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

Copyright Notice

1 Introduction

Pi Presents is a toolkit for creating and deploying multi-media interactive displays 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 browser either on a Pi or remotely.

The Pi Presents toolkit supports the types of show commonly seen in museums; each of these can be configured to meet 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 ‘hyperlinked’ 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

- Run a selection of shows at different times, days of the week, and special days in a year.

- 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). Receive email alerts reporting the status of deployed Pi Presents applications.

- 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.

- Will operate on a Model 5 or 4 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 can be as simple as inserting the USB stick.

- Sound can output to HDMI, A/V, USB or Bluetooth 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

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

https://github.com/dddomodossola/remi

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 and GTK4.

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 and still the best 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.

2 Installation

The installation instructions are in the README.md file. This version of Pi Presents requires Raspberry Pi OS Bookworm or later and should always use the latest Raspberry Pi OS release and MUST BE INSTALLED AND RUN FROM THE PIXEL DESKTOP.

This manual assumes that the user is pi hence directories are stored in /home/pi. However Pi Presents will run under other Users.

3 Try Some Examples

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

N.B. Pi Presents must be run from the PIXEL desktop . From a terminal window opened in the pipresents directory type:

```
python pipresents.py –p pp_mediashow_1p6
```

You will see a continuous looping show in a small window. The small window is useful for development purposes. You can expand the display to nearly full screen by pressing the F1 key. 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_1p6 –f
```

If the -f command option is used the cursor is removed while Pi Presents is running. Display `blanking can be disabled in the preferences>raspberry Pi Configuration menu.`

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 position and font sizes, and to resize images.

The pipresents-gtk-examples repository has examples of how to use Pi Presents. To run them just use their directory name in the command above.

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 icon)
- F1 key toggles the display between windowed and near fullscreen.

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 and the look of individual tracks. The files can be viewed in a text editor but it is much better to edit them using the Pi Presents editor described in Section 4.

Other optional directories in the profile, pp_io_config, pp_track_plugins, customise other aspects of Pi Presents. Media can also be stored in a directory in a profile.

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_1p6
  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 the additional information that can be added to the primary content.

- pp_menu_1p6
  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_1p6
  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 P1-18) or pressing a button (closing a contact connected to 0 volts on P1-11) will play the tracks and then wait for the trigger in order to repeat the show.

  The example uses the gpiozero.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_1p6
  A single run 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. gpiozero.cfg in the profile uses p1-11 for this task.

- pp_onerandom_1p6
  When triggered plays a single track randomly chosen from the medialist .

- pp_magicpicture_1p6
  A still picture of a person is seen in a picture frame. When a button is pressed or PIR triggered the picture comes to life as a video is played. The effect is achieved by pausing the video just after the first frame. I don’t have a video of a person so Suits will have to do! In the example the PIR is replaced by the Return key.

- pp_interactive_1p6
  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 track

- pp_presentation_1p6

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_1p6
  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_liveshowempty_1p6
  Demonstrates the increased flexibility in handling liveshows with an empty tracks directory. An empty live tracks directory will play a 'list empty' track and execute the 'on empty' and 'on not empty' Show Control commands.

- pp_radiobuttonshow_1p6
  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 Radiobutttonshow profile allow for both keyboard keys and buttons to be used to select a track. A gpiozero.cfg file is included in the profile to bind four GPIO pins to entries and one to the Stop Operation.

- pp_hyperlinkshow_1p6
  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 gpiozero.cfg file.

- pp_audio_1p6
  This mediashow demonstrates the capabilities of the audio player. You will need speakers connected to hdmi and USB ports to fully appreciate it. It plays tracks to different speakers; it also demonstrates control of volume from keyboard (8 and 9) or GPIO pins and mute (keys 1 and 2). For this reason it has keys.cfg and gpiozero.cfg files in the profile,

- pp_web_1p6
  Note: Currently using the GTK4 the Webkit engine crashes Pi Presents if some profile items such as Show Text are used with web tracks. (Free beer for anyone that can tell me why !!)
-
  A demonstration of the web page rendering capabilities of Pi Presents. Ideally an internet connection is required.

  N.B Page rendering is via the Webkit browser engine not a full browser. This has fewer security features and should be used with care.

- pp_concurrent_1p6

This application demonstrates concurrent show playing capabilities. There are two mediashows running simultaneously by adding their show references to the Start Show. 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_1p6
  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 16.4

- pp_multidisplay_1p6
  This application demonstrates concurrent show playing capabilities where many shows display content on different displays. 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. In the example:
  - background - provides the static background on HDMI0
  - audio - plays the background audio
  - text - is a changing text only mediashow on HDMI0
  - slideshow – multimedia automatic slideshow on HDMI0. The slideshow may be paused by a click area on HDMI1, the pause acts seperately to that pausing the menu tracks on HDMI1
  - menu – interactive menu on HDMI1. The menu has associated click areas on HDMI1
  - One of the menu options is a Video Track that is displayed on HDMI0 having first stopped the slideshow

  The show is set up for remote control by OSC, see Section 16.4

- pp_multitouch_1p6
  As pp_multidisplay_1p6 except the menu is on the DSI0 touchscreen

- pp_touchscreen_1p6
  A good method to use a small touchscreen (DSI0) to control the playing of tracks on a large monitor (HDMI0). Event Show Control commands communicate between a Master and Slave Radiobuttonshow.

- pp_artmediashow_1p6
  A mediashow which features gapless transitions without freezing the picture at the end of video tracks. Has limited control features but could be useful for ‘artistic’ applications.

- pp_artliveshow_1p6
  A liveshow which features gapless transitions without freezing the picture at the end of video tracks. Has limited control features but could be useful for 'artistic' applications.

- pp_subshow_1p6
  Demonstrates how to use subshows to segment a larger mediashow.

- pp_timeofday_1p6
  Opens and closes two mediashows at different times of day and then exits Pi Presents. This is all controlled by the schedule tabs in the Shows. To make this demonstration work whatever the day and time you test it, the date and time is simulated by setting "simulated-time" to "yes" in the Start Show.

- pp_trackplugin_1p6
  Demonstrates the operation of track plugins. Track plugins are Python modules with a defined API which allow you to produce dynamic displays for example counters, clocks, and displays from data scraped from the web. Before using the liveshow copy the .cfg files from the /media directory of the profile to the /pp_home/pp_live_tracks directory.

- pp_showcontrol_1p6
  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 16.4

- pp_showcontrolevent_1p6
  Demonstrates the use of the event Show Control command to pause and un-pause a track using pause-on and pause-off commands

- pp_clickarea_1p6
  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 in a real application there would be two GPIO buttons mounted adjacent to the left edge of the screen and a gpiozero.cfg file to associate the buttons with the symbolic names.

- pp_animate_1p6
  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 gpiozero.cfg file in the profile. BEWARE: Ensure P1-11 can safely be used as an output before running this.

- pp_shutdown_1p6
  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.

- pp_videoplayout_1p6
  A demonstration of a show that can be used as a video playout system in theatres or television stations. See Section 5.4.5.2.

- pp_counters_1p6
  Demonstrates the commands for control of counters and use of the track plugin which can access and display values of counters

- pp_quiz_1p6
  Demonstrates the use of counters and a Hyperlinkshow to implements a quiz

- pp_osc_1p6
  Demonstrates the show control commands for remote control via OSC

- pp_beep_1p6
  Demonstrates Beep Show Control Command and Animation command, and Show Control on Event

- pp_html_1p6
  Demonstrates HTML Text Panels. HTML text allows most of the formatting power of HTML to be used in Pi Presents

3.1 Terminating Pi Presents

You can terminate Pi presents by pressing Ctrl+Break, or clicking on the 'Close' icon. 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 6.7

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 15.2.6

4 The Pi Presents Profile Editor

The web based editor is browser based so can be used from a remote computer in addition to local use. Using the web based editor remotely is described in Section 18.3

Profiles can be created and edited using the Pi Presents editor. For use on the Pi the command is:

```
python pp_web_editor.py
```

This will open the Chrome browser and display the editor gui. However there seems to be a bug in the March 24 issue of Bookworm which causes this command to hang. I solved it by making Firefox the default browser and then making Chrome the default browser again. Failing this start the editor with the -r command line option and right click on the url displayed in the terminal window and selecting Open URL

N.B If using a User other than pi you will need to edit the paths in /pipresents/pp_config/editor.cfg first.

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. In the directory browser click on the text of the example (e.g. pp_mediashow_1p6) and press OK. Do not open the directory.

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

The selected entries are shown with a green background. 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_web_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. Most fields other than Font and Colour fields are validated. If you edit the .json files with a text editor you are on your own!

Profile>Copy To – Copies a profile

Profile>Delete – Delete a profile

Profile>New from Template - lists templates for all types of show. The templates have example tracks so they will run un-edited.

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 copied show

Show> Edit - Duplicates the Edit Show button

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

Show>View Backup - Shows the parameters of the show and its associated tracks from the backup made when the profile issue was last updated.

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 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 Dir
Similar to Add 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>Copy
Copies a track

Track>Delete
Deletes the track from the medialist, no going back!

4.1.6 OSC Menu

OSC>Create
Creates an empty osc.cfg file in the profile in the directory pp_io_plugins

OSC>Edit
Edits osc.cfg.

4.1.7 Tools Menu

Tools>Update All
Update the version of all profiles in the current home + offset directory, see Section 21.

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 'Track>Add File' and 'Track>Add 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 /1p5_examples defaults ‘open’ to /pp_home/pp_profiles/1p5_examples

4.1.9 Editor Command Line Options

one of : -r --remote -l --local -n --native	If none of these are specified then -l is assumed. local – the editor server is run and a browser is opened on the RPi (ip=127.0.0.1) remote – the editor server is run with the IP address of the RPi (e.g. 192.168.1.67) it can be accessed remotely from a browser on another computer using http://192.168.1.67:8082 native – the editor is run natively on the RPi without a browser. Requires qt5 and pyview to be installed (which I failed to achieve)

4.2 Making Profiles

Application data is kept is 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.

To make profiles portable all media is best kept either under the directory /pp_home or alternatively inside a profile. In both cases it is recommended that media is stored in a sub-directory, by convention named media. Media stored like this is referenced by relative file paths in a profile e.g. +/media/myimage.jpg or @/media/myimage.jpg

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 optional directories /pp_io_config which contains Input/output configuration files such as gpiozero.cfg and pp_track_plugins which contains track plugins such as krt_time.py. 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.

To achieve portability media tracks should be stored under the /pp_home directory, preferably in sub-directories or alternatively in a profile, by convention in a directory called media. If tracks are stored in these locations 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'

If the track is in the profile then the file reference will look like '@/media/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'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 the remote option of the editor. The editor runs on the Pi but the gui appears in a browser on the PC.

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.

Insert the USB stick into a Pi and run Pi Presents with the -o option set to /media/pi/STICKNAME. STICKNAME is the drive name.

4.2.5 Developing Profiles using a Monitor with Different Pixel Dimensions

The fake-display-dimensions option in /pipresents/pp_config/pp_display.cfg 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 Raspberry Pi OS.

- 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 medialist which is associated with the show, think of the medialist 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.

- MPV Video/Audio - plays videos or audio tracks using MPV
- Image - displays images in many different formats (.jpg etc.) (uses GTK4)
- Message - a quick way to display text.
- Webkit Web – displays html content using the Webkit browser engine

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 as Show tracks.

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 shows transitions between tracks are gapless (only minimal 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 on older models of RPi 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 and its content, format, and position is configurable in the show profile.

Concurrent Shows

Pi Presents can run two or more shows concurrently, see Section 7.4. 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 may 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 each subshow 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 and Track Text Font. 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 Menushow

- 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 medialist until a period of time has elapsed, then it ends.	repeat	start	none	Show Timeout = h:m:s
Continuous show that repeats the medialist at intervals	repeat	start	none	Repeat Interval = h:m:s
Run a medialist triggered by an input	repeat	input	none	
Run a medialist with each track waiting for, and triggered by an input	repeat	input	none	Trigger for Next = Input
Run a medialist triggered by an input. Inhibit re-triggering for a period of time after the medialist 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 medialist when an input is pressed and released	repeat	input	input	In gpiozero.cfg use activated-name and de-activated name of a pin for the two triggers.
'Magic Picture'. Start a video and pause it after its first frame. Unpause using the go command and then repeat.	repeat	start	none	Video requires 'freeze at start' = 'after first frame'. Also the go command to be used
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 commands with the Symbolic names of Input Events is in the Controls field.

Artmediashow

Artmediashows provide full gapless playback; there is little gap between tracks and tracks do not pause 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, the second is triggers. 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 GPIO with a useful set of bindings can be enabled easily by copying a prepared gpiozero.cfg file to a profile from /pipresents/pp_resources/pp_templates:

The out of the box settings are:

	'Out of the box' For GPIO Pin see Section 15.2.2			**Effect**
Command	**Symbolic Name**	**Key**	**GPIO Pin.** See .cfg file	
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.
(repeat)				repeat the current track
pause-on				pause for tracks that support pause.
pause-off				unpause for tracks that support pause.
mute				mute for tracks that support mute.
unmute				unmute for tracks that support mute.
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: • 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 moves to the next track for a repeating show.
(go)	<not bound>	<not bound>	<not bound>	unpause a video track that is ‘frozen at start’
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.

inc-volume	<not bound>	<not bound>	<not bound>	For MPV video/audio tracks, increase the volume by one step
dec-volume	<not bound>	<not bound>	<not bound>	For MPV video/audio tracks, decrease the volume by one step

The 'out of the box' key for each operation is set in /pipresents/pp_io_config/keys.cfg and the GPIO pin in gpiozero.cfg in the templates directory These bind keys of GPIO inputs to symbolic names. The bindings can be modified as described in Section 15.2. 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. Triggers provide a different control paradigm than the Up and Down 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
  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 Start 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 medialist/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 – At the end of a track the show waits and 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 trigger is inhibited while a track is being played.

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

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

Field	Examples	Values
Display	HDMI0	HDMI0,HDMI1,DSI0,DSI1 - The display that the show and any subshows are to be displayed on
Medialist	show1.json	Filename of the medialist file containing the tracks for the mediashow. By default it is the same as the Show Reference but can be changed.
(Trigger For Start)	start	How the media show proceeds after it is started and at the beginning of a repeat: • start - continue without waiting • input - wait for an input event.
(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 15.2
(Trigger for Next)	continue	How to trigger the next track • input – wait for an input event • continue – continue without waiting
(Trigger for Next Parameter)		If Trigger For Next is 'input': A symbolic name that is bound to an input event as described in Section 15.2
Sequence		sequence of tracks: • ordered or reverse - played in the order of the medialist (mediashows only) • ordered - played in the order of the livelist (liveshows only) • reverse - played in the reverse order of the livelist (liveshows only) • shuffle - play tracks in a random order. For liveshow this is a true shuffle so a track will not be replayed until all the tracks in the live tracks directories have been played. For mediashow the next track is chosen randomly.
Repeat/Single-run	repeat	How the media show is repeated: • 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.

Field	Examples	Values
(Trigger For End)	none	How the end of a show can be triggered. See the table above for more detail. • none – the end trigger is not operative • input – Ends after an input event as described in Section 15.2
(Trigger for End Parameter)	my-end	If Trigger For End is input – A symbolic name that is bound to an input event as described in Section 15.2
(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*300	Values: • blank • x1+y1+width*height • width*height x1, y1 is the top left coordinate of the rectangle If x1 and y1 are omitted then the rectangle is centred on the display. 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.

Field	Examples	Values
Empty List Track		Not used for Mediashow or Artmediashow
Escape Track		Not used for Mediashow or Artmediashow
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)	bold 30pt Helvetica	Examples: 20pt arial bold 10px helvetica bold italic 10pt arial Pi Presents uses the CSS font descriptor specified here: https://www.w3.org/TR/css-fonts-3/#propdef-font This describes many other font characteristics that could be used.
(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 or transparent. Colour names are specified in the link below. Pi Presents also allows 8 hex digit colour values, the last 2 digits being the opacity. A value of 0 for opacity is transparent. Pi Presents uses the CSS color descriptor specified here: https://www.w3.org/TR/css-color-3/#color This describes many other colour values that could be used.
(Hint Justify)	right	left, right, center – Justify lines in a block of text
(Hint x)	500	Position of left of text. If blank then the text is centred in the x direction on the Show Canvas
(Hint y)	900	Position of top of text. If blank then the text is centred in the y direction on the show canvas
Eggtimer Tab		Eggtimer text is displayed when the next track is being loaded
EggTimer Text	Loading….	A single line of text
Eggtimer Text Font	bold 30pt Helvetica	See Hint Text above
Eggtimer Text Colour	white	See Hint Text above
Eggtimer Text x Position	100	distance of the start of the text from the left of the screen (pixels) If blank then the text is centred in the x direction on the show canvas

Field	Examples	Values
Eggtimer Text y Position	100	distance of the top of the text from the top of the screen (pixels) If blank then the text is centred in the y direction on the show canvas
Eggtimer Justify	right	left, right, center – Justify lines in a block of text
Notices Tab		
(Trigger Wait Text)	Waiting…	Text to show when Waiting for Trigger
Notice Text x Position		distance of the start of the text from the left of the screen (pixels) If blank then the text is centred in the x direction on the show canvas
Notice Text y position		distance of the top of the text from the top of the screen (pixels) If blank then the text is centred in the y direction on the show canvas
Notice Text Colour		See Hint Text above
Notice Text Font		See Hint Text above
Notice Justify	right	left, right, center – Justify lines in a block of text
Show Background and Text Tab		
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		See Hint Text above for values. 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 Location	+/media/ mytextfile	If not blank the file containing the Show Text, takes priority over Show Text field
Show Text x Position	100	distance of the start of the text from the left of the screen (pixels) If blank then the text is centred in the x direction on the show canvas
Show Text y Position	100	distance of the top of the text from the top of the screen (pixels) If blank then the text is centred in the y direction on the show canvas
Show Text Type	html	plain – plain text html – a panel of html rendered by the webkit browser engine. See Section 5.4.4.4

Field	Examples	Values
Plain Text Font	bold 30px Helvetica	See Hint Text above
Plain Text Colour	white	See Hint Text above
Plain Text Justify	right	left, right, center – Justify lines in a block of text
HTML Text Height	300	The height of the HTML text panel
HTML Text Width	300	The width of the HTML text panel
HTML Text Background Colour	#ff0000	The background colour of the html text panel. Vaues as Hint Text including transparent. Value is overiden by any background-color descriptor in the html
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.
Pause Timeout	5	seconds. If not blank and greater than 0 a paused track will automatically unpause after the timeout.
Track Text Font	bold italic 30 pt Helvetica	See Hint Text above
Track Text Colour	white	See Hint Text above
Track Text x Position	100	distance of the left end of the text from the left of the screen (pixels) If blank then the text is centred in the x direction on the show canvas
Track Text y Position	100	distance of the top of the text from the top of the screen (pixels) If blank then the text is centred in the y direction on the show canvas
Track Text Justify	right	left, right, center - Justify lines in a block of text
Transition	cut	cut. Type of transition between tracks. Currently not used.
Image Window	fullscreen	For any image track in the show a viewport in which to show the image. See section 5.4.4.1
Image Aspect Mode	warp	How the image is transformed to fit into the Image Window. See section 5.4.4.1
MPV Player Audio	A/V	HDMI0, A/V, USB, USB2, bluetooth in addition HDMI1 for Pi4 and greater.

Field	Examples	Values
MPV Player Volume	100	Volume of video track (0 -> 100).
MPV Window	100*200	A viewport in which to show the video. See Section 5.4.5.1 for values
MPV Aspect Mode	warp	How the video is transformed to fit into the Image Window. See Section 5.4.5.1 for values
MPV freeze at Start	no	Freeze a video at the start of the track. • no – no freeze • before-first-frame – the video is paused before the first frame is shown • after-first-frame - the video is paused after the first frame is shown To continue after the freeze use the 'go' command.
MPV Freeze at End	yes	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.
MPV Player Options		Additional command line options for the MPV Player, each option separated by a comma. An option takes the form option=value. Option is the MPV command line option without the preceding -- The option and value is not checked by Pi Presents and may cause MPV to crash.
Webkit Web Window	200+200+600*400	See Section 5.4.6
Webkit Zoom	1.0	See Section 5.4.6
Webkit Freeze at End	yes	See Section 5.4.6
Webkit Options		See Section 5.4.6
Controls Tab		
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
Show Control Tab		
Show Control at Beginning	open audio1	See Section 8

Field	Examples	Values
Show Control at End	close audio1	See Section 8
Show Control on Event	pp-pause open audio1	See Section 8.4
Disable Show Control on Event	no	See Section 8.4

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 detail 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
Display	HDMI0	HDMI0, HDMI1, DSI0, DSI1 - The display that the show and any subshows is to be displayed on
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.

Field	Examples	Values
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
Show Control Tab		
See Mediashow		

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 two Live Tracks directories - and is therefore limited to video/audio, image, and web tracks, and their track plugins. The tracks from the two directories are merged and, if sequence = ordered or reverse, 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, List Empty Track, and Escape Track; all other tracks in it will be ignored.

Actions on the list of tracks being empty:

a. Using List Empty Track Escape Track

- If the liveshow's repeat field is 'Single Run' then the liveshow exits.
- If the liveshow's repeat field is 'Repeat' and List Empty Track' is blank then an error will be generated.
- If the liveshow's repeat field is 'Repeat' and the 'List Empty Track' is not a show it is run whenever the list of tracks to be played is empty. The track is played once and then the content of the list of tracks is re-evaluated.
- The List Empty Track may be a show. If so the show should be 'single run' or there should be some other method of exiting from the show so that the list of liveshow tracks can be re-evaluated. If the list is then empty the Escape track is run once. This track must not be a show. Its purpose is to allow the liveshow to be stopped or exited if it is a subshow of another show.
- The limitation of this method is that the empty track/show uses the same Show Canvas as the liveshow.

b. Using Show Control on Empty

- The 'Show Control on Empty' and 'Show Control on Not Empty' fields allow other shows to be opened or closed depending on whether there are tracks to be played.

It is unlikely that both methods will be required in the same application.

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 and Artliveshow have 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 pp_live_tracks directory in pp_home
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 –l command line option. Can be used for synchronisation, (see below)
(List Empty Track)		Not ArtLiveshow. The Track Reference of a track that will be run when the list of tracks in a liveshow is empty. See text
(Escape Track)		Not ArtLiveshow. The Track Reference of a track that will be run when returning from an Empty Track which is a show. See text.

Field	Examples	Values
Show Control Tab		
		See Mediashow
Show Control on Empty	open show1	See Section 7.27.2
Show Control on Not Empty	close show1	See Section 7.2

Synchronisation of Uploads

Out of the box it is recommended that Pi Presents be stopped when updating livelists. This is because a track could be being played while it is being modified. Synchronisation allows livelists to be updated while a Liveshow is playing. This is achieved by having a cache directory on the Raspberry Pi to which a new livelist is uploaded then using a 'lock' to alert PP to the new files. PP will move the files to the livelist when it sees the lock when it starts and then only when it is between tracks. The supplier of tracks must use the lock to ensure it does not modify the cache while the transfer is in progress.

To enable synchronisation the file pp_livelistfetcher.py must be modified to change the mode and specify the cache directory. Full instructions are in the file. Syncing is available only for Live Tracks Directory 2.

Using symlinks (from Github)
I have the folder |available_media| which contains all the media I can use for the show and am simply symlinking them in the |live_tracks_cache| (which are then getting copied to |live_tracks| after touching the .lock-file).

This prevents huge writes to the sd-card in case of frequent changes to the displayed media (since symlinks are smaller than a whole file) which will lead to longer lifetime.

Using Rclone (from Matt)
Rclone with onedrive is a great addition to the liveshow feature when trying to get around network drives and working on collaborative drives.

I just wanted to follow up from a message that was sent last week in regards to a 403 error that we kept receiving. We were trying to get a OneDrive folder to work with the Liveshow feature using chromium browser but kept getting hit with network issues. The work around that we found worked was to use rclone to create an auto syncing folder from a OneDrive onto the PI and have Pi Presents pull from that folder for a Liveshow. It auto updates the folder once an hour and keeps the contents from erroring out when deleted or added.

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.
go	<blank>	unpause a video track that is 'frozen at start'
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
pause-on	<blank>	pause for tracks that support pause
pause-off	<blank>	unpause for tracks that support pause
mute	<blank>	mute for tracks that support mute
unmute	<blank>	unmute for tracks that support mute
inc-volume	<blank>	For MPV video tracks increase the volume by one step
dec-volume	<blank>	For MPV video tracks decrease the volume by one step

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

Field	Examples	Values
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
Display	HDMI0	HDMI0,HDMI1,DSI0,DSI1 - The display that the show and any subshows is to be displayed on
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.
Controls in Subshows	no	yes/no If no (normal Radiobuttonshow operation) pressing a button or key while a track is playing will start playing the selected track. However any tracks which are shows cannot be controlled so, for example, a mediashow needs to be single-run. If yes and the track to be played is a show track then input events will be passed down to the show. This allows the show to be controlled (using its own controls) but it will not be possible to immeadiately select another track while the show is playing.
Eggtimer Tab		
		See Mediashow
Show Background and Text		
		See Mediashow
Track Defaults Tab		
		See mediashow
Controls Tab		

Field	Examples	Values
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.
Show Control Tab		
See Mediashow		

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, and no-command.
- 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.
go	<blank>	unpause a video track that is ‘frozen at start’
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
pause-on	<blank>	pause for tracks that support pause
pause-off	<blank>	unpause for tracks that support pause
mute	<blank>	mute for tracks that support mute
unmute	<blank>	unmute for tracks that support mute
no-command	<blank>	Clicking it has no effect. Use for legends for ‘soft keys’ to display a ‘Click Area’ that is not enabled.
inc-volume	<blank>	For MPV video tracks increase the volume by one step
dec-volume	<blank>	For MPV video tracks decrease the volume by one step

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 or pp-onend. 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 Controls field can then have a command to service it e.g.

pp-onend goto mynextpage
or
pp-onend repeat

Commands associated with pp-onend are limited to exit, home, return, call, goto, jump, and repeat

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 track timeout would.

To ensure that the screen goes back to the First Track if the user leaves the screen there is a track timeout. If activated the Hyperlinkshow goto's to 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 to 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 Print 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 in addition to keyboard keys:

- 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 text and/or 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 rectangle or circle 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 10

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
Display	HDMI0	HDMI0,HDMI1,DSI0,DSI1 - The display that the show and any subshows is to be displayed on
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		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. Usually this track will do whatever is required and 'goto' the First or Home Track after a short time. If the track is an message or image track it can have no content and zero duration and be used just to reset animation etc..

Field	Examples	Values
Show Timeout	60	h:m:s, If there is no activity on the Home Track Pi Presents automatically exits to the parent show (or none if it is a top level show). 0 for no timeout.
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
Show Control Tab		
See Mediashow		

5.2.6 Start Show

Do not delete the Start Show. The Start show is not a show; it is a part of the profile that specifies things that are common to all shows:

It 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.

- Whether the time of day scheduler is enabled and the schedule for things that are common to shows (See Sect. 7.3)

- Controls for simulated time that can be used to test the time of day scheduler (See Sect. 7.3).

- The colour of the background that is displayed behind shows.

- Storage and initial value of counters (See Sect. 13)

Field	Examples	Values
Schedule Tab		
Start Shows	show1 show2	A list of Show References separated by spaces. These are shows that are started when Pi Presents starts.
Enable Scheduler	no	yes/no Enables the time of day scheduler. Leave as No unless you have set up the time of day scheduler. See Section 7.3
		See Section 7.3 for additional fields
Simulate Time Tab		
Simulate Time		yes/no Use a simulated time and date when testing the time of day scheduler. Leave as No unless you want to set up the time of day scheduler. See Section 7.3
		See Section 7.3 for additional fields
Background Tab		
Background Colour		The colour of the background when no shows are displayed.
Counters Tab		
Store Counters	no	If set to Yes then counters and their values are stored between runs of Pi Presents
Initial Values	fred = 1 bill = 2	If storage is enabled and the command line has the option --counters then the counters are populated from this field
Show Tab		
type	start	Must always be start, not editable.
title	Start Show	Text describing the show, displayed in the editor.
Show	start	must be 'start', not editable

Reference		

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
- mpv video/audio - a track played by MPV.
- message - displays lines of text
- webkit web – uses the Webkit browser engine 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 GTK4. For performance reasons image size is best limited to the 1 megapixel region for older models of the RPi.

- mpv video/audio- a video or audio track played by the MPV Media Player.

- message - displays lines of text. Can also used as a dummy to display a blank screen to allow reset of animation etc. It provides a simple slide preparation facility. If you want something better use Powerpoint or similar and export as .jpg displaying as an image.

- webkit web - renders a web page from a file or URL using the Webkit browser engine.

- 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 Locations

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

- A 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.

- A leading @ sign in file paths allows tracks to be specified relative to the specific profile directory. e.g. @/mymedia/myfile.jpg will access the /mymedia diirectory in the current profile.

- 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 medialist entries will have labels defined in the Track Reference field. If the Track Reference is blank then the track is included in a Menu, or Mediashow, 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

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 medialist. The Show To Run field specifies the Show Reference of the show.

5.4.4 Image Track

Image tracks are rendered by GTK4. Image tracks can be paused with the Pause Command. Images can appear in a window, 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		
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

Field	Example	Values
		special.
Location	+/media/river.jpg	The filename of the track which may be blank.
Thumbnail	+/media/ river_tn.jpg	The optional filename of a thumbnail image to be displayed by a menushow.
Duration	5	Seconds with resolution of 0.1 secs. 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)
Image Window	fullscreen 200+200+400*300 400*300	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
Image Aspect Mode	warp	How the image is transformed to fit into the Image Window. 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 or transparent. Colour names are specified in the link below. Pi Presents also allows 8 hex digit colour values, the last 2 digits being the opacity. A value of 0 for opacity is transparent. Pi Presents uses the CSS color descriptor specified here: https://www.w3.org/TR/css-color-3/#color This describes many other colour values that could be used.
Plugin Configuration File	+/media/krt-time.cfg	File which specifies the name and parameters of the Track Plugin.
Pause Timeout	5	seconds. If not blank and greater than 0 a paused track will automatically unpause after the timeout.
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 7.1
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 Location	+/media/mytextfile	If not blank the file containing the Track Text, takes priority over Track Text field
Track Text x Position	100	distance of the left end of the text from the left of the screen (pixels) If blank then the text is centred n the x direction on the show canvas
Track Text y Position	100	distance of the top of the text from the top of the screen (pixels) If blank then the text is centred in the y direction on the show canvas
Track Text Type	html	plain – plain text html – a panel of html rendered by the webkit browser engine. See Section 5.4.4.4
Plain Text Font	bold 30pt Helvetica	Examples: 20pt arial bold 10px helvetica bold italic 10pt arial Pi Presents uses the CSS font descriptor specified here: https://www.w3.org/TR/css-fonts-3/#propdef-font This describes many other font characteristics that could be used.
Plain 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 or transparent. Colour names are specified in the link below. Pi Presents also allows 8 hex digit colour values, the last 2 digits being the opacity. A value of 0 for opacity is transparent. Pi Presents uses the CSS color descriptor specified here: https://www.w3.org/TR/css-color-3/#color This describes many other colour values that could be used.
Plain Text Justify	right	left, right, center - Justify lines in a block of text
HTML Text Height	300	The height of the HTML text panel
HTML Text Width	300	The width of the HTML text panel
HTML Text Background Colour	#ff0000	The background colour of the html text panel
Display Show Text	yes	yes,no Allow or inhibit the display of show text for this track

Field	Example	Values
Pause Text	Paused…..	Text displayed when track is paused
Pause Text Font	Helvetica 30 bold	See Plain Text Font above
Pause Text Colour	white	See Plain Text Colour above
Pause Text x Position	100	distance of the left end of the text from the left of the screen (pixels) If blank then the text is centred in the x direction on the show canvas
Pause Text y Position	100	distance of the top of the text from the top of the screen (pixels) If blank then the text is centred in the x direction on the show canvas
Pause Text Justify	right	left, right, center - Justify lines in a block of text
Show Control Tab		
Show Control at Beginning	open audio1	See Section 7.2
Show Control at End	close audio1	See Section 7.2
Animation Tab		
Animation at Beginning	1 out1 state on 2 out1 state off 1 out2 state on 6 out3 state off	See Section 14
Clear Animation	no	yes/no See Section 14
Animation at End	0 out2 state off	See Section 14

5.4.4.1 Image Window

The Image Window controls where an image is presented spatially. If Image Window is blank then the value from the show is used. The window constrains the image using the rule specified in image Aspect Mode

Image Window has 3 formats in addition to blank:

- fullscreen
  The Image Window has the same dimensions and position as the Show Canvas

- x1+y1+width*height (pixcels)
  x1 and y1 define the position of the top left corner of the Image Window relative to the Show Canvas. Width and height are the dimensions of the window.

- width*height
  A window of dimensions width and height is centred in the Show Canvas

5.4.4.2 Image Aspect Mode

Image Aspect Mode determines how the image fits in the Image Window,

5.4.4.3

- 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.

- shrink
  As fit except that the image is not expanded if it is smaller than the window.

- 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.

- clip (not implemented yet)
  The image is displayed such that it fits in the specified window maintaining its aspect ratio. The image is shrunk or expanded as required. If the window's and image's aspect ratios are different part of the image will be cropped.

5.4.4.4 HTML and Plain Text

The Message Text, Track Text, and Show Text allow the type of text to be plain or HTML. The content of these fields can only be obtained from the Text field or from local files, not from a URL.

Plain – the text colour, font and justification can be specified. The text overlays other items without an opaque background panel

HTML – The text is HTML rendered by the Webkit browser engine. It appears in a rectangular panel the height and width of which can be specified.

The HTML Text Background Colour field allows the panel to have a background colour or to be transparent. If there are any CSS backgound-color statements in the HTML these will override this field.

5.4.5 MPV Video/Audio Track

A track played by the MPV Media Player. MPV is primarily a command line video/audio player. Pi Presents uses the python MPV library to control MPV. Pi Presents should play any track that MPV can play. Video tracks can be paused with the Pause command. 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.

MPV does not use a GPU hence CPU usage is high:

720p (1280*720) videos use typically 35% CPU on a Pi5 and ???? CPU on a Pi4
HD (1920*1080) videos use typically 65% CPU on a Pi5 and 75% CPU on a Pi4
HEVC (3840*2160 videos typically use 90% CPU on a Pi5

The MPV video track can play audio tracks. It is better if Freeze-at-End is set to No

The MPV video track can play streams, rtsp://.... etc.

Field	Example	Values
Track Tab		
Type	mpv	
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. If url contains a colon (:) preceded by more than 1 letter, it will be treated as a URL. Else, it will be considered as a file path.
Thumbnail	+/media/film _tn.jpg	The optional filename of a thumbnail image to be displayed by a menushow.
Freeze at Start	before-first-frame	Freeze a video at the start of the track. • no – no freeze • before-first-frame – the video is paused before the first frame is shown • after-first-frame - the video is paused after the first frame is shown To continue after the freeze use the ‘go’ command. If blank the value in the parent show is used
Freeze at End	yes	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.
MPV Player Audio	default	HDMI0, A/V, USB,USB2, bluetooth in addition HDMI1 for Pi4 or greater. If blank then the value in the parent show is used.
MPV Player Volume	100	Volume of video/audio track (0 -> 100). If blank then the value in the parent show is used.
MPV Player Max Volume	100	Maximum video volume. MPV could distort if source audio is great and real time controls push the volume too high. This allows the volume to be limited.
MPV Subtitles	no	yes/no – Whether sub-titles should be displayed.
MPV Speaker	left	left/right/stereo/5:1. If blank then the value in the parent show is used.
MPV Player Options		Additional command line options for the MPV Player, each option separated by a comma. An option takes

		the form option=value. Option is the MPV command line option without the preceding -- The option and value is not checked by Pi Presents and may cause MPV to crash. If blank then the value in the parent show is used.
MPV Window	fullscreen 100+100+400*300 400*300	A viewport in which to show the video. See Section 5.4.5.1 for values. If blank then the values are taken from the parent show.
MPV Aspect Mode	warp	How the video is transformed to fit within the MPV Window. If blank then the value is taken from the parent show. See Section 5.4.5.1
Alternate Video Display	HDMI0	An alternative display on which the video track is displayed. All other items are displayed on the display determined by the Show. The video covers the whole of the display using the transformation rule in MPV Aspect Mode. This feature id designed particularly for Video Playout (Section 5.4.5.2)
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 or transparent. Colour names are specified in the link below. Pi Presents also allows 8 hex digit colour values, the last 2 digits being the opacity. A value of 0 for opacity is transparent. Pi Presents uses the CSS color descriptor specified here: https://www.w3.org/TR/css-color-3/#color This describes many other colour values that could be used. 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	yes	yes/no yes - the background image of the parent show is

Background Image		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
Pause Timeout	5	seconds. If not blank and greater than 0 then a paused track will automatically unpause after the timeout.
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 7.1
Text Tab		
		See Image Track. Does not have Pause Text
Show Control Tab		
		See Image Track
Animation Tab		
		See Image Track

5.4.5.1 MPV Window

The MPV Window controls where a video is presented spatially. If MPV Window is blank then the value from the show is used. The window constrains the video uisng the rule specified in MPV Aspect Mode

MPV Window has 3 formats in addition to blank:

- fullscreen
  The MPV Window has the same dimensions and position as the Show Canvas

- x1+y1+width*height (pixels)
  x1 and y1 define the position of the top left corner of the MPV Window relative to the Show Canvas. Width and height are the dimensions of the window.

- width*height
  A window of dimensions width and height is centred on the Show Canvas

MPV Aspect Mode

MPV Aspect Mode determines how the video fits in the MPV Window,

- fit
  The video is displayed such that it fits in the specified window maintaining its aspect ratio. The image is shrunk or expanded as required. If the window's and video's aspect ratios are different this will result is showing black bars

- shrink
  As fit except that the video is not expanded if it is smaller than the window. If the window's and video's aspect ratio are different this will result is showing black bars

- warp
  The video is displayed such that it fits in the specified window without maintaining its aspect ratio. The image is shrunk or expanded as required.

- clip (not implemented yet)
  The video is displayed such that it fits in the specified window maintaining its aspect ratio. The image is shrunk or expanded as required. If the window's and video's aspect ratios are different part of the video will be cropped.

5.4.5.2 Video Playout

Pi Presents can be used as a video/audio playout system in theatres or television stations. The key aspect is that, when cued, videos are loaded and paused just before the first frame so that pressing the g key, which executes the Go command, causes the video to start immediately. In the example pp_videoplayout_1p6 the Radiobuttonhow displayed on HDMI1 and its video/audio tracks use the Alternate Video display to display the video on HDMI0. Freeze at Start is set to 'before-first-frame' and the go command is used to play the video.

The show could be a Menu or a Mediashow instead of a radiobuttonshow.

5.4.6 Webkit Web Tracks

Web tracks are rendered by the Webkit browser engine. This is that part of a browser that displays the HTML content.

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.

Because the browser engine takes a noticeable time to load the WEb Track has its own language, Browser Commands which allows a sequence of web pages to be loaded without reloading the browser.

Field	Example	Values
Track Tab		
Type	webkit	
Title		Displayed on a menu and in the editor

Field	Example	Values
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	http://www.google.co.uk +/media/hello_world.html	The filename of the track: • an internet url (including http:// etc.) • a full pathname to a file • a relative path to a file located under pp_home or a profile
Thumbnail	+/media/river_tn.jpg	The optional filename of a thumbnail image to be displayed by a menushow.
Duration	5	Seconds with resolution of 0.1 secs. If 0 then the web page is displayed until terminated by user input. If blank then the value in the parent show is used.
Freeze at End		yes/no. If yes then the web track remains displayed past the Duration until the following track is loaded. This covers up the gap between tracks. If no the track finishes at Duration. If the field is blank the value in the parent show is used.
Webkit Window	fullscreen 0+0+400*800 400*800	A viewport in which to show the web page. If blank then the value in the parent show is used. See Section 5.4.6.1 for values
Webkit Zoom	0.5 2.0	A positive real value. The content of the browser window is scaled by this factor. If blank then the value is taken from the parent show.
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.
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

Field	Example	Values
Background Colour		Use the Colour Chooser to select a colour which will return a six digit hex number, or type in a colour name or transparent. Colour names are specified in the link below. Pi Presents also allows 8 hex digit colour values, the last 2 digits being the opacity. A value of 0 for opacity is transparent. Pi Presents uses the CSS color descriptor specified here: https://www.w3.org/TR/css-color-3/#color This describes many other colour values that could be used. 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 5.4.6.2Error: Reference source not found
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 7.1
Text Tab		
		See Image Track
Show Control Tab		
		See Image Track
Animation Tab		
		See Image Track

5.4.6.1 Webkit Web Window

The Web Window controls where a page is presented spatially. If Web Window is blank then the value from the show is used.

Web Window has 3 formats in addition to blank:

- fullscreen
  The Web Window has the same dimensions and position as the Show Canvas

- x1+y1+width*height (pixels)
  x1 and y1 define the position of the top left corner of the Web Window relative to the Show Canvas. Width and height are the dimensions of the window.

- width*height
  A window of dimensions width and height is centred on the Show Canvas

x1,y1 specifies the position of the web window. width and height specify a window in which the web page is presented. If the page content is larger than the window scroll bars appear. Webkit Zoom can be used to reduce the size of the web page.

5.4.6.2 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.

If there are browser commands the duration of the track is the minimum of the time it takes to execute the browser commands and the Duration field in the profile.

Pi Presents steps through the list of browser commands. If there is no Loop command it exits after the last command. There may be one Loop command. If it is present Pi Presents loops to the location of the command one or more times.

Command	Description
load <arg>	Load the web page specified by <arg>. <arg> may be: • an internet URI • a full pathname to a file • a relative path to a file located under pp_home or the profile
refresh	refresh the currently loaded web page
wait <arg>	wait <arg> seconds
Loop <arg>	A single loop command is allowed. If a loop command is present execution will resume here after the last command of the script. The optional integer argument specifies the number of loops. If blank then looping will be continuous.

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

```
loop
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

Ensure the Duration field is at least 60

Example: Refresh the initially loaded web page at 20 second intervals 5 times
wait 20
loop 5
refresh
wait 20

Ensure the Duration field is at least 120

5.4.7 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.
Thumbnail	+/media/ river_tn.jpg	The optional filename of a thumbnail image to be displayed by a menushow.
Duration	5	Seconds with resolution of 0.1 secs. If 0 then message is displayed until terminated by user input. 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 or transparent. Colour names are specified in the link below. Pi Presents also allows 8 hex digit colour values, the last 2 digits being the opacity. A value of 0 for opacity is transparent. Pi Presents uses the CSS color descriptor specified here: https://www.w3.org/TR/css-color-3/#color This describes many other colour values that could be used. 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	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
Message Tab		
Message Text	Welcome	The text to be displayed.
Message Text Location	+/media/ mytextfile	If not blank the file containing the Message Text, takes priority over Message Text field. This field or text from the file cannot be modified by a track plugin.
Message Text x Position	100	distance of the left end of the text from the left of the screen (pixels) If blank then the text is centred on the show canvas
Message Text y Position	100	distance of the top of the text from the top of the screen (pixels) If blank then the text is centred on the show canvas
Message Text Type	html	plain – plain text html – a panel containing text, images and formatting controlled by a subset of html. See Section 5.4.4.4
Plain Text Font	bold 30pt Helvetica	Examples: 20pt arial bold 10px helvetica bold italic 10pt arial Pi Presents uses the CSS font descriptor specified here: https://www.w3.org/TR/css-fonts-3/#propdef-font This describes many other font characteristics that could be used.
Plain 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 or transparent. Colour names are specified in the link below. Pi Presents also allows 8 hex digit colour values, the last 2 digits being the opacity. A value of 0 for opacity is transparent. Pi Presents uses the CSS color descriptor specified here: https://www.w3.org/TR/css-color-3/#color This describes many other colour values that could be used.

		If blank then the value in the parent show is used.
Plain Text Justifiy	left	left/center/right. Text justification.
HTML Text Height	300	The height of the HTML text panel
HTML Text Width	300	The width of the HTML text panel
HTML Text Background Colour	#ff0000	The background colour of the html text panel
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 7.1
Text Tab		
		See Image Track. Does not have Pause Text
Show Control Tab		
		See Image Track
Animation Tab		
		See Image Track

5.4.8 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.

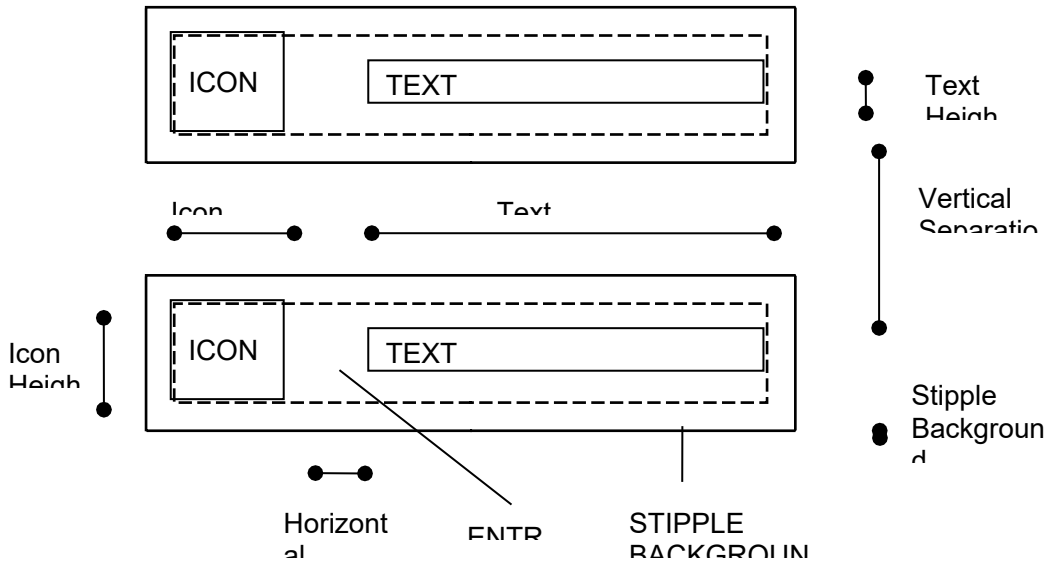
Field	Example	Values
Type	show	
Title	My Other Show	Displayed on a menu and in the editor
Track Reference		A label for the track - blank if the track is included in a menu or mediashow. Non-blank if the track is to be triggered from a symbolic name or is special.
Show to Run	myothershow	Show Reference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

5.4.9 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

Menu Window

The menu window determines the position of the menu in the Show Canvas. Unlike other tracks the width and height do not determine the width and height of the menu. Instead the width and height are determined by the menu's geometry. The width and height is used to draw the red rectangle around the menu when Guidelines are enabled so that the geometry can be adjusted to fit within the menu's intended extent.

Note: At this time only x1+y1+width*height is of practical use.

- x1+y1+width*height
  x1 and y1 determine the top left corner of the menu. width and height together with x1 and y1 determine the location of the Guidelines.

- width*height
  The menu is displayed centred on the Show Canvas. This will work only if width and height have been manually adjusted to match the actual dimensions of the menu by using the Guidelines

- fullscreen
  The menu is centred on the Show Canvas. This will work only if the width and height have been manually adjusted to match the actual dimensions of the menu by using the Guidelines

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_1p4 If this is not specified then an error message will be shown.
-c --cursor	Normally the cursor is hidden when in fullscreen mode (-f). --cursor displays the cursor in fullscreen mode.
-f --fullscreen	Run Pi Presents in full screen mode.
-o --home <arg>	Location of the Pi Presents 'data home' directory e.g. /media/pi/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. • 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. The pp_editor validate menu option will detect many Profile Errors. Tell me about ones that are not detected by the editor. Log output is also displayed in the terminal window if Pi Presents is started from one, and is also reported in the file /pipresents/pp_log/pp_log.txt 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.
-d --debug <arg>	Using the debug option with <arg> gives finer control of logging, see Section 22
-l --liveshow <arg>	Liveshow tracks are always played from the directory pp_live_tracks

	in the data home specified in the –o option defaulting to /home/pi/pp_home/pp_live_tracks If the –l option contains a directory path this is used as a second source for liveshow tracks. e.g. /home/pi/mylivetracks
--counters	When counter storage is enabled initialise counters from the Start Show, Counters tab.
--manager	Used by Manager for Pi Presents to disable terminal messages. Should not be used by users.
-n --nonetwork	When the Time of Day Scheduling is required and this option is omitted Pi Presents will wait up to 5 seconds for the LAN (network) to be available. If the option is included with no argument then it will not wait, this is useful of you have a RTC and do not require ntp time. Non-intuitively if you need to wait for a different time then use --nonetwork n to wait n seconds.

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/pi/KINGSTON

Raspberry Pi OS 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.

Raspberry Pi OS 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 gpiozero.cfg file is present in the /pp_io_config directory in a profile.

If the gpoizero.cfg file from /pipresents/pp_resources/pp_templates is copied to a profile, then pins of the P1 connector (the 40 pin 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 15.2.2

Pin of P1 connector	Symbolic Name defined in gpiozero.cfg	Command in the Controls Field of Tracks/Shows
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-shutdownnow	This is a special symbolic name and does not need to be used in the Controls Field. See Sect 6.7 It is configured in gpiozero.cfg such that a press for 5 seconds is required. Closes Pi Presents and shuts 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 Pull-up is set to up and Activated-name set to the required Symbolic Name.

- 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 Pull-up = up and Deactivated-name to the symbolic name.

Inputs can be changed from normally open to normally closed and vice versa by changing the de/activated name which is used for triggering as described in Section 15.2.2

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. If you have problems with contact bounce increase the Bounce Time of the appropriate pin by modifying gpiozero.cfg (See Section 15.2.2).

6.5 Screen Blanking and Cursor

If the -f command line option is used the cursor will not be shown while Pi Presents is running. If you want to override this and show the cursor use the -c option in addition.

Screen Blanking is controlled by the preferences>raspberrypi configuration menu of the RPi OS.

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.

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.

For Bookworm you will need to edit the file :

/home/pi/.config/wayfire.ini

add the following lines to the start:

[autostart]
pp-gtk = $HOME/pipresents/autostart.sh

then edit the file:

/home/pi/pipresents/autostart.sh

for the profile you want to run and optionally to activate a virtual environment

Finally alter the permissions of autostart.sh to make it executable by the Owner. Using the File Manager right click on the file and select Properties>Permissions.

6.7 Shutdown the Raspberry Pi from the GPIO

Providing a suitable gpiozero.cfg has been employed, (see next section) press the Shutdown button (P1-12) 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

Exiting/Restarting Pi Presents and Shutdown of the RPi

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

- Bind a key, pin or click area to the internally defined symbolic name pp-shutdownnow as described in Section 15.2. An input event with this symbolic

name will cause the Pi to shut down immediately. For GPIO the appropriate pin can also be configured such that the event is not sent unless the pin is held in a particular state for a period of time. (See table below)

- Use the Show control command 'shutdownnow' to shut down the Pi immediately from within a Show, or 'reboot' to reboot the RPi, See Section 7.2
- Use the Time of Day Scheduler
- Remotely via an OSC command

Exit Pi Presents
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 Table 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 or the Close icon on the Pi Presents window

Restarting Pi Presents

There is one way to restart Pi Presents:

- Bind a key, GPIO pin, or click area to the internally defined symbolic name pp-restartpipresents. (see table below)

The function uses the method here: https://blog.petrzemek.net/2014/03/23/restarting-a-python-script-within-itself

You must make pipresents.py executable.

Note the requirement to flush any externally connected files.

Note: These are special symbolic names and do not require entries in the Controls field of Shows or Tracks.

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.
pp-restartpipresents	Restart Pi Presents having closed all shows. The restart use the original command line options.

6.8 Python Virtual Environments

Creating an environment and installing libraries

In order to install and use python libraries using pip it is necessary to install a Virtual Environment in the /pipresents folder. To do this type:

```
python -m venv --system-site-packages venv
```

which will create a virtual environment called venv.

Now activate it with the command:

```
source venv/bin/activate
```

You can now install libraries from pip using:

```
pip install .....
```

Running Pi Presents that uses pip installed libraries

First activate the virtual environment

```
source venv/bin/activate
```

then run Pi Presents as normal.

Autostart

The file autostart,sh has options for activating a virtual environment.

7 Controlling Tracks and Shows

7.1 Controlling Track Movement in Individual Shows

Control of an individual show uses commands such as up, down, play, 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, gpiozero.cfg) the physical keys are bound to symbolic names. e.g. for gpio activated-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 I/O plugins 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. gpiozero.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

For outputs a similar system is employed. Animation commands include symbolic names. Output device drivers use configuration files such as gpiozero.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 only 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. The bindings in the Controls field of a top level show are not inherited by its subshows; each show must have its own set of Control bindings.

7.2 Opening and Closing Shows

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 to open or close a show e.g.

  open myothershow
  close myothershow

- Open and Close a show remotely using the Open Sound Control (OSC) 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 that open and close shows are:

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 of the show is not running.
closeall	Closes all shows. Pi Presents is still running.
openexclusive <show-ref>	Closes all shows (including the show that the command was sent from, if applicable) and opens the specified show.

7.3 Time Of Day Scheduler

The time of day scheduler opens and closes shows at a specified time and date. It will also trigger any other Show Control command at a specified time and date. The scheduler is enabled and controlled by a profile using the Schedule tab in the Start Show and the Schedule tab in every other show.

The Schedule tab of the Start Show allows the Time of day scheduler to be enabled. It also controls functions of Pi Presents common to all shows.

Every other show in a profile has a Schedule Tab which controls the schedule for that show. Each has 4 fields - everyday, weekday, monthday and specialday. Each has zero or more match criteria (day) with an associated list of times (times) There is an example in the example profile pp_timeofday_1p6. The aim is to eliminate the need to type in the date of every day of the year, instead everyday is used as a base and then exceptions to those are added either for days of the week, days of the month, or special days.

Pi Presents prepares a list of tasks for today when Pi Presents is started and at every midnight. For each show it considers each field in the order below and each set of day/time lines, matching today's date with the match criteria:

- everyday always matches. (the day line must have the word everyday)
- 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 open and close commands are remembered. If a later set of day/time lines or field also matches it overrides the previous matches for the specified show. A set of day/time lines with a day line but an empty list of times can be used to delete times from a

previous match. When all matches are done the list of tasks 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 show-ref of the show.

If Pi Presents is started in the middle of a day then it will try to catch up by going through today's list of tasks for each show (not for the Start show) and starting the task that should be running at the current time. Catchup can sometimes be problematic as tasks are started late and may overlap another later task or be truncated; for this reason catchup can be disabled for individual shows.

The Start Show 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 all the Show Control Commands with parameters. (It is possible to use open and close show commands in this section, specifying the show-ref, instead of in individual shows.) There is no catchup for the Start show, all previous and current tasks for the day are just ignored.

Gotchas:

- When using the Time of Day scheduler it is best not to include shows in the Start Shows field of the Schedule Tab of the Start Show, or to use Show Control commands to open or close shows. These are not taken account of by catch up and can result in unexpected results.

- Any commands in the Schedule tab of the Start show are not subject to catch up

- The scheduler should be time zone agnostic. Be careful with shows that run across midnight. I have not investigated the effect of changing to/from Daylight Saving Time.

There is a logging option specifically for testing the schedule (See Section 6.1). This shows time as Time of Day and events that are of interest.

To test the schedule with a real clock is difficult thus the Simulate Time tab in the Start Show can be used to enable a simulated time for exercising the schedule (and only the schedule). If enabled, then the simulated time you wish to start the test should be specified. The date need not have consistent day/month/year as each of the match criteria are independent.

The scheduler requires that time of day be available when Pi Presents starts; to this end Pi Presents waits for up to 5 seconds for the LAN network and hence for time from a ntp server to be available. This primarily allows wifi to connect. You can control this wait with the --nonetwork command line option.

7.4 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.

7.4.1 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 Show Control Commands

Show Control commands were first introduced to open and close shows from other shows, hence the name. They have gradually been expanded to implement a number of miscellaneous functions which are described below. Show control commands can be executed in a number of places:

- At the beginning and end of Tracks
- At the beginning and end of Shows
- When a liveshow's livelist becomes empty or not empty
- As a result of receiving an input event with a symbolic name (Section 8.4).

8.1 The Commands

Command	Use	Section
open	open a show	8.1
close	close a show	8.1
openexclusive	close all shows then open a show	8.1
closeall	close all shows	8.1
exitpipresents	exit Pi Presents	
shutdownnow	exit PP and shutdown the Rpi	
reboot	exit PP and reboot the Rpi	
beep	Play a short audio file	8.5
vibe	Plays a haptic vibe	8.6
monitor	control the monitor using xrandr commands	8.3
cec	control the monitor using CEC commands	8.3
osc, OSC	send an OSC command	16
event	generate an input event	8.2
counter	control counters	13
backlight	Control the backlight of the official touchscreen	9.3.1
animate	The arguments are any data used by Animation commands without the leading delay parameter. Note: For tracks the animation field is more flexible	14

8.2 Sending Events between Shows

The Event show control command allows a show to generate input events which are sent to all running shows. As for user generated events in Section 7.1 the shows and tracks receive these input events and convert them into the Commands that control the show/track.

Command	Use
event <symbolic name>	send an input event with the symbolic name to all running shows.

8.3 Controlling the Monitor

Show control commands can be used to control the monitors. Two alternative control protocols are supported.

Command	Action
monitor on <monitor-name>	Sends the xrandr --output display_name] --preferred command to the monitor. This allows the video to the monitor. display_name can be DSI0,HDMI, HDMI0, HDMI1
monitor off <monitor-name>	Sends the xrandr --output [display_name] --off command to the monitor. This inhibits the video to the monitor. display_name can be DSI0,HDMI, HDMI0, HDMI1
cec standby <device>	Sends the 'echo "standby <device>" | cec-client –s' command to the hdmi output (See below) The optional <device> is the device number which defaults to 0
cec on <device>	Sends the 'echo "on <device>" | cec-client –s' command to the hdmi output (See below) The optional <device> is the device number which defaults to 0
cec scan	Sends the 'echo scan | cec-client -s -d 1' command to the hdmi output A list of devices on the CEC bus will be displayed in the terminal window for debugging purposes (See below)
cec as	Sends the 'echo "as" | cec-client –s' command to the hdmi output Makes the Pi's CEC adapter the active source

You may need to install cec-utils - sudo apt install cec-utils

On a Pi 4 HDMI CEC currently works only from the hdmi0 port

8.4 Show Control on Event

The Show Control on Event field of a Show allows Show Control Commands to be associated with the symbolic name of input events. The association is for this show and its media tracks only, not for any subshow resulting from the use of Show Tracks.

The Show Control Controls field contains a symbolic name preceeding a Show Control Command e.g.

pp-pause open myshow

When an input event is received it is tested against the list of Show Control Controls and the commands associated with the all matches in the list executed. The input event is then considered for Track Controls.

8.5 Beeps

The beep command plays a short audio file to the audio device. There is no control of the playing.

beep <file> <device>

The file to be played is specified in the command. It can either be under the pp_home directory or under the profile directory:

+/myfile.wav - the file is under the pp_home directory

&/myfile.wav – the file is under the current profile directory

Acceptable file types are uncompressed audio files playable by the linux PAPLAY command (.wav and others) or, if mpg123 is installed, .mp3 files.

```
sudo apt install mpg123
```

The device field is optional. If not specified the audio device is that selected on the Raspberry Pi OS taskbar. The valid values are HDMI0, A/V, USB, USB2, bluetooth and for Pi4 onwards HDMI1.

8.6 Vibes

The vibe command plays a haptic vibration using the DRV2605 Haptic Controller. The command is:

vibe ,<sequence>

<sequence> is of the form 16,-100,16

In order for the Vibe Show Control command to function a suitable DRV2605.cfg file must be present in the profile in the directory pp_io_config. Only the library and device name is used from the config file.

Use of the DRV2605 is explained fully in Section 15.2.9 and in the DRV2605.cfg file in /pipresents/pp_resources/pp_templates.

The DRV2605 can play only one vibration at a time. A sequence produced by a Show Control command, in particular Show Control on Event, will stop use of audio to vibe and any sequence.

9 Setting Up Displays and Touchscreens

9.1 Setting Up Raspberry Pi Displays

The Pi4 onwards has 2 HDMI outputs (Pi3 and earlier 1 HDMI), an A/V output, and 2 Official DSI connected touchscreen monitors. Any combination of the two DSI monitors and the two HDMI monitors can be used by Pi Presents.

Pi Presents can cannot use the A/V monitor. If you require A/V then HDMI to A/V convertors are available at a reasonable price.

The 'Display' field in a Show entry in a profile can be used to select which Display to use for the Show and a field in screen.cfg can be used to determine which Displays click areas areas are to be defined for.

To achieve correct automatic configuration of Displays by Pi Presents the following should be observed:

- Connect all required screens before powering up the Pi. The Raspberry Pi OS does not cope well with changes of monitor. It is best to power down the

Pi before changing monitors. After restart to use the Screen Configuration menu item to ensure that the correct monitors are active.

- If a HDMI screen is not required unplug it from the Pi. It may not be sufficient to turn the monitor off as the Pi powers a small part of the monitor that reports its presence.

- Use the Raspberry Pi OS Screen configuration menu option to arrange the connected monitors - HDMI0 (HDMI-1), HDMI1 (HDMI-2), and DSI0 (DSI-1) in any configuration, provided they are not overlapping and either the left or top sides line up. The order does not matter for Pi Presents, just choose what best suits your development setup.

- Use the Raspberry Pi OS Screen Configuration menu option to select the rotation of the displays.

Information about PP's interpretation of display configuration can be obtained by using the -d option in the Pi Presents command line or the command:

```
python pp_displaymanager.py
```

When Pi Presents is running, the way the Displays are shown depends on the --fullscreen command line option.

- If the --fullscreen (-f) option is in use then the displays will be displayed full screen without decoration

- If the mode is not fullscreen then the single monitor, or HDMI0 for multiple monitors, will be displayed in a resizable window. The size of the resizable window can be adjusted in /pipresent/pp_config/pp_display.cfg

9.1.1 Running without Displays

An application using audio only and automatically started at Boot requires no display. Disconnecting both DSI and HDMI will cause Pi Presents to generate an error unless the display used for all shows in the application is set to NODISPLAY.

NODISPLAY will also allow the application to be displayed over VNC. The size of the display seen over VNC can be set by using the RPi OS Screen Configuration menu item, when using VNC and with no displays connected, to select the size of the display.

9.2 Specifying the Display to Use for a Show

Show entries in a profile have a 'Display' field which determines the display on which the show and any of its subshows are displayed. The Display field is used only if the Show is top level; subshows use the same Display as the top level show.

9.3 Setting Up Touchscreens

Touchscreens are made from an ordinary monitor with a resistive or capacitive surface which measures the position of the touch. For the official touchscreen both monitor and touchscreen data is transferred by the DSI connector. For HDMI monitors there is usually a separate USB connector for the touch surface. The touch facility requires a software driver which may or may not be provided by the manufacturer.

The touch surface requires calibration. This is achieved automatically by the Raspberry Pi OS when the position or orientation of monitors are changed.

The Official touchscreen has a backlight. Software must be installed to control the backlight as described in Section 9.3.1.

9.3.1 Controlling the Backlight

Show Control commands can control the backlight of the Official touchscreen. To enable this the python library rpi-backlight must be installed and a udev rule set. Details of how to do this are here:

https://rpi-backlight.readthedocs.io/en/latest/installation.html

The library must be installed in a virtual environment (Section 6.8).

Once installed the backlight can be controlled by linux commands. Check that the backlight works by using the rpi-backlight commands.

Raspberry Pi OS remembers the brightness setting through power cycles. Pi Presents initially uses these settings and, when it exits, will set the backlight power to On and restores the original brightness. During use of Pi Presents it can be controlled by the following Show Control Commands.

backlight on backlight off	Turn the backlight on or off
backlight set <level>	Set the brightness to a value between 0 and 100
backlight inc <delta>, backlight dec <delta>	Increment or decrement the brightness by <delta>
backlight fade <level> <time>	Fade the backlight brightness to <level> over <time> seconds

10 Using Touchscreens and Soft Buttons

See Section 9 for setting up the Pi to use touchscreens

10.1 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.

10.2 Click Areas

In its screen.cfg file Pi Presents allows the definition of click areas. These are rectangular or circular 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 15.2.10

10.3 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 rectangle 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

11 Setting Up Audio

The Pi Presents profile editor references audio outputs by the names of the outputs screen printed on the circuit board - HDMI0, HDM1 and A/V (not Pi5). It also allows two USB sound cards and a bluetooth output.

Pi Presents uses Pulseaudio and there is a need to convert Pi Presents names to Pulseaudio sink names. This is achieved in a Pi Presents configuration file:

/pipresents/pp_config/pp_audio.cfg.

The Pulseaudio sink names are different between models of the Pi and, for outputs other than the internal ones, are not known until the audio device is connected so you will need to edit this file if adding a USB or bluetooth device or if HDMI audio does not work. Details of how to do this are in the file.

Information about PP's interpretation of the Pi’s audio configuration can be obtained by using the -d option in the Pi Presents command line or by running python pp_audiomanager.py'

Pi Presents tests if an audio device is connected if it appears in a profile and the profile is run. When using a Touchscreen or when no HDMI displays are connected, maybe for testing, a ‘no connection’ error can be generated. The A/V audio device is always connected so can be used in these circumstances.

12 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, web, and image tracks. The media file types that a Liveshow recognises are in the first few lines of the file pp_definitions.py.

The Pi Presents Manager Section 18 can upload or import tracks; alternatively 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.

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.

13 Counters

Counters are intended for quizzes but could have other uses. All types of track have facilities to create, set, increment, decrement and delete counters. Currently the value of counters is displayable only by writing track plugins which requires a little knowledge of Python (or some copy and paste of the examples).

Counter commands are stored in the ‘Show Control Begin’ and ‘Show Control End’ fields of tracks and shows. Counter commands are of the form:

counter [name] [command] [parameters]

The counter field must be present to differentiate counter commands from other types of command in Show Control fields

The commands are:

command	parameters	example	use
set	name value	counter fred set 0	Create a counter called name and set its value to value. If the counter already exists just set its value.
inc	name value	counter fred inc 1	increment name by value (which must be positive)
dec	name value	counter fred dec 2	decrement name by value (which must be positive)
delete	name	counter fred delete	delete the counter name

Counters are stored in a Python dictionary called counters in the Python Class CounterManager which is defined in the file pp_countermanager.py

There are two examples of track plugins which use methods in this class to display counters, these are in /pipresents/pp_track_plugins:

- krt_counters.py

  This track plugin shows the use of all the display methods exposed by CounterManager

  - get_counter(name) – return the value of counter name as a string
  - str_counters() – returns a string with the name/value of all currently defined counters.
  - print_counters() – print the name/value of all currently defined counters to the terminal window.

  The example profile pp_counters_1p6 uses this plugin

- krt_quiz.py

  Used by the example profile pp_quiz_1p6 to provide its prettier output.

Out of the box counters and their values are destroyed when Pi Presents exits. The next section describes how counters and their values can be stored.between runs of Pi Presents.

Storing Counters

The value of counters can be stored in a file between runs of Pi Presents. They are stored in the profile in the file counters.cfg. This is a text file and can be read or edited by a text editor.

Storage is enabled by setting ‘Store Counters’ to Yes in the Counters Tab of the Start Show. When enabled all counter values will be stored in the file each time they are modified and the counters will be loaded from the file when Pi Presents starts.

To define the counters and set them to an initial value the --counters option can be used in the command starting pipresents.py. If this option is provided the file counters.cfg is populated from the ‘Initial Values’ field of the Counters tab. The format of this file is as used by python’s configparser:

fred = 1
bill = -1

There are other possibilities for initialising counters such have having a special show controlled by a hidden button or the time of day scheduler.

14 Animation Control

All types of track have facilities to control animation. Commands included in the 'Animation at Beginning' and 'Animation at End' fields are sent to I/O Plugins (Sect. 15) in order to drive physical outputs.

An example of animation commands is shown below. This applies to GPIO state control but there are other animate command parameter types.

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 real number or 0.
- Symbolic Name - The name of the output as defined in the gpiozero.cfg or other .cfg file in the profile (Section 15.2.2).
- Parameter Type
- Parameters - 1 or more fields compatible with Parameter Type.

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 commands are put in a queue for firing at the appropriate time. The time is not affected by pausing a track.

Every command is offered to every active I/O plugin. If the Symbolic Name and Parameter Type matches any of those implemented by the plugin then the command will be executed using the provided parameters.

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 an output 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.

14.1 Timing Problems with Animation

If Pi Presents is to be autostarted when the Pi is booted the setting of the Pi's clock just after the Pi is connected to the internet can cause animation timing to be incorrect. To stop this:

a. Turn off wi-fi and/or disconnect the Ethernet cable

b. In the Raspberry Pi Preferences menu select 'Do not wait for the network'

c. Add the -n option to the Pi Presents command line so that PI Presents does not wait for the network.

15 Input/Output (I/O) Plugins

In Pi Presents Input and Output is independent of the core execution of shows and tracks. Developers can interface to their own input/output devices by writing additional I/O plugins. These plugins have a standard API with the core and will use configuration files to enable and configure them.

An input plugin accepts an input from a physical device, pre-processes it and produces an input event with a symbolic name, some also allow input values to be accessed by track plugins. An output plugin takes output commands (e.g. out1 state on) having a symbolic name and parameters which it uses to generate the physical output.

I/O plugins are Python modules that should be placed in the directory /pipresents/pp_io_plugins. An I/O plugin needs to be configured with a .cfg file. These files are stored in the /pp_io_config directory in a profile. There can be more than one .cfg for a plugin, for example pp_inputdevicedriver.py may have a .cfg file for each make of remote control.

Most I/O plugins have a standard API and standard .cfg files. There are currently I/O plugins with standard configuration files and standard API for:

Device	Provided Configuration File	Direction	I/O Plugin	Use
GTK Keyboard	keys.cfg	Input	pp_kbddriver.py	Interfaces with a keyboard using the GTK4 API, not a totally standard API. This replaced the Tkinter driver in earlier versions of Pi Presents.
GPIO	gpiozero.cfg	Input/ Output	pp_gpiozerodriver.py	Interfaces the GPIO general purpose pins using GPIO Zero This replaces gpio.cfg which does not support Pi5.
Input Device	osmcremote.cfg	Input	pp_inputdevicedriver.py	Interfaces with a specific remote controls, keypads etc. using evdev
Serial device	serialdriver.cfg	Input/ Output	pp_serialdevicedriver.py	Interface with serial devices usually using a USB to RS232 convertor
I2C devices	i2c.cfg	Input/ Output	pp_i2cdriver.py	Interfaces with I2C devices
RFID Tag	pn532driver.cfg	Input	pp_pn532driver.py	Interfaces with PN532

Reader				based readers
DRV2605 Haptic Controller	DRV2605.cfg	Output	pp_DRV2605driver.py	Controls ERM and LRA Haptic actuators

There are two I/O plugins that have a non-standard configuration file and a non-standard API, however their elements are stored as above:

touchscreen	screen.cfg	Input	-	Enables touch/click sensitive areas on a touchscreen/monitor and configures the look of the touch sensitive areas.
remote control using OSC	osc.cfg	Input/Output	-	Allows Pi Presents to control or be controlled by other units. Configuration file is generated using the editor.

15.1 Enabling a Standard I/O Device

To enable an I/O device a 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.

When Pi Presents starts it will look in the /pp_io_config directory in the profile and will use any .cfg files found there. It will then look in /pipresents/pp_io_config and will use a .cfg files found there if a .cfg file with the same name has not been found in the profile.

Out of the box keys.cfg is present in /pipresents/pp_io_config so that the GTK4 keyboard driver is enabled.

The .cfg files are all text files which can be edited by a text editor. 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

or if you want them to be used by all profiles to /pipresents/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.

15.2 Configuring Inputs and Output Drivers

Section 7.1 describes the input system of Pi Presents and how external physical events are converted to input events with symbolic names. Most of the responses are configurable using the files described in this section.

When editing I/O 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.

All standard I/O configuration files must have a section called [DRIVER]. This must contain the following fields:

Field	Example	Use
title	GPIOZERO	Text which is used when reporting activity of the I/O plugin in error reports and logs.
enabled	yes	yes/no. An I/O plugin becomes active if it has a .cfg file in the appropriate directory and enabled = yes . This field enables an I/O plugin to be made inactive without removing its configuration file.
module	pp_gpiozerodriver	The name of the python module which implements the I/O plugin, without .py. The file must be in /pipresents/pp_io_plugins There can be more than one .cfg file referring to the same module, for example if there are two remotes with different key bindings.

The [DRIVER] section can contain other fields used by a specific I/O plugin.

15.2.1 Interfacing with GPIO using pp_gpiozerodriver.py

NOTE: This driver is implemented using the GPIO Zero library

https://gpiozero.readthedocs.io/

GPIO Zero will function on Pi5 in addition to all other models of Pi. It is the preferred driver for GPIO. It replaces pp_gpiodriver,py based on RPI.GPIO which does not support Pi5 or later models.

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 file in /pipresents/pp_resources/pp_templates/gpiozero.cfg is an example which, when copied to the/ pp_io_config directory in a profile enables GPIO for controlling Pi Presents. The pin settings are those used in some of the examples. It can be used as a basis for your own profiles by copying it to the profile and editing it.

The .cfg file maps physical P1 connector (the 40 pin connector) GPIO PHYSICAL input and output pins to the symbolic names of inputs and outputs. It also configures the input pins.

A section for every general purpose pin must be present in the file. Pins not in the template file are not general purpose pins. A pin with direction=none is ignored.

Inputs

NOTE: Unlike earlier GPIO drivers, gpiozero use 'active' and 'inactive' rather then rising and falling. A button is active if it is pressed and inactive when released. The input voltage for active and inactive is determined by the pull-up or active-state fields.

If you connect a button between an input and 0 volts (the preferred connection) then the pull-up field should be set high so that the internal pull up resistor is connected between the input and +3.3volts. In this case 'active' is when the input is connected to 0 volts by pressing the button.

If an internal pull up resistor is not required the pull-up should be set to none and active-state set to high or low as appropriate.

The relevant fields in the .cfg file are:

- pull-up - internal pull up. Values are up/down/none. up to +3v, down to 0v. If pull-up is up activated is 0v, if pull-up is down activated is +3V

- active-state - high/low, high means +3v=active, low means 0v=active. Needs to be set if and only if pull-up is none

Each input can generate an event having the specified symbolic name in any of three ways:

- When an input pin changes from inactive to active (activated-name)
- When an input pin changes from active to inactive (deactivated-name)
- When a pin is held in the active state for a period of time (held-name)

The relevant fields are:

- activated-name - symbolic name of Activated input event, if blank no event is generated.

- deactivated-name - symbolic name of Deactivated input event, if blank no event is generated.

- held-name - symbolic name of the Activated Hold event generated after the hold time, if blank no event is generated.

The held-name event is generated after hold-time seconds. If hold-repeat is yes then the held-name event is repeated after repeat-time seconds.

The relevant fields are:

- hold-time - time in seconds (float) that an input must be held active before the Hold event is generated

- hold-repeat - yes/no repeat the Hold event after repeat_time

- repeat-time - time in seconds (float) that an input must be held active before the Hold event is re-generated

Mechanical contacts such as a push button can bounce on and off for a period of time as the connection is made or broken. Bounce-time can remove this false state change. A value of 0.1 seconds is typical.

- bounce-time - input must be steady for this number of seconds (float) for a state change to be registered.

An output can be follow an input. If you need to invert the linkage then set active-high in the output pin.

- linked-output - board name of output pin that follows the input

Outputs

NOTE: Unlike earlier GPIO drivers GPIO Zero uses on and off to mean active or inactive. active-high determines whether On is 0V or 3.3V

- If active-high = high the On State produces +3v
- If active-high =low the On State produces 0 volts

Raspberry Pi initialises GPIO outputs to 0 volts so it is best to set active-high to high and to design relays etc. for positive logic (which many of those on the market are not). The outputs are reset to 0 volts when Pi Present exits.

The fields are:
- name - symbolic name to be used in output command
- active-high - high/low

Two animation command use the gpiozerodriver:

GPIO State control

The command is of the form:

delay output_name parameter_type parameter_value

e.g. 0 LED state on

- The output_name is compared against the name field of all pins having direction = out
- The parameter_type field should be state for GPIO state control. The animation command will be executed only if the output_name and parameter_type match.
- the parameter value must be on or off, if not an error will be flagged.

Blink a GPIO Output

The command is of the form:

delay output_name parameter_type parameter_values

e.g 0 LED blink 0.5 0.5 5 to blink a LED 5 times at 1Hz

- The output_name is compared against the symbolic name field of all pins having direction = out

- The parameter_type field should be blink for blink control. The animation command will be executed only if the output_name and parameter_type match.

There are three fields in the parameter_values:

on_time, off_time and n

- on_time - the time is seconds (float) that the LED is on

- off_time - the time in seconds (float) that the LED is off

- n - the number of cyles to be produced. If zero the blinking will continue until it is stopped by the state off animation command

15.2.2 Interfacing with GPIO using pp_gpiodriver.py

NOTE: This driver is implemented using RPI.GPIO library which is obsolescent and does not work with Pi5. gpiozerodriver.py should be used instead.

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 out of the box is defined in the file gpio.cfg.

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

The .cfg file maps physical P1 connector GPIO input and output pins to the symbolic names of inputs and outputs used by the Pi Presents examples. It also configures the input pins and sets up P1-12 as the shutdown button.

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.

De-bouncing
De-bouncing is applied to each pin individually using the threshold parameter. This number of contiguous samples with no change of input value must be detected for a change of state to be reported. De-bouncing is used to eliminate mechanical contact bounce. It might also help with PIR false triggering.

Linked Outputs
The optional linked-output and linked-invert fields allow a gpio output to be directly connected to a gpio input. See /pp_resources/pp_templates/gpio.cfg for details

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). The outputs are reset to 0 volts when Pi Present exits.

When sending the animation command:

delay output_name parameter_type parameter_value

- The output_name is compared against the name field of all pins having direction = out

- The parameter_type field should be state as this is Parameter Type supported by pp_gpiodriver.py. The animation command will be ignored for any other state even if the ouput_name matches.

- the parameter value must be on or off, if not an error will be flagged.

15.2.2.1 A more responsive alternative - pp_pigpiodriver.py

NOTE: This driver is implemented using a library which is obsolescent and does not work with Pi5. gpiozerodriver.py, which may not be as responsive, should be used instead.

pp_pigpiodriver.py provides the same functionality as pp_gpiodriver.py but is implemented using Joan's pigpio (http://abyz.me.uk/rpi/pigpio/) instead of RPI.GPIO. Since the inputs are sampled every 5uS., using DMA, instead every 10mS. it is far more responsive to short button presses, particularly when running multiple copies of MPV to display streaming cameras.

The file pigpio.cfg in /pipresents/pp_resources/pp_templates is an example of a configuration file for the driver. All fields for the pin sections are the same as in gpio.cfg except that 'threshold' is replaced by 'steady'.

steady - is the time in mS. that the input must stay at the same level for a state change be recorded, see http://abyz.me.uk/rpi/pigpio/python.html#set_glitch_filter

In order to use this driver the pigpio daemon must be run after boot up by the command sudo pigpiod, either from the terminal or in autostart.

15.2.3 Interfacing with a Remote Control or Keypad using pp_inputdevicedriver.py

Note: Since converting input to GTK4 I have found that the wireless remote controls and keypads I have tested do not require pp_inputdevicedriver.py. To find the keycodes and keynames a device produces run pp_displaymanger.py and add the appropriate keynames to keys.cfg. My testing does not include the OSMC remote mentioned below.

Remote controls etc. that interface with Linux userland by providing a file in /dev/input and which provide key presses should be able to interface with Pi Presents using the pp_inputdevicedriver.py I/O plugin. The template file osmcremote.cfg configures this plugin to work with the official OSMC remote.

https://osmc.tv/store/product/osmc-remote-control/

The [DRIVER] section must contain the following additional fields:

Field	Example Content	Use
device-name	HBGIC Technology Co., Ltd. USB Keyboard Mouse	The name of the device producing the inputs. Only inputs from this device will cause an input event to be generated. (But see Duplicate Events below). Run the program input_device.py to find this name for your remote. This program requires python-evdev which will need to be installed using apt or pip, the latter under a virtual environment (See Section6.8)
key-codes	KEY_STOP,KEY_PLAYPAUSE,KEY_DOWN, ….	The list of key codes that Pi Presents is to respond to. Run the program input_device.py to find the key codes for your remote.

tick-interval	50	The interval between polls of the remote in mS.

There are also sections for each button, the content of which is similar the gpio.cfg. Details in /pipresents/pp_resources/pp_templates/osmcremote.cfg. Some buttons may produce more than one code, a section is required for each code.

NOTE: if you are getting two input events from some remote control buttons read the GTK4 Keyboard Section 15.2.6

15.2.4 Interfacing with a Serial Link using pp_serialdriver.py

Serial links such as RS232 are often used for the control of projectors. The RPi provides an inbuilt serial link which is configured for factory debugging. It is difficult to use and will almost certainly require voltage level convertors. A better way is to use a USB to RS232 adaptor. Adaptors such as this are exposed to Linux as files like /dev/ttyUSB0 and can use the pyserial python library.

The pp_serialdriver.py I/O plugin uses the pyserial library to drive serial devices. It has been tested with a usb connected device interfacing with a Windows XP machine and Windows 98 machine. The template file serialdriver.cfg in /pipresents/pp_resources/pp_templates is an example of using the I/O plugin. In the example the serial link is used to drive a projector but the configuration will require modification and validation for your projector.

Details for using the configuration data are in the file:

/pipresents/pp_resources/pp_templates/serialdriver.cfg

The driver allows both input and output.

Input
Characters received from the link are matched against characters and strings defined in the configuration file. If a match occurs an input event with a symbolic name, also defined in the configuration file, is generated:

- As each character is received it is matched against the configuration data to generate and input event.
- When an end of line character is detected the characters received after the previous end of line are matched against the configuration data to generate an input event.

Events can be generated from any-character, specific-character, any-line and specific-line

Output
Output to the serial link uses animation commands. There are two ways to use the command, determined by the value field in the configuration data:

- preset – the characters to be output are defined in the configuration data. The configuration data defines the message to be sent for each combination of name, parameter type and parameter value and the driver translates this into the message to be sent.

For example the command ‘0 projector state on’ will be translated into a sequence of bytes 02 00 00 00 01

- explicit – the characters/bytes to be output are in the animation command. Examples are:

  0 serial-send bytes “02 00 00 00 01”
  0 serial-send string “the cat sat on the mat”

15.2.5 Interfacing with I2C devices using pp_i2cdriver.py

Many devices can interface with the RPi using the I2C bus. pp_i2cdriver.py supports a number of devices which may be of use to Pi Presents users. However the main use of this driver is to provide an example of how to interface your own device with Pi Presents.

To use the provided pp_i2cdriver.py it will be necessary to enable I2C in RPi Preferences menu.

The following devices are supported by pp_i2cdriver.py

Device	Input/Output	Symbolic Name	Parameter Type
Adafruit MCP4725 DAC	Output	dac	set – set the DAC to a ‘percentage’*** of 3.3 volts mirror – mirror the named ADC input (analog1,analog2,analog3) fade – ramp the output from a to b in t seconds. a and b are expressed as ‘percentages’***.
Pimoroni Four Letter Phat	Output	fourletter	string – display a four letter string num-string – display a 4 digit number with decimal point blank – blank mirror-voltage – display the voltage applied to the adc input mirror-percentage – display the percentage*** value of the ADC input countdown – countdown in minutes and seconds from the seconds specified.
Pimoroni Scroll HD LED matrix display	Output	scrollhd	scroll - scroll single or multiple lines of text. static – display static text static-high – a high brightness variant of static

			blank - blank
Pimoroni Automation Phat (ADS1015 ADC)	Input		The three analog input channels of the HAT are read continually at 100mS intervals. The results are used in the mirror methods and are available to track plugins using the I/O plugin manager's get_input method. The keys are: analog-1volts analog2-volts analog3-volts analog1-percentage analog2-percentage analog3-percentage Note: The digital inputs, digital outputs and relay of the HAT are GPIO based and controlled using pp_gpiozerodriver.py

***Percentage – A number between 0 and 100 which is the percentage of 3.3 volts applied to the ADC input.

The file /pipresents/pp_resources/pp_templates/i2c.cfg has details of implemented commands and methods.

Libraries used by I2C I/O plugin.
The Scroll HD and Four Letter PHats use the Pimoroni libraries. Some of the methods are just adaptions of the Pimoroni examples of using these devices. The libraries are already in the Raspberry Pi OS image.

The Automation PHAT and Adafruit DAC libraries are not required as Pi Presents uses its own libraries for these which are in in pp_i2cdevices.py, however it may be necessary to install the smBus module.

15.2.6 Configuring GTK4 Keyboard Keys using pp_kbdriver.py

NOTE: In pipresents-gtk and later keyboard input is implemented by GTK4 instead of Tkinter. The new implementation is compatible with the existing, however there are changes which should be implemented if modifying keys.cfg

- The names of common conditions do not appear to have changed from Tkinter definitions. The use of <> to surround non-printing characters is not required, however legacy keys.cfg files using them will work.

- Keys affected by bind-printing are now limited to the extended ASCII range of characters. Previously they may have included all unicode charaters. The range can be modified in pp_kbddriver.py.

Much of the foundation of Pi Presents is implemented using GTK4. This has a native keyboard/mouse interface which is implemented in Pi Presents by pp_kbddriver.py

and configured by keys.cfg. This interface is the only interface enabled out of the box so it has a file in /pipresents/pp_io_config/keys.cfg

The [DRIVER] section of the .cfg file should have the following additional fields:

Field	content	use
bind-printing	yes/no	If yes then all printing keys automatically bound to symbolic names as described below

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

condition = symbolic name e.g. Return = pp_play

The conditions are defined in

https://gitlab.gnome.org/GNOME/gtk/-/blob/main/gdk/gdkkeysyms.h

In addition to these bindings the extened ASCII printing characters are automatically bound to the symbolic name pp-key-x if 'bind-printing' = yes

e.g the 'a' key produces pp-key-a and ! produces pp-key-exclam

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

The default keys.cfg in /pipresents/pp_io_config has the following bindings:

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

Duplicate Events

The GTK4 Keyboard device driver has a quirk in that it accepts inputs from all devices in /dev/input. As a result a remote control button press that causes an event from pp_inputdevicedriver.py may also generate an event from pp_kbddriver.py. It may be necessary to remove bindings in one or other of the .cfg files or to set bind-printing to no.

The duplicate events can be seen by enabling debugging in Pi Presents with the –d command line option.

15.2.7 Advanced Interfacing with a Keyboard using pp_kbddriver_plus.py

This I/O plugin enhances the standard pp_kbddiver.py I/O plugin by allowing Pi Presents to:

- allow strings, in addition to single characters, to trigger events
- allow lines of text to be provided to track plugins

Details for using the configuration data are in the file:

/pipresents/pp_resources/pp_templates/keys_plus.cfg

The use of pp_kbddriver_plus.py conflicts with pp_kbddriver.py. If using the driver then disable pp_kbddriver.py by one of the following:

- Add a keys.cfg file to the profile with enabled = no
- Disable the driver in /pipresents/pp_io_config/ keys.cfg
- Remove keys.cfg from /pipresents/pp_io_config

The example pp_kbddisplay_1p6 demonstrates the use of the driver to accept lines of text as events and to display the text using a track plugin.

Characters received from the link are matched against characters and strings defined in the configuration file. If a match occurs an input event with a symbolic name, also defined in the configuration file, is generated:

- As each character is received it is matched against the configuration data to generate an input event.
- When an end of line character is detected the characters received after the previous end of line are matched against the configuration data to generate an input event.

Events can be generated from any-character, specific-character, any-line and specific-line

15.2.8 Producing short sounds with pp_aplaydriver.py

The pp_aplaydriver.py I/O plugin allows sounds to be played from Animation commands. Alternatively short sounds can be played by the Beep Show Control Command (Section 8.5) using the same PAPLAY and mpg123 applications

Details for using the plugin's configuration data are in the file:

/pipresents/pp_resources/pp_templates/aplay.cfg

The animation command is of the form:

0 beep1 beep

1. beep1 corresponds to a name in the aplay.cfg file
2. the parameter type field must be beep

Acceptable file types are uncompressed audio files playable by the linux PAPLAY command (.wav and others) or, if mpg123 is installed, .mp3 files. Install mpg123 with:

```
sudo apt install mpg123
```

15.2.9 Producing Haptic Effects with pp_DRV2605driver.py

The DRV2605 Haptic Controller controls ERM or LRA actuators to produce Haptic effects. This driver implements the Vibration Sequence Generation and Audio to Vibe modes of the controller. The effects are played using Animation commands or Show Control Commands (Section 8.6)

Details for using the plugin's configuration data are in the file:

/pipresents/pp_resources/pp_templates/DRV2605.cfg

The animation command is of the form:

```
0 DRV2605 sequence-name  sequence1
0 DRV2605 audio-vibe-name vibe1
```

See the example config file for the complete set of parameter types.

To use the controller I2C must be enabled and the Pimoroni DRV2605 driver must be installed by creating a virtual environment (Section 6.8) and in it:

```
pip install drv2605
```

Pi Presents does not use the Pimoroni driver, just an I2C driver which is installed, with the DRV2605 driver. Instead an extended version of the Pimoroni driver is in /pipresents/DRV2605_lib.py.

Pimoroni sell the DRV2605 Controller, ERM actuators and LRA actuators. I have tested Pi Presents with the ERM actuator but not the LRA actuator.

Code of the original Pimoroni Library is here:
https://github.com/pimoroni/drv2605-python

and the DRV2605 datasheet is here:

https://www.ti.com/lit/ds/symlink/drv2605.pdf

15.2.9.1 Loop Modes

The TI datasheet is unclear about whether to use Open Loop or Closed Loop feedback mode. For this reason the Loop Mode is settable in the .cfg file.

My best reading of the datasheet is:

Use	Loop Mode
ERM Sequence with TS2200 A library	Open Loop
ERM Sequence with other ERM Libraries	Closed Loop
LRA Sequence (1 library which is set internally by Pi Presents)	Closed Loop
Audio to Vibe for ERM or LRA	Closed Loop

15.2.9.2 Using Audio to Vibe with Raspberry Pi

The A/V output of the Raspberry Pi is via a 4 pin jack. The use of the tip and rings is non standard.

The audio output of the Raspberry Pi is tied to 0 Volts via a 1.8K resistor. It is therefore necessary to use a 1uF capacitor as directed in the datasheet. Pi Presents sets AC Coupling so the input to the DRV2605 is biassed to 1.8 volts.

In my experiments, the A/V output when playing suits-shorts.mkv was typically 100 mV p-p. It was necessary to set the Max Audio Input Voltage to 0.025 in the .cfg file in order to feel the beat of the music. Using a cheap USB audio dongle gave similar output levels and a less noisy output.

15.2.10 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:

- displays - A set of displays that the click area can appear on. e.g. HDMI0 DSI0. The click area will actually be displayed if its symbolic name is included in the controls for the show or track.

- 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 display, provided the display name is in the displays field. When an area is clicked the symbolic name will identify an input event. This input event will be offered to all shows/tracks in both displays. You may need click areas with different symbolic names if you want to be display selective.

- points - The definition of a rectangle or circle in the form

  shape x1+y1+w*h

x1, y1 are the coordinates of the top left corner of the click area. w,h are the width and height, all in pixels.

Shape is rectangle or circle. For circle equal width and height will produce a circular button. Non-equal width and height will produce a rectangle with rounded corners.

- fill-colour, outline-colour

  Specifies the look of the rectangle. For details of colour see the background colour field in a show or track. The field cannot be blank, use transparent for a transparent item.

- text, text-font, text-colour

  If text is not blank then the text is written centred in the rectangle. Text-font and text-colour are as for the text in shows or tracks.

- image
-

  If image is not blank an image file to be used in the click area. Paths relative to pp_home (+) and a profile (@) are supported or specify the complete path. The image will be warped to fit the width and height specified in points.

15.2.11 Interfacing with a RFID tag reader using pp_pn532driver.py

RFID tag readers read the data on cards such as credit cards. These cards have a small microchip which is powered by the RFID tag reader which then interrogates the card and can read or write data from/to the card. Every card has a unique Identification number (UID) and this is the only information used by Pi Presents.

There are a number of card readers, there are also different standards of card. I chose a reader based on the PN532 chip which reads cheap tags. The tags may be in different forms - credit card sized cards, key fobs, and probably most useful to museums, paper stickers.

The pp_pn532driver.py I/O plugin uses the pynfc and libnfc libraries to interface with the PN532. It has been tested using the PN532's I2C interface but should work with serial/usb and SPI.

The template file pn532.cfg in /pipresents/pp_resources/pp_templates is an example of using the I/O plugin. In the example two tag codes are defined and bound to the symbolic names of events.

To obtain the tag codes first install nfc-tools as described below, then run the command nfc-poll. You will get a reply something like this:

```
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  04
       UID (NFCID1): 76  35  a2  ac
      SAK (SEL_RES): 08
```

The tag code is derived from the UID, which will probably be an 8 digit or 14 digit hex number, by removing the spaces.

Details for using the configuration data are in the file:

/pipresents/pp_resources/pp_templates/pn532.cfg

When a tag is presented to the reader the tag Id is read and matched with the tags defined in the various sections of the .cfg file. If a match is found an input event is generated with the appropriate symbolic name. An event can also be generated when the tag is removed. Hysteresis is applied by using the threshold parameter because when a card is moved towards the reader multiple events can be produced.

There is an example pp_rfidioplugin_1p6 which use a radiobuttonshow with two tags triggering two tracks.

Installing the RFID Tag Reader

I bought this kit:

XCSOURCE® NXP PN532 NFC RFID Module V3

It works well, except that the detection distance is only a couple of centimetres. I used the I2C interface but it should work with SPI or UART.

I found that most of the instructions on the Web to install the software are out of date. The instructions here work well:

https://blog.stigok.com/2017/10/12/setting-up-a-pn532-nfc-module-on-a-raspberry-pi-using-i2c.html

This installs libnfc and nfc-tools which enables you to use the command line to read tag Id's. You also need the pynfc library. There is a copy of this in the pipresents/pynfc directory which has been modified to correct a bug. As it is inside /pipresents it does not need to be installed. For interest, the original is at https://github.com/ikelos/pynfc

15.3 Writing I/O Plugins

An I/O plugin is a python 3 module which:

- For inputs translates physical inputs to the RPi into events with symbolic names
- For outputs translates animation commands having symbolic names and parameters into physical outputs

I/O plugins should be placed in the directory /pipresents/ pp_io_plugins where they will be available to all profiles.

15.3.1 Configuring and Registering an I/O plugin

I/O plugins will be registered if there is an I/O plugin configuration file in the pp_io_config directory of a profile. Registration is implemented by pp_iopluginmanager.py. The module reads the [driver] section of the configuration file and if enabled = yes imports the module specified in module = .

The I/O configuration file can also be used to configure an I/O plugin module. This allows one module to be used with different devices, e.g. different wireless remote controls. Configuration data can be added to the [driver] section or further sections can be created if desired.

The path to the plugin configuration file is passed to the I/O plugin module so that the module can read the additional parameters.

15.3.2 Class

The class name must be the same as the name of the file

The I/O plugin output method may be called from many different objects hence variables used by the output side should be class variables.

15.3.3 Methods

init

```
def init(self,filename,filepath,widget,pp_dir,pp_home,pp_profile,button_callback=None)
```

init() is called once by the I/O plugin manager when Pi Presents starts. The method should:

- read configuration data from the [driver] section of the I/O configuration file in the profile
- initialise the physical device
- indicate using a state variable that initialisation is successful and the plugin is active.
- if activation is successful return ‘normal’,’a message for the log’
- if activation fails return ‘error’,’a message for the error pop up and log’

Arguments:

filepath – the path to the I/O plugin configuration file.

filename - leaf of filepath, used for logging

widget - the instance of an object onto which to hang the Tkinter after and after_cancel calls

pp_dir – path to /pipresents

pp_home – path to /pp_home

pp_profile – path to current profile

callback- the remote function to be called to generate an event from a physical input. callback has two parameters:

- symbolic name of the event
- a text string identifying the source of the event. The plugin should read it from the [driver] section

start

def start(self):

start() is called once by the I/O plugin manager when Pi Presents starts. The method should:

- If appropriate enable any interrupt action
- If the input is polled scan the inputs once and then call widget.after() to schedule a further scan.

Scanning the physical inputs must not be blocking because Pi Presents use cooperative scheduling.

Reception of an interrupt or detection of a physical input when polling should execute the callback function:

self.button_callback(symbolic_name,source)

get input

get_input(self,key,driver_ref='')

get_input is used by track plugins to obtain values of inputs. The key will be a string which can be used to identify the input required.

Use of get_input has revealed that the key field by itself is not a convenient way to access a particular input because key might have the same name in more than one I/O driver. The optional driver_ref field allows a specific driver to be selected for the get_input operation thus making the key local to that driver.

To use this the I/O plugin configuration file must have a driver-ref field with a driver reference name. If not all I/O plugins are queried with the key.

e.g. If the driver-name = caldav in an I/O configuration file then
get_input((<start value>,<end value>),'caldav') will query only that I/O for the key which in this case is a tuple.

The method returns 2 arguments:

- Found – True/False
- Value – the value

terminate

def terminate(self)

terminate is called by the main program when Pi Presents closes for any reason. It should shut down any interrupts and cancel any polling timers

is active

allows PI Presents to determine the whether the plugin is active

def is_active(self):
    returns whether the i/O plugin is active (True/False)

handle output event

This method uses animation commands of the form:

    name param_type param_value_1 “param value 2”

to generate physical outputs

def handle_output_event(self, name, param_type, param_values, req_time)

    name – the symbolic name specified in the animation command

    param_type – the parameter type specified in the animation command

    param_values – a list of parameter values specified in the animation command. There may be 0 or more parameter values. If a parameter has embedded spaces then it must be contained in “”

    req_time – the time at which the command was required to be executed, Use for logging only, the animation manager ensures the output is made at the correct time.

Every animation command is offered to this and every other active I/O plugin. If the name and param_type matches any of those implemented by the plugin then the command should be executed using the provided parameters and ‘normal’ returned

The parameter values should be checked for correctness (e.g. state is on or off). If illegal parameter values are provided then ‘error’ should be returned.

The function returns two parameters - status,message

    status – ‘normal’/’error’
    message – a text string which will be displayed in the error pop up and used for logging.

15.3.4 Example

Components:

- pp_exampledriver.py in the pp_io_plugin directory
- exampledriver.cfg in /resources/templates directory
- pp_ioplugin_1p4 example profile in the Pi Presents examples github repository

The example plugin has all the elements of a plugin but does not do any physical I/O:

- The input generates an event with the symbolic name tick at an interval defined in the exampledriver.cfg configuration file.

- The output prints the message contained in the animation command to the terminal window.

16 Remote Control using OSC

Version 1.3.1 allows Pi Presents to control instances of Pi Presents on other computers or to control or be controlled by a 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. For Pi Presents a Master is a unit that sends commands to control Slaves so:

- A Master Unit (OSC Client) sends commands to Slaves

- A Slave Unit (OSC Server) receives commands from Master Units and acts on them.

Depending on its configuration Pi Presents may be both a Master and a Slave.

The implementation in this version of the software (1.3.1c) is much improved from the initial experimental version. Improvements include:

- Many slaves per master
- A particular unit can be both a master and a slave
- A slave can be controlled by many masters, replies are returned to the master sending the commands.

- It continues to use 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.

- Commands now use Pi Presents Show Command syntax, converting them internally to the messages required by OSC.

- Configuration is much simplified

The implementation does not support:

- Pattern matching language to specify multiple recipients of a single message
- High resolution time tags
- "Bundles" of messages whose effects must occur simultaneously

16.1 Sending and Receiving Commands via OSC

In Pi Presents terms OSC commands are sent by Master units and received by Slave units. To send an OSC command, place the appropriate command in the Show Control field of a track or show. Other than configuring OSC, receiving messages is automatic.

Other than the send, loopback, and server-info commands, all OSC commands have the same effect as they would have if used on the local machine.

An OSC Show Control command has three or more fields:

- osc – all osc commands have osc or OSC as a first field
- unit – the OSC name of the unit to which the command is to be sent, the name should also appear in the 'OSC Names of slaves' field of the configuration file.
- command – open, openexclusive, close, closeall, event, monitor, animate, exitpipresents, shutdownnow, reboot, send, loopback, server-info
- parameters – zero or more parameter values

  e.g. osc unit1 open myshow

16.2 OSC Fundamentals

OSC commands are converted by Pi Presents into messages. The OSC software does a further conversion into an efficient binary format for transmission. The messages have an OSC address field and zero or more data fields. e.g.

    /pipresents/unit1/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. unit1 must appear in the list of slave units in the configuration file.

- /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 a show-ref as an argument.

Pi Presents Show Command	OSC Command	Use
osc unit1 open myshow	/pipresents/unit1/core/open myshow	Open the referenced show
osc unit1 close myshow	/pipresents/unit1/core/close myshow	Closes the referenced show
osc unit1 openexclusive myshow	/pipresents/unit1/core/ openexclusive myshow	Close all running shows then open the referenced show
osc unit1 closeall	/pipresents/unit1/core/ closeall	Closes all running shows
osc unit1 event pp-stop	/pipresents/unit1/core/event pp-stop	Generates an input event with the referenced symbolic name
osc unit1 exitpipresents	/pipresents/unit1/core/ exitpipresents	Exits Pi Presents
osc unit1 shutdownnow	/pipresents/unit1/core/ shutdownnow	Shut down the RPi
osc unit1 reboot	/pipresents/unit1/core/reboot	Reboot the RPi
osc unit1 animate out1 state on	/pipresents/unit1/core/ animate out1 state on	This provides an output pass-through system. The arguments are any data used by Animation commands without the leading delay parameter.
osc unit1 monitor on	/pipresents/unit1/core/ monitor on	Turn the monitor on.
osc unit1 monitor off	/pipresents/unit1/core/ monitor off	Turn the monitor off
osc anyunit send /myprotocol/anything/you /like arg1 2	/myprotocol/anything/you/ like arg1 2	Send the osc message specified to ‘anyunit’. Can be used to send commands to types of unit other than Pi Presents units. All arguments are sent as strings.
osc unit1 loopback	/pipresents/unit1/system/ loopback	Causes the slave to return a blank loopback-reply message to the master. The reply appears in the terminal window.
osc unit1 server-info	/pipresents/unit1/system/ server-info	Causes the slave to report information about itself in a server-info-reply message. The reply appears in the terminal window.

16.3 Configuring OSC

OSC is enabled and configured by the presence of an osc.cfg file in the directory /pp_io_config in a profile. The file is a non-standard I/O plugin configuration file. pp_web_editor.py has a menu option to create, and edit osc.cfg files in the profile.

This Unit

Field	Example	Use
OSC Name of this unit	my-pi	The /unit part of the address of this unit. Used in matching messages sent to this unit.
IP of This Unit		Normally this field should be blank and Pi Presents will obtain this value from the network. If this fails an IP can be specified here. If your Pi is connected to two networks it may be necessary to choose a preferred network in web.cfg (Section 18.1). The IP used to listen for messages from a master when slave is enabled, and replies when master is enabled.

Fields for a Master Unit

Field	Example	Use
Master Enabled	yes	yes/no, Enable/disable this unit as a Master
Listening Port for replies from slaves.	9001	Port used to listen for replies from the slave units. Slaves reply on the same port as they listen hence the Listening port of the Master must be the same as the Listening Port of all its Slaves
OSC Names of slaves	slave1 slave2	A space separated list of OSC names that is used as a lookup table to determine the IP's of slave units from the OSC command.
IP's of Slaves	myhostname 192.168.1.102	A space separated list of IP's of the slave units having the same order as the OSC Names of slaves. The IP may be the numerical IP of the unit. I have also found the hostname, or hostname.lan also works. hostname.local may work.

Fields for a Slave Unit

Field	Example	Use
Slave Enabled	yes	Enable/disable this unit as a slave
Listening Port for commands from a master	9001	Port that the slave listens to for commands from a master unit. All slaves must listen for commands on the same port which must be the same as the Listening port of the Master Unit.

16.4 oscremote and oscmonitor

pp_oscremote.py and pp_oscmonitor.py are two stand-alone programs which I developed to test the OSC interface of Pi Presents. They run on Windows (requires python to be installed), Linux, or Raspberry Pi OS. They contain code which could be

used as the basis for other applications. Anyone interested in writing an IOS or Android App for controlling Pi Presents?

oscremote sends commands to Pi Presents while oscmonitor monitors and displays OSC messages that Pi Presents has sent. Out of the box oscremote is configured to send commands to the unit /pipresents using port 9001 and oscmonitor to listen to master units on port 9002. When first running these programs you will need to complete other configuration fields.

By running them from different RPi's you can run Pi Presents, a remote, and a monitor. (or just 2 of them). Three example profiles have been set up as demonstrations of OSC that can be run in this way out of the box.

- pp_osc_1p6 is a simple profile to demonstrate some of the key points. Its osc.cfg sets up Pi Presents as a master and a slave. The show mediashow is in the Start show and sends a number of different commands from the show control fields of the track which can be monitored by oscmonitor. Replies to the loopback and server-info commands can be seen in the terminal window running Pi Presents. Use oscremote to close and open the show mediashow.

- pp_multiwindow_1p6 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 using open show and close show commands.

- 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 oscmonitor.py

By changing the configuration you can run oscremote and oscmonitor as a pair. You can also run oscmonitor and oscremote on one machine and Pi Presents on a second.

pp_oscremote.py
Before using oscremote.py it needs to be configured using the options>edit menu to configure the communication. Out of the box it is set up to communicate with the osc unit pipresents with the hostname raspberrypi using port 9001.

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 animation 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. To do this a /pp_home/pp_profiles directory containing the profile will need to be presents and the – o command line option of pp_oscremote.py used to set the path to pp_home (default is /home/pi).

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.

pp_oscmonitor.py
Just displays messages it receives in the Status window. Out of the box it is configured to receive messages on port 9002.

17 Track Plugins

Track plugins are python 3 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 plugin name may be relative, if the plugin name is preceded by:

- @ - the plugin is in the profile in the directory pp_track_plugins
- + - the plugin is in pp_home in the directory pp_track_plugins
- no prefix – the plugin is in /pipresents/pp_track_plugins (legacy location)

Placing track plugins inside /pipresents is now discouraged; ideally put them in the profile using @

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
  https://web.archive.org/web/20200315112731/http://effbot.org/imagingbook/pil-index.htm.
  or

  http://effbot.org/imagingbook/pil-index.htm.

- Tkinter canvas operations are documented at

  https://web.archive.org/web/20200315112731/http://effbot.org/tkinterbook/canvas.htm
  or
  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.

- MPV video - the plugin will be supplied with a file containing a video. A video track must be returned that is playable by VLC. 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 MPV Video Window to window it.

- web - the plugin will be supplied with a file containing html code (whatever chromium browser supports for rendering). A html file should be returned that is playable by the browser. 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.

18 Remote Management

NOTE: pp_manager.py was developed before the Raspberry Pi used VNC to remotely display the RPi's desktop. VNC is a much better solution to the remote management of Pi Presents.

pp_web_editor.py is now the only editor for Pi Presents profiles. The original pp_editor.py was retired many years ago.

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.

Both programs are based on Remi. The library does not implement security strategies, and so it is advised to not to use them across unsafe public networks.

In addition when loading data from external sources, consider protecting the programs, and also Pi Presents from potential javascript injection before displaying the content directly.

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

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

  The presumed workflow of the manager is that you will upload media or liveshow files from a remote computer to the Pi, or import them from a USB stick. Once in the Pi media and livetracks can be deleted or renamed. Profiles

can similarly be uploaded or imported; in addition they can be created or edited on the Pi and then downloaded to the remote computer.

- Web Editor - pp_web_editor.py. The Pi Presents profile editor described in Sect. 4 used in its remote mode.

18.1 Setting up for Remote Use

Before first using the two programs:

- 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]		
media_offset	/media	The offset from the pp_home directory to which media will be imported or uploaded. So /media will store media in home/[user]/pp_home/media
livetracks_offset	/ pp_live_tracks	The offset from the pp_home directory to which livetracks will be imported or uploaded. So /pp_live_tracks will store media in /home/[user]/pp_home/ pp_live_tracks the default live tracks directory.
profiles_offset		The offset from the pp_profiles directory to which profiles will be imported 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. /test_profiles will store profiles in /home[user]/pp_home/pp_profiles/test_profiles
options		When starting Pi Presents from the Manager append the specified options. Sect 6.1 The –p and –o options should not be specified. The Manager automatically adds these.
autostart_path	/my_profile	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 auotstarting Pi Presents..So /my_profile will start the profile in /home/[user]/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/ [user]/ pp_home	The path of the Pi Presents Home Directory. It must end in pp_home.
import_top	/media/pi	This field limits the files for the following actions to those in this directory and below:

		• Source for importing media and livetrack files. • Source for importing profiles As a security measure this option limits the files that can be seen to those below the top. It also determines the starting point for the import File Selection dialog.
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]		These fields are also used by the email system.
unit	My Pi	A name which appears on the Manager's screen and on emails. Useful if you have more than one Pi under management.
force_ip		The ip address of Raspberry Pi that is running the Editor or Manager. For normal use this field can be left blank as the IP address of the Pi is automatically determined. If this field is specified auto detection will not be used.
preferred_interface		If wifi and wired Ethernet are both connected the Pi will have two IP addresses. If you have two interfaces and wish to specify which one should be used populate this field. Values on my Pi are wlan0 and eth0.

18.2.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 the server's 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 List 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 by downloading the log.

Menus

- media>import - 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.
- media>manage - delete or rename media files
- livetracks>import - Copy livetrack files from the selected directory into pp_home into the sub-directory defined by livetracks_offset.
- livetracks>upload - Upload liverack files from the browser into pp_home into the sub-directory defined by livetracks_offset.
- livetracks>manage - delete or rename livetracks files
- profile>import – 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.
- profile>manage – delete or rename a profile.
- editor>run – Run the Web editor. This will start the editor's server on the Pi. Click the link to access the editor which will 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
- options>email – edit the Email Alert options (see Sect. 19.1)
- logs>Download Log – Download the Pi Presents log file
- logs>Download Stats – Download the Pi Presents statistics file
- Pi>Reboot/Pi>Shutdown - reboot or shutdown the RPi

Using Autostart

If the Manager's 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 Raspberry Pi OS's autostart file (Sect. 6.6) should have the command:

```
usr/bin/python3  /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

18.3 Using the Web Editor

When managing Pi Presents remotely the Web Editor is normally started and stopped from the Manager however it can be run independently :

- Use the command python3 pp_web_editor.py -r 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

19 Email Alerts

Pi Presents and the Manager can send alerts by email.

email_with_ip	Sent by the Manager when it starts. The email contains the IP address of the Pi for use by the remote browser.
email_at_start	Sent by Pi Presents when it starts.
email_on_error	Sent when Pi Presents exits when a Fatal or Profile error is detected. The message contains the error text.
email_on_terminate	Sent when Pi Presents exits either by it being terminated from the Pi or pressing Exit on the Manager
log_on_error	(not implemented) Attach the log file to the error email.

19.1 Setting Up Email Alerts

Before first using email alerts:

- The file /pipresents/pp_config/pp_web.cfg must be edited using a text editor. The [network] section fields preferred_interface and force_ip need to be set. For normal use they should be blank.

- The file /pipresents/pp_config/pp_email.cfg must be edited using a text editor. Entries in the section [email-editable] can subsequently be edited by the Manager.

Field	Example	Description
[email]		
server		url of the server to be used to send the emails. The software has been tested only with gmail. Since the username and password is not encrypted it is better not to use your normal email account. Set up a gmail account especially for the purpose. You will need to disable the advanced security feature in the gmail account.
port		port of the server used to send the emails
username		username and password for the server used to send the emails.
password		
[email_editable]		
email_allowed		A global enable. yes to enable all types of email alert, otherwise no
to		A list of email addresses to which the emails will be sent, one address per. line.
sender		An email address to go in the From: field of the email. If the field is missing or is blank then username is used.
email_with_ip email_at_start email_on_error email_on_terminate		set to yes to enable the alert, otherwise no.
log_on_error		Not implemented. Set to yes to enable the log as an attachment, otherwise no.

19.2 Using Email Alerts

having set up the pp_email.cfg file alerts will be sent automatically. The editable part of pp_email.cfg can be modified using the Manager.

20 Hardware and Operating System Requirements

RPi OS Bookworm has two alternative Display Environments, Wayland (default) and X11 (legacy)

Full support for presents-gtk is available only for RPi OS Bookworm using the Wayland Display Environment. This is available only on RPi Model 4 and RPi Model 5.

However pipresents-gtk using Bookworm is likely to work reasonably well on any model of RPi that has adequate RAM providing you switch to the X11 Display Environment using sudo raspi-config.

My development effort is concentrated on Wayland. Features that do not work on X11, such as the touchscreen, are not given priority.

There are other differences between models which will affect your choice:

- Model 4 or 5 is required if you want 2 displays.
- Pi5 supports 2 DSI official touchscreens
- Pi5 has no A/V audio and the A/V video requires soldering on a socket.
- Pi 5 has a different internal implementation of GPIO the pp_gpiozerodriver.py I/O plugin must be used instead of pp_gpiodriver.py. pp_gpiozerodriver.py works on all models of Pi

21 Updating Pi Presents

21.1 Updating Profiles

As Pi Presents develops new fields are 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 warn 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 and leave a backup in pp_profiles.bak. There may 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 directory then you can use the 'Tools>Update All' menu option of the editor to update them.

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

21.2 Updating Pi Presents

For safety take a copy of the pipresents directory 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.

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.

22 Debugging, Statistics, Bug Reports and Feature Requests

22.1 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 is CSV and suitable for reading into a spreadsheet. The separator is ; it can be changed in pp_statsrecorder.py.

Statistics logging is enabled by the –d command line option as described in Section 22.2.

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 and 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, profile. Track details are track type, track reference, track title, location, profile.

Example:
"Date";"Time";"Show Type";"Show Ref";"Show Title";"Command";"Track Type";"Track Ref";"Track Title";"Location";profile
"2016-02-01";"15:02:43";"";"";"";"start";"";"";"";"/pp_interactive_1p4";"/pp_interactive_1p4"

"2016-02-01";"15:02:53";"mediashow";"mediashow";"Mediashow";"play child";"menu";"mymenu";"Menu";"";;"/pp_interactive_1p4"

"2016-02-01";"15:02:58";"menu";"mymenu";"Menu";"play";"image";"";"A Stunning River Scene";"+/media/river.jpg";"/pp_interactive_1p4"

The writing of statistics to a device is separated out from the remainder of Pi Presents and is in the file pp_statsrecorder.py. This module can be re-written if another statistics recorder such as InfluxDB is required.

22.2 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 allow 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 log suitable for debugging profiles
10000	16	A log suitable for debugging Pi Presents.
100000	32	A log 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 22.1
100000000	256	A log that is suitable debugging Time of Day scheduling.

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-gtk/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.

23 Gotchas and Known Problems

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/python3 /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 Raspberry Pi OS Wheezy and its content is different for Raspberry Pi OS Bullseye.

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. If Pi Presents freezes when fullscreen CTRL-ALT-DELETE might make the task bar visible.

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

```
killall mpv
```

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.

My video/audio track does not play with Pi Presents
Try playing it using MPV from the command line or from the RPi OS menu if the option is present.

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!

Pi Presents produces lots of GTK and webkit warnings in the terminal window

GTK warnings can be removed by prefacing the python pipresents.py command as follows:

```
GTK_A11Y=none python pipresents.py -p .....
```

However this does not remove webkit warnings.

All warnings can be removed by following the command with text which redirects errors to the null device. This does not inhibit normal Pi Presents output but will inhibit python error reports so use with care,

```
python pipresents.py -p pp_mediashow_1p6 -d 2>/dev/null
```