

| +Track changes base is 1.3.1a

Pi Presents

**A Multi-media Interactive Display and Animation Toolkit
for Museums, Visitor Centres and more.....**

Running on the Raspberry Pi

Ken Thompson

<http://pipresents.wordpress.com>

Contents

1	Introduction	6
1.1	Acknowledgements	8
2	Installation	9
3	Try Some Examples	9
3.1	Terminating Pi Presents	13
4	The Pi Presents Profile Editor	13
4.1	Using the Editor.....	13
4.1.1	The Displays	13
4.1.2	Profile Menu.....	14
4.1.3	Show Menu	14
4.1.4	Medialist Menu	14
4.1.5	Track Menu	15
4.1.6	OSC Menu	15
4.1.7	Tools Menu	15
4.1.8	Options Menu.....	16
4.1.9	Editor Command Line Options	16
4.2	Making Profiles	16
4.2.1	Making Profiles Portable	17
4.2.2	Using the SD Card for Profiles	17
4.2.3	Using the Editor on a Windows PC	18
4.2.4	Using a USB Stick for Profiles	18
4.2.5	Developing Profiles using a Monitor with Different Pixel Dimensions..	18
5	The Components of Pi Presents	19
5.1	Introduction	19
5.2	Shows	21
5.2.1	Mediashow and Artmediashow.....	21
5.2.1.1	Controls and Commands.....	23
5.2.1.2	Triggers.....	24
5.2.1.3	Fields	25
5.2.2	Menu	31
5.2.2.1	Controls and Commands.....	31
5.2.2.2	Fields	31
5.2.3	Liveshow and Artliveshow	32
5.2.4	Radiobuttonshow	33
5.2.4.1	Controls and Commands.....	33
5.2.4.2	Fields	34
5.2.5	Hyperlinkshow.....	35
5.2.5.1	Controls and Commands.....	36
5.2.5.2	Touchscreens and Soft Buttons	37
5.2.5.3	Fields	38
5.2.6	Start Show	39
5.3	Medialists	40
5.4	Tracks	40
5.4.1	Track Location Names	41
5.4.2	Anonymous and Labelled tracks	41
5.4.3	Show Track	41
5.4.4	Image Track.....	41

5.4.4.1	Image Window	44
5.4.5	Video Track.....	45
5.4.6	Audio Track.....	46
5.4.7	Web Tracks.....	48
5.4.7.1	Browser Commands.....	50
5.4.7.2	Full screen Integrated Browser Displays	51
5.4.8	Message Tracks.....	52
5.4.9	Show Track.....	54
5.4.10	Menu Track.....	54
6	Black Box Operation	58
6.1	Command Line Options.....	58
6.2	Specify a Profile	59
6.3	Specify a Home Directory.....	60
6.4	Using GPIO to Control Pi Presents.....	60
6.5	Disable Screen Blanking	61
6.6	Start Pi Presents when Power is applied to the Pi	61
6.7	Shutdown the Raspberry Pi from the GPIO	61
7	Exiting Pi Presents and Shutdown of the RPi	62
8	Controlling Shows.....	62
8.1	Controlling Track Movement in Individual Shows	62
8.2	Opening and Closing Shows and Show Control	63
8.3	Time Of Day Scheduler	64
8.4	Concurrent Shows.....	65
8.4.1	Control with Concurrent Shows	66
8.4.2	Limitations on Concurrency.....	66
9	Touchscreens and Soft Buttons	67
9.1	Controlling Shows with a Touchscreen.....	67
9.2	Click Areas.....	67
9.3	Soft Buttons	68
10	Run-Time Commands.....	68
11	Providing Dynamic Content in a Liveshow.....	69
12	Animation Control	70
13	Input/Output Drivers.....	71
13.1	Enabling an I/O Device	71
13.2	Configuring Inputs and Output Drivers	72
13.2.1	Configuring GPIO Pins.....	72
13.2.2	Configuring Touch/Click Areas	73
13.2.3	Configuring Keyboard Keys.....	74
14	Remote Control using OSC (very beta).....	74
14.1	OSC Messages in Pi Presents.....	75
14.2	Sending and Receiving OSC Messages	76
14.3	Configuring OSC	76
14.4	oscremote.py and oscmonitor.py	77
15	Track Plugins	78
16	Remote Management.....	80
16.1	Setting up for Remote Use.....	80
16.2	Using the Manager	81

16.3	Using the Web Editor	83
17	Hardware Requirements.....	83
18	Updating Pi Presents.....	84
19	Debugging, Statistics, Bug Reports and Feature Requests	84
19.1	Statistics Production	84
19.2	Debugging Profiles	85
20	Gotchas and Known Problems	86
21	Converting Version 1.2 to 1.3.....	88

Copyright Notice

This manual and the Pi Presents software are copyright Ken Thompson. For licence conditions see:

<https://github.com/KenT2/pipresents-gapless/blob/master/licence.md>

Raspberry Pi is a trademark of the Raspberry Pi Foundation

<http://www.raspberrypi.org>

1 Introduction

Pi Presents is a multi-media interactive display toolkit with animation control facilities. It was originally intended for Museums and Visitor Centres but has already found uses in hospitals, shops, schools, art installations, libraries and more.....

Until the Raspberry Pi arrived buying or constructing something even as simple as an interactive audio player was expensive. The Raspberry Pi with its combination of Linux, GPIO and a powerful GPU is ideal for black box multi-media applications; all it needed was a program to harness its power in a way that could be used by non-programmers.

There are a number of digital signage applications for the Raspberry Pi which are limited to slideshows and must take their media from the internet. Pi Presents is different; it is a toolkit which allows construction of many types of interactive presentation applications by using a simple to use editor running on a Pi, a Linux box or a Windows PC..

The Pi Presents toolkit supports [the](#) types of show commonly seen in museums; each of these can be configured to meet the your individual requirements.

Some uses:

- Animation or interpretation of exhibits by triggering a sound, video, or slideshow from a button, keyboard or other GPIO input.
- While playing media, GPIO outputs can be used to control external devices such as lights, animatronics etc.
- A multimedia slideshow for a visitor centre. Images, videos, audio tracks, and messages can be displayed. Repeats can be at intervals or at specified times of day.
- Slideshows to be interrupted by the visitor and a menu of further content presented.
- A show for kiosks where content can be initiated by pressing one of a number of buttons.
- Facilities to construct shows commonly seen on touch screens in museums.
- Construct a 'powerpoint' like multi-media presentation where progress through slides is manually controlled by buttons or keyboard.
- Basic digital signage with the ability upload image, video or audio tracks over a network for display in a repeating show.
- Multi-window displays, displaying several types of content on a single screen

- Construct a network of Pi Presents applications controlled through the Open Sound Control (OSC) protocol or control Pi Presents from remote computers, tablets or smartphones.
- [Manage Pi Presents and edit profiles remotely using any browser \(Local Area Network required\).](#)
- Foreign language support. All text shown to the audience is configurable.

Once set up Pi Presents is easy to use and does not require a network:

- Shows can be prepared using a simple to use editor on a Windows PC, the Pi, or another Linux box.
- Will operate on a Model A or A+ Raspberry Pi (but see hardware requirements).
- No need to modify the Pi's SD card after initial installation. All media and configuration options can be kept on a removable USB stick. Installing an updated application is as simple as inserting the USB stick.
- Video output can use HDMI or composite selected using the normal Pi setup procedure. As Pi Presents can be started and shutdown without a keyboard a monitor is not required.
- Audio track output is selectable between HDMI and analogue and can be output from left, right or stereo speakers on a per track basis. A number of tracks can be played simultaneously to different speakers each with presettable volume.

Pi Presents, out of the box, runs as a desktop application on the Raspberry Pi using the keyboard for control. However with a little bit of Linux magic, which is explained in this manual, it can be made to run as a black box application with control from GPIO or remotely via OSC. Black box features include:

- Disabling screen blanking and the mouse pointer.
- Full screen operation without window decorations
- Completely headless operation without a keyboard, mouse or buttons (except perhaps a shutdown button if you do not want to risk SD Card corruption)
- Black box control can be by buttons, PIR, or any source of digital inputs. It is straightforward to make your own controls.
- Mouse compatible touchscreens are supported if drivers are available for the Raspberry Pi. Alternatively a keyboard and mouse can be used. Pi Presents has been designed such that it is possible to add drivers written in Python for other forms of input and output such as RS232 and I2C.
- Automatic start up when power is applied to the Pi

- Safe shutdown without using keyboard or mouse
- Automatic opening and closing of shows and shutdown of the Pi at specified times of day. The time of day scheduler is programmable for different schedules on different days of the week, or month, or for special days in the year.
- Remote control of a Pi Presents application using Open Sound Control from another Pi Presents or any computer with a suitable OSC application.

1.1 Acknowledgements

Douglas Otwell for dbus control of omxplayer

<https://github.com/Douglas6/omxcontrol>

Davide Rosa for Remi – Remote Interface, a toolkit for writing browser based gui's in Python

<https://github.com/dddodomodossola/remi>

Johannes Baiter (jbaiter) for pyomxplayer based on Noah's pexpect

Numerous people on the Raspberry Pi forum and websites, particularly StackOverflow, who have unwittingly provided me with solutions to many technical problems and taught me Python.

ActiveState for a free Python systems for Windows

Bullets from <http://www.enterprise-dashboard.com/tag/red-green-yellow-alert/>

Icons from <http://www.fatcow.com/free-icons>

Buttons from <http://www.freewebsitebuttons.com>

The Raspberry Pi Foundation for the first affordable machine that plays video, audio and has GPIO.

Having been introduced to open source software through the Raspberry Pi I was amazed at the time and effort which so many people put in to produce software for others, so I thought it worthwhile to do the same with Pi Presents. I am glad I did because the feedback I have had from users and potential users has given me so many good ideas for extensions to Pi Presents. Needless to say I have been so busy producing Pi Presents that I have not had time to use it for my own intended projects, however one day I am sure I will (I now have two) ☺.

2 Installation

The installation instructions are in the README.md file. Pi Presents is aimed at the latest Foundation Raspbian release ([Jessie](#)) and MUST BE INSTALLED AND RUN FROM THE LXDE DESKTOP.

3 Try Some Examples

Download the examples from the `pipresents-gapless-examples` github repository as described in the README.md file.

N.B. Pi Presents must be run from the LXDE desktop (you must have executed `startx` either from the command line or set 'boot to desktop' in `raspi-config`) . From a terminal window opened in the `pipresents` directory type:

```
python pipresents.py -p pp_mediashow_1p3
```

You will see a continuous looping show in a small window. The small window is useful for development purposes; you can adjust its size by editing `pipresents.py`, around line 45. You can expand the display to nearly full screen by using the appropriate window decoration. If however you want to see the show truly full screen without borders then use the `-f` command option:

```
python pipresents.py -p pp_mediashow_1p3 -f
```

The mediashow example and all the other examples are designed for a 1920*1080 display. If you have a smaller resolution display some of the text and part of the larger images will disappear off the edges of the display. Pi Presents does not automatically adjust for screen resolution; you will need to use the Pi Presents editor to adjust text font sizes, and to resize images external to Pi Presents.

The `pipresents-gapless-examples` repository has examples of how to use Pi Presents.

You can use the bash script `examples.sh` to execute the examples, just type `./examples.sh` from a terminal window open in the `/pipresents` directory or copy the `examples.desktop` file from `/pipresents` to your desktop. You will need to make `examples.sh` executable.

The examples use the following keys:

- Up or Down Cursor- move through a menu.
- Up or Down Cursor - Move through a mediashow show. Can also be used to skip to the next or previous track in an automatic mediashow. The movement circles around at the start and end of a show.
- Return –Play the selected menu entry or a child track.
- Escape – Stops the current track/show in an intuitive manner.
- Spacebar - in an image, video, or audio track, toggle pause.
- CTRL-BREAK always exits Pi Presents. (Duplicated by the Close window decoration or Alt+F4)

All examples use a selection of media tracks which are in `/home/pi/pp_home/media`. The profiles are in `/home/pi/pp_home/pp_profiles`. In each profile the

pp_showlist.json file specifies the look and feel of each show; other files specify the media to use. The files can be viewed in a text editor but it is much better to edit them using Pi Present's own editor described in Section 4.

Other optional files in the profile, screen.cfg, gpio.cfg, keys.cfg, osc.cfg and schedule.json customise other aspects of Pi Presents.

How the profiles work is explained in detailed later so do not worry if you do not understand them, just enjoy what Pi Presents is capable of.

The examples:

- **pp_mediashow_1p3**
The profile you have just run. The show will start immediately, progress automatically, and then repeat. Use the Up and Down cursor keys to skip tracks. The example demonstrates the types of track and additional information that can be added to the primary content.
- **pp_menu_1p3**
The example demonstrates all the menu formats available in Pi Presents. The second level menus, and any show which runs from another show are called subshows. Use the Up and Down cursor keys to traverse the menu then press Return to Start a track or sub-menu, and Escape to terminate it.
- **pp_exhibit_1p3**
This demonstrates how to trigger a mediashow from a PIR. When started, the screen will be blank. Triggering the PIR input (opening a contact connected to 0 volts on P1-11) or pressing a button (closing a contact connected to 0 volts on P1-18) will start a one shot mediashow and then wait for the trigger in order to repeat the show.

The example uses the gpio.cfg file in the profile which has pins P1-18 and P1-11 bound to the symbolic name PIR. This name is then associated with Trigger for Start in the mediashow.

The example also uses the Return key as a trigger input so you can see the effect without using the GPIO pins. The keys.cfg file in the profile overrides the default keys.cfg and binds the Return key to the symbolic name PIR.

BEWARE: Do not use the GPIO pins until you have read Section [6.4](#)

- **pp_openclose_1p3** (New in Version 1.3)
A single shot mediashow with a different method of triggering. Useful if you want a track to play when opening a box or door, and stop when it is closed.
- **pp_onerandom_1p3** (New in Version 1.3)
When triggered plays a single track randomly chosen from the medialist .
- **pp_magicpicture_1p3** (New in Version 1.3)
A still picture of a person is seen in a picture frame. When a button is pressed the picture comes to life as a video is played. The effect is achieved by pausing the video on the last frame so when the show is first started the track must be played once (the aim is that this is done by staff at the beginning of the day). I don't have a video of a person so Suits will have to do!

- **pp_interactive_1p3**
This combines a mediashow and a menu. The mediashow is run continuously but every track has a hint showing that a menu can be triggered by the Return key. Pressing Escape from the menu returns to the mediashow. A track or show which is accessible from any track in a mediashow is called a child.
- **pp_presentation_1p3**
The mediashow is configured as a manually controlled presentation. Use the Up and Down cursor keys to traverse through it. Pi Presents cannot use Powerpoint presentations but Powerpoint presentations can be saved as jpeg images.
- **pp_liveshow_1p3**
While the Liveshow is running place some video files, audio files, or images into the directory /home/pi/pp_home/pp_live_tracks and watch them miraculously appear. A liveshow is a mediashow without pre-defined media.
- **pp_radiobuttonshow_1p3**
Think of the navigation method in a Radiobuttonshow as being like a car radio channel changer. Press keys 1-4 or GPIO buttons to play a track and Escape to return to the main screen.

Controls defined in the Radiobuttonshow profile allow for both keyboard keys and buttons to be used to select a track. A gpio.cfg file is included in the profile to bind four GPIO pins to entries and one to the Stop Operation.

- **pp_hyperlinkshow_1p3**
The Hyperlinkshow works something like a set of web pages with links between them and Back and Home buttons. All of the 'story telling' applications in museums appear to use this navigation technique.

The show is aimed at touchscreens. The on screen buttons are touch sensitive areas which, for testing purposes, are also sensitive to mouse clicks. The file screen.cfg in the profile defines the buttons and the symbolic names of their input events. If using 'Soft Keys' the show could be controlled by GPIO pins by binding pins to the symbolic names in a gpio.cfg file.

- **pp_audio_1p3**
This mediashow demonstrates the capabilities of the audioplayer. You will need speakers connected to hdmi and analogue ports to fully appreciate it. It plays tracks to different speakers; it also demonstrates the use of Run-Time controls to control volume from keyboard (8 and 9) or GPIO pins. For this reason it has keys.cfg and gpio.cfg files in the profile,
- **pp_web_1p3**
A demonstration of the web browser capabilities. Ideally an internet connection is required. Because the integration between Pi Presents and the browser is not good, it will take a long time for the browser to open and, if you need to interact with Pi Presents when the browser is displayed you will need to click on its window first.
- **pp_concurrent_1p3**

This application demonstrates concurrent show playing capabilities. There are two mediashows running simultaneously. The mediashow showing images must be controlled from the keyboard while the concurrent mediashow containing audio tracks has its controls disabled and is played continuously.

- **pp_multiwindow_1p3** (New in Version 1.3)
This application demonstrates concurrent show playing capabilities where many shows display content. There are 4 mediashows and a menu running simultaneously. The menu must be controlled from the keyboard while the concurrent mediashows have their controls disabled and are played continuously. The number of concurrent shows is limited only by the power of the Pi. In the example:
 - background - provides the static background
 - audio - plays the background audio
 - text - is a changing text only mediashow
 - slideshow – multimedia automatic slideshow
 - menu – interactive menu

The show is set up for remote control by OSC, see Section 14.4

- **pp_artmediashow_1p3**(New in Version 1.3)
A mediashow which features gapless transitions without freezing the picture at the end of video tracks. Has limited features but could be useful for 'artistic' applications.
- **pp_artliveshow_1p3**(New in Version 1.3)
A liveshow which features gapless transitions without freezing the picture at the end of video tracks. Has limited features but could be useful for 'artistic' applications.
- **pp_subshow_1p3**
Demonstrates how to use subshows to segment a larger mediashow.
- **pp_timeofday_1p3** (Greatly enhanced from Version 1.2)
Opens and closes two mediashows at different times of day and then exits Pi Presents. This is all controlled by the schedule.json in the profile. To make this demonstration work whatever the day and time you try it the date and time is simulated by setting "simulated-time" to "yes" in schedule.json. Make a backup copy of schedule.json and then alter the date and time to see the effects. Note the morning opening only on Xmas day, just like English pubs!
- **pp_plugin_1p3** (Plugin API has been modified for Version 1.3)
Demonstrates the operation of track plugins. Track plugins are Python modules with a defined API which allow you to produce dynamic displays particularly clocks and from data scraped from the web. Before using the liveshow copy the .cfg files from the /media directory to the /pp_live_tracks directory.
- **pp_showcontrol_1p3**
Demonstrates the use of Show Control to start and stop one concurrent show from another. The show is set up to control another Pi by OSC, see Section 14.4
- **pp_clickarea_1p3**

Demonstrates use of click areas in each type of show and the use of Soft Buttons. In the Radiobuttonshow click areas are selectively disabled by individual tracks. Soft Buttons has a keys.cfg file which allows keys 1 and 2 to select the two tracks. When deployed properly there would be two GPIO buttons mounted adjacent to the left edge of the screen and a gpio.cfg file to associate the buttons with the symbolic names.

- `pp_animate_1p3` (Animation commands have changed for version 1.3)
Demonstrates the use of Animation Control by setting P1-11 to On at the start of a track and to Off at the end. To achieve this it has a gpio.cfg file in the profile. BEWARE: Ensure P1-11 can safely be used as an output before running this.
- `pp_shutdown_1p3`
A mediashow that demonstrates how to use Show Control to shut down the Pi. It displays a message track and then when the Down Cursor is pressed starts a track that shuts down the Pi immediately.

3.1 Terminating Pi Presents

You can terminate Pi presents by pressing Ctrl+Break or Alt+F4, or clicking on the 'Close' icon exits Pi Presents. Terminate is aimed at aborting Pi Presents even in abnormal situations. For other ways of closing Pi Presents or shutting down the RPi see Section 7

Some keyboards do not have a BREAK key; it is possible to configure another key to exit Pi Presents by editing the keys.cfg file to bind another key to the pp-terminate symbolic name as described in Section 13.2.3

4 The Pi Presents Profile Editor

Profiles can be created and edited using the Pi Presents editor.

```
python pp_editor.py
```

When using the editor you will need to supply show references, track references and file names. It is advisable not to create names beginning with pp- or pp_ to avoid clashes with names used by Pi Presents.

4.1 Using the Editor

4.1.1 The Displays

Select one of the example profiles using the Profile>Open menu. GOTCHA: On a Linux system remember to open the selected profile directory before pressing OK.

- The top left panel displays the shows in the profile

- The bottom left panel shows the medialists in the profile, click on one of the entries to select it.
- The right-hand panel shows the tracks in the selected medialist.

The selected entries are shown in red. Click the 'Edit Show' button adjacent to the left-hand panel to edit the selected show and the 'Edit' button adjacent to the right-hand panel to edit the selected track.

Edits are saved to disc when OK is pressed, use Cancel if you want to exit without saving.

You can run pipresents.py and pp_editor.py concurrently from two terminal windows so the effect of any edits can quickly be seen. You will need to restart pipresents.py to see the effect of the updates made by the editor.

If you are going to edit the examples then first make a copy of them by using the File Manager to access

```
/home/pi/pp_home/pp_profiles
```

and to copy a profile directory.

4.1.2 Profile Menu

Profile>Open - displays a directory viewer to select a profile for editing. The profiles displayed are those in the home + offset directory defined in the Options>Edit menu.

Profile>Validate - Validate the Profile. Font and Colour fields are currently not validated and if you edit the .json files with a text editor you are on your own!

Profile>New from Template - lists templates for all types of show. The templates have example tracks so they will run un-edited but some will be missing configuration files. Examples of these can be copied from:

```
/home/pi/pipresents/pp_config
```

and

```
/home/pi/pipresents/pp_resources/pp_templates
```

4.1.3 Show Menu

Show>Add - Add a new show, and a medialist with the same name. The show type of an existing show cannot be edited.

Show>Copy To - Copy the show, creating an empty medialist for the show

Show> Edit - Duplicates the Edit Show button

Show>Delete - Deletes the show from the profile, no going back!

4.1.4 Medialist Menu

Medialist>Add - Add a new medialist. The .json extension is added if not included.

Medialist>Copy To – Copy the medialist

Medialist>Delete - Deletes the medialist from the profile, no going back!

4.1.5 Track Menu

Track>Add from File

Adds one or more track entries containing media files to the end of the medialist. These may be images, videos or audio tracks. The editor automatically calculates the file location and track type. If either of these is unacceptable or the media file extension is rejected then use Track>New to add a blank track of the required type.

The list of extensions that are used to select track type is in the first few lines of the pp_definitions.py source. Please raise a bug report if these are inadequate.

Track>Add from Dir

Similar to Add from File, except that all eligible tracks in the directory are added to the medialist.

The list of eligible extensions that is used to select tracks and their types is in the first few lines of the pp_definitions.py source. Please raise a bug report if these are inadequate.

Track>New

Adds a blank track of the selected type to the end of the medialist.

Track>Edit

Duplicates the Edit Button

Track>Delete

Deletes the track from the medialist, no going back!

4.1.6 OSC Menu

OSC>Create OSC Configuration

Creates an empty osc.cfg file in the profile

OSC>Edit OSC Configuration

Edits osc.cfg. You will be asked to select:

- master - this Pi Presents is controlling one other computer. Multi-clients to follow.
- slave - this Pi Presents is controlled by another computer
- master+slave – To follow

OSC>Delete OSC Configuration

Deletes osc.cfg by renaming it to osc.cfg.bak

4.1.7 Tools Menu

Tools>Update All

Update the version of all profiles in the current home + offset, see Section 18. If you wish to see which profiles are being updated then run the editor from a terminal window using the -d option.

4.1.8 Options Menu

Options>Edit

Edit the editor options:

- Pi Presents Home Directory**
 This is an important option which must be set to the directory in which any profile directories and relative media is to be assembled. Out of the box it is set to /home/pi/pp_home. If you choose an alternative setting for this then ensure that the directory /pp_profiles is created inside the chosen /pp_home.
- Initial Media Directory**
 This is just a helper setting to define where 'Add Track' and 'Add from Dir' start their browsing.
- Offset for current profiles**
 This advanced option allows profiles to be separated into directories under /pp_home/pp_profiles and readily accessed so an offset of /1p3_examples defaults 'open' to /pp_home/pp_profiles/1p3_examples

4.1.9 Editor Command Line Options

-d --debug	Run time errors give alerts and are reported to the terminal window if open. Turning on debugging will in addition provide a trace of the operation of the editor in the terminal window. Errors and the trace are also recorded in the /pipresents/pp_logs/pp_log.log file.
--forceupdate	Update a profile even if the editor and profile major issues (e.g. 1.2) are the same (for beta testers only)

4.2 Making Profiles

Application data is kept in a directory called /pp_home. /pp_home must contain a directory /pp_profiles in which profiles are stored.

A profile is a directory which contains the information needed to configure a single application of Pi Presents. It must contain a pp_showlist.json file and one or more medialist (.json) files. The pp_showlist.json file contains a number of sections, a 'start' show section and a number of sections each defining a user generated show.

Each section in the pp_showlist.json file defines the look and feel of a show and how they link together. Medialist (.json) files in a profile define the content each show and some per track look and feel information.

For portability reasons all media is best kept under the directory /pp_home, but this is not essential. Media can be stored anywhere under the /pp_home directory using sub-directories if necessary.

The default location of /pp_home is /home/pi; this will need to be overridden by command line options if you are using a USB stick.

A profile may optionally contain other configuration files such as gpio.cfg and schedule.json. A readme.txt file can be included to allow the profile to be documented.

4.2.1 Making Profiles Portable

Profiles and their media can be moved between different drives without breaking the references between shows and their media tracks. This is necessary if a profile is prepared on a Pi on a SD card and moved to a USB Stick for operational use; also if the profiles are to be prepared on a Windows machine.

To achieve portability media tracks should be stored under the /pp_home directory, in sub-directories if desired. If tracks are stored in this location before the profile is prepared then the Profile Editor will automatically compute the appropriate track reference:

- If the file is below the home directory (...../pp_home), as defined in the Options>Edit menu then the file reference will look like '+/track_to_play.mp4'
- Otherwise the reference will look like '/home/pi/mymedia/track_to_play.mp4'; an absolute reference.

Absolute references do have their uses, for example in specifying internet url's e.g. http://www.mysite.com/track_to_play.mp4

4.2.2 Using the SD Card for Profiles

This is very useful for developing applications on a Pi. To use the SD card for developing and running profiles:

- Create a directory /pp_home in the User Pi's home directory and inside that create a directory called pp_profiles. You can also create a directory in /pp_home called say, media, to hold media files.
- In the Editor Options set the Pi Presents Data Home directory to /home/pi/pp_home and the Initial Directory for Media to /home/pi/pp_home/media
- Copy media files to the /media directory
- Using the editor create a new profile, call it myprofile, and edit it.
- To run the profile type:

```
python pipresents.py -p myprofile
```

from a terminal window open in the pipresents directory.

When running Pi Presents from a desktop shortcut, or from an autostart file it is best to specify the full path of pipresents and also the full path of the data home directory. e.g.

```
python /home/pi/pipresents/pipresents.py -o /home/pi -p myprofile
```

4.2.3 Using the Editor on a Windows PC

Use on a PC requires a Python system to be installed. I use the free Activestate Python <http://www.activestate.com/activepython>. There is a good installation guide here <http://www.richarddooling.com/index.php/2006/03/14/python-on-xp-7-minutes-to-hello-world/>.

To install the editor on a Windows PC download the Zip File from <https://github.com/KenT2/pipresents-gapless> (just click on the Zip button). Extract the contents of the folder pipresents-next-master to a folder called pipresents.

Either download pipresents-gapless-examples and extract the folder to your top level folder (c:\users\ken on my Windows 7 machine) or create a 'pp_home' folder here with a folder 'pp_profiles' inside.

Open pp_editor.py in the Idle editor and run it. Alternatively you can double click on pp_editor.py or create a desktop shortcut to pp_editor.py; you may need associate '.py' files with python.exe.

Use of the editor is the same as on the Pi, it is obviously important that all track references are to media files stored under pp_home.

Profiles created on a PC can be transferred to the Pi using a USB Stick, alternatively you can ftp them across a network using a PC ftp program such as Filezilla or WinSCP.

4.2.4 Using a USB Stick for Profiles

To run profiles from a USB stick copy /pp_home to the top level of a USB stick from its location either on a Pi or on the PC.

Insert the USB stick into a Pi and run Pi Presents with the -o option set to /media/STICKNAME. STICKNAME is the drive name. Windows seems to convert lowercase drive names to upper case.

The editor will probably run on any Linux PC. If you are using such a beast you probably do not need your hand held when transferring profiles.

4.2.5 Developing Profiles using a Monitor with Different Pixel Dimensions

The `-s` `--screensize` command line option assists with developing applications on a host monitor that has different pixel dimensions to the target monitor. Use of this option will cause Pi Presents to use the provided dimensions instead of the host screen dimensions that it obtains from Raspbian.

- All layout operations such as centring and warping images will use the provided target dimensions.
- If fullscreen mode is not in use a yellow rectangle will show the dimensions of the target screen.
- If fullscreen mode is in use then the display will be the size of the target monitor placed at the top left of the host monitor. If the target dimensions are greater than the host dimension then parts of the display will disappear.

5 The Components of Pi Presents

5.1 Introduction

Shows and Tracks

The Pi Presents toolkit has two building blocks - shows and tracks.

A show plays tracks; the list of tracks to be played is contained in a medialog which is associated with the show, think of the medialog as an enhanced playlist.

The toolkit currently has seven types of show each optimised for a different purpose:

- Mediashow - plays a sequence of tracks, usually automatically, but progress can be manually controlled. Transitions between tracks are gapless but this is achieved by freezing videos and images at the end of the track while the next track is loaded.
- Artmediashow – plays a sequence of tracks as in Mediashow with full gapless capability. The next track is loaded while the previous track is playing hence the tracks do not freeze at their end. The Artmediashow has no triggering capability and does not support Child tracks or Subshows
- Hyperlinkshow - Implements the touchscreen functionality that you see in many museums.
- Radiobuttonshow - A simple kiosk show showing a navigation screen - press one of many buttons to play a track.
- Menu - Similar to a Radiobuttonshow but the track selection is by traversing a menu using 'cursor' keys or a single button.
- Liveshow - A Mediashow like show with dynamic remotely sourced content.
- Artliveshow – uses remote content as in Liveshow and has full gapless capability. The next track is loaded while the previous track is playing hence

the tracks do not freeze at their end. The Artliveshow has no triggering capability and does not support Child tracks or Subshows

The toolkit currently has 'players' for five types of track each playing a different type of media. All 'players' allow the primary media to be displayed in a window with an optional background of plain colours, images, and text annotations. Players allow animation to be controlled, track plugins to be written in Python, and for other concurrent shows to be opened and closed.

- video - plays videos using omxplayer
- audio - plays audio tracks using mplayer
- image - displays images in many different formats (.jpg etc.) (Python Imaging Library)
- message - a quick way to display text.
- Web – runs an embedded web browser (uzbl)

Subshows and Child Tracks

In Pi Presents a show can be a track of its parent show; this is called a subshow. Subshows have a number of uses which include:

- Dividing a long mediashow into segments. Each segment is a show and a parent mediashow contains the segment shows in its medialist.
- A menu or radiobuttonshow might have mediashows as entries rather than single tracks
- Multi-level menus

Subshows can be nested to any depth; the limit is probably the confusion that it will cause the audience when using them. Subshows are tracks so appear in the medialist.

Child tracks are tracks used in a special way by mediashows and liveshows. If a child track is specified in the profile it can be initiated while running any track in the parent show returning to the next track in the parent when finished. Child tracks, like any track can be a show and are sometimes referred to as Child Shows..

The Egg Timer

In all shows transitions between tracks are gapless (no black gaps between tracks).. This is achieved by freezing videos and images at the end of the track while the next track is loaded. For manual operation this means there is a delay of up to 2 seconds after a control is pressed. The 'Egg Timer' can be used cover this delay by displaying text while the loading takes place. The text appears above the track display so its content, format, and position is configurable in the show profile.

Concurrent Shows

Pi Presents can run two or more shows concurrently, see Section 8.4. The concurrent shows appear to run in parallel. This has many uses which include:

- Providing a background audio track to a slideshow.
Use two mediashows, one with the slideshow and the other with the audio tracks. The latter will need the controls disabled if there is customer interaction with the former.
- Dividing the screen into areas (Show Canvases) each showing a different Show. Perhaps an automatic slideshow in one area, time of day provided by a show with a track plugin in another, and a user controlled menu of pictures in a third. Controls will need to be disabled for all but one of the shows. For more advanced applications Pi Presents can be configured so that each concurrent show and its subshows has their own sets of controls.
- Being really thrifty and doing two completely different tasks with the same Pi, perhaps a slideshow in the Reception and a dummy talking in a museum exhibit triggered by a PIR

Each concurrent show can have subshows.

5.2 Shows

Shows have fields which control the sequencing and look of the show. They also have a number of fields which provide default values for all the tracks in the show e.g. Duration, Transition, OMXPlayer Audio; tracks will use the values in the show if their corresponding fields are left blank.

All shows have the following fields:

- Title - Text displayed in the editor and in the entries of a menu show
- Show Reference - A label which allows other shows to reference this show. Can be any alpha-numeric string without spaces.
- Medialist - this is the name of a file which appears in the medialist panel. It must have the extension .json. Every show must have a medialist to define the tracks in the show. The same medialist can be used by more than one show.

5.2.1 Mediashow and Artmediashow

Think of a mediashow as a slideshow that can play tracks of different types - videos, audio tracks, images, and even animation control sequences.

Mediashows have a number of fields to define the control of the show, e.g. Trigger for Start, Repeat/Single and Sequence.

A track can be associated with a mediashow such that the track is accessible from any track in the show. These are termed 'child tracks'. The Child Track parameter specifies the track reference of the child track, which may be a track or a show. Associated with the use of a Child Track is 'Hint Text' that is displayed only when the Child Track field is not blank.

The Repeat/Single field control how mediashows run:

- repeat - the mediashow repeats until it is stopped.
- single-run - the mediashow runs once and then exits. Single run is primarily for use in subshows and with Show Control

Other fields allow mediashows to be customised for many applications. The table below shows the Trigger and other field settings for common uses of mediashows and liveshows.

Task	Repeat	Trigger for Start	Trigger for End	Other Fields (where non-zero)
Continuous show	repeat	start	none	
Continuous show that repeats the medalist until a period of time has elapsed, then it ends.	repeat	start	none	Show Timeout = h:m:s
Continuous show that repeats the medalist at intervals	repeat	start	none	Repeat Interval = h:m:s
Run a medalist triggered by an input (was single shot)	repeat	input	none	
Run a medalist triggered by an input. Inhibit re-triggering for a period of time after the medalist completes.	repeat	input	none	Repeat Interval = h:m:s
Play a track triggered by an input. If sequence = shuffle a random track will be played each time.	repeat	input		Track Count Limit = tracks
Start and stop playing a medalist when an input is pressed and released	repeat	input	input	In gpio.cfg use rising edge and falling edge of a pin for the two triggers.
'Magic Picture'. Trigger a video and freeze it at its end while waiting for the repeat trigger	repeat	input-persist	none	
Use a mediashow as a subshow or as an item of a menu, radiobuttonshow etc.	single-run	start	none	

Mediashows and Liveshows can also be controlled by the user using Commands such as Up, Down and Stop. The association of these with the Symbolic names of Input Events is in the Controls field.

Artmediashow

Artmediashows provide full gapless playback; there is no gap between tracks and tracks do not freeze at their end to cover loading of the next track. Artmediashows have some limitations and do not have the fields and commands which are shown in brackets in the Table below:

- Subshows
- Child Tracks
- Start, Next and End Triggers.
- Web Tracks

5.2.1.1 Controls and Commands

The first means of controlling mediashows and liveshows is by using their Commands. Out of the box these Commands are bound to a set of symbolic names and these in turn are bound to a set of keyboard keys. In addition control of Pi Presents by GPO with a useful set of bindings can be enabled easily by copying a prepared gpio.cfg file to a profile from /pipresents/pp_resources/pp_templates:

The out of the box settings are:

Command	'Out of the box'			Effect
	Symbolic Name	Key	GPIO Pin.	
up	pp-up	Cursor Up		Previous track in mediashow or liveshow, or up for menu
down	pp-down	Cursor Down		Next track in mediashow or liveshow, or down for menu
play	pp-play	Return		Start an entry in a menu, or start a child show.
pause	pp-pause	Spacebar		toggle pause for tracks that support pause.
stop	pp-stop	Escape		Stop playing a track and, if a show is in its quiescent state and is not a top level show, return to the parent show. Quiescent state: <ul style="list-style-type: none"> • Liveshow/Mediashow – at all times • Menu – displaying menu. If it is in its quiescent state and a top level show then stop closes a single-run show but restarts a repeating show.
null	<not applicable>	<not applicable>	<not applicable>	inhibits the control with the same symbolic name that has been defined in the show.
exit	pp-exit	<not bound>	<not bound>	Stop playing a track and exit to the parent show or, if a top level show, close the show.
omx-*	<not bound>	<not bound>	<not bound>	execute a runtime commandl for the video player (Section 10)
mplay-*	<not bound>	<not bound>	<not bound>	execute a runtime command for the audio player (Section 10)

(uzbl-*)	<not bound>	<not bound>	<not bound>	execute a runtime command for the web browser (Section 10)
----------	-------------	-------------	-------------	--

The 'out of the box' key for each operation is set in /pipresents/pp_config/keys.cfg and the GPIO pin in gpio.cfg. The bindings can be modified as described in Section 13.2. The binding of the Command to the symbolic name is specified in the Controls field of the show. It is unlikely you will want to modify these but you may wish to override them for an individual show as described below.

Remember that if more than one show is running concurrently the input event is passed to and potentially executed by each concurrent show. This may be undesirable, for example if a mediashow of audio tracks is running as a background to a manually advanced slideshow; we do not want the up and down operations to change audio track, so Disable Controls is used with the audio show.

If Disable Controls = yes for the show then all Commands are disabled for the show and it must run automatically or be controlled by triggers as described in the next section. For more advanced situations individual Commands can be deleted or bound to alternative symbolic names for a specific show by using the Controls field of a show.

Commands can be placed in the Controls field of the tracks in addition to the Controls field of the mediashow. The controls in a track are merged with and override those in the show. This is generally not required but could be used where, for example it is required that the audience watches the whole of a video. In this case using the null commands would inhibit up/down/stop for the track

5.2.1.2 Triggers

Mediashows and liveshows have triggering facilities. The facilities are designed for use when the show is played without the full user interaction provided by their Commands

Immediately after a mediashow is run, or it repeats, the show uses the Trigger For Start field to decide what to do next, After each track Trigger For Next can be employed to control movement to the next track. Trigger for End determines when the mediashow finishes

Triggers can be applied to sub-shows and child shows. Triggers are not inhibited by 'Disable Controls'.

5.2.1.2.1 Trigger for Start

Trigger For Start can take the following values:

- start
The mediashow continues to the first track.
- input /input-persist

The mediashow waits for an input event before running the first track of the show. The symbolic name of the input event must be included in the Trigger for Start Parameter field.

Text to display while waiting for a Star Trigger is in the Notices Tab of a mediashow or liveshow.

5.2.1.2.2 *Trigger for End*

Trigger for End causes the show to exit (single-run) or repeat (repeat). It can take the following values:

- none
The end trigger is not operative the show will exit or repeat at the end of the medialog/live tracks directory.
- input – the end trigger is generated when an input with a symbolic name matches that in the Trigger for End Parameter field.

5.2.1.2.3 *Trigger for Next*

The Trigger For Next field allows individual tracks in a mediashow to be triggered by an input event:

- None - The trigger is not operative the show will continue to the next track at the end of the previous.
- Input - The show moves to the next track when it is triggered by the input event having the symbolic name specified in Trigger for Next Parameter field. The track duration can be used to set a longstop on the trigger, or if set to 0 to allow forward movement by trigger only.

5.2.1.3 **Fields**

Fields for Mediashow and Liveshow. Fields not used for Artmediashow and Artliveshow in brackets

Field	Examples	Values
Show Tab		
Type	(art)mediashow	Cannot be edited
Title	My First Show	Text describing the show. Displayed in the editor and on menus
Show Reference	show1	A 'label' by which the show is referenced by other shows. Any text without spaces
Medialog	show1.json	Filename of the medialog file containing the tracks for the mediashow. By default it is the same as the Show Reference but can be changed.

Field	Examples	Values
(Trigger For Start)	start	How the media show proceeds after it is started and at the beginning of a repeat: <ul style="list-style-type: none"> • start - continue without waiting • input - wait for an input event. • input-persist – wait for an input event. Input-persist displays the immediately previous track while waiting (videos are frozen at their end)
(Trigger for Start Parameter)	PIR	If Trigger For Start is 'input', or 'input-persist': A symbolic name that is bound to an input event as described in Section 13.2
(Trigger for Next)	continue	How to trigger the next track <ul style="list-style-type: none"> • input - respond an input event • continue – continue without waiting Input can be used in one of two ways: If the duration of a track is zero the mediashow will move forward only when the trigger is received. If the duration of a track is non-zero then the mediashow will move forward automatically and also when the trigger is received.
(Trigger for Next Parameter)		If Trigger For Next is 'input': A symbolic name that is bound to an input event as described in Section 13.2
Sequence		sequence of tracks: <ul style="list-style-type: none"> • ordered - played in the order of the medialist • shuffle - play tracks in a random order
Repeat/Single	repeat	How the media show is repeated: <ul style="list-style-type: none"> • repeat – Wait for Trigger for Start. Tracks in the medialist are then played once and then the medialist repeats by waiting for Trigger for Start. • single-run - Waits for Trigger for Start. Tracks are then played once and then the show closes. Note: There is no natural end to a shuffled set of tracks.
(Trigger For End)	none	How the end of a show can be triggered. See the table above for more detail. <ul style="list-style-type: none"> • none - the end trigger is not operative • input - Ends after an input event as described in Section 13.2 • duration - end the show after a period of time.

Field	Examples	Values
(Trigger for End Parameter)	5:00	If Trigger For End =: <ul style="list-style-type: none"> input - A symbolic name that is bound to an input event as described in Section 13.2
(Track Count Limit)	1	If non-zero the show closes or repeats after Count tracks have been played. Intended fo use where the Track Count Limit is less than the number of anonymous tracks in the medialist.
(Repeat Interval)	30	h:m:s The show repeats/ends at Repeat Interval or at the end of the medialist, whichever is last. If the medialist ends before Repeat Interval then a blank screen is displayed. Interval = 0 always repeats/ends when medialist is complete. Repeat Interval=0 does not work for shuffle as a shuffled medialist never finishes. For shuffle and Repeat Interval>0 the show always continues until Repeat Interval as a shuffled medialist does not finish
(Show Timeout)	1:30:20	h:m:s If non-zero end the show after the specified period if nothing else has ended it.
Show Canvas	100 100 400 600	The top left and bottom right coordinates of the show canvas. Show canvas helps the user divide the screen when displaying more than one concurrent show. If the field is not blank the Show Canvas dimensions controls the placing and size of images and videos taking the place of the dimensions of the screen. The Show Canvas determines the origin of the images (top left) however it is not a window and does not crop the image or video so, if the bottom right co-ordinates of the image are larger than the Show Canvas it can overlap other show canvases.
Child Track Tab		
(Child Track)	child-track	Non-blank to enable a Child Track and to specify the track reference of the track.
(Hint Text)	Press Play to..	A single line of text
(Hint Font)	Helvetica 30 bold	Use the Font Chooser to select a font, or type one in: See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm

Field	Examples	Values
(Hint Colour)	white	Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
(Hint x)	500	Position of left of text
(Hint y)	900	Position of top of text
Eggtimer Tab		Eggtimer text is displayed when the next track is being loaded
EggTimer Text	Loading....	A single line of text
Eggtimer Text Font	Helvetica 30 bold	Use the Font Chooser to select a font, or type one in: See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Eggtimer Text Colour	white	Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Eggtimer Text x Position	100	distance of the start of the text from the left of the screen (pixels)
Eggtimer Text y Position	100	distance of the top of the text from the top of the screen (pixels)
Notices Tab		
(Trigger Wait Text)	Waiting...	Text to show when Waiting for Trigger
Live Tracks Empty Text		Text to show when a liveshow has no tracks to display
Notice Text x Position		distance of the start of the text from the left of the screen (pixels)
Notice Text y position		distance of the top of the text from the top of the screen (pixels)
Notice Text Colour		Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Notice Text Font		Use the Font Chooser to select a font, or type one in: See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Show Background and Text Tab		

Field	Examples	Values
Background Image	+ /media/image .jpg	If not blank the file name of an image which is used as the background for any tracks in the show. The image is warped to fit the Show Canvas. If you do not want background images to be warped then edit them in advance to be the size of the Show Canvas.
Background Colour		Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. See http://effbot.org/tkinterbook/tkinter-widget-styling.htm The background colour is set to black when Pi Presents starts. The value in this field is used by a track in the show if the value in the track's field is blank. If the resulting Background Colour is blank the background colour is not changed.
Show Text	Pictures from my holiday	Show text is overlayed on all tracks in the show. If not blank the text to be displayed.
Show Text Font	Helvetica 30 bold	Use the Font Chooser to select a font, or type one in: See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Show Text Colour	white	Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Show Text x Position	100	distance of the start of the text from the left of the screen (pixels)
Show Text y Position	100	distance of the top of the text from the top of the screen (pixels)
Track Defaults Tab		Tracks played in the show need some configuration if not supplied in the individual tracks
Duration	10	seconds. How long a track having no natural end is displayed. A value of 0 displays continuously.
Transition	cut	cut. Type of transition between tracks. Currently not used.
Image Rotation	90	Rotates the image
Image Window	original 10 100 fit fit 1010 40 40 fit ANTIALIAS	For any image track in the show a viewport in which to show the image. See section 5.4.4.1

Field	Examples	Values
Video Audio	hdmi	hdmi/local/both <blank>. Sound output channel for any video track played by omxplayer from the show. If blank then the channel is set automatically by the presence of a hdmi monitor. (can be forced in raspi-config)
Video Audio Volume	0	Volume of audio for the video track (-60 -> 0 dB).
Video Window	original warp warp 10 100 200 700	For any video track in the show a viewport in which to show the video. <ul style="list-style-type: none"> • original – use OMXPlayer default behaviour • warp – scale to the size of the screen without maintaining aspect ration • warp followed by two x,y, pairs (top left, bottom right) – the viewport window. The video is scaled to this size without preserving the aspect ratio.
Freeze at End	yes	yes/no If yes the video will be frozen near the last frame until the next track is ready to be displayed.
Video Player Options		Other options for omxplayer (care required to avoid have a nice day!)
Audio Player Audio	local	hdmi/local. Sound output channel for any audio track played by MPlayer from the show.
Audio Player Speaker	stereo	left/right/stereo. Speaker for any audio track played by MPlayer in the show.
Audio Player Volume	0	Volume of audio track (-60 -> 0dB) played by MPlayer in the show.
Audio Player Options		Other options for MPlayer (care required to avoid rejection!)
(Web Window)	warp warp 10 100 200 700	For any web track in the show a viewport in which to show the web page. <ul style="list-style-type: none"> • warp - scale to the size of the screen without maintaining aspect ration • warp followed by two x,y, pairs (top left, bottom right) - the viewport window. The web page is scaled to this size without preserving the aspect ratio.
Controls Tab		

Field	Examples	Values
Controls	pp-play play pp-up up	Defines the bindings between symbolic names of input events and commands See Table above and Section 6.4 One binding per line each with the format: symbolic-name command Bindings are NOT inherited by subshows.
Disable Controls	no	yes/no If 'yes' Commands (e.g. play, pause, up, down, stop) are disabled. This is a quick way to inhibit all controls for a concurrent show. For more selective control use the Controls field

5.2.2 Menu

A menu show uses the Title and Thumbnail fields of tracks and of shows to automatically generate a menu on the screen. The Up and Down Commands can then be used to scroll the menu and the Play Command to play the track or show.

A menushow needs an associated menu track in its medialist which defines the layout of the menu:

- Display, thumbnails, bullets or Titles in many combinations
- Align the menu vertically or horizontally
- Create multi-column, multi-row menus
- Alter the separation, colours and font of the menu items

5.2.2.1 Controls and Commands

Menu Commands are the same as those for Mediashow and are described in details in Section 5.2.1.1

5.2.2.2 Fields

Field	Examples	Values
Show Tab		Essential information
Type	menu	Cannot be edited
Title		Text describing the show, displayed in the editor.
Show Reference	mymenu	A 'label' by which the show is referenced by other shows. Any text without spaces

Field	Examples	Values
Show Canvas		See Mediashow
Medialist	mymenu.json	Filename of the medialist file containing the tracks for the menu.
Show Timeout	59	h:m:s, If there is no activity on the menu Pi presents automatically close the show. 0 for no timeout.
Track Timeout	20	h:m:s, If non-zero tracks launched from the menu will be stopped after this time and control will return to the menu.
Menu Track	menu-track	The track reference of the track that will display the menu content. Must be of type Menu Track
EggTIMER Tab		EggTIMER text is displayed when the next track is being loaded.
		See mediashow
Show Background and Text Tab		Show text is overlaid on all tracks other than message tracks opened from the menu.
		See mediashow
Track Defaults Tab		Tracks played from a menu need some configuration if not supplied in the individual tracks
		See Mediashow
Controls Tab	pp-down down	Defines the bindings between symbolic names of input events and commands One binding per line each with the format: symbolic-name command Bindings are not inherited by subshows. See Table above and Section 5.2.1.1
Disable Controls	no	yes/no If 'yes' the commands are disabled. This is a quick way to inhibit all controls for a concurrent show. For more selective control use the Controls field

5.2.3 Liveshow and Artliveshow

A Liveshow is identical to a Mediashow except that the content is dynamically supplied from directories of media, the Live Tracks directories, and is therefore limited to video, image, and audio tracks, and their track plugins. The tracks from the two directories are merged and, if sequence = ordered, sorted by the leaf of the file name.

See Section 5.2.1 for details of commands, triggers and fields.

A medialist must be associated with a Liveshow. It is used for the Child Track and all other tracks in it will be ignored.

Artliveshow

Artliveshows provide full gapless playback; there is no gap between tracks and tracks do not freeze at their end to cover loading of the next track. Artliveshows have some limitations and do not have the fields which are shown in brackets in the Table in Section 5.2.1.3:

- Start, Next and End Triggers.
- Child Tracks

The Liveshow has the following fields additional to those of the Mediashow:

Field	Examples	Values
Live Tracks Directory 1		The full path of a Directory containing tracks for this show. If blank the tracks are taken from the <code>pp_live_tracks</code> directory in <code>pp_home</code>
Live Tracks Directory 2		The full path of a Directory containing tracks for this show. If blank the tracks are taken from directory specified in the <code>-l</code> command line option.

5.2.4 Radiobuttonshow

A Radiobuttonshow provides the sort of show facilities that are in many kiosks namely:

'Start with an initial display which might include some text inviting the user to press buttons or touch the screen to initiate a track. While playing the track pressing another button will play another track. At the end of a track or when Stop is pressed revert to the initial display.'

In Pi Presents the initial display can be an image, message, video or audio track, or a show. Playing of other tracks is by means of commands in the Controls field of the Radiobuttonshow e.g.

```
but1 play myimagetrack
but2 play myvideotrack
```

5.2.4.1 Controls and Commands

Each control has three fields separated by spaces:

- symbolic name - the symbolic name of the input event that will play the show or track.
- command – see below
- track - the Track Reference or Show Reference of the track or Show to be played.

Command	Argument	
play	track-ref	Play the track specified by track-ref
return	<blank>	return to the first track of the show
stop	<blank>	stop the current track and, if show is the quiescent state and this is not a top level show, then close the show.

		The quiescent state is when the first track is showing.
exit	<blank>	close the Radiobuttonshow
null	<blank>	inhibits the control with the same symbolic name that has been defined in the show.
pause	<blank>	toggle pause on video, audio, or image tracks
omx-*	<blank>	execute a runtime control for the video player (Section 10)
mplay-*	<blank>	execute a runtime control for the audio player (Section 10)
uzbl-*	<blank>	execute a runtime control for the web browser (Section 10)

Commands can be placed in the Controls field of the tracks in addition to the Controls field of the radiobuttonshow. The controls in a track are merged with and override those in the show. This is generally not required but could be used where, for example it is required that the audience watches the whole of a video. In this case using the null command would inhibit the symbolic names that play other tracks or stop the track.

5.2.4.2 Fields

Field	Examples	Values
Show Tab		
Type	radiobuttonshow	Cannot be edited
Title	My Show	Text describing the show displayed in the Editor and menu
Show Reference	myradioshow	A 'label' by which the show is referenced by other shows. Any text without spaces
Show Canvas		See Mediashow
Medialist	mymedia.json	Filename of the medialist file containing the tracks for the Radiobuttonshow. All tracks should have a Track Reference.
First Track	myfirsttrack	The Track Reference of the track that will form the initial display for the show.
Show Timeout	1:59	h:m:s, If there is no activity on the First Track Pi Presents automatically returns to the previous show. 0 for no timeout.
Track Timeout	20	h:m:s If non-zero tracks launched from the radiobuttonshow will be stopped after this time and control will return to the show.
Eggtimer Tab		
		See Mediashow
Show Background and Text		
		See Mediashow

Field	Examples	Values
Track Defaults Tab		
		See mediashow
Controls Tab		
Controls	myname play mytrack	Defines the bindings between symbolic names of input events and commands One binding per line each with the format: symbolic-name command [arg] Bindings are NOT inherited by subshows. See Table above for list of commands
Disable Controls	no	yes/no If 'yes' all Commands are disabled. This is a quick way to inhibit all controls for a concurrent show. For more selective control use the Controls field.

5.2.5 Hyperlinkshow

A Hyperlinkshow provides the type of show facilities that are used in touchscreen displays in museums:

'Start with an initial page with some introductory text, video or image and a selection of on screen buttons which allow the user to move to another page. Each page can have a different selection of buttons to link to other pages. When in any page other than the initial page additional buttons allow the user to go back to the previous page or to return to the initial page.

All of the touchscreen displays that I have tried in my researches seem to work this way, or are Radiobuttonshows.

Every page in the example above is a track in a Hyperlinkshow. Each has a Controls field which contains commands implementing the movement between tracks and back.

e.g.track 'story1b' might contain the following controls:

```
next-name call story1c
alternative-name call alternative1
back-name return
home-name home
```

xxx-name is the symbolic name of an input event. This example says either go forward to 'story1c' or to 'alternative1', return to the previous track (which was

probably 'story1a'), or return to the home track which probably means going back through 'story1a' without displaying it.

When executing the call command Pi Presents remembers where it has come from in the 'path' (essentially a stack) so the return command can go back one removing the current track from the path. Think of nested call and return of subroutines in a programming language.

There is a special track called the Home Track. The home command can skip back along the path until it arrives at the Home Track so it is not necessary to know how far you have gone to get back to a known starting point.

Each control has three fields separated by spaces:

- symbolic name - the symbolic name of the input event that will trigger the command
- command - call, return, goto, jump, exit, null, repeat, pause, no-command, and runtime controls
- track - the Track Reference of the track to be played

5.2.5.1 Controls and Commands

Command	Argument	Effect
call	Track Reference	play Track Reference and add it to the path
return	<blank>	return 1 back up the path removing the track from the path, stops at Home Track.
return	number	return n tracks back up the path removing the track from the path, stops at Home Track.
return	<Track Reference>	return to Track Reference removing tracks from the path, goes through Home Track if necessary.
home	<blank>	Return up the path to Home Track removing tracks from the path
jump	<Track Reference>	play Track Reference forgetting the path back to Home Track
goto	<Track Reference>	play Track Reference, forget the path
repeat	<blank>	Repeat the current track without resetting the timeout.
null	<blank>	inhibits the Link with the same symbolic name that has been defined in the show.
exit	<blank>	end the Hyperlinkshow
pause	<blank>	toggle pause on video, audio, or image tracks
no-command	<blank>	Clicking it has no effect. Use for legends for 'soft keys' to display a 'Click Area' that is not enabled.
omx-*	<blank>	execute a runtime control for the video player (Section 10)
mplay-*	<blank>	execute a runtime control for the audio player (Section 10)
uzbl-*	<blank>	execute a runtime control for the web browser

Commands can be placed in the Controls field of the Hyperlinkshow in addition to the Controls field of tracks. This is a convenience to save typing because the commands from the show are used in every track and the commands from the track are merged with them. Sometimes this is not desirable and using the null command in the track will cancel the command with the same symbolic name in the show.

The goto command does not remember where it has come from. It is used in special circumstances such as timeout. It is also an alternative way to call/return for implementing a back button. If used every back has to be a goto. It is not a good idea to mix call/return and goto in the same part of a Hyperlinkshow.

If a track, such as a video or a timed image, ends naturally there needs to be a way for Pi Presents to execute a Command in order to determine what to do next. This is achieved by Pi Presents generating an internal input event with the symbolic name pp-onend; the Links field can then have a command to service it e.g.

```
pp-onend goto mynextpage
or
pp-onend repeat
```

Commands associated with pp-onend cannot be pause, no-command, omx-, mplay-, or uzbl-

There is a second special track the First Track. Pi Presents always starts the show here. In many applications Home Track and First Track will have the same Track Reference. They may however be different. This was designed such that First Track would start the show with an image or video to entice the user to touch a button to move to the Home Track in order to start the interactive sequence of tracks. Once past the Home Track pressing home, return, or jump would not return him to First Track. However the timeout would.

To ensure that the screen goes back to the First Track if the user leaves the screen there is a timeout. If activated the Hyperlinkshow goto's the Timeout Track. This could have the same Track Reference as the First Track but a separate Timeout Track allows for reset of, say, animation before goto'ing the First Track.

Tracks in Hyperlinkshows can be shows. These shows have their own set of commands including stop which will return to the hyperlinkshow.

When developing a hyperlinkshow application it is sometimes useful to see the path of pages that Pi Presents has remembered. To enable this use Debug Path.

5.2.5.2 Touchscreens and Soft Buttons

The nature of Hyperlinkshows is such that a different set of controls is needed for each track of the show. Pi Presents has two ways to provide these controls:

- Click Areas

Click areas are aimed at touchscreens. A 'click area' is an area of the screen which is mouse click sensitive. Although they are called 'click areas' they can

best be used with touchscreens and might better be called touch areas. Touching the area of a touchscreen or clicking with a mouse in the area will cause an input event with a symbolic name defined in screen.cfg.

Optionally associated with a click area is an image, maybe of a button, which is displayed when the click area is enabled. Alternatively a click area can be transparent and the area defined as a polygon to encompass the shape of the object behind.

- **Soft Buttons**

Soft buttons are a poor man's touchscreen. They were used a lot in real time control applications before touchscreens were developed.

Soft buttons are a row of unmarked keys or buttons arranged along the edges of a display screen. The legends for the buttons are software controlled and appear adjacent to the button on the edge of the display area of the screen. Thus different button positions and legends can be applied to each track of a Hyperlinkshow.

Pi Presents will allow both of these techniques to be used to control Hyperlinkshows as describes in Section 9

5.2.5.3 Fields

Field	Examples	Values
Show Tab		
Type	hyperlinkshow	Cannot be edited
Title	My Hyperlink Show	Text describing the show displayed in the editor and menu.
Show Reference	myhyperlinkshow	A 'label' by which the show is referenced by other shows. Any text without spaces
Medialist	mymedia.json	Filename of the medialist file containing the tracks for the Hyperlinkshow. All tracks should have a Track Reference
First Track	myfirsttrack	The Track Reference of the track that will form the initial display for the show.
Home Track	myhometrack	The Track Reference of the track that will form the root of the call/return path.
Timeout Track		<p>The track displayed when a timeout occurs. Pi Presents does not return directly to the Home track as there may be a need to reset animation or stop concurrent shows.</p> <p>Usually this track will do whatever is required and 'goto' the First or Home Track after a short time. If the track is an audio track it can have no sound and zero duration.</p>
Show Timeout	60	h:m:s, If there is no activity on the Home Track Pi Presents automatically returns to the previous show. 0 for no timeout.

Field	Examples	Values
Track Timeout	30	h:m:s. When playing a track if no user input is received or the track does not naturally finish before the timeout then goto the Timeout Track. A value of 0 disables the timeout function.
Show Canvas		See Mediashow
Eggtimer Tab		
		See mediashow
Show Background and Text Tab		
		See Mediashow
Track Defaults Tab		
		See mediashow
Controls Tab		
		See Mediashow
Disable Controls	no	yes/no If 'yes' Internal Operations (play, pause, up, down, stop) and Run Time controls are disabled. This is a quick way to inhibit all controls for a concurrent show. For more selective control use the Controls field.
Controls	name call mytrack	Defines the bindings between symbolic names of input events and commands One binding per line each with the format: symbolic-name command Bindings are NOT inherited by subshows. See Table above for a list of commands

5.2.6 Start Show

The Start show specifies the shows to be run when Pi Presents starts. Each show will run concurrently with other shows. Input events will be passed to all concurrent shows. The Start shows field may be blank in which case shows must be started by other means. The Start Show must be present even if there are no shows to start.

Field	Examples	Values
-------	----------	--------

type	start	Must always be start, not editable.
title	Start Show	Text describing the show, displayed in the editor.
Show Reference	start	must be 'start'
Start Shows	show1 show2	A list of Show References separated by spaces.

5.3 Medialists

Medialists are similar to playlists in a media player in that they define the tracks to be played. How they are played is defined by the show that the list is associated with. Each entry in a medialist is a 'track'. Tracks can be of various types:

- image - a still image
- video - a track played by OMXPlayer.
- audio - an audio track played by MPlayer
- message - displays lines of text
- web – runs a browser to display a URL
- show - shows can be used as tracks allowing nesting to any depth.
- menu-track – a special track to define the format of a menu

A medialist is generally associated with a single show. However in Pi Presents they have been kept separate from the remainder of the show specification so that the same medialist could be used in two different shows.

5.4 Tracks

Each type of track has fields describing how to display the track, some of these override the corresponding fields in the associated show.

The type of track describes the primary content of the track; primary content can be displayed in front of an image or plain coloured background and have text annotations.

- image - a still image. Image types are those that can be rendered by the Python Imaging Library. For performance reasons image size is best limited to the 1 megapixel region.
- video - a track played by OMXPlayer. Playable video formats depend on the codec licences purchased from the Foundation.
- audio - an audio track played by MPlayer. Any track type playable by MPlayer should be playable. The audio track type is very flexible as it has extended duration handling features which makes them useful for sequencing animation or shows.
- message - displays lines of text. Can also used to display a blank screen. It provides a simple slide preparation facility. If you want something better use Powerpoint or similar and export as .jpg displaying as an image.

- web- displays a URL in a browser. Browser integration into Pi Presents is not optimal and this type of track should be used with care. There are many other browser based digital signage systems.
- show - shows can be used as tracks allowing nesting to any depth.
- menu – used to specify the layout of a menu for the Menu Show.

5.4.1 Track Location Names

Track locations which specify files can be relative or absolute. Relative file names allow profiles to be portable. See Section 4.2.1

- The leading + sign in file paths allows tracks to be specified relative to the /pp_home directory. If a plus sign is not present then the path must be absolute. It is recommended that media files be stored under /pp_home if the profiles are to be portable.
- Absolute references do have their uses, for example in specifying internet url's e.g. http://www.mysite.com/track_to_play.mp4

5.4.2 Anonymous and Labelled tracks

Some medalist entries will have labels defined by the field Track Reference. If the Track Reference is blank then the track is included in a Menu, Mediashow or Liveshow, these are called anonymous tracks. If the label is not blank then the track can be referenced by Commands used in Radiobuttonshows and Hyperlinkshows, or for a special purpose. Special purposes include:

- The Child of a mediashow or liveshow.
- The Menu Track of a Menu
- First, Home and Timeout tracks in Hyperlinkshow or Radiobuttonshow

The field definitions for each type of medalist track follow in the next section.

5.4.3 Show Track

The tracks in a show can themselves be shows. These are called sub-shows. Show tracks allow sub-shows to be included in a medalist. The Show To Run field specifies the Show Reference of the show.

5.4.4 Image Track

Image tracks are rendered by the Python Imaging Library. (See hardware requirements for recommended image size). Image tracks can be paused with the Pause Command. Images can appear in a window, be rotated, can have an image or colour as a background, can be overlaid with text; Track Text is overlaid on a per track basis, Show Text per show.

Field	Example	Values
Track Tab		

Field	Example	Values
Type	image	
Title		Displayed on a menu and in the editor
Track Reference		A label for the track - blank if the track is to be included in a menu or mediashow. Non-blank if the track is to be triggered from a symbolic name or is special.
Location	+ /media/sarename.gif	The filename of the track which may be blank.
Thumbnail	+ /media/sarename_tn.jpg	The optional filename of a thumbnail image to be displayed by a menushow.
Duration	5	Seconds. If 0 then image is displayed until terminated by user input. If blank then the value in the parent show is used.
Transition	cut	cut. (Not used)
Rotation	90	degrees. If blank then the value in the parent show is used.
Image Window	original 10 100 fit fit 1010 40 40 fit ANTIALIAS	A viewport in which to show the image. If blank then the value in the parent show is used. See section 5.4.4.1
Background Image	+ /media/image.jpg	The file name of an image which is used as the background for the main image. If blank then the value in the parent show is used.
Display Show Background Image	yes	yes,no yes - the background image of the parent show is displayed if the track Background Image is blank no - the background image of the parent show is not displayed
Background Colour		Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. See http://effbot.org/tkinterbook/tkinter-widget-styling.htm If blank then the value in the parent show is used.
Plugin Configuration File	+ /media/krt-time.cfg	File which specifies the name and parameters of the Track Plugin.
Controls Tab		
Controls	myname null	Bindings of symbolic names to commands. The bindings defined here for the track are merged with and override bindings for the associated show. See Section 8.1
Text Tab		

Field	Example	Values
Track Text	Picture of Taj Mahal	If not blank the text displayed with the image, overlaying it if necessary.
Track Text Font	Helvetica 30 bold	Use the Font Chooser to select a font, or type one in. See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Track Text Colour	white	Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Track Text x Position	100	distance of the left end of the text from the left of the screen (pixels)
Track Text y Position	100	distance of the top of the text from the top of the screen (pixels)
Display Show Text	yes	yes,no Allow or inhibit the display of show text for this track
Pause Text	Paused.....	Text displayed when track is paused
Pause Text Font	Helvetica 30 bold	Use the Font Chooser to select a font, or type one in. See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Pause Text Colour	white	Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Pause Text x Position	100	distance of the left end of the text from the left of the screen (pixels)
Pause Text y Position	100	distance of the top of the text from the top of the screen (pixels)
Show Control Tab		
Show Control at Beginning	open audio1	See Section 8.2
Show Control at End	close audio1	See Section 8.2
Animation Tab		
Animation at Beginning	1 out1 state on 2 out1 state off 1 out2 state on 6 out3 state off	See Section 12
Clear Animation	no	yes/no See Section 12
Animation at	0 out2 state off	See Section 12

Field	Example	Values
End		

5.4.4.1 Image Window

The Image Window controls how an image is presented spatially. It contains a number of fields separated by spaces. If Image Window is blank then the value from the show is used.

The first field is one of original, shrink, fit or warp

- original
The image is displayed at its original size. The command has optionally two arguments the x,y, coordinates of the top left corner of the displayed image. If the arguments are omitted then the image is displayed centred on the screen.

e.g. original, original 100 100
- fit
The image is displayed such that it fits in the specified window maintaining its aspect ratio. The image is shrunk or expanded as required. The command has optionally four arguments the x,y, coordinates of the top left corner and bottom right corners of the displayed image. If the arguments are omitted then the image is displayed centred on the screen.

e.g. fit, fit 100 100 1000 500
- shrink
As fit except that the image is not expanded if it is smaller than the window.

e.g. shrink, shrink 100 100 1000 500
- warp
The image is displayed such that it fits in the specified window without maintaining its aspect ratio. The image is shrunk or expanded as required. The command has optionally four arguments the x,y, coordinates of the top left corner and bottom right corners of the displayed image. If the arguments are omitted then the image is warped to be displayed full screen.

e.g. warp, warp 100 100 1000 500, warp 100 100 1000 500 BICUBIC

For fit, shrink and warp an optional filter, one of NEAREST, BILINEAR, BICUBIC, ANTIALIAS can be specified as the fifth argument, see

<http://effbot.org/imagingbook/image.htm>

thumbnail and resize sections. If the argument is omitted NEAREST is used.

e.g. fit 100 100 1000 500 BICUBIC

5.4.5 Video Track

A track played by OMXPlayer. Pi Presents should play any track that OMXPlayer can play (but see hardware requirements Section 17). Video tracks can be paused with the Pause Internal Operation. Videos can appear in a window, can have an image or colour as a background which can be overlaid with text; Track Text is overlaid on a per track basis, Show Text per show. The video itself cannot be overlaid with text because of OMXPlayer limitations.

Field	Example	Values
Track Tab		
Type	video	
Title	The Film	Displayed on a menu and in the editor
Track Reference		A label for the track - blank if the track is included in a menu or mediashow. Non-blank if the track is to be triggered from a symbolic name or is special.
Location	+myvideos/film.mp4	The filename of the track.
Thumbnail	+media/film_tn.jpg	The optional filename of a thumbnail image to be displayed by a menushow.
Freeze at End		yes/no If yes, the last frame of the track is displayed when the track ends (the video is paused just before its end). If no, the video ends normally. If blank the value is taken from the show.
Video Player Audio	local	hdmi/local/both. If blank then the value in the parent show is used.
Video Player Volume	0	Volume of video track (-60 -> 0 dB). If blank then the value in the parent show is used.
Video Player Options	-t 1	Additional command line options for omxplayer. If blank then the value in the parent show is used.
Video Window	original warp warp 10 10 500 1000	A viewport in which to show the video. If blank then the values are taken from the parent show. <ul style="list-style-type: none"> original - use OMXPlayer default behaviour warp - scale to the size of the screen without maintaining aspect ratio warp followed by two x,y, pairs (top left, bottom right) - the viewport window. The video is scaled to this size without preserving the aspect ratio.
Background Colour	red	Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name.

		See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm . If blank then the value in the parent show is used.
Background Image	+/images/back.jpg	The filename of an optional image that is displayed in the background, can be blank. The video, Show Text and Track text is displayed above the image. If blank then the value in the parent show is used.
Display Show Background Image	yes	yes,no yes - the background image of the parent show is displayed if the track Background Image is blank no - the background image of the parent show is not displayed
Plugin Configuration File	+/media/weather_ny.cfg	File which specifies the name of and parameters for the Track Plugin
Seamless Loop		yes/no. yes enables the omxplayer -loop option to repeat the track continuously. For (art)liveshow Seamless Loop is always 'no'. An artmediashow with one track might perform better than using Seamless Loop. Note: a looping track has no end.
Controls Tab		
Controls	myname null	Bindings of symbolic names to commands. The bindings defined here for the track are merged with and override bindings for the associated show. See Section 8.1
Text Tab		
		See Image Track. Does not have Pause Text
Show Control Tab		
		See Image Track
Animation Tab		
		See Image Track

5.4.6 Audio Track

A track played by MPlayer. Pi Presents should play any audio track that MPlayer can play. Sound can be directed to HDMI or analogue and to either of the analogue speakers. Audio tracks can be paused with the Pause Command.

Audio tracks can have an associated display. The display can have a coloured or image background; Track Text is overlaid on a per track basis, Show Text per show.

The audio track has enhanced duration control making it suitable as a 'dummy' track for controlling animation or concurrent shows. It can have zero duration and run without playing any media.

Field	Example	Values
Track Tab		
Type	audio	
Title	The Music	Displayed on a menu and in the editor
Track Reference		A label for the track – blank if the track is included in a menu or mediashow. Non-blank if the track is to be triggered from a symbolic name or is special.
Thumbnail	+ /media/sarena me_tn.jpg	The optional filename of a thumbnail image to be displayed by a menushow.
Location	+ /tracks/music.mp3	The filename of the track. The location is optional. If blank the timing of the track is taken from the Duration field which must not be blank.
Duration	5	<p>Seconds.</p> <p>If duration is blank then the track ends when the audio file ends or is stopped. The track must be stopped by the user if there is no audio track.</p> <p>If zero the Location field should be left blank so no audio track is played. Animation and Show Control are executed. Zero duration is aimed at use of the track for show or animation control.</p> <p>If greater than zero the 'track' ends at the stated time. If a file has been specified in Location the playing of the track may be cut short or there may be period of silence after the audio file. This is primarily aimed at Animation and Show Control</p> <p>For (art)liveshow Duration is always Blank.</p>
Audio Player Speaker	left	left/right/stereo. If blank then the value in the parent show is used.
Audio Player Audio	local	hdmi/local. If blank then the value in the parent show is used.
Audio Player Volume	0	Volume of audio track (-60 -> 0 dB). If blank then the value in the parent show is used.
Audio Player Options		Additional command line options for mplayer. If blank then the value in the parent show is used.
Background Colour	red	Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name.

		See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm). If blank then the value in the parent show is used.
Background Image	+/images/back.jpg	The filename of an optional image that is displayed in the background. The video, Show Text and Track text is displayed above the image. If blank then the value in the parent show is used.
Display Show Background Image	yes	yes,no yes – the background image of the parent show is displayed if the track Background Image is blank no – the background image of the parent show is not displayed
Plugin Configuration File	+/media/weather_ny.cfg	File which specifies the name of and parameters for the Track Plugin
Controls Tab		
Controls	myname null	Bindings of symbolic names to commands. The bindings defined here for the track are merged with and override bindings for the associated show. See Section 8.1
Text Tab		
		See Image Track
Show Control Tab		
		See Image Track
Animation Tab		
		See Image Track

5.4.7 Web Tracks

Web tracks are rendered by the Webkit based uzbl browser.

You will have installed the uzbl browser when installing Pi Presents. You can use it in from the desktop like an ordinary program.

Web pages can be played from the internet or from the local file system. They can appear in a window or be fullscreen, can have an image or colour as a background and Track and Show Text. Track Text is overlaid on a per track basis, Show Text per show.

The integration of uzbl into Pi Presents is not as good as the other players; it is best used in non-interactive displays:

- The browser is loaded each time a web track is played; this can take up to 15 seconds and is not properly handshaked with Pi Presents. If you want to display a succession of web pages the Web Player in Pi Presents has a Browser Command script which allows a sequence of web pages to be displayed, with optional looping.
- Unlike other players, when the browser is in use 'focus' is on the browser window and not on Pi Presents. This means that keyboard inputs are sent to uzbl and not to Pi Presents until the Pi Presents window is clicked on. GPIO inputs are not affected.
- The browser window has scrollbars and title bar. Use of the browser affects the display of the task bar. Section 5.4.7.2 describes how to correct these.

Field	Example	Values
Track Tab		
Type	web	
Title		Displayed on a menu and in the editor
Track Reference		A label for the track - blank if the track is to be included in a menu or mediashow. Non-blank if the track is to be triggered from a symbolic name or is special.
Location	www.google.co.uk +/media/mywebpage.html	The filename of the track: <ul style="list-style-type: none"> • an internet url • a full pathname to a file • a relative path to a file located under pp_home
Thumbnail	+/media/sarename_tn.jpg	The optional filename of a thumbnail image to be displayed by a menushow.
Duration	5	Seconds. If 0 then the web page is displayed until terminated by user input. If blank then the value in the parent show is used.
Web Window	warp warp 10 100 200 700	A viewport in which to show the web page. If blank then the value in the parent show is used otherwise: <ul style="list-style-type: none"> • warp - scale to the size of the screen without maintaining aspect ration • warp followed by two x,y, pairs (top left, bottom right) - the viewport window. The web page is scaled to this size without preserving the aspect ratio.
Background Image	+/media/image.jpg	The file name of an image which is used as the background for the web page. If blank then the value in the parent show is used.

Field	Example	Values
Display Show Background Image	yes	yes,no yes - the background image of the parent show is displayed if the track Background Image is blank no - the background image of the parent show is not displayed
Background Colour		Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. See http://effbot.org/tkinterbook/tkinter-widget-styling.htm If blank then the value in the parent show is used.
Plugin Configuration File	+ /media/weather_ny.cfg	File which specifies the name of and parameters for the Track Plugin
Text Tab		
		See Image Track. No Pause text
Browser Commands Tab		
Browser Commands	wait 10 refresh	Commands which load URL's etc. See Section 5.4.7.1
Controls Tab		
Controls	myname null	Bindings of symbolic names to commands. The bindings defined here for the track are merged with and override bindings for the associated show. See Section 8.1
Text Tab		
		See Image Track
Show Control Tab		
		See Image Track
Animation Tab		
		See Image Track

5.4.7.1 Browser Commands

The Browser commands field contains 0 or more browser commands. Each command is on a new line. Some commands have an argument which is separated from the command by a space

Command	Description	uzbl Command used
---------	-------------	-------------------

load <arg>	Load the web page specified by <arg>. <arg> may be: <ul style="list-style-type: none"> • an internet URI • a full pathname to a file • a relative path to a file located under pp_home preceeded by + 	uri
refresh	refresh the currently loaded web page	reload_ign_cache
wait <arg>	wait <arg> seconds	-
exit	Normally the script loops after the last command is executed. Exit, as the last command, will end playing of the track.	-
loop	A single loop command is allowed. If a loop command is present execution will resume here after the last command of the script.	-
uzbl <arg>	Execute a uzbl browser command. The commands are defined here: http://www.uzbl.org/readme.php <arg> is the uzbl command together with its arguments. These commands will allow tailoring of the display of individual web pages	

Example: repeating two web pages. Looping will continue until the track duration is exceeded.

```
wait 20
load www.google.co.uk
wait 20
load www.museumoftechnology.org.uk
```

Example: display two pages then exit to the next track.

```
wait 20
load www.google.co.uk
wait 20
load www.museumoftechnology.org.uk
wait 20
exit
```

Example: Refresh the initially loaded web page at 20 second intervals

```
wait 20
refresh
```

5.4.7.2 Full screen Integrated Browser Displays

Out of the box uzbl will show a title bar. To remove this, edit the file /home/pi/.config/openbox/lxde-rc.xml to include the following just before the statement </applications>

```
<application name="uzbl*">
  <decor>no</decor>
```

</application>

A reboot of the Pi is necessary for the edit to take effect. NOTE: This requires confirmation if using a Raspbian release of 25/12/2014 or later (new UI).

When the browser is displayed the underlying Pi Presents window changes to reveal the task bar. To correct this, adjust the Taskbar and minimize it:

- Right click on the Taskbar and select Panel Setting
- Select the Appearance Tab. Select Solid colour and, choose black with 100% opacity.
- Select the Advanced Tab
- Set 'minimise panel when not in use' to On
- Set 'Size when minimized' to 2 pixels

Scrollbars are displayed when the content of a web page is larger than the window. These can be removed as described here. (not tested)

<http://www.uzbl.org/wiki/hide-scrollbars>

Alternatively use the uzbl zoom_out command in a Browser Command script to adjust the size of the page so scrollbars are not required.

5.4.8 Message Tracks

Message tracks display text against a coloured background or optional image. They do not need a media file to be specified as the text is contained in the Message Text field.

Field	Example	Values
Track Tab		
Type	message	
Title	A Message	Displayed on a menu and in the editor
Track Reference		A label for the track - blank if the track is included in a menu or mediashow. Non-blank if the track is to be triggered from a symbolic name or is special.
Message Text	Welcome	The text to be displayed.
Thumbnail	+ /media/sarena me_tn.jpg	The optional filename of a thumbnail image to be displayed by a menushow.
Duration	5	Seconds. If 0 then message is displayed until terminated by user input. If blank then the value in the parent show is used.
Text Font	Helvetica 30 bold	Use the Font Chooser to select a font, or type one in: See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Text Colour	white	Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a

		colour name. See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Justification	left	left/center/right. Text justification.
Message x Position	100	If blank then the message is centred in the screen. If non-blank the distance of the left end of the text from the left of the screen (pixels)
Message y Position	500	If Message x Position is specified then distance of the top of the text from the top of the screen (pixels)
Background Colour	red	Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm If blank then the value in the parent show is used.
Background Image	+/images/back.jpg	The filename of an optional image that is displayed instead of a plain background. The message is displayed on top of the image. If blank then the value in the parent show is used.
Display Show Background Image	yes	yes,no yes - the background image of the parent show is displayed if the track Background Image is blank no - the background image of the parent show is not displayed
Plugin Configuration File	+/media/weather_ny.cfg	File which specifies the name of and parameters for the Track Plugin
Controls Tab		
Controls	myname play track34	Bindings of symbolic names to commands. The bindings defined here for the track are merged with and override bindings for the associated show. See Section 8.1
Text Tab		
		See Image Track. Does not have Pause Text
Show Control Tab		
		See Image Track
Animation Tab		
		See Image Track

5.4.9 Show Track

A show can be a track in another show. Uses might be:

- A mediashow made up of smaller mediashows
- A menu of mediashows
- Menus with sub-menus
- A Liveshow run from a Mediashow which provides an initial screen.

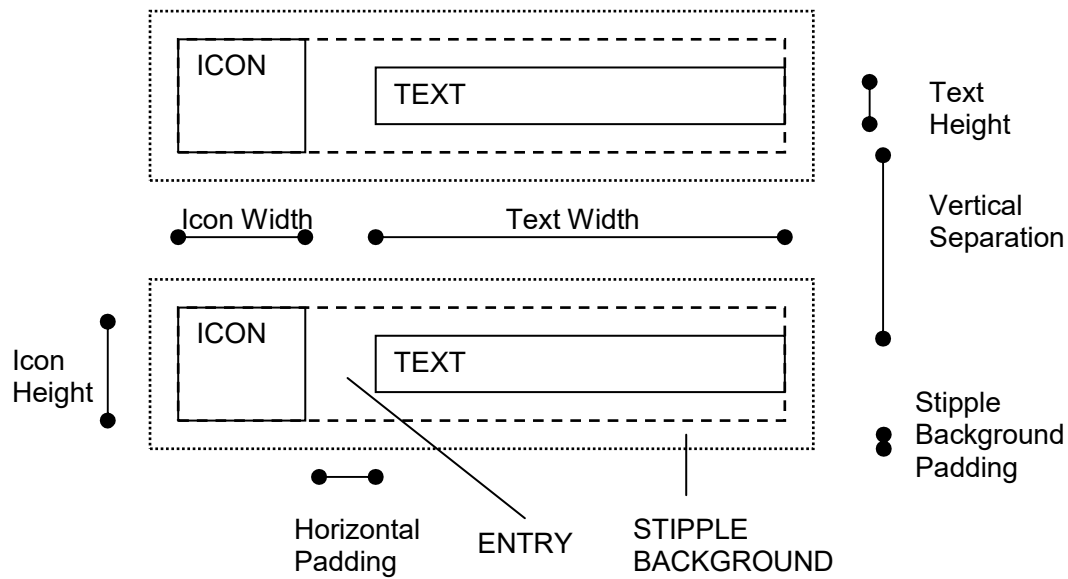
Field	Example	Values
Type	show	
Title	My Other Show	Displayed on a menu and in the editor
Track Reference		A label for the track - blank if the track is included in a menu or mediashow. Non-blank if the track is to be triggered from a symbolic name or is special.
Show to Run	myothershow	Show Reference of the show to be run
Thumbnail	+ /media/sarename_t n.jpg	The optional filename of a thumbnail image to be displayed by a menushow.

5.4.10 Menu Track

A menushow needs an associated menu track in its medialist which defines the layout of the menu. A menu track provides it. Its Track Reference should appear in the Menu Track field of a Menu show.

The menu formats that can be produced are extensive:

- Display, thumbnails, bullets or Titles in many combinations
- Align the menu vertically or horizontally
- Create multi-column, multi-row menus
- Alter the separation, colours and font of the menu items



Menu Geometry Definitions

Field	Example	Use
Track Tab		
Type	menu	
Title	Menu Track	Displayed on a menu and in the editor
Track Reference	menu-track	A label for the track.
Background Colour		see mediashow - Background Colour for menu
Background Image	+ /media/image.jpg	If not blank The file name of an image which is used as the background for the menu.
Display Show Background Image	yes	yes,no yes - the background image of the parent show is displayed if the track Background Image is blank no - the background image of the parent show is not displayed
Entry Font	Helvetica 30 bold	Use the Font Chooser to select a font, or type one in: see: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Entry Colour	white	Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. see: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Selected Entry Colour	red	colour when the entry selected
Plugin Configuration File	+ /media/krt-time.cfg	File which specifies the name and parameters of the Track Plugin.

Geometry Tab		
Menu Window		<p>fullscreen, one or two pairs of x,y, coordinates fields are separated by spaces.</p> <ul style="list-style-type: none"> • fullscreen - the menu is displayed full screen. • one x,y, coordinate - the menu's top left corner • two x,y, coordinates - the second pair of coordinates specifies the bottom right corner of a bounding box for the menu.
Direction	vertical	vertical, horizontal - the direction of laying out and traversing the menu.
Rows	5	If Direction = vertical the number of rows in the menu.
Columns	1	If Direction = horizontal the number of columns in the menu.
Icon Mode		<p>thumbnail,bullet,none</p> <ul style="list-style-type: none"> • thumbnail - if the track has a thumbnail then this is displayed otherwise, if it is an image track the image is warped to the icon size otherwise a thumbnails is displayed which depends on the type of track. • bullet- the icon specified in Bullet is displayed. • none - the menu displays only text .
Text Mode		<p>none,right,below</p> <ul style="list-style-type: none"> • right - the text is displayed to the right of the icon • below - the text is displayed below the icon
Bullet	+ /media/bullet.jpg	The filename of the bullet. Absolute or relative.
Icon Width	100	Width of the icon. Images will be warped to fit into the defined size.
Icon Height	100	Height of the icon. Images will be warped to fit into the defined size.
Horizontal Padding	10	When Text Mode is Right the horizontal distance between the Icon and the Text
Vertical Padding	10	When Text Mode is Below the vertical distance between the Icon and the Text
Text Width	200	Width of the text. The text is wrapped to fit into this width.

Text Height	50	Height of the text. Used to calculate the positions of entries. However text is not constrained to fit into this height. The user must adjust the font size to suit.
Horizontal Separation	20	The distance between the bottom of one menu entry and the top of the next.
Vertical Separation	20	The distance between the right edge of one menu entry and the left edge of the next.
Stipple Background		yes/no. Display a stippled rectangle decoration behind the menu entry.
Stipple Background Padding		The distance that the stipple background extends from the menu entry.
Guidelines		<p>never,auto,always</p> <p>If Menu Window has two x,y pairs then coloured rectangles are displayed on the screen to help design the menu.</p> <ul style="list-style-type: none"> • White rectangles are the menu entry. • The Blue rectangle is the calculated extent of the menu. • The Red rectangle is the bounding box as defined by Menu Window <p>Auto displays the rectangles only when the extent of the menu is outside the bounding box.</p>
Text Tab		
Track Text	This menu	Multi-line text
Track Text Font	Helvetica 30 bold	<p>Use the Font Chooser to select a font, or type one in:</p> <p>See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm</p>
Track Text Colour	white	<p>Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name.</p> <p>See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm</p>
Track Text x	100	distance of the start of the text from the left of the screen (pixels)
Track Text y	800	distance of the top of the text from the top of the screen (pixels)
Hint Text	To Play, press return	Menus do not have children; the hint is a general purpose field which might be used for displaying instructions.
Hint Font	Helvetica 30 bold	<p>Use the Font Chooser to select a font, or type one in:</p> <p>See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm</p>

Hint Colour	white	Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name. See: http://effbot.org/tkinterbook/tkinter-widget-styling.htm
Hint x	100	distance of the start of the text from the left of the screen (pixels)
Hint y	800	distance of the top of the text from the top of the screen (pixels)
Controls Tab		
Controls	myname null	Bindings of symbolic names to commands. The bindings defined here for the track are merged with and override bindings for the associated show. See Section 8.1
Show Control Tab		
		See Image Track
Animation Tab		
		See Image Track

6 Black Box Operation

There are a number of things to set up to make Pi Presents into a full screen, auto starting, GPIO controlled application. You do not need to use them all.

6.1 Command Line Options

`python pipresents.py -h` will show the command line options

Options	
-p --profile <arg>	Name of the profile to be used. e.g. pp_mediashow_1p3 If this is not specified then an error message will be shown.
-b --noblack	Disable screen blanking.
-f --fullscreen	Run Pi Presents in full screen mode.
-o --home <arg>	Location of the Pi Presents 'data home' directory e.g. /media/USBSTICK or /home/pi/my_data If this option is not specified it defaults to the users home directory.
-d --debug	Fatal (system) errors and Profile errors produce alerts on the display. On acknowledgment Pi Presents exits because it cannot continue.

	<ul style="list-style-type: none"> • Profile errors are errors detected in the profile. • Fatal errors are errors in the Pi Presents software. These could be a side effect of an undetected Profile error. <p>The pp_editor validate menu option will detect many Profile Errors. Tell me about ones that are not detected by the editor.</p> <p>Log output is also displayed in the terminal window if Pi Presents is started from one, and is also reported in the file</p> <p style="text-align: center;">/pipresents/pp_log/pp_log.txt</p> <p>If the -d option is not used then the Log output is Fatal Errors, Profile errors, and Warnings. if the -d option is used then log output also includes a log of the operation of Pi Presents suitable for debugging profiles.</p> <p>Warnings are detected intermittent problems that Pi Presents can recover from. Most of them are due to random crashes by omxplayer.</p>
-d --debug <arg>	Using the debug option with <arg> gives finer control of logging, see Section 19
-v --validate	Validate the profile when Pi Presents is started.
-l --liveshow <arg>	<p>Liveshow tracks are always played from the directory pp_live_tracks in the data home specified in the -o option defaulting to</p> <p style="text-align: center;">/home/pi/pp_home/pp_live_tracks</p> <p>If the -l option contains a directory path this is used as a second source for liveshow tracks.</p> <p style="text-align: center;">e.g. /home/pi/mylivetracks</p>
-s --screenize <arg>	This command line option allows shows to be developed using a monitor which has a different pixel dimensions than the target monitor. <arg> is width*height where width and height are positive integers. See Section 4.2.5 for use of this option.
--manager	Used by Manager for Pi Presents to disable terminal messages. Should not be used by users.

6.2 Specify a Profile

The --profile (-p) command line option specifies the profile to use. This is the name of the profile directory in the pp_profiles directory. If the option is omitted an error will be produced.

The prefix pp_ means nothing special other than denoting files provided by the developer.

6.3 Specify a Home Directory

The `--home (-o)` command line option specifies the location of Pi Presents data home. If the option is omitted it will default to the users home directory.

USB sticks can be used to hold profiles and media. They will be assigned mount points in the `/media` directory.

e.g. `/media/KINGSTON`

Raspbian will auto mount USB sticks if they are present at power on, or plugged in afterwards. If Pi Presents is started at power on it takes up to 10 seconds for the drive to be mounted; Pi Presents allows for this.

Raspbian will compute the mount point from the name of the drive on the stick. If preparing the USB Stick on a Windows machines it appears Windows converts all entered drive names to upper case.

6.4 Using GPIO to Control Pi Presents

GPIO control is enabled when a `gpio.cfg` file is present in the `/pp_io_config` directory in a profile.

`sudo` need not (and must not) be used.

If the `gpo.cfg` file from `/pipresents/pp_resources/pp_templates` is copied to a profile, then pins of the P1 connector are bound to the following symbolic names. These names are used in examples by mediashows, menus and special commands. The bindings can be modified as described in Section 13.2.1

Pin of P1 connector	Symbolic Name	Command
P1-15	pp-down	down
P1-16	pp-up	up
P1-18	pp-play	play.
P1-22	pp-pause	pause
P1-7	pp-stop	stop
P1-11	PIR	Used as a mediashow Start Trigger in some examples
P1-12	pp-shutdown	Press for 5 seconds to close Pi Presents and shut down the Pi.

Using this file the GPIO pins are configured as edge triggered inputs with internal pull-up resistors and require the following device characteristics:

- Push buttons should be mechanical, push to make (normally open), and be connected between the GPIO pin and 0 volts. The contact will close when the button is pressed so it is set to falling edge trigger.
- I have based the PIR entry on PIR's used in UK burglar alarms. These have Normally Closed relay contacts which should be connected between the GPIO pin and 0 volts. The relay contact will open when movement is detected

so the entry is set to rising edge trigger and, because it is a relay contact, the a pull up resistor is configured to avoid radio signals causing false triggering.

Inputs can be changed from normally open to normally closed and vice versa by changing the edge which is used for triggering as described in Section [13.2.1](#)

A 330 ohm resistor is series with the button or PIR is recommended to protect the Pi should the inputs inadvertently be used as outputs.

GPIO Pin ----- 330R ----- contact ---- 0 volts

Be very careful not to connect a GPIO pin to the +5volt pin; it is likely to fry your Pi

There is software contact de-bouncing which is set with a small hysteresis. If you have problems with contact bounce increase the THRESHOLD of the appropriate pin by modifying gpio.cfg (See Section [13.2.1](#)).

6.5 Disable Screen Blanking

To disable screen blanking you must first install xset which is part of the x server utilities package, this should [already be installed in Raspbian](#).

```
sudo apt-get install x11-xserver-utils
```

You can then use the --noblack command option to disable screen blanking.

6.6 Start Pi Presents when Power is applied to the Pi

This will function only if you have set 'boot to desktop' using raspi-config.

- If you are using Raspbian Jessie edit the file

[/home/pi/.config/lxsession/LXDE-pi/autostart](#)

Note: The directory .config is already present in the image but you will need to select 'Show Hidden Files' in the File Manager to see it.

Add the following line below the last line to start Pi Presents with the required options:

```
/usr/bin/python /home/pi/pipresents/pipresents.py -o /home/pi -p  
myprofile
```

6.7 Shutdown the Raspberry Pi from the GPIO

Press the Shutdown button for more than 5 seconds. Pi Presents will exit and safely shut down the Pi.

The RPi can also be shut down immediately in other ways as described in Section 7

7 Exiting Pi Presents and Shutdown of the RPi

There are many ways to exit Pi Presents and shut down the Pi from Pi Presents:

- The operation of pressing a GPIO button for 5 seconds is achieved by binding a GPIO pin to the special symbolic name pp-shutdown. See Section [13.2.1](#). Only a pin can be bound to pp-shutdown as it needs to be asserted for 5 seconds as an anti-tamper measure.
- Bind a key, pin or click area to the internally defined symbolic name pp-shutdownnow as described in Section [13.2](#). An input event with this symbolic name will cause the Pi to shut down immediately. (See below)
- Use the Show control command 'shutdownnow' to shut down the Pi immediately from within a show, See Section [8.2](#)
- Use the Time of Day Scheduler
- Remotely via an OSC command

There are many ways to exit Pi Presents

- Bind a key, GPIO pin, or click area to the internally defined symbolic name pp-exitpipresents. (see below)
- Use the Show control 'exitpipresents' command to exit Pi Presents from within a show.
- Use the Time of Day Scheduler
- Remotely via an OSC command
- When developing use Ctrl+Break, Alt+F4 or the Close icon on the Pi Presents window

Symbolic name	Use
pp-exitpipresents	Exits Pi Presents having closed all shows.
pp-shutdownnow	Exits Pi Presents and the safely shuts down the Raspberry Pi.

8 Controlling Shows

8.1 Controlling Track Movement in Individual Shows

Control of an individual show uses commands such as up, down, play and stop call return to control the movement between individual tracks in a show. There needs to be some method of connecting these commands to the physical input devices such as keys and GPIO buttons. This is a two stage 'binding' or association process:

In configuration files associated with a particular input device (e.g. keys.cfg, gpio.cfg) the physical keys are bound to symbolic names. e.g. for gpio falling_name = myname

In the Controls field of shows and tracks the symbolic name is bound to the command e.g. myname up.

To re-iterate in more technical detail. In Pi Presents the physical devices that are employed are separated from the core operations on shows and tracks. The physical devices have drivers which take the physical inputs and produce **input events** which are identified by **symbolic names**. The shows and tracks receive these **input events** and convert them into the Commands that control the show.

The two associations (bindings) take place in:

- Configuration files associated with the drivers, e.g. gpio.cfg. These define which inputs produce input events and their symbolic names.
- The Controls field in shows and tracks. These associate symbolic names of input events with the Commands, Run-Time Controls, and Triggers used by shows and tracks.

Each type of show has a set of commands which are defined in the section describing the show. Mediashows and Menus have commands such as play, pause, up and stop. Hyperlinkshows have commands such as call and return.

In addition to Commands Mediashows and Liveshows have Triggers which are used in some shows as an alternative to commands. These are generally used by gpio to trigger simple responses to buttons or PIR's

All shows also have Run-Time controls which are used by tracks when they are running to allow user control of volume etc.

For outputs a similar system is employed. Animation commands include symbolic names. Output device drivers use configuration files such as gpio.cfg to convert the symbolic names to physical outputs.

Controls and Subshows

A show can have sub-shows; input events are passed down from show to subshow so they affect the currently lowest level show and its tracks.

A top level show is started when Pi presents starts, from Show Control commands, from the time of day scheduler or remotely via OSC; it has no parent. In version 1.2 the bindings in the Controls field of a top level show were inherited by its subshows. In version 1.3 this is not the case and each show must have its own set of Control bindings.

8.2 Opening and Closing Shows and Show Control

There are four methods to open and close shows:

- Start one or more shows by including their show references in the Start Shows field of the Start Show. All shows specified in the Start Shows field of the Start Show will be run when Pi Presents starts.
- Open or close shows using the Time of Day Scheduler.
- Use a Show Control command in the Show Control field of a track to open or close a show e.g.

```
open myothershow
close myothershow
```

- Open and Close a show remotely using the Open Sound Control protocol

Shows opened by one method can be closed by any other method.

Only one instance of a show can be running at a time. Attempts to open a show that is already running will be ignored.

It is possible to start Pi Presents with no shows running. Additionally if all shows are closed Pi Presents will continue running with a blank black screen to allow remote control or time of day scheduler to open further shows.

Show Control Commands are placed in the Show Control fields of tracks.

Command	Use
open <show-ref>	Opens the specified show. The command will be ignored if the show is already running
close <show-ref>	Closes the specified show. The command is ignored if the show is not running.

In addition to opening shows Show Command fields can be used to close Pi Presents or shutdown the RPi (Section 7). The field is also used for remote control commands using the OSC protocol (Section 14)

8.3 Time Of Day Scheduler

The time of day scheduler opens and closes shows. It will also exit Pi Presents or shutdown the Raspberry Pi. The scheduler is enabled and controlled by the presence of the file schedule.json in the profile.

The schedule.json file is a text file which can be edited by Leafpad. However json is a difficult and unforgiving format so edit with care; I hope to provide an extension to pp_editor.py soon. There is a template schedule in

```
/pipresents/pp_resources/pp_templates
```

which you can copy to the profile and edit.

If changing these files be aware that there is little checking of the content of this file by Pi Presents. If you modify the file run pipresents.py from a terminal window using simulated time so that any Python error messages can be displayed.

schedule.json has a 'show-ref' section for each show to be controlled. There is an additional pseudo show section called pp-core which is used to control Pi Presents.

Each show-ref section has 4 sub-sections everyday, weekday, monthday and specialday. Each has a number match criteria (day) with an associated list of times (times). There is an example in the example profile pp_timeofday_1p3.

Pi Presents prepares a schedule for today when Pi Presents is started or at midnight. For each show-ref it considers each sub-section in the order below matching today's date with the match criteria:

- everyday always matches.
- weekday – matches if today's day of the week is in the list of weekdays
- monthday – matches if today's day of the month is in the list of numbers
- specialday – matches if the today's date is in the list of dates

When a match occurs the times are remembered and their associated commands are **remembered**. If a later sub-section also matches it overrides the previous matches for the specified show. A show with an empty list of times can be used to delete times from a previous match. When all matches are done the schedule for today is prepared by sorting the times.

Times can be hh:mm:ss or hh:mm using a 24 hour clock. Commands are open and close and are associated with the name specified in the show-ref section.

The pp-core show-ref section is different; it is used to control Pi Presents. The matching process is the same and a schedule for today is produced. The commands that can be used only in this section are:

exitpippresents – exit Pi Presents
shutdownnow – exit Pi Presents and shutdown the RPi safely.

If Pi Presents is started in the middle of a day then it will try to catch up by going through today's schedule for each show and starting the show immediately should the show need to be running at the current time. This can result in exit of Pi Presents or shutdown of the RPi immediately if pp-core times are set inappropriately.

To test the schedule with a real clock is difficult thus the field simulate-time can be set to yes instead of no. If yes then the simulated time you wish to start the test should be specified. If simulate-time is yes then today's schedule will be displayed on the terminal if Pi Presents has been started from it.

The scheduler should be time zone agnostic. I have not investigated the effect of changing to/from Daylight Saving Time.

8.4 Concurrent Shows

Pi Presents can run two or more shows concurrently. The shows appear to run in parallel. All concurrent shows use the same screen area but the Show Canvas field of a show can help to separate shows to different parts of a screen. There are some limitations on concurrent shows due to the power of the RPi and limitations of the operating system. Some uses of concurrent shows:

- Providing a background audio track to a slideshow.

Use two mediashows, one with a manually controlled slideshow and the other with the audio tracks. The latter will need the controls disabled using Disable Controls if there is any customer interaction with the former.

- Using a single display for a number of independent shows, maybe a slideshow to the left, a user controlled menu to the right and a strip showing the current time and date at the top.
- Being really thrifty and doing two completely different tasks with the same Pi, perhaps a slideshow in the Visitor Reception with Child show facilities, and a dummy talking in a museum exhibit triggered by a PIR using a single shot mediashow.

8.4.1 Control with Concurrent Shows

Pi Presents can run two or more shows concurrently. The concurrent shows appear to run in parallel and Input Events are passed to all concurrent shows. This is essential but if not managed carefully can lead to some undesirable effects. Most of the common situations can however be addressed by Disable Controls.

For example, if running a manually controlled mediashow displaying images in parallel with a mediashow providing background music; using Up and Down to move through the images should not skip audio tracks. The solution is to set Disable Controls = Yes in the audio show.

More complicated scenarios can be addressed by editing or deleting the bindings of the Controls for a specific show. Using the Controls field of a show a Command can be bound to a different symbolic name for each show and hence be triggered by a different input. Alternatively individual Commands can be deleted for that show.

8.4.2 Limitations on Concurrency

Concurrent shows have some limitations due to limits built into Linux, MPlayer and OMXPlayer:

- If two show display overlapping images or text the last object to be displayed will overlay older objects.
- Video tracks will always overlay images and text
- If two video tracks overlap then there is likely to be flickering.
- If there is more than one audio or video track to be played that might overlap in time then local audio must be used. This is because omxplayer does not use the alsa mixer. The problem is particularly severe for the artmediashow and artliveshow as the next track is loaded during the playing of the previous track, not at its end. The problematic combinations are complex, the only safe way is to direct all audio to the local headphone socket.
- HD videos may stutter while an audio track is being played.

- Animation outputs are delayed if they are coincident with the start of an audio track or image. The output event will not be missed but will be delayed.
- If CPU load is high omxplayer can take so long to load a video that it times out.

9 Touchscreens and Soft Buttons

9.1 *Controlling Shows with a Touchscreen*

All types of show can be controlled by touchscreens and use soft buttons. Touchscreens or soft buttons are almost essential for Hyperlinkshows, a useful alternative to gpio buttons in Radiobuttonshows and can be used to replace cursor or gpio button control in mediashows, liveshows and menus.

- Click Areas

Click areas are aimed at touchscreens. A 'click area' is an area of the screen which is mouse click sensitive. Although they are called 'click areas' they can best be used with touchscreens and might better be called touch areas. Touching the area of a touchscreen or clicking with a mouse in the area will cause an input event.

Optionally associated with a click area is an image, maybe of a button, which is displayed when the click area is enabled. Alternatively a click area can be transparent and the area defined as a polygon to encompass the shape of the object displayed as part of the background image.

- Soft Buttons

Soft buttons are a poor man's touchscreen. They were used a lot in real time control applications before touchscreens were developed.

Soft buttons are a row of unmarked keys or buttons arranged along the edges of a display screen. The legends for the buttons are software controlled and appear adjacent to the button on the edge of the display area of the screen. Thus different buttons and legends can be applied to each track of a Hyperlinkshow.

Pi Presents allows both of these techniques to be used to control any type of show.

9.2 *Click Areas*

In its screen.cfg file Pi Presents allows the definition of click areas. These are polygonal areas of the screen which are touch or mouse click sensitive. A touch or click will produce an input event identified by a symbolic name.

Click areas can have text, coloured backgrounds, outlines and images.

The presence of a screen.cfg file in the /pp_io_config directory in a profile enables click areas and contains the click areas to be used for every show and track in the application. The click areas to be displayed on each track and show are determined by the symbolic names in the Controls field of the track or show. Configuration of Click Areas is described in Section 13.2.2.

9.3 Soft Buttons

Soft buttons need both a gpio button to be associated with the required Command as described elsewhere and a passive on screen legend for the button configured as described here. The passive legend must appear only on the track that requires it so it needs to be included in screen.cfg as a click area and included in the Controls field of the track with a special command to disable it.

For example to set up a soft button to control movement from a page in a Hyperlinkshow:

- Use the symbolic name bound to a gpio button in the Controls field of the Track.

```
my-button call pig-video
```

- Mount the button next to the screen and set up a 'click area' in screen.cfg, with the symbolic name my-button-legend. Specify its 'points' such that it is displayed next to the button. This click area will display the legend, say 'See a Pig', but must not be clickable.
- To make the click area appear when the track is being displayed but for it not to respond to clicks use it in the Controls Field of the track and give it the command 'no-command'

```
my-button-legend no-command
```

10 Run-Time Commands

OMXPlayer, Mplayer and the uzbl browser have a number of run-time commands which are defined for omxplayer and mplayer in:

<https://github.com/popcornmix/omxplayer>

<http://www.mplayerhq.hu/DOCS/man/en/mplayer.1.html#INTERACTIVE%20CONTROL>

Only operations with a single character from the list of operations in the references are allowed. The single character must be preceded by omx- or mplay- respectively. Multi-character operations are not currently supported.

For example:

```
videovolup omx-+
videovoldown omx--
```

included in the controls field of a show or track adds volume control to omxplayer. videovolup is a symbolic name which can be bound to a key, gpio pin etc. Keyboard keys naturally have repeats; for GPIO pins gpio.cfg can enable repeating for buttons.

For the uzbl browser used to display web tracks the commands useable for run-time controls are defined here:

<http://www.uzbl.org/readme.php>

The command strings should be preceded by uzbl-

For example:

```
largerweb = uzbl-zoom_in
```

will make the content of the browser larger.

11 Providing Dynamic Content in a Liveshow

Pi Presents was not intended for the dynamic supply of media however by popular demand I have included a facility where a 'Liveshow' can play a set of tracks which change during the running of the show .

The New> Liveshow template and New>Artliveshow are working Liveshows. The tracks to be played should be placed in the Live Tracks Directory1 which out of the box is /home/pi/pp_home/pp_live_tracks

Liveshows play audio, video and image tracks. The media file types that a Liveshow recognises are in the first few lines of the file pp_definitions.py.

Tracks could be ftp'ed into the Live Tracks Directory1 using Filezilla or some such.

Advanced Use

Using the -l command line option of Pi Presents it is possible to have a second location containing live tracks; the Live Tracks Directory2. The location of this directory is specified by the -l command line option. The files in the two live tracks directories are combined and sorted by their leaf name.

The directory could be on a remote fileserver (I have not tried this). Alternatively the complete profile for a show could be held on a remote fileserver.

From Version 1.3.1 concurrent shows are allowed hence it is necessary for allow each liveshow to access its own directory. The liveshow profile now has two fields for this purpose, Live Tracks Directory1 and Live Tracks Directory2. If these are not blank their location overrides the default location.

12 Animation Control

All types of track have facilities to control animation. Using commands included in the 'Animation at Beginning' and 'Animation at End' fields an external output can be turned on or off synchronised with the start or end of tracks, with optional delay.

An example of animation commands is shown below.

animate at beginning	1 out1 state on 2 out1 state off 1 out2 state on 6 out3 state on
Clear Animation	no
Animation at End	0 out2 state off

A command has four or more fields separated by spaces and terminated with a newline:

- Delay - seconds as a positive integer or 0.
- Symbolic Name - The name of the output as defined in the gpio.cfg file in the profile (Section 13.2.1).
- Parameter Type
- Parameter - 1 or more fields compatible with Parameter Type.

Pi Presents currently supports GPIO on/off output. The fields are designed to allow extension of this to other types of output. For example pulse width modulation (PWM)

```
0 pwm1 pwm 90 50
```

might set a PWM channel to 90% mark at 50Hz.

GPIO On/Off Output

If the Symbolic Name is to provide on/off GPIO then Parameter Type is 'state' and there must be one Parameter with value 'on' or 'off'

The state of GPIO outputs before Pi Present starts is defined in the Device Tree for the Raspberry Pi. I believe that all outputs are set to off but be careful if there is anything of safety concern such as a mechanical device attached to an output.

All outputs are set to the off state when Pi Presents starts. Commands in Animation at Beginning are executed at the beginning of a track and Animation at End at the end of a track. When the commands are executed the required changes to the GPIO are put in a queue for firing at the appropriate time. The time is not affected by pausing a track.

Animation commands in the queue are not forgotten at the end of a track so animation can be extended over multiple tracks. A side effect of this is that it is possible for a GPIO change to happen at the wrong time if the duration of a track is indeterminate. If you want to avoid this and ensure outputs are in a defined state at the end of a track set Animate Clear to yes. If yes then, before Animation at End is

executed, the queue will be cleared of those events that were commanded by the track but not fired.

13 Input/Output Drivers

In Pi Presents Input and Output is independent of the core execution of shows and tracks. This will eventually allow extension to different input/output devices by writing additional I/O plugins. These plugins will have a standard API and will use configuration files to enable and configure them. The API is not yet designed in detail but the functionality that makes the drivers independent and configurable is.

An input driver accepts inputs from a physical device, pre-processes it and produces input events with symbolic names. An output driver takes output command (e.g. out1 state on) having a symbolic name and parameters and generates the physical output.

There are currently drivers for:

Device	Configuration File	Use
keyboard	keys.cfg	Controls which keyboard keys are used by Pi Presents.
gpio	gpio.cfg	Enables gpio. Controls which gpio pins are used by Pi Presents for inputs and outputs and configures the pins
touchscreen	screen.cfg	Enables touch/click sensitive areas on a touchscreen/monitor and configures the look of the touch sensitive areas.
remote control using OSC	osc.cfg	Allows Pi Presents to control or be controlled by other units.

13.1 Enabling an I/O Device

To enable an I/O device the appropriate configuration file must be placed in a /pp_io_config directory in a profile. There are sample configuration files in /pipresents/pp_resources/pp_templates which can be copied and edited. These files also contain information, additional to that here, on how to use them.

keys.cfg is treated slightly differently to make it easy for beginners. There is a keys.cfg file in /pipresents/pp_config which is used as a fallback if there is no keys.cfg in the profile. This file can be copied to the profile and edited to override the fallback.

The .cfg files are all text files which can be edited by Leafpad. Do not edit the files inside /pp_templates; if you wish to change their content then copy the file to a profile.

e.g. inside /home/pi/pp_home/pp_profiles/myprofile/pp_io_config

If editing these files be aware that there is little checking of the content of these files by Pi Presents. If you modify the file run pipresents.py from a terminal window first so that any Python error messages can be displayed.

13.2 Configuring Inputs and Output Drivers

Section 8.1 describes the input system of Pi Presents and how external events are converted to input events. Most of the responses are configurable using the files described in this section.

When editing configuration files you will need to supply symbolic names. It is advisable not to create names beginning with pp- to avoid clashes with names used by Pi Presents.

13.2.1 Configuring GPIO Pins

BEWARE: Accidentally using a pin as an output with the output shorted to ground or +3.3 volts might fry your Pi, use a series resistor on every input and output for protection.

The configuration of the GPIO used by Pi Presents is defined in the file `gpio.cfg`. The file in `/pipresents/pp_resources/pp_templates/gpio.cfg` is an example which, when copied to a profile, configures Pi Presents for the buttons and the PIR described in this manual and used by the examples.

`gpio.cfg` maps physical Pi GPIO input and output pins to the symbolic names of inputs and outputs used by Pi Presents. It also configures the input pins.

A section for every pin must be present in the file. A pin with `direction=none` is ignored.

Inputs

Each pin can generate an event having the specified symbolic name in any of four ways:

- rising edge - An event with the symbolic name specified in 'rising-name' is generated when the input changes from 0 to 1 (0 volts to 3.3 volts)
- falling edge - An event with the symbolic name specified in 'falling-name' is generated when the input changes from 1 to 0 (3.3 volts to 0 volts)
- one state - An event with the symbolic name specified in 'one-name' is generated at 'repeat' intervals while the input state is '1' (3.3 volts). The first event happens after 'repeat' interval. If you want the input to respond immediately set the rising edge event to the same symbolic name.
- zero state - An event with the symbolic name specified in 'zero-name' is generated at 'repeat' intervals while the input state is '0' (0 volts). The first event happens after 'repeat' interval. If you want the input to respond immediately set the falling edge event to the same symbolic name.

If you do not want the event to be generated leave the symbolic name blank

For the purposes of this manual and the examples gpio.cfg is set up to allow normally open push buttons connected to ground (0 volts) and a PIR with a normally closed relay contact connected to ground.

Outputs

The logical 'ON' state produces +3.3 volts. The logical 'OFF' state produces 0 volts

Pi Presents initialises GPIO outputs to 0 volts so it is best to design relays etc. for positive logic (which many of those on the market are not).

13.2.2 Configuring Touch/Click Areas

The file screen.cfg defines the areas of the screen that will become mouse click or touch sensitive. Click areas are not limited to Hyperlinkshows; they can be used for all types of show.

The file consists of a number of sections each with a unique name. The name can be anything but must be unique within the file.

All fields in each section must be present. The fields of each section are used as follows:

- name - The symbolic name of the click area. Each command in the Controls field of a track or show has a symbolic name. When the track in a show is played the click areas in the Controls field are displayed on the screen and when an area is clicked the symbolic name will identify an input event.
- points - this is a set of x y pairs that defines the points of a polygon bounding the area. The polygon is automatically closed so a quadrilateral will have 4 (not 5) x,y pairs. There must be at least three pairs of points. For details of this and other attributes see <http://effbot.org/tkinterbook/canvas.htm>
create_polygon For rectangular click areas the points may be specified as

x1+y1+w*h

where xi, y1 are the coordinates of the top left corner of the click area and w,h are the width and height, all in pixels.

- fill-colour, outline-colour

Specifies the look of the polygon. Use a blank field for transparent

- text, text-font, text-colour

If text is not blank then the text is written centred in the polygon.

- image, image-width, image-height

An image to be used as a button. Paths relative to pp_home are supported (+/) or specify the complete path. The image will be warped to fit - image-width and image-height and centred on the polygon.

13.2.3 Configuring Keyboard Keys

Keyboard keys are bound to symbolic names in the file `keys.cfg`. The file has a number of lines with the format

`condition = symbolic name`

The conditions are defined in effbot.org/tkinterbook/tkinter-events-and-bindings.htm in the <Return>, a, and <Shift-Up> sections. The conditions and the symbolic names are case sensitive.

In addition to these bindings the printing characters on the keyboard, (the ones obeying the <Key> condition in the reference), are automatically bound to the symbolic name `pp-key-x`

e.g the 'a' key produces `pp-key-a`

Automatic binding of a printing key can be overridden by a line such as `a = pp-pause`.

Note: <any control> = `pp-shutdown` will not work as `pp-shutdown` needs to be asserted for 5 seconds. It is useful for gpio only. However you can shut down the Pi from Pi Presents using a show control command or by binding a key to the special symbolic name `pp-shutdownnow`.

The fallback `keys.cfg` in `/pipresents/pp_config` has the following bindings:

Symbolic Name	Bound Key	Command
<code>pp-down</code>	Cursor Down	<code>down</code>
<code>pp-up</code>	Cursor Up	<code>up</code>
<code>pp-play</code>	Return	<code>play</code>
<code>pp-pause</code>	Spacebar	<code>pause</code>
<code>pp-stop</code>	Escape	<code>stop</code>
<code>pp-terminate</code>	CTRL-BREAK	Abort Pi Presents.

14 Remote Control using OSC (very beta)

Version 1.3.1 allows Pi Presents to control instances of Pi Presents on other computers or to be controlled by any computer that supports the Open Sound Control protocol, including smartphones and tablets.

OSC is a lightweight flexible protocol which was originally intended to control musical instruments. It can do much more than this and is supported by many multi-media equipments.

The implementation in this version of the software is experimental. It works, has been reliable in a small number of tests, but is subject to change as I develop it further,

hopefully with input on how to use it from yourselves. Known deficiencies and question marks are:

- The current version supports only one slave per master and a particular unit can be a master or a slave, not both.
- It uses the UDP protocol not TCP. This makes it fast and compatible with more third party systems but UDP is what is called an unreliable protocol – messages are not guaranteed to arrive or be in the order they were sent; whether this matters for small systems on a local LAN remains to be seen.
- Symbolic names giving device independence are not supported. The raw command is required in the Show Control field.

14.1 OSC Messages in Pi Presents

OSC messages are sent by master units and received by slave units. The messages have an address field and zero or more data fields. For efficient transmission these are encoded into a binary format and decoded by the receiver.

To Pi Presents an OSC message is just a text string containing an address and zero or more arguments e.g.

```
/pipresents/myunit/core/open myshow
```

The OSC address is hierarchical, for example

```
/pipresents/unit1/core/open
```

This means:

- /pipresents - the message is for equipments that can understand Pi Presents type OSC messages. All other units will ignore this message.
- /unit1 - message is for Pi Presents unit1 and will be ignored by any other units.
- /core - the message is for the core of Pi Presents, the part that controls shows and controls tracks via input events.
- /open is the command to open a show. The complete message will have the show-ref as an argument

Sub-address	Data	Example	Use
/core/open	show-ref	/core/open myshow	Open the show show-ref
/core/close	show-ref	/core/close myshow	Closes the show show-ref
/core/event	symbolic name	/core/event pp-stop	Generates an input event with the symbolic name
/core/exitpipresents	<blank>		Exits Pi Presents
/core/output	animation command without	/core/output out1 state on	This provides a simple output passthrough system. The arguments are any data used by the Animation

	the initial delay		field without the leading delay parameter.
SYSTEM			Pi Presents responds to these. Sent only for testing by <code>pp_oscremote.py</code>
/system/loopback	<blank>	/system/loopback	Causes the slave to return a message to the master.
/system/server-info	<blank>	/system/server-info	Causes the slave to report information about itself.

14.2 Sending and Receiving OSC Messages

To send an OSC message place it in the Show Control field of a track. Other than configuring OSC, receiving messages is automatic.

14.3 Configuring OSC

OSC is enabled and configured by the presence of an `osc.cfg` file in the directory `/pp_io_config` in a profile. `pp_editor.py` has a menu option to create, edit and delete `osc.cfg` files in the opened profile. When using the editor you will be asked whether you want the unit to be a master or slave.

Fields for Master Unit

Field	Example	Use
Name of this unit	master-pi	The /unit part of the address of this unit. Used in matching messages sent to this unit.
IP of This Unit	192.168.1.101	Used to set up the server used by this unit to listen for messages. Slave must send reply messages to this address
Listening Port of this Unit	9001	Used to set up the server used by this unit to listen for messages. Slave must send reply messages to this address
Name of Controlled Unit	slave-pi	The /unit part of the address of the unit that is controlled by this unit (<code>/pipresents/slave-pi</code>).
IP of Controlled Unit	192.168.1.102	Used to send messages to the slave unit
Listening Port of Controlled Unit	9002	Used to send messages to the slave unit

Fields for Slave Unit

Field	Example	Use
Name of this unit	slave-pi	The /unit part of the address of this unit. Used in matching messages sent to this unit.
IP of This Unit	192.168.1.102	Used to set up the server used by this unit to listen for messages. master must send messages to this address
Listening Port of this Unit	9002	Used to set up the server used by this unit to listen for messages. master must send messages to this address
Controlled by Unit	master-pi	The /unit part of the address of the unit that controls this unit (<code>/pipresents/master</code>). Used to send replies to

		loopback and server-info
IP of Controlled By Unit	192.168.1.101	Used to send replies to loopback and server-info
Listening Port of Controlled by Unit	9001	Used to send replies to loopback and server-info

14.4 oscremote.py and oscmonitor.py

pp_oscremote.py and pp_oscmmonitor.py are two stand-alone programs which I developed to test the OSC interface of Pi Presents. They run on Windows, Linux or Raspbian. They contain code which could be used as the basis for other applications; however this is my first attempt at socket programming so beware! Anyone interested in writing an Iphone or Android App for controlling Pi Presents?

oscremote.py sends commands to Pi Presents while oscmonitor.py monitors and displays OSC messages that Pi Presents has sent. Out of the box they are both configured to talk to the unit /pipresents using localhost (127.0.0.1). By altering the configuration you can connect oscremote.py and oscmonitor.py together.

By running them from different terminal windows you can run Pi Presents and a remote or a monitor on a RPi. Two examples have been set up as demonstrations of OSC that can be run in this way out of the box.

pp_multiwindow_1p3 has an osc.cfg file which allows it to be controlled by oscremote.py; just remove all the shows from the Start Show and control them from oscremote.py.

pp_showcontrol.py has an osc.cfg file which allows it to send output commands from the show control field of the two image tracks to mirror control of the audio show. These can be monitored by oscremote.py (remove shows from Start Show).

I have not yet tried sending commands between two copies of Pi Presents running on different machines. It should work though.

oscremote.py

Before using oscremote.py it needs to be configured using the OSC>edit menu to configure the communication. Out of the box it is set up to communicate with Pi Presents using localhost (127.0.0.1).

The buttons each generate all or part of an OSC message in the Message to Send field; you will need to add show refs, and output commands and press Send. Messages sent and any replies appear in the status field.

The profile>select menu option allows a Pi Presents profile to be selected; remember to set the path to data home in the options first. The shows in the profile are shown in the Shows tab; selecting one of these before the open or close button saves you remembering or typing the show-ref, nothing more.

oscmmonitor.py

Just displays messages it receives in the Status window. Out of the box it is configured to receive from Pi Presents using localhost (127.0.0.1).

15 Track Plugins

Track plugins are python code modules with a documented interface specification that can be coded by the user to enhance the displays of Pi Presents. Track Plugins allow dynamic display of information such as time, weather, and tickers. It is likely that information will be scraped from web sites or from RSS feeds.

Track plugins are executed by Players. Their primary use is to display dynamic content. This can be achieved in two ways:

- By modifying or replacing the media to be played.
- By writing dynamic information direct to the Pi Presents display

Track plugins are stored in the directory /pipresents/pp_track_plugins in .py files. It is recommended that the plugin file name is preceded by your initials so that there are no clashes with python modules (e.g. time.py will clash with the standard time module.)

Using Plugins

If the plugin is required for a track then the 'Plugin Configuration File' field of a Player should contain the name of a Plugin Configuration File e.g.

+ /media/krt_image_text.cfg.

Relative paths are allowed. The media produced by the plugin must match the type of the track to be played.

The Plugin Configuration File allows the same plugin to be called with different parameters. It must contain at least

```
[plugin]
plugin = pluginname
type = image      # required if the plugin is to be used in a liveshow
```

pluginname is the name of the python module containing the plugin (the filename without .py).

The configuration file may define further parameters which will be available to the plugin code via the dictionary plugin_params:

e.g.

```
[plugin]
plugin = krt_image_text
type = image      # required if the plugin is to be used in a liveshow
# optional
text = text to display
```

The plugin code can read the value of text so by using a number of configuration files all calling the same krt_image_text plugin you can have a different text in each track.

You can use plugins in Liveshows. To do this the plugin must be specially written. To use a plugin in a liveshow just copy the plugin configuration file to `pp_live_tracks`. To be used in liveshows plugin configuration files must specify the track type so that Pi Presents knows which type of track to play.

Writing Plugins

As of Version 1.3.1 the track plugin API has changed and improved. The changes are necessary so that the plugin way of displaying information matches the pre-loading required for gapless transitions. API details are in `pp_example_plugin.py`

I have provided three examples of plugins

- `pp_example_plugin.py`
This contains the API documentation. The example is long as it addresses all the types of track that a plugin might be used for, and also liveshows.
- `krt_image_text.py`
An example that adds text to an image.
- `krt_time.py`
Modifies the screen directly to display the current time.

The examples use:

- The Python Imaging Library (PIL). The handbook is at <http://effbot.org/imagingbook/pil-index.htm>.
- Tkinter canvas operations are documented at <http://effbot.org/tkinterbook/canvas.htm>

The functional interface varies slightly depending on the type of the track:

- `image` - the plugin will be supplied with the path to a file containing an image (e.g `picture.jpg`). An image track file must be returned and must be able to be displayed using PIL and Tkinter.
- `audio` - the plugin will be supplied with a file containing audio. An audio track can optionally be returned that is playable by `mplayer`. Blank is allowed and no audio will be played.
- `video` - the plugin will be supplied with a file containing a video. A video track must be returned that is playable by `omxplayer`. The plugin can also write directly to the canvas but be aware that the video will appear over the top of the text so use `Video Window` to window it.
- `web` - the plugin will be supplied with a file containing html code (whatever `uzbl` supports for rendering). A html file should be returned that is playable by `uzbl`. The plugin can also write directly to the canvas but be aware that the browser will appear over the top of the text so use `Web Window` to window it.
- `message` - the plugin will be supplied with the text that would have been displayed, this can be modified and returned. The plugin can also write directly to the canvas.

16 Remote Management

The programs `pp_manager.py` and `pp_web_editor.py` are used for the remote management of Pi Presents from a web browser running on any computer on the same network as Pi Presents. Each program runs a web server serving an interactive web page that looks similar to the normal gui of the Pi Presents editor.

- Manager - `pp_manager.py` provides facilities to select and run Pi Presents profiles. It is also possible to upload and download profiles and media.

The Manager can also replace the normal autostart mechanism (Sect. 6.6) with a mechanism that can remotely modify the profile to be auto-started.

- Web Editor - `pp_web_editor.py` is a near clone of the Pi Presents Profile editor.

16.1 Setting up for Remote Use

Before first using the two programs:

- The files `/pipresents/exit_pp.sh` and `/pipresents/exit_ed.sh` must be made executable.
- The file `/pipresents/pp_config/pp_web.cfg` must be edited using a text editor. Entries in the section `[manager-editable]` can subsequently be edited by the Manager.

Field	Example	Description
[manager-editable]		
<code>media_offset</code>	<code>/media</code>	The offset from the <code>pp_home</code> directory to which media will be copied or uploaded. So <code>/media</code> will store media in <code>...../pp_home/media</code>
<code>profiles_offset</code>		The offset from the <code>pp_profiles</code> directory to which profiles will be copied or uploaded. Generally left blank. This offset also determines the directory for the Manager's profile selection list. e.g. Specifying an offset e.g. <code>/test_profiles</code> will store profiles in <code>.....pp_home/pp_profiles/test_profiles</code>
<code>options</code>		When starting Pi Presents from the Manager append the specified options. Sect 6.1 The <code>-p</code> and <code>-o</code> options should not be specified. The Manager automatically adds these.
<code>autostart_path</code>	<code>/my_profile</code>	If not blank the Manager will autostart Pi Presents when it is started and before its web server is started. The field specifies the profile to be used when autostarting Pi Presents..So <code>/my_profile</code> will start the profile in

	pp_home/pp_profiles/my_profile
autostart_options	-fb	When autostarting Pi Presents appends the specified options. (Sect 6.1) The -p and -o options should not be specified. The manger automatically adds these.
[manager]		
home	/home/pi/pp_home	The path of the Pi Presents Home Directory. It must end in pp_home.
source-top	/media/USBSTICK	This field limits the files for the following actions to those in this directory and below: <ul style="list-style-type: none"> • Source for copying media files. • Source for copying profiles <p>Copying is expected to be from a USB stick and this option limits the files that can be seen to those on the stick.</p> <p>It also determines the starting point for the copy file selection dialog.</p>
unit =	Cafe Pi	A name which appears on the Manager's screen. Useful if you have more than one Pi under management.
port =	8081	The port used by the editor. Should not need to be changed.
username =		If a username and password is specified then to use the Manager it will be necessary to login
password =		
[editor]		Configure the Web Editor
port = 8082		The port used by the editor. Should not need to be changed.
username =		If a username and password is specified then to use the Web Editor it will be necessary to login
password =		
[network]		
ip	192.168.1.108	The ip address of Raspberry Pi that is running the Editor or Manager. Mouse over the Network Icon on the Task bar to find the ip If you want to use the Manager and Web Editor from the Pi itself then 127.0.0.1 will work and removes the need to edit the configuration if the dynamic ip changes.

16.2 Using the Manager

Start the Manager by the command `python pp_manager.py` from a terminal window opened in the `pipresents` directory. This will start the Manager's server.

The Manager can be accessed from a browser using its IP address and port e.g. `192.168.1.108:8081`

The manager will show a list of profiles from the profiles directory.

Buttons

- Start Button – Starts the profile selected from the displayed list of profiles.
- Exit Button – Exits Pi Presents
- Refresh Button – Refreshes the list of profiles; required if a profile is added or deleted from the Pi rather than from the Manager.

The line above the buttons displays the Unit and the run state of Pi Presents. When Pi Presents has exited an exit code is displayed in brackets:

- 100 – Normal Exit
- 101 – Pi Presents Closed by external event (usually from the Pi)
- 102 – Pi Presents exited due to a runtime error. The error message can be accessed in /pipresents/pp_logs/pplog.txt

Menus

- media>copy - Copy media files from the selected directory into pp_home into the sub-directory defined by media_offset.
- media>upload - Upload media files from the browser into pp_home into the sub-directory defined by media_offset.
- profile>copy – copy a profile directory into pp_home/pp_profiles into the sub-directory defined by profiles_offset.
- profile>upload - upload a profile directory into pp_home/pp_profiles into a directory defined by profiles_offset. The profile directory must be archived into a .zip file before upload. The Manager will automatically unzip the archive.
- profile>download – download the selected profile to the browser. The profile is downloaded as a zip archive.
- editor>run – Run the Web editor. This will start the server on the Pi. To access the editor open a new browser tab at the IP address and port of the editor e.g. 192.168.1.108:8082
- editor>exit – Exit the Web Editor
- options>manager/ options>autostart – Edit the Manager's configuration

Using Autostart

If the autostart_path configuration entry is not blank then, when the Manager is started, Pi Presents will be run with the configured profile and options.

To achieve this Raspbian's autostart file (Sect. 6.6) should have the command:

```
usr/bin/python /home/pi/pipresents/pp_manager.py
```

instead of a command to run Pi Presents.

After the profile has been run the Manager's server will be started and can be accessed in the normal way via a browser. It will show that the auto-started profile is running.

It is intended that the Manager be run continually in parallel with Pi Presents. If this is consuming too much processor power then edit `update_interval=0.1` (secs) to say 0.5 in the last line of `pp_manager.py`

16.3 Using the Web Editor

The Web Editor is normally started and stopped from the Manager however it can be run independently :

- Use the command `python pp_web_editor.py` from a terminal window opened in the `/pipresents` directory. This will start the Web Editor's server.

In either case the Editor can be accessed from a browser using its IP address and port e.g. `192.168.1.108:8082`

The Pi hosted editor has not been removed (yet!). Both editors use the same configuration file `pp_editor.cfg`.

The Web Editor provides the same functions as the Pi hosted editor, except updating the version of a profile. There are a few minor differences:

- Navigation when selecting files and directories. To open a directory click on its icon. To select a directory click on its name, to select a file click on its name.
- When editing a show or track clicking on the Select File button will swap the editor with a file selection window instead of a pop-up.
- Colour names can be used but if the colour chooser is used the name will be converted to the rgb hex code.
- The colour chooser is not supported by Internet Explorer or Epiphany
- Browsers layout pages differently. IE and Epiphany seem to be the main culprits for showing different layouts. The Manager and Editor have been developed using Firefox. Open source Firefox for Linux is called Iceweasel and runs well on B2 or B3 models. Install it with `sudo apt-get install iceweasel`.

17 Hardware Requirements

Pi Presents can be used with either [Model 1](#), [Model 2](#) or [Model 3](#) Pi's. However for a better performance for videos or images a [Model 2](#) or [Model 3](#) Pi is preferred. The GPIO pins have been chosen such that they are version agnostic. 256MB Pi's can be used provided image size kept to the 1 megapixel region.

If you wish to play videos 128MB of GPU memory must be used.

Display of images is slow with earlier Pi's. A one megapixel images takes a couple of seconds to display. The one 10 megapixel image I tried took 10 seconds to display and crashed a 256MB Pi. Larger images, greater than the screen pixel dimensions, will do nothing to improve the picture and will take longer to display even on 512MB

machines; I use the brilliant Faststone Photo Resizer <http://www.faststone.org/FSResizerDetail.htm> to reduce the size of images on a Windows PC.

Use the HDMI or headphone output for audio. Amplification and volume control will need to be provided in external hardware to suit the application.

18 Updating Pi Presents

For safety take a copy of the pipresents folder and any data before doing updates.

Download Pi Presents from github and install it as described in the README.md file. Data stored in /home/pi/pp_home will not be affected. Do not store your data or modify any files in the pipresents directories as they may be overwritten.

As Pi Presents develops new fields will be added to the profile definition and others deleted. To control this, the three elements - Pi Presents, the editor and the profiles - must have the same version. Pi Presents will object if a profile with the wrong version is used. To correct this open the profile in the editor, it will automatically update the version of the profile and its fields. There will be a few residual update tasks that cannot be automated; read the release notes to identify these.

If you have many profiles to update in the same /pp_home then you can use the 'tools>update all' menu option of the editor to update them.

The profiles in pipresents-examples will be kept compatible with the latest version of Pi Presents. Beware, re-installing these might overwrite profiles you have made.

When updating Pi Presents read the release notes. You may need to update the configuration files and carry out a few tasks that cannot be done automatically.

19 Debugging, Statistics, Bug Reports and Feature Requests

19.1 Statistics Production

Pi Presents will output events to the file /pipresents/pp_logs/pp_stats.txt. The content of the file is suitable for analysing to produce statistics on the use of an application. The file contents CSV and suitable for reading into a spreadsheet. The separator is ; it can be changed in pp_utils.py.

Statistics logging is enabled by the `-d` command line option as described in Section 19.2.

When Pi Presents starts, if the file pp_stats.txt does not exist it will be created and a header line written. If the file exists it will not be deleted, further event rows will be appended to those from the previous run of Pi Presents.

There are four forms of event lines denoted by the command field:

- Start – Indicates the Pi Presents has been started. Date/time and the name of the profile are written.
- start trigger, next trigger – Indicates that a mediashow or liveshow has received a start or next trigger. The date/time and show details are written.
- play child – Indicates that a mediashow or liveshow has received an event to play a child track. Content is as for the events below.
- other commands such as play, call – Menushows, Radiobuttonshows and Hyperlinkshows write date/time, show details and track details when user instigated events are detected.

Show details are show type, show reference, show title. Track details are track type, track reference, track title and location.

Example:

```
"Date";"Time";"Show Type";"Show Ref";"Show Title";"Command";"Track Type";"Track Ref";"Track Title";"Location"
```

```
"2016-02-01";"15:02:43";"";"start";"";"1p3_examples/pp_interactive_1p3"
```

```
"2016-02-01";"15:02:53";"mediashow";"mediashow";"Mediashow";"play child";"menu";"mymenu";"Menu";"
```

```
"2016-02-01";"15:02:58";"menu";"mymenu";"Menu";"play";"image";"";"A Stunning River Scene";"+/media/river.jpg"
```

19.2 Debugging Profiles

The `-d` command line option allows a trace of the operation of Pi Presents to be output to the terminal window and to a log file as described in Section 6.1. The `-d` option has an argument which allows fine control of the log output

Value (Binary)	Value (decimal)	Log Output
1	1	Fatal (System) Errors
10	2	Profile Errors
100	4	Warnings
1000	8	A Trace suitable for debugging profiles
10000	16	A Trace suitable for debugging Pi Presents.
100000	32	A Trace suitable for debugging Pi Presents with instances of Player and Shower
1000000	64	Memory leak monitoring for debugging Pi Presents
10000000	128	Write events suitable for statistics production to stats.txt. See Section 19.1

Without the `-d` option Pi Presents uses a value of 7. With the `d` option but no value specified a value of 15 is used.

Reporting of uncollectable garbage is permanently on. This may result in a message being reported to the terminal when Pi Presents is closed. They indicate that Pi

Presents software is not deleting tracks or shows correctly. I am interested in these reports.

In addition to the trace most of my debugging is by the use of Print statements in the python code. I may have inadvertently left some of these statements in the code resulting in messages on the terminal even if debugging is turned off.

Bug Reports and Feature Requests

Please use the Github Issues Tab <https://github.com/KenT2/pipresents-gapless/issues> to report bugs and ideas for extensions.

I am keen to improve Pi Presents and your input on real world experiences and requirements would be invaluable to me, both minor tweaks to the existing functionality and major improvements.

20 Gotchas and Known Problems

When not fullscreen the Pi Presents window is too small or too large

Edit the following lines in pipresents.py (around line 40).

```
self.nonfull_window_width = 0.6 # proportion of width
self.nonfull_window_height = 0.6 # proportion of height
self.nonfull_window_x = 0 # position of top left corner
self.nonfull_window_y=0 # position of top left corner
```

When using autostart, or running from a desktop shortcut my profile is not found.

In any of these situations it is best to use the full path of pipresents and the data home in commands e.g.

```
/usr/bin/python /home/pi/pipresents.py -o /home/pi -p myprofile
```

Also be aware that the location of the autostart file changed at the 25/12/2014 release of Raspbian [Wheezy and its content is different for Raspbian Jessie](#).

Pi Presents locks up in full screen, how do I escape.

This is usually caused by Python reporting an exception due to incorrect configuration data. To avoid this use validation in the editor and try the show, not full screen, and running from a terminal window so you can Ctrl+C out if Ctrl+Break fails. If this fails, try Ctrl+Z then closing the terminal window.

When Pi Presents crashes it sometimes leaves omxplayer mplayer or uzbl running. You can see this using top. To remove these processes use:

```
killall omxplayer omxplayer.bin mplayer uzbl-core
```

This may leave zombie processes. To remove these it seems necessary to close the terminal window.

Pulling the power has potential for corruption, so the ideal solution to a seemingly complete lockup is to SSH into the Pi from another machine, run `top` (`top -upi`) and kill the python process with the `k xxxx` command. You may need to kill an `omxplayer` process as well.

My video/audio track does not play with Pi Presents

Try playing it using `OMXPlayer` or `MPlayer` from the command line.

Unclutter does not hide the cursor at all times

Azizar reported that he edited `/etc/default/unclutter`, adding

```
-grab
-noevents
```

I have built a gpio input or output device and it is not working with Pi Presents.

There are two stand alone GPIO test programs in the `pipresents` directory `input_test.py` and `output_test.py` so you can test you I/O before blaming Pi Presents! For B+ and 2b you will need to modify the python code to add the additional pins (just uncomment the second definition of pins)

Permission to write to `pp_log.txt` or `pp_stats.txt` is denied.

The log files [are](#) created with the user permissions currently employed. If `sudo` was used in the command that created the file it is owned by root. Delete the file with user root and then create with user Pi or run `pipresents.py` without `sudo`.

When using the editor on Windows I get the message "failed to save medialist, trying again"

When using Windows the medialist is intermittently not saved the first time. However if you try again it will probably succeed. Ideas on why this happens are welcome. Code is in `pp_medialist.py` at around line 266.

Pi Presents crashes after playing videos for a few hours

There seems to be some timing problem in the interface with `OMXPlayer` or with `OMXPlayer` itself. I have tried to detect these crashes and allow Pi [Presents](#) to [continue](#). If you have such problems and can reliably reproduce them without using Pi Presents please report them on the [omxplayer github](#).

21 Converting Version 1.2 to 1.3

There are significant changes between the versions. pp_editor.py will insert and delete fields in the profile but the following will require manual intervention

Change	Reason
Pi Presents	
It is now not necessary to use the <code>-o</code> command line option when using sudo (from 1.3.1h sudo is not required)	Pi Presents detects the use of sudo and compensates
The <code>-g</code> command line option has been removed. GPIO is enabled by the presence of a <code>gpio.cfg</code> file in the profile (not in data home or <code>/pipresents/pp_home</code>).	Enables future user coded I/O plugins.
Shows	
Links field and Controls Field have been combined into a field called Controls. The functionality is unchanged. Controls bind Symbolic Names to Commands, the words Internal Operation and Link are not used any more, both are called Commands.	Make all shows operate in the same way to ease understanding.
In Version 1.2 Subshows and Child Shows inherited their Controls from their parent show. All shows now require their own set of Controls. Newly created mediashows and liveshows have a useful set pre-defined but you will need to insert the following into existing mediashows, liveshows and menus: pp-down down pp-up up pp-play play pp-stop stop pp-pause pause	In 1.2 the parent and subshow could not have different types of control (e.g. links and internal operations) so a mediashow called from a Hyperlinkshow could not be controlled.
The Show and Track Background are now 'warped' to the size of the Show Canvas	Makes it easier to modify the size of show canvas windows in multi-window applications
The Interval, Show Timeout and Track Timeout fields now use hh:mm:ss instead of secs. Currently >59 seconds is allowed but this might change.	
Commands and Symbolic Names	
pp-exit pre-defined symbolic name is now pp-exitpresents	avoid clashes of terminology
In keys.cfg pp-exit is now pp-terminate	avoid clashes of terminology
Mediashow	

<p>The triggers and other show control fields have been re-organised and expanded</p> <ul style="list-style-type: none"> interval and singleshoot needs to be replaced by repeat. Time of Day Triggers are replaced by the Time of Day Scheduler. Review all mediashows and liveshows for correct functioning as there are many modifications. 	Much improved mediashow triggering and control.
<p>The has-child field has been removed. Instead there is a Child field that contains the track-reference of a track in the medialist if a child is required. Will require the child field to be completed.</p>	Recognition that a 'child show' is actually just a track that is accessible from all tracks in a mediashow. Being a track in can be a Show Track.
<p>Manual progress has been removed. Use tracks with 0 duration instead. Use freeze at end if you do not want videos to auto advance.</p>	Manual made the code even more complex and presentations not a high priority for Pi Presents.
<p>The input-quiet trigger type has been removed. Instead delete the trigger text in the notices tab.</p>	
<p>Some fields have been moved between tabs</p>	Make shows consistent.
<p>Start Trigger cannot now be triggered by Play or Down commands. If trigger by a key is required then bind start trigger to a key.</p>	
<p>Liveshow</p>	
<p>In version 1.2 the liveshow and mediashow had completely different code. In version 1.3 much of the code is common which means all the features of mediashow are available.</p> <p>Review liveshows to ensure they function correctly</p>	Enhancement
<p>Menu</p>	
<p>In version 1.2 the layout of the menu was determined by the Show Profile.</p> <p>In Version 1.3 there is a Menu Type track in the medialist which determines the layout of the menu. pp_editor will do the conversion and insert a 'menu track' into a copy of the medialist called xxx – menu1p3. The conversion retains the old medialist file which can be deleted.</p>	All shows are structured the same making software maintenance easier.
<p>The menu background is now specified in the show profile. The menu background track can be deleted.</p>	All shows are structured the same making software maintenance easier.
<p>The location of default bullet is now /pipresents/pp_resources/bullet.png</p>	
<p>Click Areas</p>	
<p>screen.cfg has additional mandatory fields allowing a click area to have an image. Add to each section is screen.cfg: image =</p>	Click areas can have button images.

image-width = image-height=	
Track Plugins	
The API has changed significantly. They will need a rewrite.	Be compatible with gapless transitions and multi-windows
Show Control	
The order of fields in the Show Control field has been reversed and the names of the commands changed. 'myshow start' > 'open myshow' 'myshow stop' > 'close myshow' 'gobdegook exit' > 'exitpipresents'	Allows OSC commands. Names changed to avoid confusion caused by command clashes.
Animation	
The order of fields in animation commands has been changed and the command fields expanded 'out1 on 10' > '10 out1 state on' The delay field is now mandatory.	Provision for additional I/O plugins with more complex command formats.
GPIO	
Whether to use GPIO is now determined by the presence of a gpio.cfg file in the /pp_io_config directory in a profile. The -g command line option is not now used.	Required to allow users to add I/O plugins (see below)
Pins for B+ and 2B will need to be added to gpio.cfg. If not a warning is given.	
Configuration Files	
In a profile keys.cfg, gpio.cfg, and screen.cfg need to be moved to a subdirectory /pp_io_config. The new osc.cfg will go there as well. schedule.json is in the profile not in the subdirectory.	These are configuration files for I/O device drivers. When the I/O plugin API is implemented a plugin will be enabled by the existence of a configuration file in the /pp_io_config directory in a profile.
/pipresents/pp_home has been removed	Having a pp_home in two places could be confusing.
The fields of resources.cfg are now in the show profile. These fields will require re-populating with your choice of text.	Allows different concurrent shows to have different messages.
Fallbacks of gpio.cfg,, screen.cfg, and keys.cfg are now not allowed in /pp_home and in /pipresents/pp_home. They must be in the profile. If necessary copy gpio.cfg and screen.cfg into the	These are configuration files for I/O device drivers. When the I/O plugin API is implemented a plugin will be enabled by the existence of a configuration file

<p>profile (in a pp_io_config directory). keys.cfg still has a fallback in /pipresents/pp_config; only need to copy this if you have modified it.</p> <p>NOTE: In keys.cfg pp-exit is now pp-terminate</p> <p>There are templates for screen.cfg, gpio.cfg and schedule.json in /pp_resource/pp_templates, and for keys.cfg in /pp_config. These can be copied to profiles and modified.</p>	<p>in the /pp_io_config directory in a profile.</p> <p>There is one exception, to make it easy for beginners there is a fallback keys.cfg in /pipresents/pp_config</p>
controls.cfg has been removed.	It is replaced by every show having its own set of controls.
Limited other resources used as fallbacks by Pi Presents are in /pp_resources.. It is best to put resources for an application in the profile or in pp_home and not modify /pipresents sub-directories.	Updating of Pi Presents is less likely to cause you problems.
Editor	
Validation has not caught up with the changes to the profiles. There should be no correct profiles that fail but there could be undetected profile errors	