically by user interaction. A search query can be, for example, browsing images in a certain category, browsing images with wallpapers or an advanced search. All queries are evaluated dynamically when requested by the user. If the user chooses to view a single image/video the user sees a static html page that is maintained and updated when necessary. Both the dynamic queries and static pages are generated from a template system and are in either English or Spanish. In addition, URL requests are rewritten by the webserver, so that, for example,
http://www.iac.es/catimage/images/archive/top100/ is rewritten to http://www.iac.es./catimage/bin/images.pl?searchtype=top100. This allows

search engines to index output generated by the Perl scripts, which would otherwise not be indexed.

4.1.1 Directory Structure

Before going into details, a general introduction to the directory structure will give a better overview. All paths will be referred to the root of the Simplicity system. Thus if the image archive is located in */net/ttwww/si/www/catimage/* then */images* refers to */net/ttwww/si/www/catimage/images*.

The following directories exist in the root :

/bin	CGI-script directory. Contains the scripts that generate the dynamic index pages.
/images	Static files for the image archive. Contains both html files and image files.
/videos	Static files for the video archive. Contains both html files and video files.
/simplicity	Contains all data files, scripts etc. that should not be publicly accessible. Contains, eg, the CSV-files and scripts for maintaining and updating the static html pages.

4.1.2 CGI-scripts

In */bin* there are three scripts and two symbolic links:

images.pl	CGI-script for index page generation for the image archive.
embargoimages.pl	Symbolic link to *images.pl* for the embargoed image archive.
videos.pl	CGI-script for index page generation for the video archive.
embargovideos.pl	Symbolic link to *videos.pl* for the embargoed video archive.
admin.pl	Administration interface for, eg, forcing the regeneration of static html files.

4.1.3 Simplicity files

In */simplicity* there are several important directories and files.

csvfiles/	Contains the CSV-files for both archives.

data/	Contains temporary data and log files that enable intelligent decision-making about updates to a static page. Also contains compiled templates to speed up page generation based on templates.
installation/	Contains configuration files and scripts make Simplicity easier to install.
lib/	Contains Perl modules that are common to the different scripts in Simplicity. However the Perl modules should only be available for the Perl scripts in Simplicity and not system-wide.
locale/	Contains language lexicons that make localisation of the Perl-scripts easier.
scripts/	Contains scripts to maintain the static files automatically.
webauth/	Password files to protect embargo pages.
webtemplates/	Contains template files for page design.

4.1.4 Static files

All the static files are located in */images* and */videos*. Both directories contain *html/en/* and *html/es/* - the English and Spanish html files respectively. Files in these directories are automatically updated by the system. Furthermore both */images* and */videos* contain some folders for the all the different display formats and videos.

4.2 Installation and Requirements

Simplicity can be installed on most Unix systems and requires Perl version 5.8 plus the following Perl modules:

CGI	(core module)
Digest::MD5	(core module)
Exporter	(core module)
Time::Local	(core module)
URI::Escape	(download from CPAN)
HTML::Entities	(download from CPAN)
Parse::RecDescent	(download from CPAN)
Inline::C	(download from CPAN)

Locale::Maketext (core modules from Perl version 5.8)
Locale::Maketext::Lexicon (download from CPAN)
HTML::Template (download from CPAN)
HTML::Template::JIT (download from CPAN)

The core modules are included in the Perl distribution while the other modules need to be downloaded and installed in their latest versions from CPAN (http://search.cpan.org). **Locale::Maketext::Lexicon** includes a script *xgettext.pl* that also needs to be available in the system path for execution.

Simplicity requires Apache version 1.3.x or 2.x.x with the option to use rewrites, multiviews and authentication. The program *msgfmt* - from *gettext* (see http://www.gnu.org/software/gettext/gettext.html) must compile the localisation files to byte-code to localise the Perl scripts.

4.2.1 Webserver configuration

The webserver configuration is in two stages. The configuration that needs to be inserted into the correct virtual host is located in */simplicity/installation/ServerSetup.txt*. It is especially important that the rewrites are located in the virtual host configuration and not a, for example, *.htaccess* file, since Apache will apply its own rewrites before passing control to the rewrites in *.htaccess* (this is avoided by placing them in the virtual host instead). Furthermore it is important that some rewrites (see the file) have `[PT]` after the pattern, to allow the request to pass through other handlers. Likewise it is important that `AuthConfig` and `Limit` can be overridden to allow *.htaccess* files to be generated dynamically to password protect embargo pages. Second, certain directories such as */simplicity/* deny public access by use of *.htaccess* files.

Further documentation on rewrites can be found in the Apache documentation.

4.2.2 Perl Modules

The newest versions of the Perl modules can be downloaded from http://search.cpan.org (search by name) and installed by running the following command in each of the downloaded and expanded directories.

```
perl Makefile.PL
make
make install
```

or by using

```
sudo perl -MCPAN -e 'install HTML::Template::JIT'
```

4.2.3 Permissions

The system is fairly sensitive to permissions issues. If some files have the wrong permissions, they may not be overwritten and the system will seem to be broken. Correct permissions can be set on the necessary files and folders by running the script */simplicity/installation/modify_permissions.sh*. In particular, the files */simplicity/data*, */images/html* and */videos/html* must be writeable by both the user running the cron-scripts and the webserver user. This is best achieved by putting the cron-script user and the webserver user into the same group and making files writable by the file owner and the file owners group.

4.2.4 Cronjob

The following cronjob needs to be run on the server by the same user as the webserver user.

```
00,05,10,15,20,25,30,35,40,45,50,55 * * * * cd
/net/ttwww/si/www/catimage/simplicity/scripts/; /usr/bin/perl
up2datecheck.pl –A images; modify_permissions.sh 1>/dev/null

00,05,10,15,20,25,30,35,40,45,50,55 * * * * cd
/net/ttwww/si/www/catimage/simplicity/scripts/; /usr/bin/perl
up2datecheck.pl –A videos; modify_permissions.sh 1>/dev/null
```

4.3 System Modules

Common Perl modules for all the scripts are located in */simplicity/lib/Simplicity*. This is a list of the modules with a brief description:

ImagesArhive.pm	Image archive specific routines and definitions.
IndexTools.pm	Generation of, eg, the Google navigation bar.
L10N.pm	Sets up of script localisation.
Search.pm	Routines for searching the CSV-files.
Site.pm	Global site definitions of various paths.
Tools.pm	Routines for easy generation of static pages.
VideosArchive.pm	Video archive specific routines and definitions.

4.3.1 Template System

The Perl scripts that produce the output use the template system **HTML::Template::JIT** that is an extension of **HTML::Template** that compiles the templates to native byte-code for faster execution. The compiled templates are stored in */simplicity/data/compiled_templates*. The actual templates are located in */simplicity/webtemplates/en* and */simplicity/webtemplates/es* depending on their language. The template language is straightforward and is documented on http://html-template.sourceforge.net so only a brief example will be given here.

Basically the contents of a variable can be inserted with:

```
<tmpl_var name="variable_name">
```

The above template code will require that the variable name `variable_name` is set somewhere in the code by the following Perl code:

```
$tpl->param('variable_name' => <somevalue> );
```

The templates are set up in *images.pl*, *videos.pl*, *staticimages.pl*, *staticvideos.pl* and *IndexTools.pm*. Each of the scripts creates a template instance in the variable `$tpl`. It is obvious from the scripts which main template file they use. The main template file may include other template files and conditional statements and is thus very simple, yet powerful, to use.

The template system should be able to detect changes to the template files and recompile the templates. However, this doesn't always work when templates included in the main template are updated. Therefore the compiled templates can be cleared from the administration interface. The first time a script is accessed after clearing the cache, the script will need to compile a new template. This compilation will take several seconds, but is a one time operation.

4.3.2 Locale System

The locale system uses the Perl modules **Locale::Maketext** and **Locale::Maketext::Lexicon** to translate the text located inside the Perl scripts more easily. Most text should be located in the templates and therefore be easy to translate, but inevitably some text must be located in the scripts, which is where the locale system comes in handy.

Localising a script consists of:

1. Loading the necessary modules (that is **Simplicity::L10N**).
2. Marking strings that need to be translated.
3. Extracting strings that need translation from scripts.
4. Inserting the strings into a common lexicon.
5. Translating the strings in the lexicon.
6. Compiling the lexicon to machine code.

Marking strings

To localise a script, load the module **Simplicity::L10N** and, somewhere in the script, before the localised strings, make a call to:

```
Simplicity::L10N::setLanguage( 'en' );
```

Afterwards mark a string for localisation by using the routine `loc()`. So, for example, to localise the string `"Results for $text"` write:

```
loc('Results for [_1]',$text)
```

This example also shows how to include variables in the string by use of `[_1]`. For most cases just surround the string with the `loc` function.

The function `loc` will replace the text with the translated text. Sometimes the text will have to be translated later, for example if calling `loc($text)` where `$text` contains a string. This string should not be translated when given to `loc`, as `loc` will try to translate the string. However the string still needs to be inserted into the lexicon. So, to mark a string for translation, but return the original string instead of the translated string, call the routine `translate` instead. For an example of its use see *ImagesArchive.pm*.

Extracting strings

Strings are extracted by the external program *xgettext.pl* as part of the **Locale::Maketext::Lexicon** module. The program is called from the command line:

```
xgettext.pl    -o=simplicity/locale/es/messages.po \
               -f=simplicity/installation/localizefiles.txt \
               -D=bin \
               -D=simplicity/lib/Simplicity \
               -D=simplicity/scripts
```

This tells *xgettext.pl* to put a lexicon in the Spanish directory

/simplicity/locale/es/messages.po, to read *localizefiles.txt* for files that need to be parsed for marked strings and that the files are located in three possible directories (by use of the `-D` option). This step needs to be repeated in the English directory to create both an English and a Spanish lexicon.

The above command will create the file */simplicity/locale/es/message.po* and */simplicity/locale/en/message.po*. If the files are already there, they will be updated with any new strings. This ensures that if the string has already been translated, it will just be reused instead of inserting a new string.

Translation of lexicons
The *messages.po* files will contain a long list of entries similar to this:

```
#: simplicity/scripts/staticimages.pl:104
msgid "Images"
msgstr "Imágenes"
```

If the string has not yet been translated, `msgstr` will be an empty string. The comment above `msgid` shows the translator where the strings were extracted from. All the empty `msgstr` have to be filled out to translate the lexicon.

Compilation of Lexicon
Once the lexicon has been translated, it must be compiled to machine code by the program *msgfmt* from *gettext*. This program is usually installed on most Unix development systems. To compile a lexicon, just run the following command:

```
msgfmt -o simplicity/locale/es/messages.mo simplicity/locale/es/messages.po
```

The process of extracting strings and running *xgettext.pl* and *msgfmt* has been automated by the script */simplicity/installation/localize.sh*. This script will extract strings with *xgettext.pl* and compile the lexicon in one step. So, if new localised strings have been added, run *localize.sh* once, translate the lexicon and run it one more time.

4.3.3 Search Modules

The Perl module *Search.pm* is responsible for all searches in the CSV-files. It uses code references and anonymous subroutines to make implementation simpler and more structured. It is strongly recommended that readers make sure they understand code references and anonymous subroutines before continuing. Further information can be found at the following to URLs:

http://perldoc.perl.org/perlsub.html
http://perldoc.perl.org/perlref.html

Basically there are four main routines in *Search.pm*. First, the subroutine `findId` will look up a single id in a CSV-file. Second, the subroutine `search` will look up several rows matching a given condition. Third, the subroutine `parseQuery` will parse a query string to make an abstract representation suitable for the fourth subroutine `freesearch` to find matching rows. Furthermore the `parseQuery` function will expand the query for certain astronomical search terms, to allow for better searching.

Both `findId`, `search` and `freesearch` will make calls to a function `splitRow` that is defined in either *ImagesArchive.pm* or *VideosArchive.pm* depending on the current archive. The function `splitRow` defines the fields for the CSV-files.

Search Queries
The query parser will parse search queries from the keyword search and the advanced search "by string". The following is an example of the syntax of a search query (a grammar is given in the source code for *Search.pm*)

```
+Pluton -Charon M51 "Some Phrase"
```

Basically a search query consists of mandatory and non-mandatory terms (no sign), where mandatory terms can be inclusive (plus sign) or exclusive (minus sign). A term is a single keyword or phrase. In the above example there are four search terms. All mandatory conditions are checked to hold first, meaning that all inclusive terms have to be in a single entry (a row in the CSV-file) and at the same time all exclusive terms must not be in a single entry. Then all non-mandatory terms are checked. All non-mandatory terms require that at least one of the terms needs to be in a single entry to have the entry included in the search result.

Query Expansion
Queries are expanded using terms like `M51` and `NGC 3370` since both the concatenated version and spaced version may exist in the CSV-file and should be found. The query expansion will expand a single term like `M51` or `'M 51'` (notice the quotes which make it a single term) to the two terms `'M51'` and `'M 51'`. So

M51 => 'M51' or 'M 51'

'M 51' => 'M51' or 'M 51'

This will expand the search field and include more entries in the final search result since we search on more terms (boolean OR). Furthermore if two search terms in a row match a certain pattern, eg,. `M 51`, the query parser will narrow the search field. Notice that `M 51` is two search terms namely `M` and `51`. If two search terms in a row match a given pattern it will be expanded. Example:

M 51 => 'M51' or 'M 51'

This narrows the search field by using more specific terms. We concatenate two search terms to produce two more specific search terms. This will result in fewer entries in the final search result.

4.3.4 Archive Modules

The two archive modules *ImagesArchive.pm* and *VideosArchive.pm* contain source code specific to the image and video archives respectively. Paths to data files, output directories, dependent templates etc. are all located in the archive module for the respective archive. Furthermore subroutines that define the fields in the CSV-file, process up-to-date checks and generate static pagesetc. are also located in the archive modules. Specific details will not be given here, since all subroutines and variables are used in some other module/script where they will be explained.

4.4 Scripts

The two modules *Tools.pm* and *IndexTools.pm* are used in the scripts responsible for static page generation, up-to-date checks and index page generation. These scripts and the roles of the two modules will be described in this section.

4.4.1 Index Page Generation

Index page generation is performed by the scripts */bin/images.pl* and */bin/videos.pl* respectively. Essentially the following happens in the script:

1. Initialisation:
    a. Path set up.
    b. Parameter extraction from query.
    c. Localisation set up.
2. Selection of index layout based on parameters.

3. Generation of search condition based on parameters.
4. Search and retrieve results.
5. Pass results and other values to template system for displaying to user.

The script takes the following parameters:

`searchtype`	Determines which kind of search to perform: a Hall of Fame search, Advanced search, Keyword search, Top 100 search etc.
`viewtype`	The main index layout type to use. Defaults to `standard`.
`string`	A query string whose meaning depends on `searchtype`.
`from`	The image/video number that the index page must contain (for browsing results).
`lang`	One of the values `en` or `es` for English/Spanish (defaults to `en`).

Index Layout

The initialisation process just ensures that the proper modules are loaded and that parameters have been extracted. Afterwards the index layout is set up to allow for different layouts of the index pages for example for Hall of Fame, View All, Wallpaper and Normal View. Since an index layout can probably be used for both the image and video archive, the module *IndexTools.pm* defines a list of index layouts, namely:

```
wallpaper_viewall
wallpaper
zoomable_viewall
zoomable
bestof
videosbestof
videosstandard_viewall
videosstandard
standard_viewall
standard
```

All layout types are defined in *IndexTools.pm* with number of rows, number of columns and whether to use one, two or three html table rows for each row of results. This allows the template system to first make an html table row with images and then a row with image titles before continuing to the next row of results. The html code for each index is defined in */simplicity/webtemplates/en/indicies.tpl* (English version).

Search Condition

The search condition for each search type might look like this:

```
$condition = sub { $row::wallpaper eq "yes" };
```

This creates an anonymous subroutine that is called for each row in the CSV-file. The subroutine must return `TRUE` to have a row included in the search results (note that if no return statement is present, then the return value is defined by the last statement in the subroutine). Inside the subroutine the individual fields can be accessed by `$row::<fieldname>`. Hence, in the example above, all rows in the CSV-file having `yes` as a value for the field `wallpaper` will be included. Alternatively, a limit condition can also be specified to retrieve only a certain number of rows (this should however only be used when the status bar is not shown as in Hall of Fame searches).

4.4.2 Up-to-date Checks

If the website has many simultaneous hits, static pages are generated for all entries in the archives to reduce the CPU load. Up-to-date checks of the static pages are performed by the script */simplicity/bin/up2datecheck.pl*.

The script is run every 5 minutes to check if it needs to generate a new set of static pages for a given archive. Furthermore the script can be run manually from the command line with the following options:

```
Usage: up2datecheck.pl [options]
         -o              trigger CSV-file check
         -a              generate all pages
         -A [archive]    archive to check
```

If no archive is given nothing happens. It is important to note that *up2datecheck.pl* only runs checks against the CSV-file and not the actual image/video files. Hence, if a user has forgotten to upload a broadcast video, the script will not regenerate the static page even though it is outdated. The administration interface described in the user guide (see Section 3.4.2) uses this script and should be used in preference to the command line version since it might cause permission issues on the generated files if the command line version is used.

The script keeps track of changes in a couple of ways. File modification times are monitored, so if the files used by a static page are changed, an update check will be carried out. Essentially changes to the templates files and scripts that generate the static pages will result in the regeneration of all static pages

in an archive. However if a CSV-file has been modified the script will only regenerate the static pages for which the data has changed.

To detect if a row has been edited the script saves a checksum for the row and compares this checksum with the new checksum next time it runs. If a row has not been changed, its checksum will be the same.

A list of embargoed entries together with the release date for each archive is also maintained to trigger regeneration and removal of password protection when the release date has been passed.

The scripts also generate an *.htaccess* file that denies access to embargoed entries. It is generated on the basis of a template located in */simplicity/scripts/templates/htaccess.tpl* and looks like:

```
<Files ~ "(IDLIST)\.(EXTLIST)">
AuthType Basic
AuthName "Embargo Images and Videos"

AuthUserFile /si/www/catimage/simplicity/webauth/webauthusers
AuthGroupFile /si/www/catimage/simplicity/webauth/webauthgroups
require group embargos admins
</Files>
```

`IDLIST` is replaced by a list of embargoed entry ids, and `EXTLIST` is replaced by a list of extensions (typically *html, txt, doc, pdf, jpg, gif, tif* and *zip*) specific for each archive (defined by the variable `$Simplicity::EXTLIST` in the archive modules).

If an entry (image/video) is removed from the CSV-file, the corresponding static html page is also deleted. All non-generated files in output directories will also be deleted, unless they begin with a full stop like *.htaccess*.

4.4.3 Static Page Generation

The *up2datechecks.pl* script, as described in previous section, controls the generation of static pages. The script will use the subroutines `generatePage`, `updateCheck` and `initUpdateCheck` defined in the archive modules to determine if the proper files exist and generates a static page if *up2datechecks.pl* finds it necessary to update a page.

Both the image and video archive will rely on the scripts *staticimages.pl* and *staticvideos.pl* in the `generatePage` subroutine to generate a single static page.

Both scripts are quite simple as they only extract a row from the CSV-file and pass the values on to the template system. The two scripts use a small set of subroutines in the module *Tools.pm* to allow hashes for the template system to be created more easily.

4.5 Photoshop Scripts

The User Guide (Section 3) documents how to install and use Microsoft Excel, Photoshop and a file transfer tool for archive administration. More detailed information on the inner workings of the Photoshop scripts is given here.

4.5.1 Images Display Formats

The website *Spacetelescope.org* uses 12 different image display sizes.

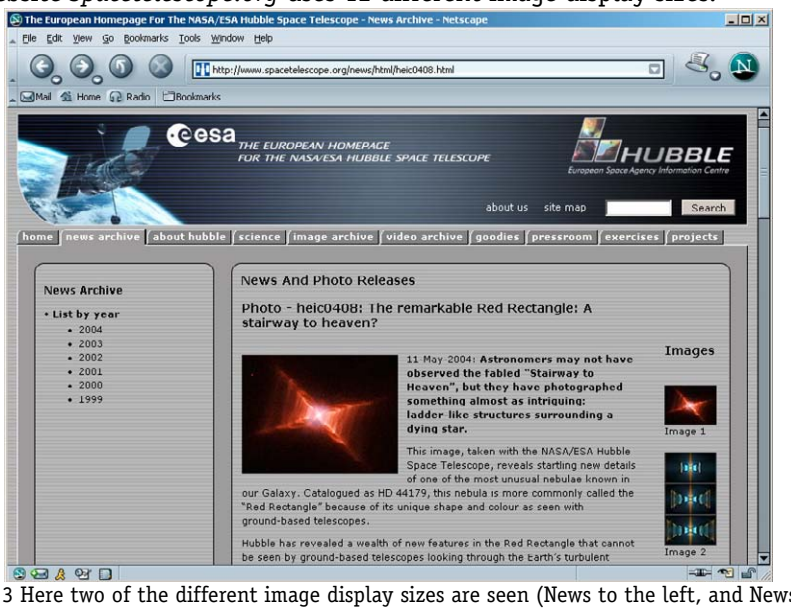

Figure 3 Here two of the different image display sizes are seen (News to the left, and Newsmini to the right).

The different display sizes that are generated from the original images are (W x H in pixels):

1. **original**: "Fullsize original" (tiff/jpg/gif). No colour correction. Original colour profile embedded.
2. **large**: "Large jpeg" (orig x orig), jpeg quality 80. No colour correction. Original colour profile embedded.

3. **screen** : "Screensize jpeg" (1280 x *), jpeg quality 60. Unsharp mask radius 1, Threshold 0, Amount 25. If the colour profile of the original image is either AdobeRGB or LStar-RGB, the image is colour corrected manually and no profile is embedded. If the colour profile is unknown, no correction is applied and no profile is embedded.
4. **medium**: "Medium jpeg" (320 x *), jpeg quality 60, unsharp mask radius 1, Threshold 0, Amount 70. If the colour profile of the original image is either AdobeRGB or LStar-RGB, the image is colour corrected manually and no profile is embedded. If the colour profile is unknown, no correction is applied and no profile is embedded.
5. **thumbs**: "Thumbnail jpeg" (max(x,y) = 122), jpeg quality 45, unsharp mask radius 1, Threshold 0, Amount 80. If the colour profile of the original image is either AdobeRGB or LStar-RGB, the image is colour corrected manually and no profile is embedded. If the colour profile is unknown, no correction is applied and no profile is embedded.
6. **mini**: "Mini jpeg" (max(x,y) = 50), jpeg quality 45, unsharp mask radius 1, Threshold 0, Amount 80. If the colour profile of the original image is either AdobeRGB or LStar-RGB, the image is colour corrected manually and no profile is embedded. If the colour profile is unknown, no correction is applied and no profile is embedded.
7. **wallpaper1**: "Wallpaper 1024 x 768", jpeg quality 80. Unsharp mask radius 1, Threshold 0, Amount 25. If the colour profile of the original image is either AdobeRGB or LStar-RGB, the image is colour corrected manually and no profile is embedded. If the colour profile is unknown, no correction is applied and no profile is embedded.
8. **wallpaper2**: "Wallpaper 1280 x 1024", jpeg quality 80. Unsharp mask radius 1, Threshold 0, Amount 25. If the colour profile of the original image is either AdobeRGB or LStar-RGB, the image is colour corrected manually and no profile is embedded. If the colour profile is unknown, no correction is applied and no profile is embedded.
9. **wallpaper3**: "Wallpaper 1600 x 1200", jpeg quality 80. Unsharp mask radius 1, Threshold 0, Amount 25. If the colour profile of the original image is either AdobeRGB or LStar-RGB, the image is colour corrected manually and no profile is embedded. If the

colour profile is unknown, no correction is applied and no profile is embedded.

10. **zoom**: No options, produced with *Zoomifyer EZ v3.0.exe*.
11. **wallpaperthumbs**: "Thumbs 122 x 92 pix" created from wallpaper1 jpegs, jpeg quality 45, unsharp mask radius 1, Threshold 0, Amount 80. If the colour profile of the original image is either AdobeRGB or LStar-RGB, the image is colour corrected manually and no profile is embedded. If the colour profile is unknown, no correction is applied and no profile is embedded.
12. **hofthumbs**: For Hall of Fame (95 x *), jpeg quality 45, unsharp mask radius 1, Threshold 0, Amount 80. If the colour profile of the original image is either AdobeRGB or LStar-RGB, the image is colour corrected manually and no profile is embedded. If the colour profile is unknown, no correction is applied and no profile is embedded.

These different display sizes are generated with the help of Photoshop CS2's built-in **Javascript** facility. A small script (for instance *Images.jsx*) calls another script (*spacetelescopemethods.jsinc*) that contains recipes for how every display size on the entire website is created for each archive. For instance, the script that creates the 15 main images that are used in the image archive, *Images.jsx*, looks like this:

```
var store = "M:/llc/heicweb/images/";


#include "spacetelescopemethods.jsinc";


// Apply the script to the active document
run( app.activeDocument );


/**
 * Runs the image processing steps
 * @param document Document to process.
 */
function run(document) {
    var originalState = document.activeHistoryState;
    var originalUnit = app.preferences.rulerUnits;
    app.preferences.rulerUnits = Units.PIXELS;
// Flatten the image, change mode to 8-bit RGB
    document.flatten();
    document.changeMode( ChangeMode.RGB );
    document.bitsPerChannel = BitsPerChannelType.EIGHT;
// Copy he original image into the original folder
    copyOriginal( document );
```

```
// Generate all 12 thumbnails except Round
    processThumbnails( document );
// Go back to the original state
    document.activeHistoryState = originalState;
// Flatten the image, change mode to 8-bit RGB to prepare for other actions
    document.flatten();
    document.changeMode( ChangeMode.RGB );
    document.bitsPerChannel = BitsPerChannelType.EIGHT;
    app.preferences.rulerUnits = originalUnit;
}
```

The script calls *spacetelescopemethods.jsinc,* which starts like this:

```
var lstarString = "Lstar-RGB.icc";
var adobergbString = "Adobe RGB (1998)";


var Algorithm = {
    None       : 0,
    Crop       : 1,
    ResizeWidth: 2,
    ResizeBox  : 3,
    Resize4kpx : 4
};

/**
 * Defines a function that can be make new display formats with.
 */
function Thumbnail( a, w, h, s, u, n, q, argbsat, argbB, argbG, argbW,
lstarSR, lstarHY, lstarHB, lstarB, lstarG, lstarW ) {
    return {
        algorithm             : a,
        width                 : w,
        height                : h,
        sharpen               : s,
        unsharpen             : u,
        name                  : n,
        quality               : q,
        AdobeRGBSatMaster     : argbsat,
        adobeRGBBlackLevel    : argbB,
        adobeRGBGamma         : argbG,
        adobeRGBWhiteLevel    : argbW,
        LStarSatRed           : lstarSR,
        LStarHueYellow        : lstarHY,
        LStarHueBlue          : lstarHB,
        LStarBlackLevel       : lstarB,
        LStarGamma            : lstarG,
        LStarWhiteLevel       : lstarW
    };
}

// Define default values
var defaults = [
0,                    //defAdobeRGBSaturationMaster  0
0,                    //defAdobeRGBBlacklevel        1
1.0,                  //defAdobeRGBGamma             2
255,                  //defAdobeRGBWhitelevel        3
```

```
0,                    //defLStarSaturationRed        4
0,                    //defLStarHueYellow            5
0,                    //defLStarHueBlue              6
0,                    //defLStarBlacklevel           7
1.0,                  //defLStarGamma                8
255                   //defLStarWhiteLevel           9
];

// Parameters for every display size except original and zoom.
// The proper display format is called by an index from 0-10
// Compare these with the description of the display sizes. The one
// showed below is the screen display format.
var thumbnails = [
Thumbnail( Algorithm.ResizeWidth, 1280, 0, false, 25, "screen" , 60, 5, 8,
0.96, 255, 6,  -5,  5,  5,0.95, 255  ),
…
];
```

4.5.2 Video Display Formats

The website uses five different types of main videos:

1. **Small QT**: 180 x 144 pixels, Quicktime Sorensen-3 encoded (needs Quicktime 4 or 5).
2. **Small MPEG**: 180 x 144 pixels, MPEG-1 encoded.
3. **Medium QT**: 360 x 288 pixels, Quicktime Sorensen-3 encoded (needs Quicktime 4 or 5).
4. **Medium MPEG**: 360 x 288 pixels, MPEG-1 encoded.
5. **H.264**: Full PAL format (720 x 576 pixels) MPEG-4 encoded. Can be played with Apple Quicktime 7.
6. **Broadcast**: 720 x 576 pixels, Uncompressed Quicktime or AVI, Broadcast quality (CCIR 601 PAL). Zipped with Winzip. This format can be imported by any broadcast video editing system and edited.

The video formats are made "pipeline-fashion" with Cleaner XL from Discreet. It is possible to use different input formats, and a huge number of output formats, sizes, effects etc. can be chosen. It is possible to make a batch list of files to process overnight. In addition the program is very fast, and has excellent preview functions.

5 Technical Customisation

This chapter assumes knowledge of the content of the previous chapter and will mainly focus on what to remember when making small changes to Simplicity. Common to all changes is that modifications to the Perl scripts will probably require new lexicons and translations. That is, run */simplicity/installation/localize.sh* once, translate the *message.po* files and run *localize.sh* again (see 4.3.2).

5.1 New Field in CSV-File.

Adding a new field in the CSV-file primarily requires that the routine that split a row in the CSV-file is updated. This subroutine is called `splitRow` and is located in either *ImagesArchive.pm* or *VideosArchive.pm* depending on which archive is being changed. The most important part is that the fields are defined in the same order in the `splitRow` subroutine as they appear in the CSV-file.

To display the newly added field on the static page, update *staticimages.pl* or *staticvideos.pl* by adding the field as a template variable. The place to be edited in the source code will look somewhat like this, where the new field `newfield` has been added:

```
# Setup field names
$tpl->param(    "ID" => $row::id,
                "PRIORITY" => $row::priority,
                "NEWFIELD" => $row::newfield,
```

Remember that some fields used to display text on the static pages must be localised and thus require two separate fields.

5.2 New Image Category/Instrument

Adding a new instrument to the list of possible instruments means editing */images/search.html.en* and */images/search.html.es* (note this only applies to the image archive). Find the drop-down box for the instrument list and add the new instrument. The value of the option is the value that must be used in the CSV-file.

Adding a category requires more steps (applies to both the image and video archive).

1. First edit *ImagesArchive.pm*, to add a new category/subcategory to the

variable `%Simplicity::ImagesArchive::Categorization` which looks like this:

```
%Simplicity::ImagesArchive::Categorization = (
   translate('Solar System')  => [translate('Planets'),translate('Sun')],
   translate('Nebulae')       => [],
```

The name used above for the category must be the same as the one used in the CSV-files.

2. Next ensure that there is a translation for the category. This is done by running */simplicity/installation/localize.sh*, then translating the category in the *messages.po* files for both English and Spanish, then running */simplicity/installation/localize.sh* again (see 4.3.2 for more details on translation).

3. Edit */images/search.html.en* and */images/search.html.es*. First add the new category to the drop-down box. Use lowercase for the value of the option tag, as in:

```
<option value="new category">New Category
```

Next add the category to the JavaScript array to the top of the file by using the examples below.

```
store['new category'] = new Array( ); // Without sub-categories

store['new category'] = new Array(    // With sub-categories
       'All',
       'all',
       'New Subcategory',
       'new subcategory'
       );
```

If there are no subcategories, just add the first line. If there are subcategories, then ensure that both the subcategory and "all" subcategory exist as shown in the example above. Each subcategory has two elements in the array. First element defines the displayed text and should be localised, while the second element is not localised and should match the subcategory added in step 1 (in lowercase).

4. Lastly add the category/subcategory to the category overview page */images/index.html.en /images/index.html.es*. Use one of the links below to display an index page for the category/subcategory (remember to change `en` to `es` for the Spanish version).

```
archive/en/subtopic/new+subcategory/
archive/es/topic/new+category/
```

Notice that space is replaced by `+` signs.

5.3 New Image Formats

To add new image display formats:

1. Update the Photoshop script and create a new droplet.
2. Make a folder for the display format.
3. Update *staticsimages.pl* to show the new format on static pages.

Add the new display format to the thumbnails array in the top of the Photoshop script, */simplicity/installation/clients/spacetelescopemethods.jsinc*.

```
Thumbnail( Algorithm.ResizeWidth,  122,    0, false, 80, "thumbs_new"
, 45,  5,  8, 0.96, 255, 6,  -5,  5,  5,0.95, 255, false  ),//2
```

This creates a display format that is 122 pixels wide and keeps that aspect ratio. The image will be saved in the folder `thumbs_new`, so create the directory */images/thumbs_new*. The rest of the parameters can be tweaked as necessary. Refer to section 4.5.1 for more information on the different display sizes that might help to customise the new format.

Next create a variable `index_thumbs_new` to hold the index of the newly added element to the `thumbnails` array. Insert the variable `index_thumbs_new` in the list in the `makeImages` function. When the *spacetelescopemethods.jsinc* script has been edited remember to distribute to all client machines and create new droplets (see 3.2.2).

Last, the display format should be added to the static pages by editing *staticimages.pl*. This is done by adding the following code next to the other downloads.

```
addDownload(   \%images,                    // Section to add download to.
               loc('Thumbs New'),           // Display name,
               "thumbs_new",                // Folder.
               "jpg",                       // Image extension
               "images",                    // Icon to use
               $Simplicity::LOCALDIR,       // Always the same
               $Simplicity::LOCALURL,       // Always the same
               1 );                         // Use special massive warning
```

The last parameter `1`, is used on large files to check whether the pixel size is above 4000x4000 pixels instead of only 20 MB. Remember to localise the script afterwards.

5.4 New Video Formats

Adding a new video format requires more files to be updated than when adding new image display formats. First decide if the new format should be downloadable only, or also have a play page. If the video is to be downloadable only thrn go through nearly the same steps as for new image formats. Just edit *staticvidoes.pl*, not the Photoshop script. Formats with a play-page should also go though this step.

Basically only the following lines are needed to add the new video format to the html pages.

```
addVideo(      \@videos,
               loc('Large H.264 MPEG-4'),    // Title (localized)
               'mp4',                        // Playable format.
               'mp4',                        // Downloadable format.
               'h264',                       // Folder/Size.
               'qt',                         // Icon.
               0,                            // 1 if has play page, 0 else.
               $Simplicity::LOCALDIR,        // Always the same.
               0,                            // Tell template to show
special note (currently: For TV use only!)
               0);                           // 1 if playpage exists but
only in one size, 0 if both small and medium exists.
```

If the video format is only downloadable - as in the above example - remember to create the folder */videos/h264* and add the extension to (`mp4`) to the variable `$Simplicity::EXTLIST` in *VideosArchive.pm*.

If the video format is to have play-pages, then additional steps are required:

1. Add the format to the variable `@PlayVideoFormats` in *VideosArchive.pm* if it there are both a small and a medium version.
2. Add the format to the variable `@PlayVideoFormatsNoSize` in *VideosArchive.pm* if there is only one size version.
3. Create directories to html play-pages and datafiles. For two-size video formats (small/medium) create */videos/html/180px/newformat* and */videos/html/320px/newformat*. For one-size video formats create */videos/html/newformat* and */videos/newformat_folder*.

Last, add a template variable in *staticvideos.pl* for the play-page:

```
      'newformat_playpage' => ($version eq 'format_ext')
```

and update the templates */simplicity/webtemplates/en/video.tpl* and *video_embargo.tpl* (both Spanish and English) to display the new play-page. Remember to localise the Perl scripts.

5.5 New Index Type

To add a new index layout type, first update *IndexTools.pm* with the new index type:

```
       elsif( $layout eq 'new_layout_type' ) {
              $Simplicity::Index::TableCols = 4;
              $Simplicity::Index::TableRows = 3;
              $Simplicity::Index::IndexLayout = 'new_layout_type';
              $Simplicity::Index::UseTitleRow = 1;
              $Simplicity::Index::UseAdditionalRow = 0;
       }
```

1. The variable `IndexLayout` describes the name of the index layout and is used to include the proper index template in the template *indices.tpl*.
2. Next, edit *indices.tpl* and add the line:

```
<tmpl_if name="index_new_layout_type"><tmpl_include
name="index_new_layout_type.tpl"></tmpl_if>
```

3. Create the file *index_new_layout_type.tpl* and use some of the other index layouts as a guide. Remember to create the file in both the Spanish and English directories.

5.6 Users and groups management

Password files in */simplicity/webauth* are used to manage access to the administration page and embargo news for users and groups. Refer to the Apache documentation for information on how to update these files.

http://httpd.apache.org/docs/2.2/programs/htpasswd.html

6 Index