PVRTune

User Manual

Filename	:	PVRTune.User Manual
Version	:	PowerVR SDK REL_3.5@3523383a External Issue
Issue Date	:	17 Apr 2015
Author	:	Imagination Technologies Limited

Contents

List of Figures

List of Tables

1. Introduction

1.1. Document Overview

The purpose of this document is to serve as a complete user manual for PVRTune and PVRPerfServer. It includes installation instructions, functionality explanations and useful tips on how to make the most out of the applications.

1.2. Software Overview

PVRTune is a performance analysis tool for the PowerVR graphics cores. It uses driver level counters and hardware registers to provide real-time data on the performance of an application running on a PowerVR graphics core. PVRTune receives performance data from either PVRPerfServer or PVRHub running on the target device.

Figure 1 illustrates the conceptual structure of PVRTune in relationship to the interactions taking place at the client software, device software and device hardware levels. The following concepts are relevant to the figure:

- **PVRPerfServer**: This is an application which runs on the target device and reads counters and registers from the driver (and also, optionally, from any application which uses the PVRScope library) and either saves the data to a PVRTUNE file or transmits the data to PVRTune over a network.
- **PVRHub**: This is an application which runs on Android or Linux and allows an instance of PVRPerfServer or PVRTrace to run on the target device. For the remainder of this document, all PVRPerfServer functionality can be accomplished using PVRHub, unless otherwise stated.
- **PVRScope**: This is a performance analysis library that retrieves graphics core counter data and works alongside PVRTune to augment it with custom markers and counters.
- **PVRTune**: This is the client-side element of the combined package. It provides a graphical representation of the real-time information being sent by PVRPerfServer, as well as providing a means to save and load PVRTUNE files.

Note: On Android, PVRHub is an application that has a user interface. On Linux, it is a set of folders and scripts. For more information regarding PVRHub, see the "PVRHub User Manual".

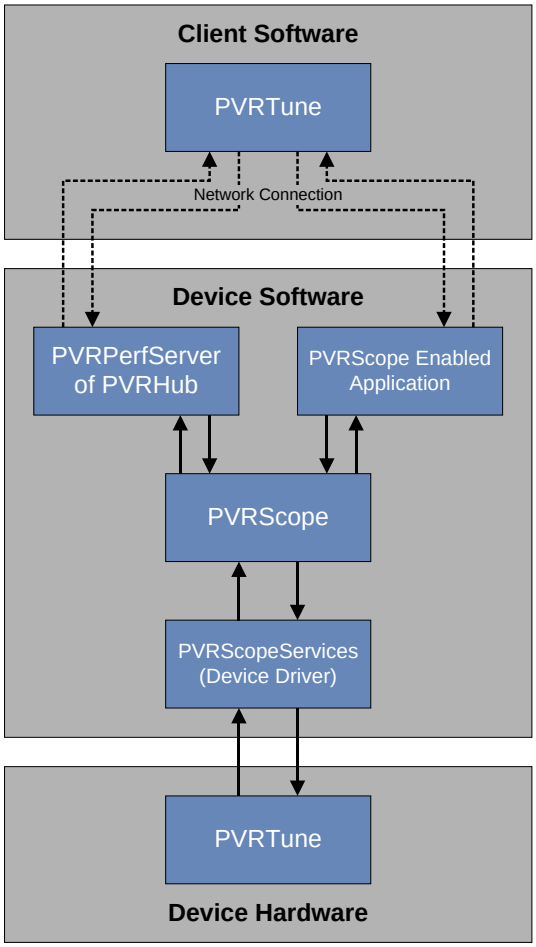

Figure 1. Structure of PVRTune

2. PVRPerfServer

2.1. Overview

PVRPerfServer is a console application for Android, Linux, Neutrino and Windows. Its purpose is to read counters and registers from the PowerVR hardware on a device and either save the data to a PVRTUNE file or transmit that data to PVRTune over a network.

2.2. Requirements

In order for PVRPerfServer to function correctly, the driver must have performance profiling enabled. In many cases, as the overhead is so small, this functionality is available by default. This can be checked by confirming the existence of `PVRScopeServices.dll`, `libPVRScopeServices.so` or an equivalent. If PVRPerfServer fails to initialise, it is likely that performance profiling support has been removed from the device drivers.

2.3. Installation

2.3.1. Package Installation

PVRTrace can be installed from the PowerVR SDK installer:

1. Navigate to the PowerVR Insider SDK webpage (powervrinsider.com) and download the SDK.
2. Launch the installer and follow the on-screen instructions.
3. Once the SDK has been successfully installed, PVRPerfServer can be found within the PVRTune folder in the install directory:

   `<InstallDir>\PVRTuneDeveloper\PVRPerfServer\<PLATFORM>\`

2.3.2. Android

PVRPerfServer functionality is obtained by installing PVRHub for Android. To install this application, run `adb install PVRHub.apk` on the local machine.

2.3.3. Linux

With the package successfully installed, copy the `PVRHub` folder to the target device and follow the instructions found in the “PVRHub User Manual”.

2.3.4. Neutrino and Windows

With the package successfully installed, copy the `PVRPerfServer` binary to the target device.

2.4. Driver Compatibility

2.4.1. Compatible DDK Ranges

On PowerVR hardware devices, PVRPerfServer is limited to supporting the following driver versions:

- 1.3.13.1589 onwards.
- 1.4.14.593 onwards.

2.5. Usage

2.5.1. Android

With the Android APK installed, open the application menu and run `PVRHub`.

2.5.2. Linux

From a command-line interface run `pvr_profile <binary>`, where `<binary>` is the desired application for profiling.

2.5.3. Neutrino and Windows

From a command-line interface run the `PVRPerfServer` executable.

Note: Only one client at a time may be connected to an instance of PVRPerfServer.

2.5.4. Command-Line Options

PVRPerfServer supports several command-line options as identified next.

Table 1. Command-line options

Option	Effect
`-h`	Show help text.
`/?`	Show help text.
`--disable-hwperf`	Disables the use of PVRScope's hardware performance functionality.
`--group=N`	On start-up, switch the hardware to the specified counter group number.
`--port=N`	Network port to use. Default is `6520`.
`--sendto="myfile"`	Instead of using the network, record data directly to the specified file.
`--t=N`	Time, in milliseconds, between counter updates. The default value is `2`. This value can be increased to reduce the CPU usage of PVRPerfServer.
`--c=N`	Time, in milliseconds, between CPU Load updates. The default value is `200`.
`--pid`	Gather data for CPU usage, memory usage and PID executable name of relevant programs.
`--pid=N,M`	Gather data for CPU usage, memory usage and PID executable name of relevant programs. In addition, track the specified PIDs.
`--periodic=1/0`	Enables/disables periodic timing tasks (for use when recording to a file).
`--graphics=1/0`	Enables/disables graphics timing tasks (for use when recording to a file).
`--qat=N`	Commands PVRPerfServer to auto-quit after a specified number of seconds

2.5.5. Run-Time Options

PVRPerfServer supports several key-presses at run-time. This is summarised in Table 2.

Table 2. Run-time hotkeys

Key	Effect
H	Show help text.
M	Send a mark. Useful for placing simple markers into the data stream, annotated with a number that increments for each mark.
Q	Quit PVRPerfServer.

3. PVRTune GUI

3.1. The Basics

3.1.1. Overview

The PVRTune GUI captures and presents the real-time information sent by PVRPerfServer and also provides a means of saving and loading PVRTUNE files for analysis.

3.1.2. Installation

PVRTune GUI can be installed from the PowerVR SDK installer:

1. Navigate to the PowerVR Insider SDK webpage (powervrinsider.com) and download the SDK.
2. Launch the installer and follow the on-screen instructions.
3. Once the SDK has been successfully installed, PVRTune GUI will be available in:

```
<InstallDir>\PVRTuneDeveloper\PVRTune\<PLATFORM>\
```

3.1.3. User Interface Layout

The default analysis interface is displayed after a connection has been established or after a PVRTUNE file has been loaded (see Section 3.3 for more information on how to connect). The PVRTune GUI displays a large amount of information (Figure 2), split into the following sections:

- **Menu bar** (Figure 2a): The section enables access to several options related to file, edit, view, connection and help.
- **Timeline area** (Figure 2b): This section displays counter data as a function of time.
- **Counter Table window** (Figure 2c): This section lists all the hardware and software counters that are in use.
- **Counter Properties window** (Figure 2d): This section is used to view counter-specific information when a counter is chosen from the `Counter Table`.
- **Find window** (Figure 2e): This section allows the user to search for a variety of items, such as counters and markers.
- **Renderstate Override window** (Figure 2f): This section allows the user to interact with the renderstate of the target device.
- **Monitor window** (Figure 2g): This section displays various loads recorded by the utility.
- **Remote Editor window** (Figure 2h): This section details data from PVRScope.
- **PID window** (Figure 2i): This section displays the connections being used in the recording.
- **Status bar** (Figure 2j): This section provides general information pertaining to the running of PVRTune.

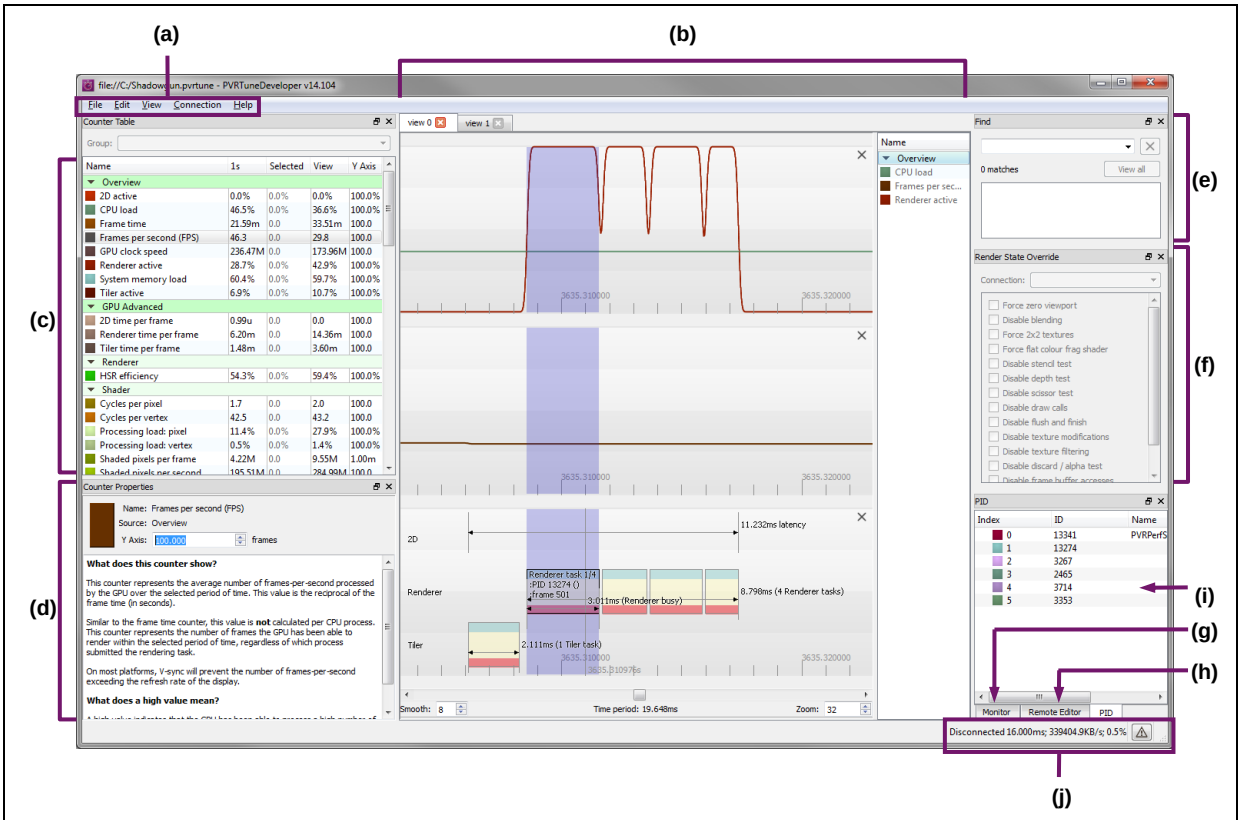

Figure 2. General layout of PVRTune GUI

3.1.4. Getting Started with Analysing an Application

There are several benefits to using PVRTune to analyse an application, and the achievement of effective analysis is largely driven by the ability to understand how to identify application bottlenecks (see Section 4). By analysing an application, it is possible to obtain improved frame rate to increase user enjoyment and application responsiveness. Furthermore, using PVRTune allows reduced render time to be obtained without increased frame rate in order to gain more idle time, thereby saving power. Analysing using PVRTune can also help increase visual quality without sacrificing frame rate.

There are two main approaches to analysing an application in PVRTune, namely through connected analysis (which displays real-time data) and offline analysis (from a saved PVRTUNE file). These main steps involved in these two approaches are summarised next.

Connected Analysis

Connected analysis is the assessment of real-time data from a target device connected using PVRPerfServer. The core steps to performing a connected analysis are as follows:

1. Ensure that the target device is using the PowerVR device drivers.
2. Boot the target device.
3. Install PVRPerfServer on the target device.
4. If necessary, initialise the PowerVR device drivers.
5. Run PVRPerfServer. If successful PVRPerfServer outputs the server name, IP address and port number.
6. Run PVRTune GUI on the host machine. It is important that both PVRTune and PVRPerfServer have matching versions.
7. If PVRPerfServer is on the same subnet as PVRTune, the target device will be broadcast in PVRTune GUI (see Section 3.3.2). If this is not the case, then manually enter the IP address of the server (see Section 3.3.1).

8. Once successfully connected, identify the bottleneck in the application by analysing the output of PVRTune (for more information, see Section 4).
9. Attempt to resolve the bottleneck. For more information on optimization techniques, see the document entitled “PowerVR Performance Recommendations”.
10. Repeat the necessary steps with the newly optimized application until performance is at the required level or no further bottlenecks can be identified.

Offline Analysis

It is possible to analyse an application without being directly connected to PVRPerfServer. This requires a PVRTUNE file of the target application to have been created prior to analysis. This approach is particularly useful when large amounts of data are being lost due to network load or high CPU usage on the client machine.

A PVRTUNE file can be created either through the use of the `--sendto=` command-line parameter for PVRPerfServer, or by saving the file of an existing tune using PVRTune GUI (see Section “Save a File”).

Note: PVRPerfServer only saves counter data for a single counter group when using the `--sendto=` parameter. By default the group is set to zero but this can be changed using the `--group=` command-line parameter.

With a PVRTUNE file created and copied across to an accessible location, perform the following steps:

1. Identify the version of PVRTune and PVRPerfServer used to create the file.
2. Run the version of PVRTune that matches the identified PVRPerfServer version.
3. Open the file in PVRTune GUI (see Section 3.3.5). PVRTune GUI will then display the tuning data.
4. Identify the bottleneck.
5. Attempt to resolve the bottleneck. For more information on optimization techniques, see the document entitled “PowerVR Performance Recommendations”.
6. Repeat the necessary steps with the newly optimized application until performance is at the required level or no further bottlenecks can be identified.

3.2. Menu Bar

3.2.1. File Menu

Figure 3 illustrates the `File` menu which provides options for opening and saving files, managing tabs and exiting PVRTune GUI.

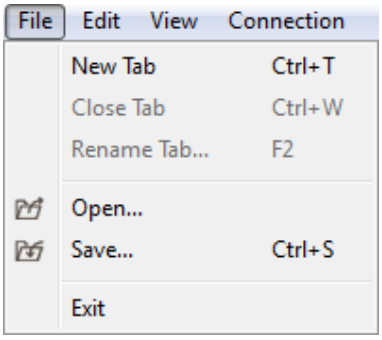

Figure 3. File menu

Open a New Tab

To open a new tab in the `Timeline area`, click `File -> New Tab` (Figure 3). This is a useful feature that allows multiple graph views to be opened at the same time.

Close a Currently Selected Tab

Closing the currently selected tab in the `Timeline area` can be achieved by clicking `File -> Close Tab` (Figure 3).

Rename a Tab

Tabs present in the `Timeline area` can be renamed on-demand. To rename a tab, select it and click `File -> Rename` Tab (Figure 3).

Open a File

To open a PVRTUNE file, click `File -> Open` (Figure 3). This will open a dialog box for browsing to the required file.

Save a File

To save a PVRTUNE file, click `File -> Save` (Figure 3). This will open a dialog box in which the name of the file can be input and the file saved.

Exit PVRTune

To close PVRTune GUI, click `File -> Exit` (Figure 3).

3.2.2. Edit Menu

Figure 4 illustrates the `Edit` menu which provides options for editing user preferences.

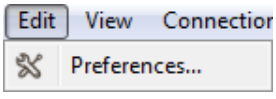

Figure 4. Edit menu

Preferences

User preferences are set by clicking `Edit -> Preferences…` in the `Menu bar`. This opens the `Preferences` dialog box (Figure 5) which displays various options to enable the customization of PVRTune GUI. The `Preferences` dialog box contains several options:

- **Background**: This option is available under the `Graph` section of the `Preferences` dialog box and allows the background colour of graph in the `Timeline area` to be changed from the default setting to either light or dark, in order to make it easier to depict lines.

- **Draw Marks across all timelines**: This option is turned off by default but can be toggled if necessary to show up marks in the `Timeline area`.
- **Hide empty timelines**: Toggle this option to show or hide empty timelines in the `Timeline area`.
- **Zoom control: invert mouse wheel**: This option is available under the `Graph` section of the `Preferences` dialog box. The mouse wheel can be used to zoom in and out of a graph in the `Timeline area`. With the option unchecked, i.e., set to default, scrolling down zooms into the area while scrolling up zooms out of it. This default setting can be inverted by ticking the `Zoom control: invert mouse wheel` checkbox.
- **Integrate connection form**: This option is available under the `User Interface` section of the `Preferences` dialog box. Unchecking this option enables the user to make the connection form (see Section 3.3) a pop up window.
- **Hide unnecessary tabs**: This option is available under the `User Interface` section of the `Preferences` dialog box and, if unchecked, an individual tab is assigned for the connection form in the `Timeline area`. This facilitates quick access to the connection form after a connection is made or with a file loaded.

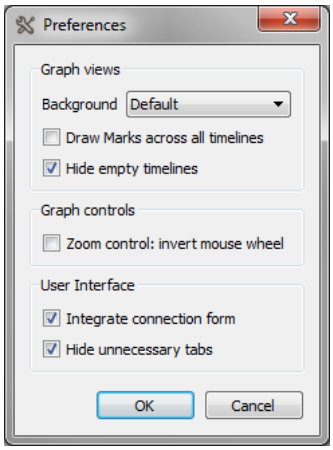

Figure 5. Preferences dialog box

3.2.3. View Menu

Figure 6 illustrates the `View` menu. The menu provides options for finding information as well as showing or hiding the various windows present in the PVRTune GUI, thereby allowing for workspace customization.

Note: The workspace can also be customized by dragging and dropping the individual dockable sections to either the right or left hand sides of the interface or as standalone windows.

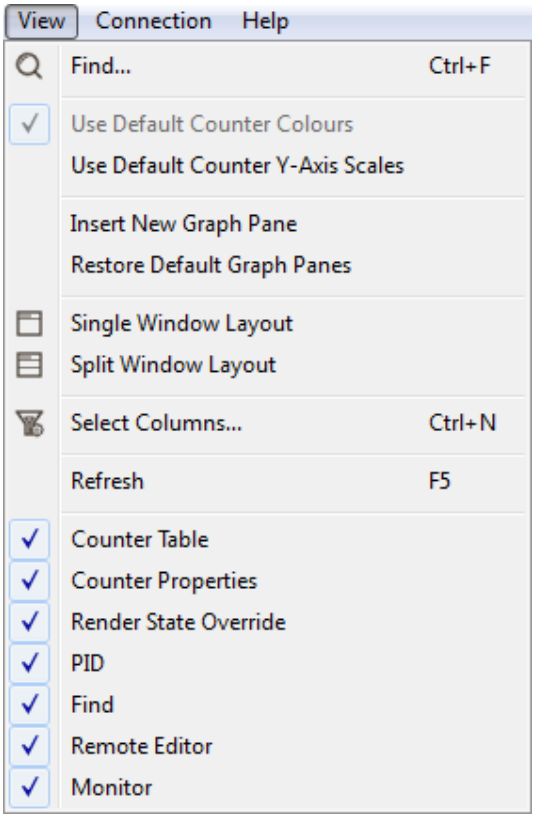

Figure 6. View menu

Find

To initiate a search, click `View -> Find...` (Figure 6). This will open the `Find` window or jump to it for carrying subsequent steps in searching for information (see Section 3.10).

Use Default Counter Y-Axis Scales

Selecting the `Use Default Counter Y-Axis Scales` option (Figure 6) informs PVRTune to use its default Y-axis values for counters. The information appears in the `Y Axis` field in the `Counter Properties` window.

Insert New Graph Pane

This option (Figure 6) allows adding a new graph within the selected view in the `Timeline area`.

Restore Default Graph Panes

This option (Figure 6) causes only the timing data to display in the `Timeline area`, hiding additional graph panes. The timing data then occupies the full space within the `Timeline area`.

Toggle Single Window Layout

To switch to a single window layout for the graph displayed in the `Timeline area` (see Section 3.5.1), toggle the option `View -> Single window Layout` (Figure 6).

Toggle Split Window Layout

To switch to a split window layout for the graph displayed in the `Timeline area`, toggle `View -> Split Window Layout` (Figure 6). This organises two graph views within the same tab.

Customize Counter Table Columns

To customize the display of columns in the `Counter Table` window (see Section 3.6.1), click `View -> Select Columns...` (Figure 6). This will open a dialog box to allow the selection of the required columns (see Section 3.6.1).

Refresh PVRTune GUI

To refresh PVRTune GUI, click `View -> Refresh` (Figure 6).

Show or Hide Counter Table Window

To show or hide the `Counter Table` window, toggle `View -> Counter Table` (Figure 6).

Show or Hide Counter Properties Window

To show or hide the `Counter Properties` window (see Section 3.6.6), toggle `View -> Counter Properties` (Figure 6).

Show or Hide Renderstate Override Window

To show or hide the `Renderstate Override` window (see Section 3.9), toggle `View -> Renderstate Override` (Figure 6).

Show or Hide PID Window

To show or hide the `PID` window (see Section 3.7), toggle `View -> PID` (Figure 6).

Show or Hide Find Window

To show or hide the `Find` window (see Section 3.10), toggle `View -> Search` (Figure 6).

How or Hide Remote Editor Window

To show or hide the `Remote Editor` window (see Section 3.8), toggle `View -> Remote Editor` (Figure 6).

Show or Hide Monitor Window

To show or hide the `Monitor` window (see Section 3.4), toggle `View -> Monitor` (Figure 6).

3.2.4. Connection Menu

Figure 7 illustrates the `Connection` menu which is used for connection management purposes. A number of options are available from this menu and these are covered in further detail in Section 3.3.

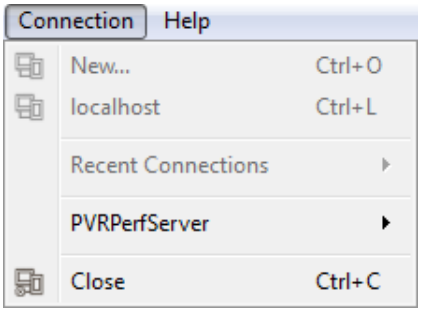

Figure 7. Connection menu

3.2.5. Help Menu

The `Help` menu is opened by selecting the appropriate option from the `Menu bar` (Figure 8). It provides options for accessing PVRTune help assets, sending feedback, viewing general PVRTune release information and checking for software updates.

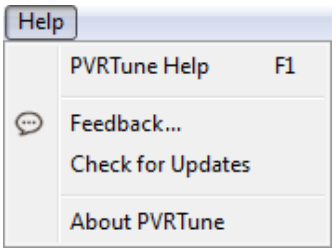

Figure 8. Help menu

View PVRTune User Manual

To view the "PVRTune User Manual", click `Help -> PVRTune Help` (Figure 8).

Submit Feedback

To provide feedback, click `Help -> Feedback…` This will open a dialog box where instructions are displayed on how to post feedback and request for support (Figure 8).

Check for Updates

As of SDK release 3.0, PVRTune is able to auto-update. However, to force-check for software updates, click `Help -> Check for Updates` (Figure 8).

About PVRTune

To view basic information about PVRTune release information such as versioning and contact details, click `Help -> About…` (Figure 8).

3.3. Connection Management

Establishing a connection is a fundamental step prior to being able to visualize hardware performance data in PVRTune GUI. The connection management capability of PVRTune provides a range of options to facilitate the task of connecting. Upon launching PVRTune, the connection form is displayed as shown in Figure 9. Notice that the various windows and options used for analysis purposes are disabled at that point, until a valid connection is made or a PVRTUNE file is loaded.

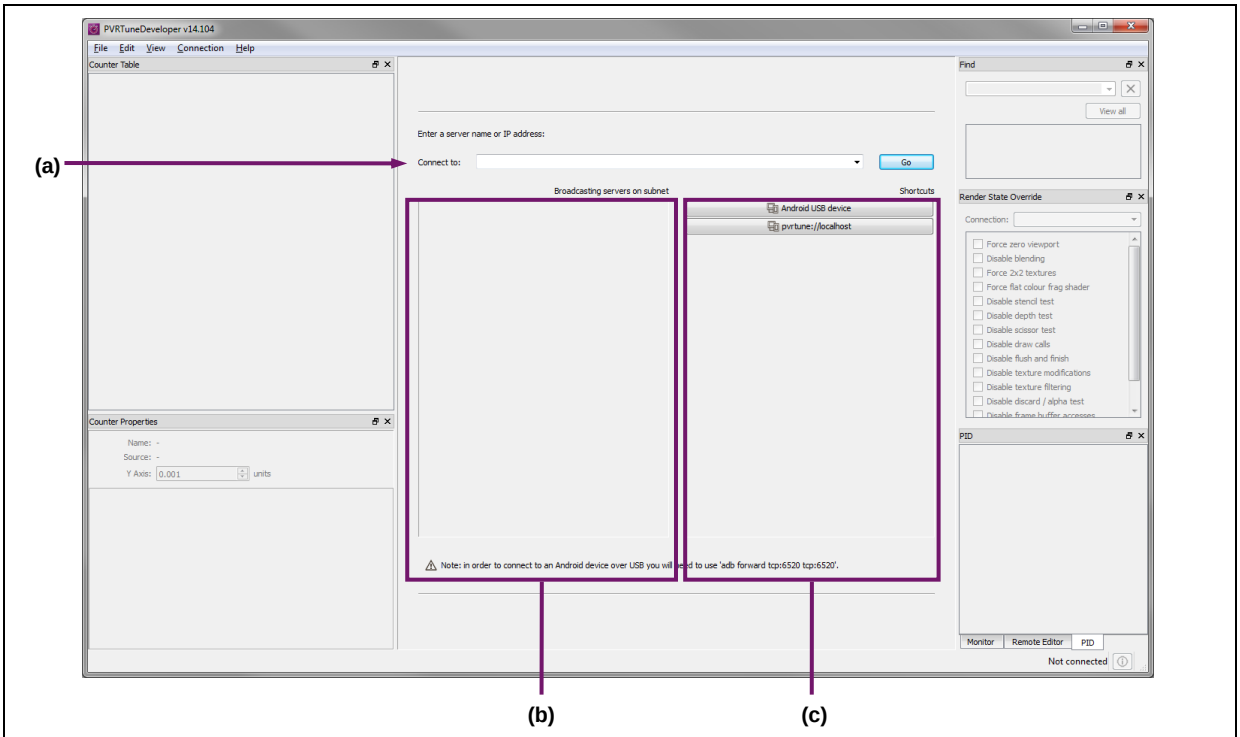

Figure 9. Connection interface displayed after launching PVRTune GUI

Note: The connection form can be customized as a pop window and can also be assigned an individual tab in the `Timeline area` *(see Section "Preferences").*

3.3.1. Connect to a Target Device

To connect to a target device, perform the following steps:

1. Enter an IP address or IP resolvable name of the target device into the `Connect to:` box (Figure 9a).
2. Click the `Go` button to start the connection. If an instance of PVRPerfServer is found, PVRTune will connect to the target device.

3.3.2. Connect Using a Broadcasting Server

When launched, and periodically after launch, PVRPerfServer broadcasts its existence to the subnet to which it is connected. PVRTune GUI lists all the instances of PVRPerfServer broadcasting on the user's subnet in the area of the connection form identified in Figure 9b. Connection can be established by selecting the desired target IP address in the list.

3.3.3. Connect via Localhost

Connecting via localhost implies the establishment of a connection between PVRTune and an instance of PVRPerfServer running on the same device that PVRTune is running on. The option for connecting via localhost will be displayed in the connection form if it can be reached.

Note: PVRTune uses TCP port 6520 to send data to and from Android devices. In order to setup a localhost connection, `adb forward` *must be used to forward TCP data across the USB connection. To run* `adb forward`*, enter the following command in a command prompt:*

```
adb forward tcp:6520 tcp:6520
```

3.3.4. Connect Using a Recent Connection

Recently used connections and PVRTUNE file paths are listed in the area of the connection form identified in Figure 9c. This area also lists recently accessed PVRTUNE files, if any. To connect using a recently used connection, select the desired connection from the list.

3.3.5. Connect Using a Saved File

A saved PVRTUNE file can be loaded by clicking `File -> Open` and browsing to the required file (see Section “Open a File”).

3.3.6. Alternative Ways of Connecting

It is possible, using the `Connection` menu (see Figure 7), to control some aspects of connection management. These are:

- **Establish a new connection**: Select the `New` option in the `Connection` menu to open a dialog box for inputting the IP address or IP resolvable name. This is essentially the same procedure discussed in Section 3.3.1.
- **Connect via localhost**: Select the `localhost` option in the `Connection` menu to connect via localhost. This is essentially the same procedure discussed in Section 3.3.3.
- **Connect using a recent connection**: Select the `Recent Connections` option in the `Connection` menu to view a list of previously accessed connections and choose the desired connection from the list. This is essentially the same procedure discussed in Section 3.3.4.

3.3.7. View Connection Status

Data related to the connection status of PVRTune is displayed in the `Status bar` area of PVRTune GUI (see Figure 2j). Table 3 lists the details of the data that is displayed, from left to right, in the `Status bar`. An example of connection status data is:

```
Receiving 11.2640s (56607, 4922.58ps 0.000sp); 337.9KB/s; 0.7%; 0
```

Table 3. Connection status information in the Status bar

Data	Description
`Status`	This indicates the current status of PVRTune. The options available are: `Connecting`, `Receiving` and `Disconnected`.
`Time`	This indicates for how long PVRTune has been receiving data or was connected (if the current status is disconnected).
`Count`	The number of times PVRTune has performed an action related to its current status (e.g., if PVRTune is receiving data, then the count indicates the number of times it has received data from PVRPerfServer).
`Count per second`	The rate at which the count value moves per second.
`Gap`	The average time period between actions as measured by the count value.
`Receive rate`	The rate at which PVRTune is receiving data.

Data	Description
DC%	The percentage progress towards automatic disconnect due to memory usage (after PVRTune has recorded 1GB of data, it disconnects from PVRPerfServer).
Group	The currently active counter group.

3.3.8. View and Modify PVRPerfServer Settings

Figure 10 illustrates the PVRPerfServer Details dialog box, which is accessed by selecting Connection -> PVRPerfServer -> Details from the Menu bar. The dialog box can be used to view the current PVRPerfServer information and modify certain settings. In other words, when PVRTune is connected with PVRPerfServer, the dialog box allows PVRTune to remotely control the counter groups that are currently active on PVRPerfServer. This in turn then activates or deactivates certain counters on the PowerVR hardware.

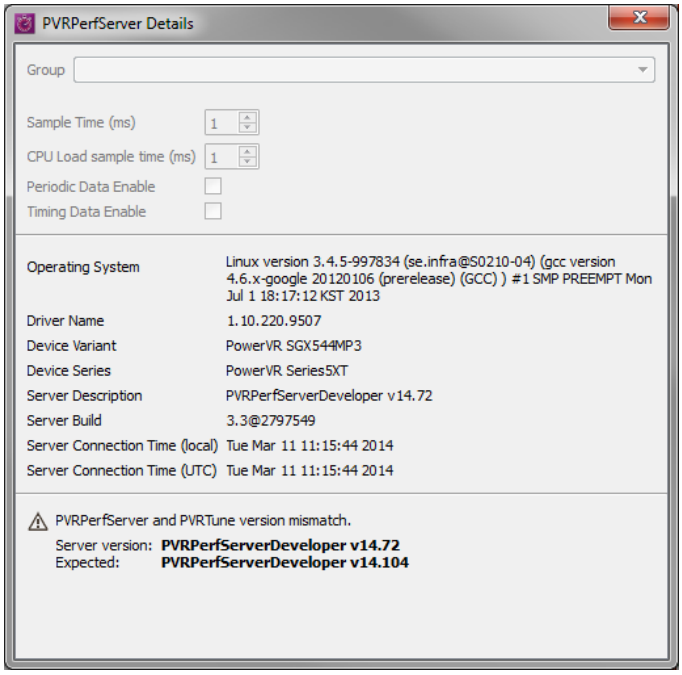

Figure 10. PVRPerfServer Details dialog box

Note: Any warning encountered in the running of PVRPerfServer and PVRTune is also captured in the dialog box.

Table 4 provides a description of the fields present in the PVRPerfServer Details dialog box.

Table 4. Information captured in the PVRPerfServer Details dialog box

Field	Description
Group	Active counter group on PVRPerfServer. Different counter groups can be selected, as appropriate, to change their settings.
Sample Time (ms)	Time, in milliseconds, between the hardware samples taken by PVRPerfServer.
CPU Load sample time (ms)	Time, in milliseconds, between the CPU Load samples taken by PVRPerfServer.

Field	Description
`Periodic Data Enable`	When this option is ticked, PVRTune receives counter updates much quicker than otherwise. This option is enabled by default.
`Timing Data Enable`	When this option is ticked, PVRTune receives Tiler or Renderer timing data. This option is enabled by default.
`Operating System`	Name and version details of the operating system of the target device.
`Driver name`	Version of the PowerVR driver being used.
`Device Variant`	Variant-specific details of the device.
`Device Series`	Series-specific details of the device.
`Server Description`	PVRPerfServer name and version number.
`Server Build`	Release and build version of PVRPerfServer.
`Server Connection Time (local)`	The local time at which PVRPerfServer became connected.
`Server Connection Time (UTC)`	The Coordinated Universal Time (UTC) at which PVRPerfServer became connected.

Note: Clicking the `Connection -> PVRPerfServer -> Send Quit Message` *option sends a message to PVRPerfServer requesting the process to exit.*

3.3.9. Close a Connection

A connection can be closed on demand when it is no longer required. To close a connection, click `Connection -> Close`. If real-time analysis is being performed, then clicking the `Close` option once will first stop connection to PVRPerfServer, allowing the data to be saved as a PVRTUNE file or for continuing the analysis as-is. Clicking the `Close` option a second time will exit the analysis and return the user to the connection form. On the other hand, in the case of offline analysis from a previously saved PVRTUNE file, clicking the `Close` option simply exits the analysis and returns the user to the connection form.

3.4. Monitor Window: Performance at a Glance

PVRTune GUI comes with a monitoring functionality which allows the user to view a high level overview of graphics core workloads, thereby giving insight on how to isolate application bottlenecks. The Monitor window displays and categorises the various loads being tracked by PVRTune, as shown in Figure 11. The dropdown box for the time period (Figure 11a) facilitates choosing over how long the loads should be averaged. In the example, this time period is set to the default `0.5s` value.

Note: Further information on how to analyse bottlenecks can be found in Section 4.

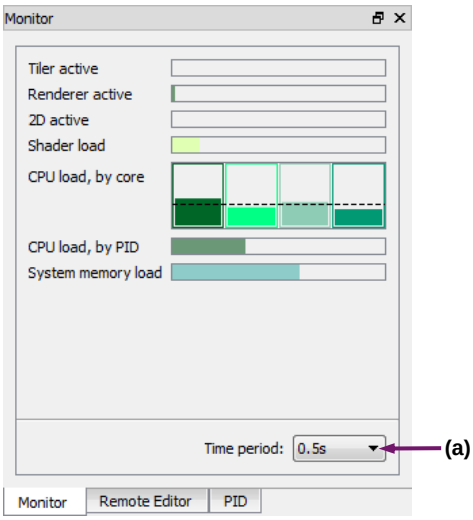

Figure 11. Monitor window

3.5. Working with the Timeline

The captured data is plotted in the `Timeline area` of PVRTune GUI and displayed as a graphical representation in order to facilitate analysis. Figure 12 illustrates an example of static data displayed after a saved PVRTUNE file is opened, using the default graph view. The information is split into three distinct graphs, which capture render timing data and counter data.

Note: If the analysis were based on real-time profiling, which is dynamic in nature, the displayed timing data would appear to be continuously changing with time.

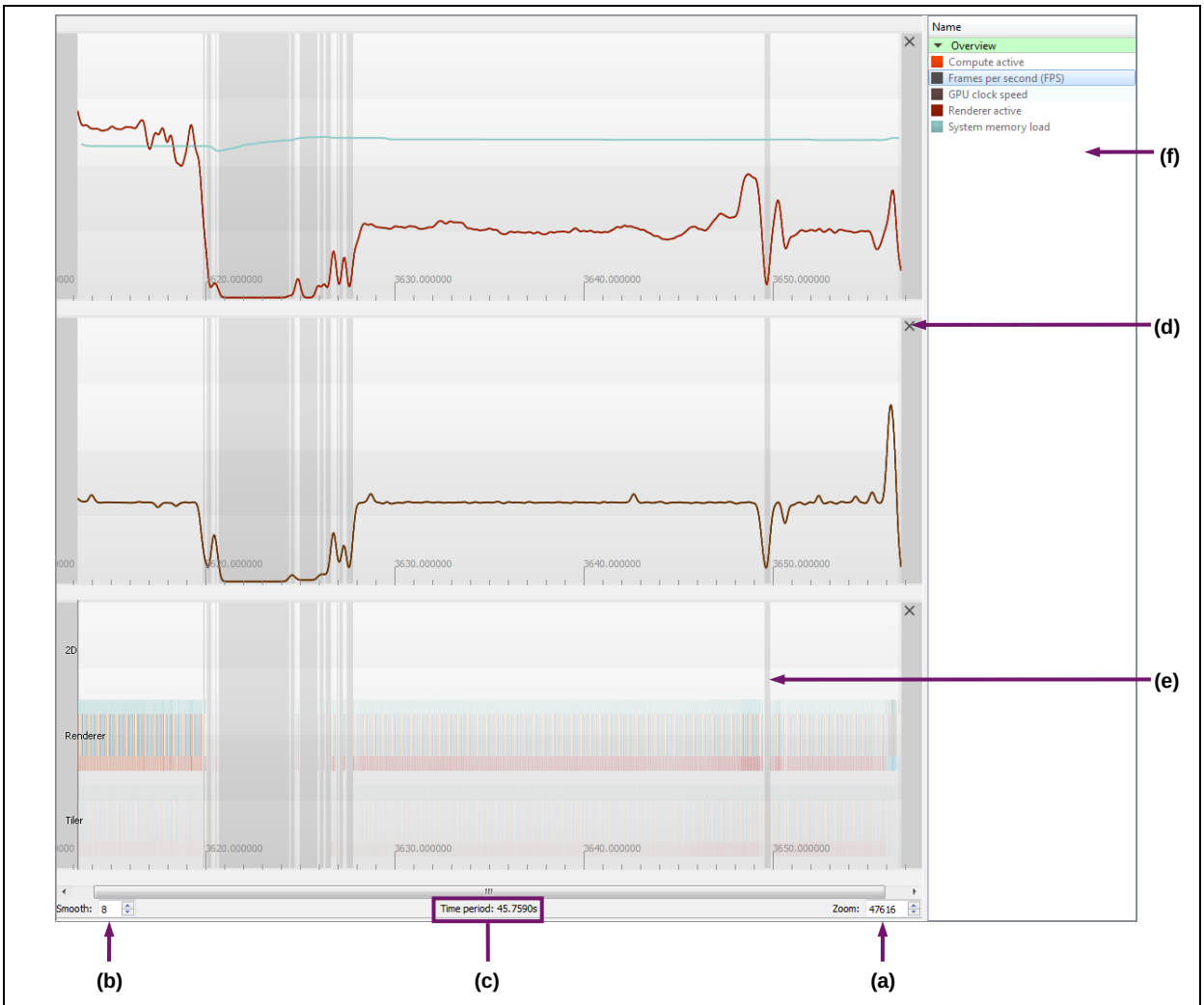

Figure 12. Graphical representation of captured data

3.5.1. Using the Graph View

A number of user actions can be performed when interacting with the graph view. These are documented next.

Zoom in and out of a Graph

Zooming in and out of a graph is a useful feature for being able to visualize details of the captured data as well as to have a more general perspective of the trend of the data as a whole. One method of zooming in and out is to use the scroll button of a mouse in the graph view. Alternatively, a zoom value can be specified in the `Zoom` field (Figure 12a). The lower the zoom value, the bigger the magnification and vice versa.

Note: By default, scrolling up decreases the size of the graph whereas scrolling down zooms into it. The mouse scroll order can be inverted by using the `Preferences` *dialog box (see Section "Preferences").*

Smooth out Counters

Counters displayed in the timeline can be smoothed out by specifying a value in the `Smooth` field (Figure 12b). The maximum value is `80`. Increasing the value will smooth out the counter and the counter plots become easier to interpret.

View Captured Data over the Entire Time Period

To adjust the graph view such that the captured data is displayed for the entire time period, perform the following:

1. Right-click a graph in the `Timeline area`. This will open an action menu.
2. From the menu, select the option called `View Selection/All`.

Note: The value of the total time period is specified at the bottom of the graph view (Figure 12c).

View the Start of the Captured Data

To adjust the graph view such that the start of the captured data is shown, perform the following:

1. Right-click a graph in the `Timeline area`. This will open an action menu.
2. From the menu, select the option called `View Earliest`.

View the End of the Captured Data

To adjust the graph view such that the end of the captured data is shown, perform the following:

1. Right-click a graph in the `Timeline area`. This will open an action menu.
2. From the menu, select the option called `View Latest`.

Select a Time Range

A specific region can be selected on a graph during analysis. This can be achieved by using the `Ctrl` key while left clicking and dragging on a graph. To deselect an already selected region, press `Ctrl` and left click anywhere on the graph. The selection of a time range is relevant when viewing averaged values over the range (see Section 3.6.1).

Change the Graph View Layout

The layout of the graph view can be quickly changed to a single or split view layout. This is carried out by using the `Single View Layout` and `Split View Layout` options from the `View` menu (see Section 0).

Change the Graph Rendering Options

During an analysis task, certain types of data can be shown or hidden on demand in the graph view. To change the data render options, perform the following:

1. Right-click a graph in the `Timeline area`, where the change needs to be made. This will open an action menu.
2. From the menu, select or remove the options `Render 25%, 50%, 75% Quarters`, `Render Timing Data` or `Render Marks`, as appropriate. The change will be reflected on the graph.

Restore the Default Graph View

After changes to the graph view have been made it is possible to revert to default settings, if required. To achieve this, perform the following:

1. Right-click a graph in the `Timeline area`. This will open an action menu.
2. Select the option `Restore Default View`.

Close a Graph in a Multi-Graph View

In a multi-graph view, as in Figure 12, it is possible to close one of the graphs by clicking its corresponding button to close it (Figure 12d). To display the graph again, restore the graph view to default.

Add and Remove Tabs

Tabs displaying graphs can be added and removed on demand. This procedure is highlighted in Section 3.2.1.

Save a Graph as an Image

PVRTune GUI comes with options to allow graphs to be saved as images. There are different modes for saving a graph as an image and the procedure is next highlighted:

1. Right-click a graph in the `Timeline area`. This will open an action menu.
2. From the menu, select the desired save settings which are one of `Save Image with HUD` (saves 'foreground' content when mouse hovering over a graph), `Save Image` (saves image in the same location as the binary) and `Save Image To…` (saves image to a user-defined location).

Pause the View during Connected Analysis

During real-time profiling, the graph view continuously changes with time since the collected data is dynamic. To pause the view at a given time, perform the following steps:

1. Right-click a graph in the `Timeline area`. This will open an action menu.
2. Select the option `Pause View`.

Add and Remove Counters to Graph

Counters can be added on demand to the timeline by selecting them from the `Counter Table` window and dragging them to the graph. This is covered in more detail in Section 3.6.2.

Note: Some counters, when added to a graph view, may not be immediately visible as a result of very low or very high Y-axis values. This can be overcome by adjusting the Y-axis scaling of the counter (see Section "Change the Y-Axis Scaling of a Counter").

View Events

Several events (also known as marks) are displayed in the timeline in addition to all of the other information. These marks are signified by bars running from the top of the graph to the bottom (Figure 12e). A full list of the supported events can be found in Section 3.5.3.

Use the Counter Legend

The `Timeline area` also contains a `Counter Legend` section which can be used for several purposes (Figure 12f). See Section 3.6.4 for more details.

3.5.2. Timing Data

Timing data can be made visible on a graph by choosing the `Render Timing Data` option from the action menu displayed when right-clicking the graph (see Section "Change the Graph Rendering Options"). By default, in a multi-graph view (e.g., using default view settings) the timing data is always displayed in the bottommost graph.

Figure 13 depicts an example of captured timing data. The timing data is arranged on several timelines, each with its own label. The timeline labelled `Tiler` represents the Tile Accelerator (TA) core time, which is a measure of the time spent in tiling/culling the frame and running vertex shaders. The timeline labelled `Renderer` represents Renderer time and is a measure of how much time is spent fetching textures, processing fragment shaders and other fragment processing tasks. These timelines represent the two main stages in the Tile Based Deferred Rendering (TBDR) process.

Note: In addition to the `Tiler` *and* `Renderer` *timelines, there can also be others depending on the hardware being profiled. A summary of hardware-specific terms is provided in Appendix A. Also consult the "PowerVR Hardware Architecture Guide" for a more detailed understanding of the PowerVR hardware architecture.*

Each block displayed in the representation of the timing data corresponds to a given task or activity within a frame and is colour coded to make the frame, process ID and render target easily identifiable. In addition to `Tiler` and `Renderer` timing data, there can also exist other sets of data for transfer tasks, 2D core time (for chips with dedicated 2D cores), compute time (for PowerVR Series6 onwards) and custom timing data sent using PVRScope.

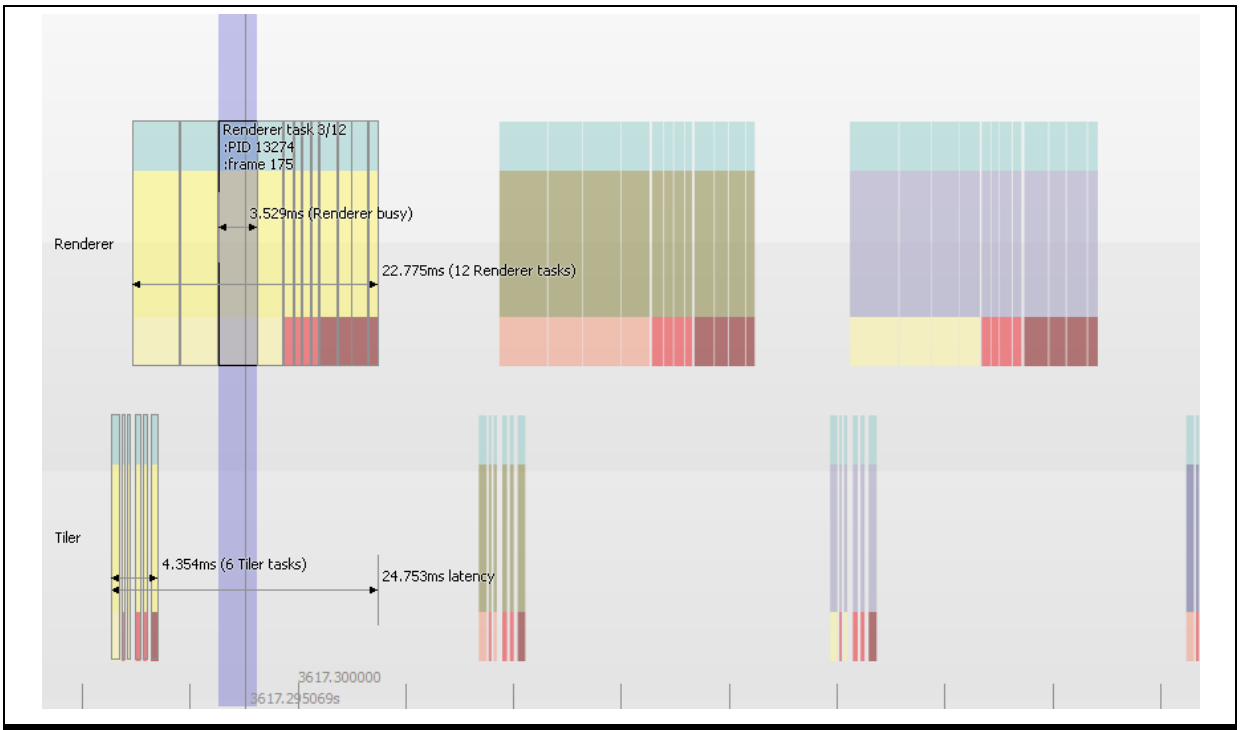

Figure 13. Example of Tiler and Renderer timing data

View Details of Timing Data

A wealth of information can be gained from the timing data by holding the mouse pointer over a task in the timeline. The information displayed this way details the process ID, the frame number, the total number of tasks on each timeline (such as Tiler and Renderer) that can be attributed to the same frame, the time spent on each task, the time spent processing a set of tasks, and the total time spent processing the frame.

Transfer Tasks

Transfer tasks represent the time spent processing tasks related to moving large amounts of memory, such as blitting or texture uploading. Figure 14 shows an example where a series of transfer tasks are displayed as timing data. These tasks are depicted as a series of grey blocks with pink tips.

Figure 14. Transfer tasks

Task Colour Coding

The displayed timing data is colour coded for convenience of interpretation. Blocks of the same colour represent a single frame (see Figure 13), where the colours are recycled every sixteen frames. In addition to the general colour of a block, its top and bottom tips are also coloured. The top tips represent the process ID, where different colours indicate different process IDs. On the other hand, the bottom tips represent the render target, where different colours indicate different render targets.

Figure 15 provides an example of task colour coding used in the display of timing data. The core colour of a task represents a single frame. Each frame is given a core colour (Figure 15a), an associated process (Figure 15b) and a render target (Figure 15c). Repeated top tip colours refer to the same process ID. In Figure 15 the tasks were generated by one process, which is indicated through the use of a single colour (pale blue) across the top tips. Repeated bottom tip colours refer to the same render targets. In the example, three render targets are present and these are indicated by the three colours used for the bottom tips of the tasks. In double or triple buffered situations the render targets are different for each frame, alternating between each of the back buffer targets.

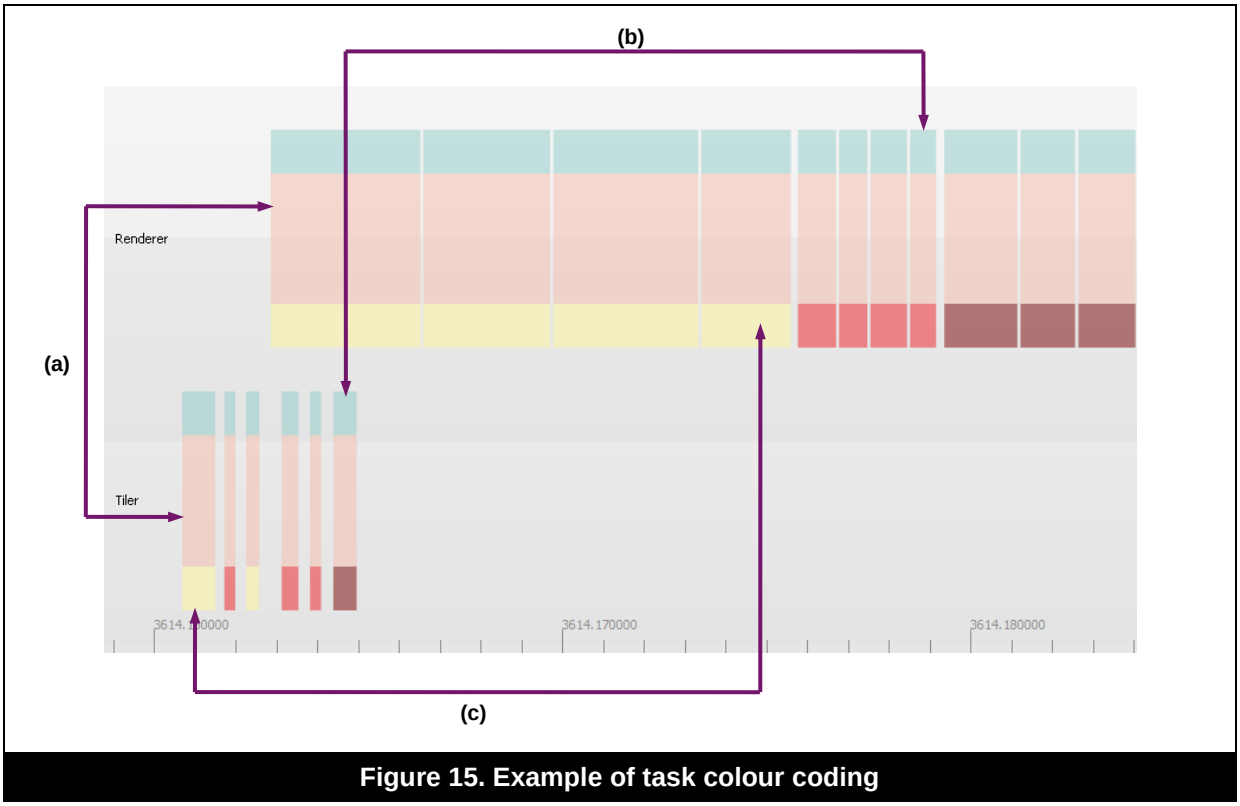

Figure 15. Example of task colour coding

Driver Timing Data

Driver timing data becomes relevant when PVRTune is used in combination with PVRTrace. The data is displayed in its own timeline and is represented as a series of tasks labelled with the thread ID that called the driver. An example of driver timing data is provided in Figure 16.

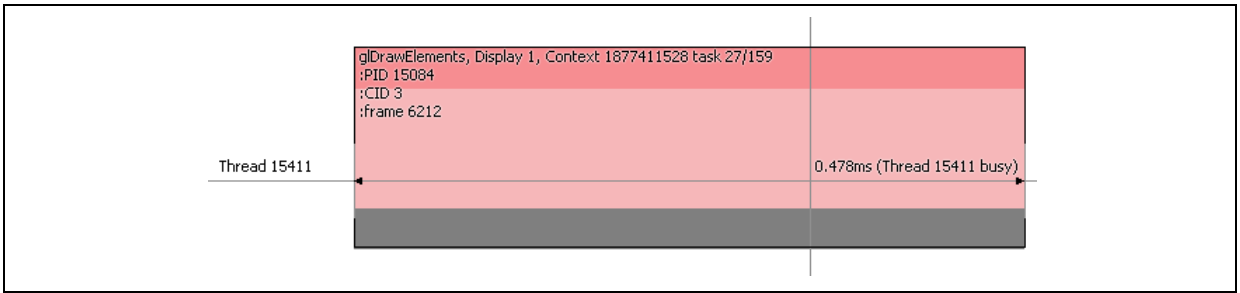

Figure 16. Example of driver timing data

Note: By default, only the most expensive calls are displayed, e.g., `glDrawElements`, `glReadPixels`, *shader compilation and texture uploads. For a more verbose output, open PVRHub, select* `Options` *and set* `API Function Timing Events` *to* `Verbose`.

Warning: Setting `API Function Timing Events` *to* `Verbose` *may affect performance.*

3.5.3. Event List

Active Counters Changed

This event represents the point at which the active counter group has been changed using some custom hardware counter group . The event appears as a vertical grey line (Figure 17).

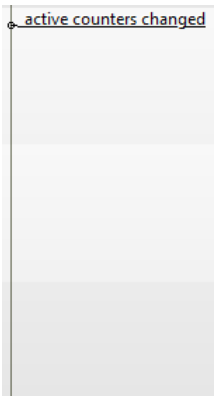

Figure 17. Example of active counters changed

Event Ordinal Reset

This event is usually hidden by `active counters changed`. The event represents a change in the ordering of the counters or counter sources read by PVRPerfServer and appears as a vertical green line (Figure 18).

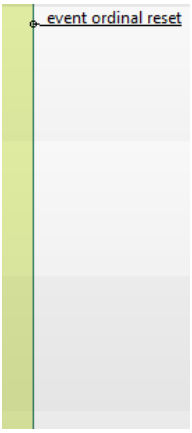

Figure 18. Example of event ordinal reset

Custom Mark

Custom marks are marks that have been sent to PVRTune either by a PVRScope enabled application or by pressing the `M` key in PVRPerfServer. A custom mark appears as a vertical red line (Figure 19).

Figure 19. Example of custom mark

Power-off Period

Power-off periods are represented by vertical grey blocks in the graph view (Figure 20). These events occur when the hardware has gone to sleep or powering down to save power due to a lack of work.

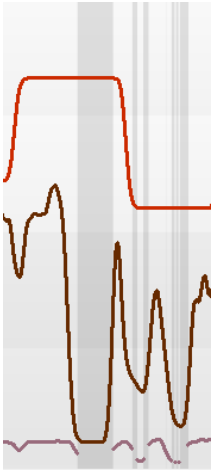

Figure 20. Example of power-off period

Data Loss Period

A data loss period is one during which PVRTune has lost data that was sent by PVRPerfServer. A data loss period is represented as a vertical green block (Figure 21). There are several possibilities to minimize data loss, such as:

- Decrease device CPU load.
- Alter the PVRPerfServer data read rate, using the `-c` command-line option.
- There could be congestion on the network the data is being transmitted on. This can be alleviated by using a less congested network or by attempting ad-hoc networking.
- Save data to a file rather than send it over the network (see Section 2.5.4 for the `-sendto=` command-line option for PVRPerfServer).
- Reduce the data quantity. This can be achieved by disabling the –`periodic=` or –`graphics=` option via the PVRPerfServer command-line (see Section 2.5.4).

Note: PVRPerfServer command-line options can also be set remotely from PVRTune (see Section 2.5.4).

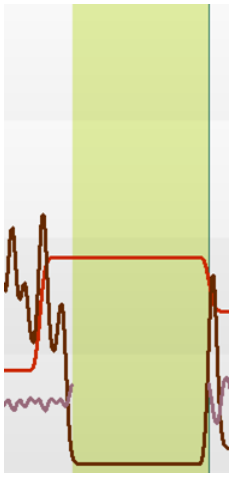

Figure 21. Example of data loss period

Hardware Reset Period

Hardware reset periods occur when the hardware locks up and stops responding, possibly due to firmware or driver error. This usually happens due to the buffer filling up because of the limited storage available. A hardware reset can be identified as a vertical pink block (Figure 22).

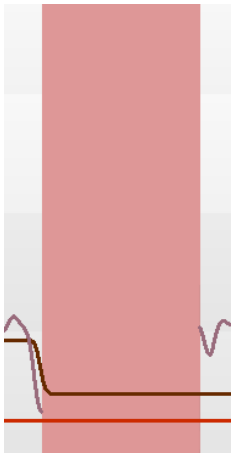

Figure 22. Example of hardware reset period

Smart Parameter Management Mode

The Smart Parameter Management (SPM) mode is used to manage parameters when it buffer-overflows. A Renderer task labelled `Renderer SPM Task` appears when this mode is activated (Figure 23).

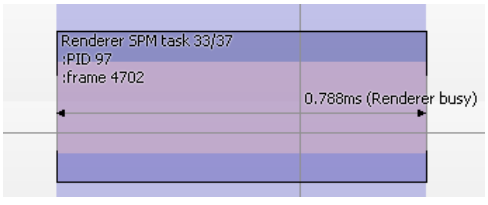

Figure 23. Example of SPM mode

3.6. Working with Counters

Counter information is listed in the `Counter Table` window of PVRTune GUI. An example of listed counters is shown in Figure 24. The counters are organised under different headings (e.g., `Overview`, `GPU Advanced`, etc.) which can be expanded or collapsed as needed. During connected analysis, the table also displays the counters available within specific counter groups. In the example, the `Group` dropdown box is disabled (Figure 24a) which is normal during offline analysis.

Note: For a full list of default PVRTune counters and their description, refer to the document entitled "PVRTune Counter List and Description".

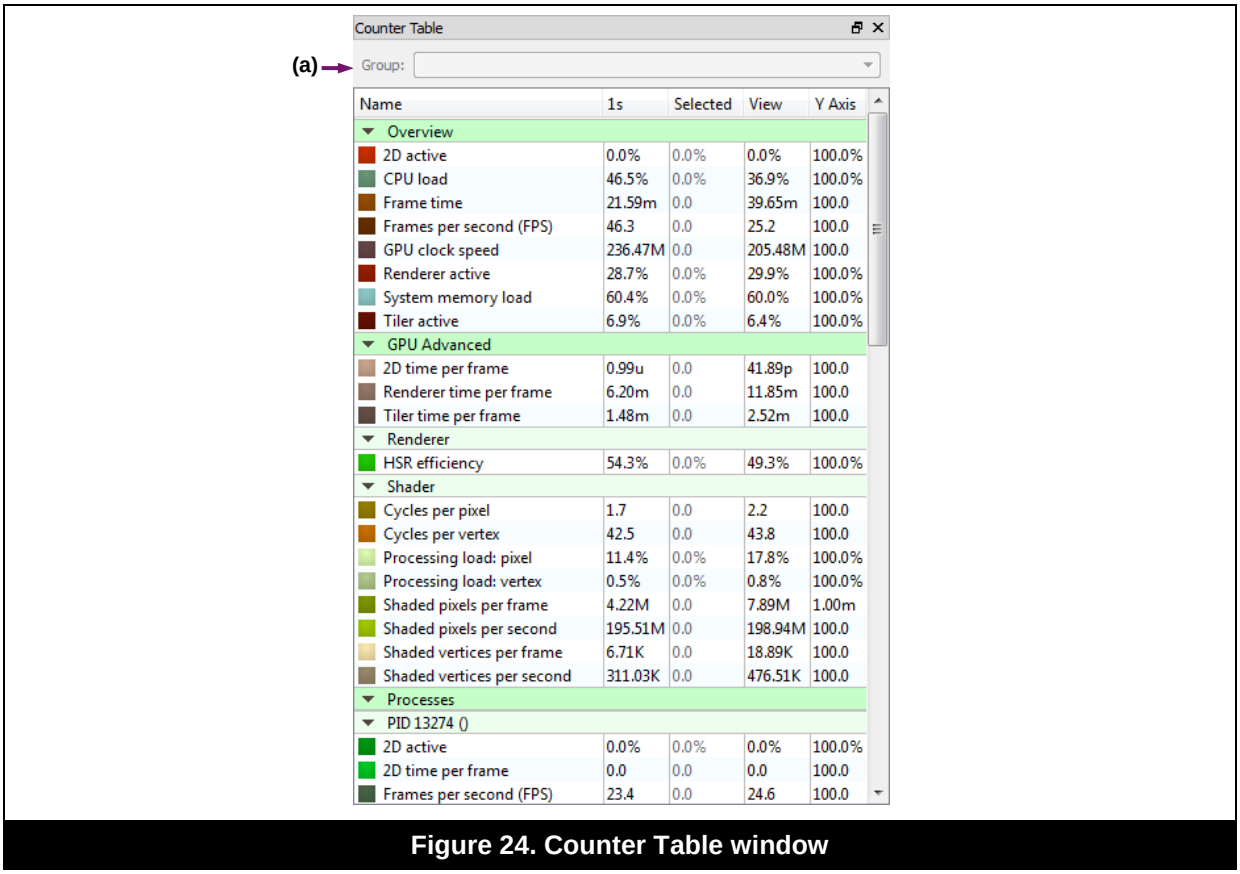

Figure 24. Counter Table window

3.6.1. Select Columns to Display

By default, counter data are tabulated following a similar format as shown in Figure 24. Additional data sets can be customized for display by right-clicking the `Counter Table` window and choosing the `Select Columns…` option, as shown in Figure 25.

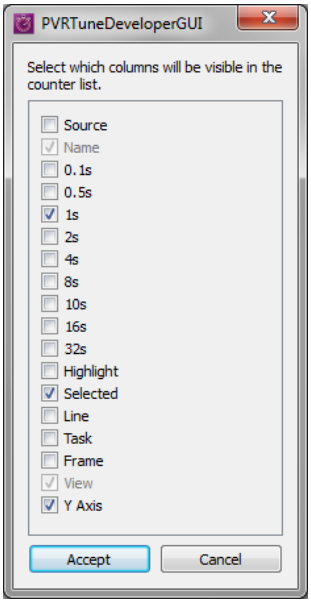

Figure 25. Dialog box for selecting and deselecting columns

Table 5 identifies a summary of the additional columns that can be selected and deselected on demand.

Table 5. Summary of columns and their description

Column	Description
`0.1s` to `32s`	These values represent time frames. The columns show the average value of counters over the most recent timespan (e.g., the `32s` column gives the average over the last 32 seconds).
`Selected`	By selecting a time range (see “Select a Time Range”) and using this checkbox, it is possible to view the average over the selected region.
`Line`	This column responds to mouse hover. It shows the values of counters at the time associated with the position of the mouse in the graph view.
`Task`	This column responds to mouse hover. It shows the average value of counters over the time frame of a single Tiler or Renderer task.
`Frame`	This column responds to mouse hover. It shows the average value of given counters over the timeframe of a single frame.
`View`	This column responds to mouse hover. It shows the average value of counters over the timeframe of the graph for which the cursor is currently over.

Add a Column of Data to the Counter Table

An additional column of data can be included in the `Counter Table` window by selecting it from the participating dialog box (Figure 25). Multiple columns can also be chosen if necessary.

Remove a Column of Data from the Counter Table

A column of data can be removed from the `Counter Table` window by selecting it from the participating dialog box (Figure 25). Multiple columns can also be chosen if necessary.

3.6.2. Add a Counter to the Graph View

During connected analysis in PVRTune GUI, counters are updated in real-time and can be plotted against the timeline to give a graphical representation of their change over time. Counters can also be added during offline analysis. To add a counter to a graph view, perform the following steps:

1. Identify the counter that is of interest from the `Counter Table` window (Figure 24).
2. Drag and drop the counter from the `Counter Table` window to an existing graph present in the `Timeline area`. This results in the addition of new plotted data on the graph. Figure 26 shows an example where the `Frames per second` counter, represented by the line graph, has been added to a graph view with rendered timing data.

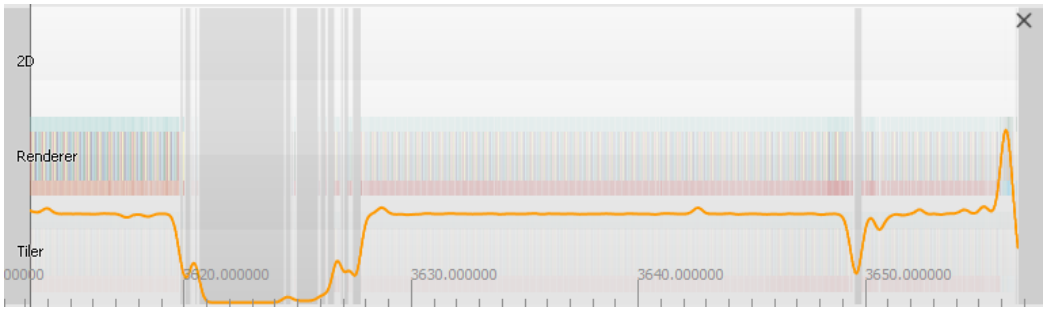

Figure 26. Adding a counter to a graph view

Note: A new graph can be created by dragging and dropping a counter to a graph placeholder available in the `Timeline area`. A graph placeholder is the narrow area of space located above or below a graph view. An example is shown in Figure 27.

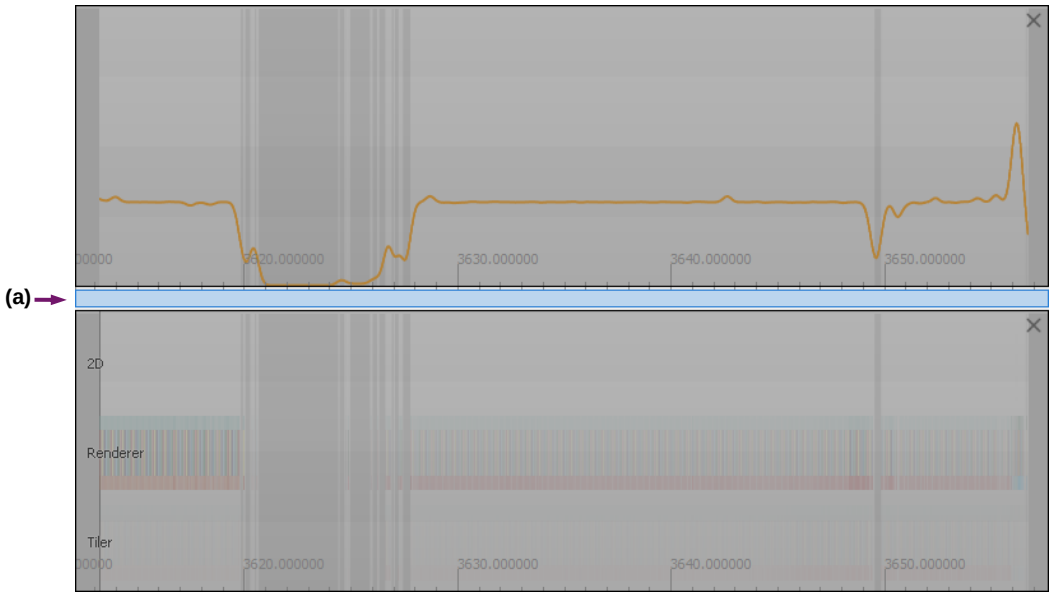

Figure 27. Creating a new graph by adding a counter to a graph placeholder

3.6.3. Remove a Counter from the Graph View

Counters plotted in the graph view can be removed when they are no longer required. To remove a counter from the graph view, perform the following steps:

1. In the `Timeline area`, identify the counter that needs to be removed.
2. Click the plot corresponding to the counter. This will highlight the plot.
3. Press `Delete` to remove the plotted counter data from the graph. The same action can be achieved by right-clicking the counter plot and selecting the `Remove Selected Counter` option from the action menu.

3.6.4. Counter Legend in the Timeline Area

The `Counter Legend` is located at the right hand side of the `Timeline area` and lists the counters that are only relevant to the plotted graphs (see Figure 12). A number of options are available to interact with the information displayed in the `Counter Legend` and these are next discussed.

Highlight a Plot Corresponding to a Counter

In order to highlight a plot for a counter displayed in the `Counter Legend`, mouse over the name of the counter.

Add a Column of Data to the Counter Legend

The data sets displayed in the `Counter Legend` can be customized on demand. In order to do this, perform the following steps:

1. Right-click anywhere in the `Counter Legend`. This will open an action menu with several options.
2. Click the `Select Column…` option. This will open a dialog box to aid the selection of columns (see Figure 25).

Note: For further information about the description of the various columns and their functions, see Table 5.

3. Select the desired column to display by using its corresponding checkbox. Multiple columns can be chosen if required.
4. Click the `Accept` button to apply the changes. The selected column is then added to the `Counter Legend`.

Remove a Counter from the Counter Legend

A counter can be removed from the `Counter Legend` when it is no longer required. To achieve this, click the counter and hit the `Delete` key. Alternatively, right-clicking the counter will open an action menu from which the `Delete Selection` option can be chosen. Removing a counter from the `Counter Legend` also removes its corresponding plot from the graph view.

Remove all Counters from the Counter Legend

All counters displayed in the `Counter Legend` can be removed on demand. To achieve this, right-click anywhere in the `Counter Legend` to open an action menu from which the `Delete All` option can be chosen. Removing all counters from the `Counter Legend` also removes their corresponding plots from the graph view.

Remove a Column of Data from the Counter Legend

To remove a column of data from the Counter Legend, perform the following steps:

1. Right-click anywhere in the `Counter Legend`. This will open an action menu with several options.
2. Click the `Select Column…` option. This will open a dialog box to aid the deselection of columns (see Figure 25).
3. Uncheck the desired column by using its corresponding checkboxes. Multiple columns can be deselected if required.
4. Click the `Accept` button to apply the changes. The deselected column is then removed from the `Counter Legend`.

3.6.5. PVRTrace Software Counters

An application is able to send custom software counters to PVRTune via PVRScope. When an application is being profiled specifically by PVRTrace, the corresponding software counters sent to PVRTune using PVRScope provide data on certain statistics within the running application.

Enable PVRTrace Software Counters

Software counters can be enabled on different operating systems. The approach to enabling these counters on Android, Linux, Neutrino and Windows is as follows:

- **Android**: PVRHub should be launched. On the `Options` screen, ensure that the setting called `Enable Software Counters` is enabled. PVRTrace Recording Libraries must be installed on the device for this to work.
- **Linux**: With PVRHub installed, run `pvr_profile <binary>` where `<binary>` is the application that is required to be profiled with software counters active.
- **Neutrino and Windows**: In the `pvrtraceconfig.json` file on the device, ensure that for `Profiling` the `SoftwareCounters` and `Enabled` options are set to `true`.

List of PVRTrace Software Counters

Table 6 lists the software counters that are available from the PVRTrace libraries.

Table 6. List of PVRTrace software counters

Software counters
Indexed draw calls
Non-indexed draw calls
Points no.
Line no. (list)
Line no. (loop)
Line no. (strip)
Total line no.
Triangle no. (list)
Triangle no. (strip)
Triangle no. (fan)
Total triangle no.
Texture uploads
Texture modifications

Software counters
Scissor calls
Viewport calls
Frame buffer accesses
Vertex shader compiles
Fragment shader compiles
Program links
Framebuffer access (bytes)
Texture uploads (bytes)
Texture modifications (bytes)
Buffer object uploads (bytes)
Buffer object modifications (bytes)
Uniform uploads
Context binds
Shader proportion: pixel
Shader proportion: vertex
Shader proportion: compute
Vertices per batch
Shader slot scheduled count: compute
Shader slot scheduled count: pixel
Shader slot scheduled count: vertex
Shader slot size: pixel
Shader slot size: vertex

3.6.6. Counter Properties

PVRTune GUI displays of a wide range of information associated with counters. By exploring the properties of a counter, it becomes possible to learn more about, e.g., its description and implications when a high value is recorded, amongst others. The most recently clicked counter is displayed in the `Counter Properties` window (Figure 28). The window captures some basic details including the counter name, associated colour and the maximum value of the Y-axis to be used on the graph view.

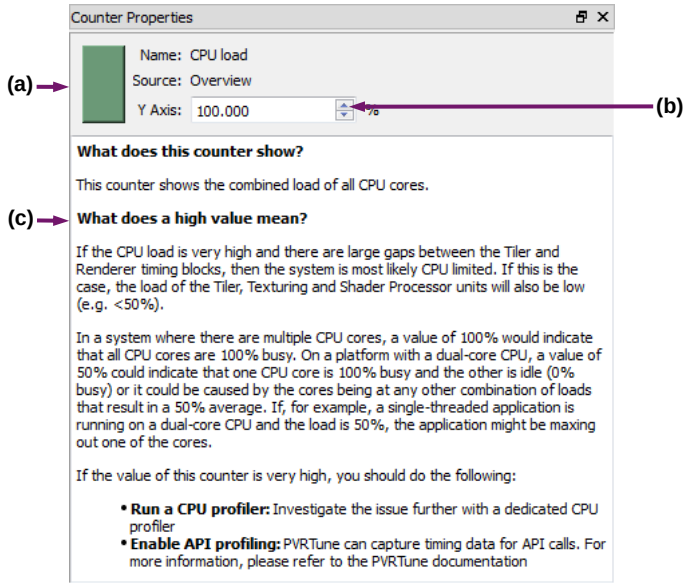

Figure 28. Counter Properties window

Change the Colour of a Counter

The user is able to customize the colour of a counter on demand by following these steps:

1. Identify the counter that is of interest either from the `Counter Table` window (see Figure 24) or from the `Counter Legend` in the `Timeline area`.
2. Select the counter by clicking it. This will display the properties of the counter in the `Counter Properties` window (Figure 28).
3. Click the colour icon associated with the counter (Figure 28a). This will open a colour picker from which a suitable substitute colour can be chosen.
4. Select the desired colour from the colour picker.
5. Click `OK` to complete the procedure.

Change the Y-Axis Scaling of a Counter

The ability to change the Y-axis scaling of a counter is a handy feature that allows small and large value plots to be scaled for visibility and interpretation. This can be achieved by using the `Y Axis` field provided in the `Counter Properties` window (Figure 28b). For particularly large value plots (such as transformations per frame) it is necessary to enter a large scale. On the other hand, for very small value plots (such as frame time) it is necessary to enter a fractional value.

View the Description of a Counter

Counter description is very important as it helps data interpretation and analysis. To view the description of a counter, perform the following step:

1. Identify the counter that is of interest either from the `Counter Table` window (see Figure 24) or from the `Counter Legend` in the `Timeline area`.
2. Select the counter by clicking it. This will display the properties of the counter in the `Counter Properties` window (Figure 28).
3. Interpret the detailed information provided (Figure 28c).

3.7. Process ID (PID) Window

The `PID` window in PVRTune GUI displays a list of the individual connections being used in the recording. It only displays the known PIDs relevant to the operation and not all the PIDs running on the system are listed.

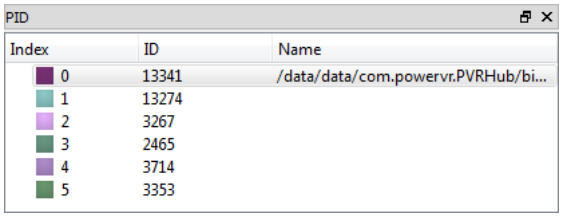

Figure 29. PID window

Note: If no name can be depicted in the `Name` *column, inspect the* `--pid` *command-line option from PVRPerfServer (see Section 2.5.4).*

3.8. Remote Editor Window

This functionality allows for real-time, remote editing of shaders and various numerical value adjustments. PVRScope-enabled applications can register data as being editable or readable by PVRTune. The value of this data can then be passed by PVRScope, via PVRPerfServer, to PVRTune. Any value that has been registered appears in the `Remote Editor` window in PVRTune GUI (Figure 30). Values can be edited using the options provided in the window and the changes are automatically sent back to the application.

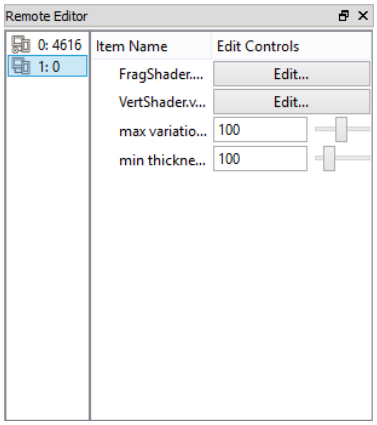

Figure 30. Remote Editor window

Note: For more information on PVRScope refer to the "PVRScope User Manual".

3.9. Renderstate Override Window

The renderstate override functionality in PVRTune permits the user to remotely control the renderstate of a target device. This provides a useful means of prototyping render changes as well as enabling the quick identification of obscure bottlenecks. The renderstate override functionality is accessible from the `Renderstate Override` window and in order to use it, PVRTrace profiling mode should be enabled.

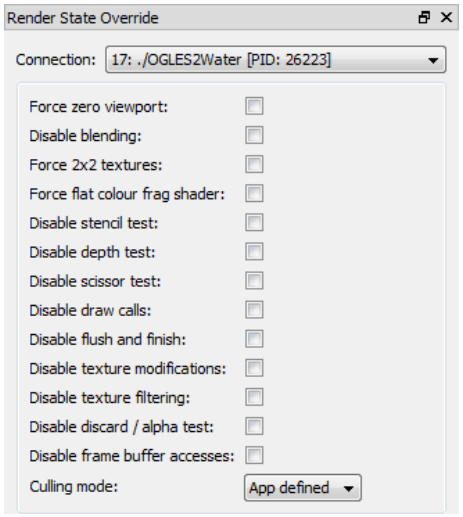

Figure 31. Renderstate Override window

Table 7 provides a list of the various renderstate override options as well as their description.

Table 7. Explanation of renderstate override options

Option	Description
`Force zero viewport`	Modifies the viewport to have zero sized dimensions.
`Disable blending`	Disables alpha blending.
`Force 2x2 textures`	Forces textures to be 2 pixels by 2 pixels in size.
`Force flat colour frag shader`	Forces the fragment shader to output a single colour.
`Disable stencil test`	Disables the stencil test.
`Disable depth test`	Disables the depth test.
`Disable scissor test`	Disables the scissor test.
`Disable draw calls`	Disables all `glDraw()` function calls.
`Disable flush and finish`	Disables all `glFlush()` and `glFinish()` function calls.
`Disable texture modifications`	Disables `glTexSubImage2D()` and `glTexSubImage3D()`. This disables the updating of texture sub-regions.
`Disable texture filtering`	Disables texture filters. All texture filters are set to the nearest point.
`Disable discard / alpha test`	Disables the alpha test.
`Disable frame buffer accesses`	Disables `glReadPixels()` function calls.
`Culling mode: App defined`	Forces the culling mode to perform culling as defined in the user application.
`Culling mode: None`	Disables culling.
`Culling mode: Back`	Forces the culling mode to perform back-face culling.
`Culling mode: Front`	Forces the culling mode to perform front-face culling.

3.10. Find Window

PVRTune GUI comes with a search capability which is helpful for quickly finding information. The `Find` window (Figure 32) allows the user to input search terms and the retrieved results are narrowed down using an incremental search approach.

A number of items can be searched this way, including activities (e.g., `Renderer`), marks (e.g., `abc`), counters (e.g., `triangles per`), frames (e.g., `40000`), PIDs (e.g., `4242`), time as per the X-axis in the graph view (e.g., `1200`), and time range by separating two numbers with a pair of dots (e.g., `1200.1..1200.2`).

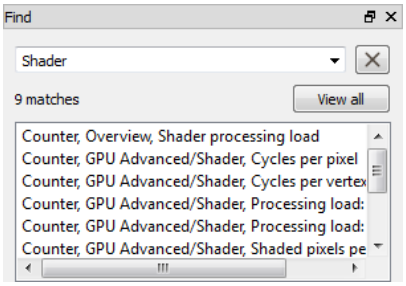

Figure 32. Find window

Note: Counters can be dragged and dropped from the `Find` window to the graph view in order to generate new counter plots.

4. Identifying Bottlenecks

The ability to identify bottlenecks is vital when analysing the performance of an application. Section 3.4 and Section 3.9 have previously explained two of the features of PVRTune for monitoring graphics core workloads and renderstate, respectively, in the effort to help identify some types of bottlenecks. This section explains how to identify bottlenecks using the graph view. The bottlenecks usually fall into one of five categories:

- CPU limited.
- Vertex limited.
- V-Sync limited.
- Fragment limited.
- Bandwidth limited.

4.1. CPU Limited

A CPU limited application is often identifiable as an application suffering from poor performance or frame rate even though the graphics core usage is not high. In PVRTune this can be very easily identified since CPU limited applications have a CPU load that is either at or near one hundred percent (Figure 33a), or is widely variable.

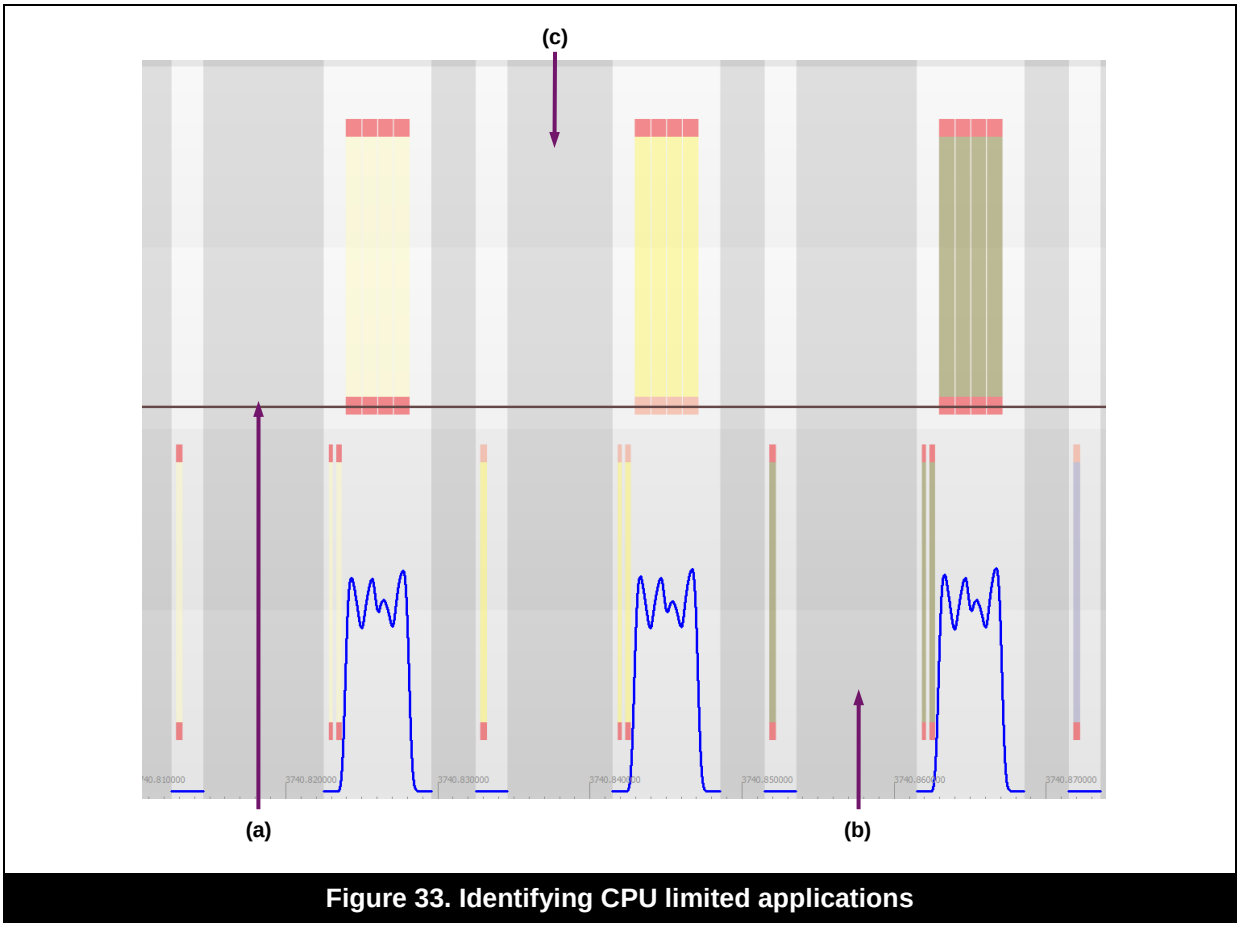

Figure 33. Identifying CPU limited applications

Other identifying factors include gaps in the Shader load, caused by the PowerVR hardware going to sleep while waiting for further instructions (Figure 33b) and regular visible gaps between frames when displaying timing data (Figure 33c).

4.2. Vertex Limited

Vertex limited applications are applications where the bottleneck comes from processing either large amounts of vertices per frame, or from the use of a complex vertex shader, or both. This can be identified by large gaps between Renderer tasks (Figure 34a) while there is little or no gap between Tiler tasks (Figure 34b).

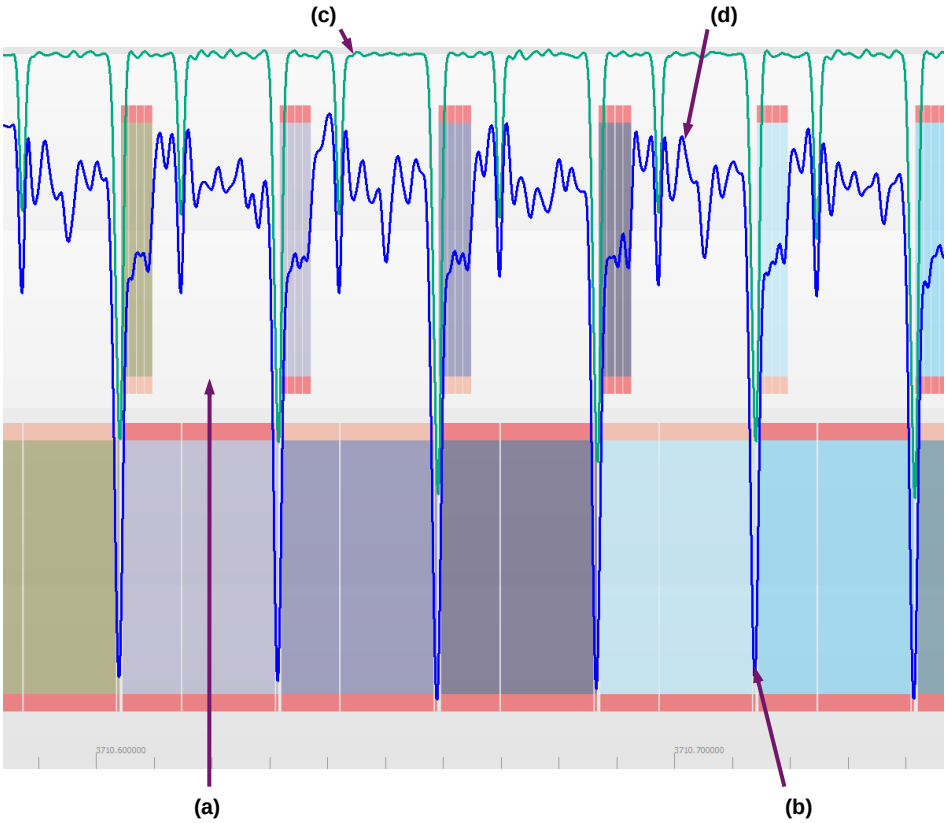

Figure 34. Identifying vertex limited applications

Further information can be gained from the `Processing load: Vertex` and the `Tiler load` counters. If `Tiler load` is high (Figure 34c) but `Processing load: Vertex` is not then the scene has too many vertices in it and the cost is coming from the tiling process. On the other hand, if `Processing load: Vertex` is high (Figure 34d) but `Tiler load` is not, then the bottleneck is likely to be in the vertex shader.

4.3. V-Sync Limited

Vertical Synchronisation (V-Sync) is a display option that forces an application to synchronize graphical updates with the update rate of the screen. This causes some frames to be slightly delayed and enforces a maximum refresh rate, but reduces screen tearing and can save power. V-Sync limited applications are often characterized by intermittent gaps between frames in the graph view, and the frame rate appears to be limited at a set maximum value. If possible, V-sync should be disabled when profiling an application as it adds noise to the PVRTune output and this makes it more difficult to diagnose where optimization work could be beneficial or if completed optimization has been successful.

4.4. Fragment Limited

Fragment limited applications are very common and occur in most scenes that have fewer vertices than the number of pixels in the framebuffer. Fragment limited applications can be identified when there is the presence of no gaps between Renderer tasks (Figure 35a), large gaps between Tiler tasks (Figure 35b), a high value of `Processing load: Pixel` (Figure 35c) or high Shader clock cycles per pixel (Figure 35d).

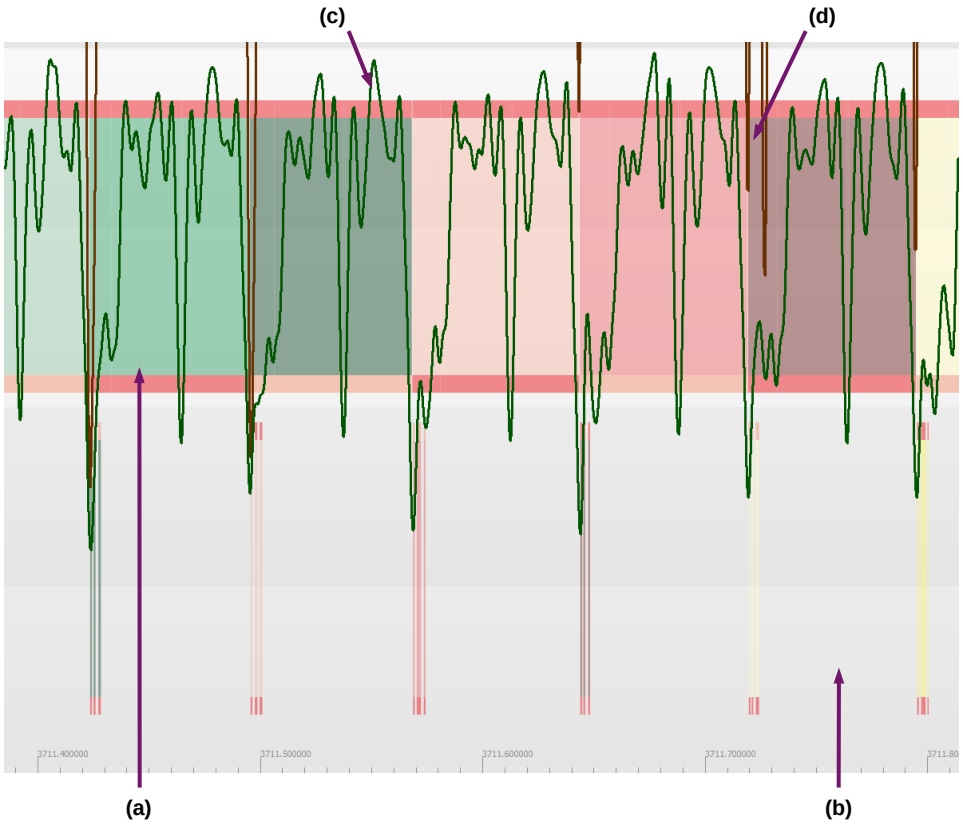

Figure 35. Identifying fragment limited applications

4.5. Bandwidth Limited

Cases of bandwidth limited applications are both hard to visualize and identify, as they may appear as other bottlenecks. Programs may be bandwidth limited if:

- Timeline shows the application to be fragment limited but the `Processing load: Pixel` is low.
- Timeline shows the application to be vertex limited but the `Processing load: Vertex` and `Tiler load` are low.

Other instances of bandwidth limitation may occur. For example, if many units are accessing memory simultaneously then the available system memory bandwidth limits can slow all operations on the hardware. This is platform specific and, as such, there is no counter to record it. As a rule of thumb, action should always be taken to reduce bandwidth use whenever possible through the correct use of texture compression, mesh optimisation and by avoiding unnecessary texture reads, etc.

Note: Bandwidth in System-on-Chip (SoC) devices is shared amongst all components of the chip. Non-graphics processor areas of the chip using large amounts of bandwidth may still cause application graphics to be bandwidth limited.

5. Contact Details

For further support, visit our forum:
http://forum.imgtec.com

Or file a ticket in our support system:
https://pvrsupport.imgtec.com

To learn more about our PowerVR Graphics SDK and Insider programme, please visit:
http://www.powervrinsider.com

For general enquiries, please visit our website:
http://imgtec.com/corporate/contactus.asp

Appendix A. Hardware Terms: Quick Reference

Table 8 identifies a list of hardware terms in order to make it easier for the user to interpret the statistics displayed in PVRTune. For a more detailed interpretation, refer to the help function in the software itself as the statistics will change depending on the hardware.

Table 8. Quick reference of hardware terms

Term	Description
Tiler	The Tiler is responsible for processing the vertices and tiling. This includes transforming vertices, culling, clipping and storing the data. The timing data contains the TA and Shader processing operations.
Renderer	The Renderer is responsible for processing the fragments within a given tile and the timing data contains all the ISP, TSP and fragment processing operations.
TA	The Tile Accelerator (TA) takes geometric data that has been transformed by the shader processor as input and clips, projects and culls it. The TA Core timing data contains vertex processing and tile setup.
TSP	The Texture and Shading Processor (TSP) performs interpolation and schedules tasks for the shader processor and texture data pre-fetches.
ISP	The Image Synthesis Processor (ISP) is responsible for per-tile Hidden Surface Removal (HSR) to ensure that the fragments processed by the TSP are only those that will affect the rendered image.
Shader	The PowerVR shader processor is a flexible, multi-threaded processor capable of executing vertex, fragment, graphics core instructions and issuing memory access operations (such as texture fetches).
Compute	Series6 only: The Compute Core is dedicated to compute tasks issued via the Compute Data Master (CDM). Compute timing data includes the programmable arithmetic handled by the shader processor.

Note: Refer to the "PowerVR Hardware Architecture Guide" for a more detailed understanding of the PowerVR hardware architecture.

Appendix B. Exported Data

B.1. Raw Data

The `Raw Data` file contains the raw data that PVRTune sources to display timing and graphing details (Table 9). Each row represents a single point in time and each column represents some specific item of data at that time.

Table 9. Raw data values and their description

Value	Description
`nHW16ClockCnt`	The value in this field represents a counter on the graphics core that increments once every 16 clock cycles. This is used to generate an estimate of the clock speed and to timestamp graphing and timing data.
`nPacketOrdinal`	This value represents the ID number of the packet sent from the driver to PVRTune. This number is incremented for every packet. Missing values indicate that data has been lost.
`nFrameNumber`	This value represents the frame number of the application whose data the given row represents.
`nClockSpeed`	This value represents the clock speed of the graphics as reported by the driver.
`eType`	This value is an integer representation of the type of message received, as described in `eType.Core`, `eType.Activity`, and `eType.Process`.
`eType.Core`	This value represents the 'core' that the given event is about, e.g., Tiler, Renderer, 2D core or the uKernel.
`eType.Activity`	This value represents the type of activity an event is about. The values identified in Table 10 are valid.
`eType.Process`	This value shows the kind of process associated with the above `eType.Activity`. If the activity is starting, `eType.Process` reports `start`. If it is ending it reports `end`, otherwise `impulse`.
`nPID`	This value represents the process ID of the application whose data the given row represents.
`nRTData`	The values used for `nRTData` are used to represent the render target being used at the time the data was taken. The specific integer used is arbitrary and not meaningful. However, shared values across activities represent shared use of a render target across those activities.
`nInfo`	This contains the value of one of a series of registers, with the specific register being based on the value of `eType` (see Table 11) and defined in the technical reference manual for the appropriate chip.
`Counter[0-8]`	Each of these fields represents the absolute value of the hardware counter mapped to this software counter. In PowerVR Series5 chips this varies by active group. In PowerVR Series5XT chips, `Counter8` always reads zero. Knowing which counter source has been programmed into which counter is possible using the active counters CSV file (see Appendix B.3). *Note: Information regarding which counter sources are directed into which counter for each group can be found in the technical reference manual for the appropriate chip.*

Table 10. eType.Activity values and their description

Value	Description
Power	Represents the system's power supply.
Event	Representative of no specific activity and used to update counter values only.
Tiler	Standard vertex work.
Renderer	Standard fragment work.
Renderer SPM	Fragment work done due to entering SPM mode.
Transfer	A transfer queue task.
2D	Work done on the dedicated 2D core.
MK Tiler	Signifies the uKernel working on a vertex related task.
MK Renderer	Signifies the uKernel working on a fragment related task.
MK 2D	Signifies the uKernel working on something for the dedicated 2D core.
MK event	Signifies the uKernel processing an event.
MK Tiler dummy	Signifies the uKernel processing a vertex task to clean up after a failure.
MK Renderer dummy	Signifies the uKernel processing a fragment task to clean up after a failure.
MK 2D dummy	Signifies the uKernel processing a 2D core task to clean up after a failure.
MK Transfer dummy	Signifies the uKernel processing a transfer task to clean up after a failure.

Table 11. eType values and registers

eType	Register
Power Start	MKIF_HOST_CTL.ui32HostClock
Tiler Start	EUR_CR_DPM_PAGE_STATUS
Tiler End	EUR_CR_DPM_PAGE_STATUS
Renderer Start	EUR_CR_DPM_PAGE
Renderer End	EUR_CR_DPM_PAGE
Renderer SPM Start	EUR_CR_DPM_PAGE
Renderer SPM End	EUR_CR_DPM_PAGE

B.2. Timing Data

The Timing Data file contains much the same information as the Raw Data file, but in a slightly more simplified form, e.g., no event or impulse data is included and the nHW16ClockCnt value is replaced with a microsecond accurate time based on the host clock.

B.3. Active Counters

The Active Counters file contains two different elements depending upon the chip. On PowerVR Series5 chips two fields are present: a microsecond accurate time based on the host clock and the active counter group at that time. This is primarily used to track counter group changes as these alter the meaning of the values of Counter[0-8] in the Raw Data and Timing Data files.

On PowerVR Series5XT chips nine fields are present: the first is a microsecond accurate time based on the host clock; fields two through to nine contain a mapping of which hardware counters map to which software counters. This is also used to track counter group changes as these alter the meaning

of the values of `Counter[0-8]` in the `Raw Data` and `Timing Data` files. The values in each of the counter fields represent the `Group`, `Bit`, `CounterBitSelect` and `SumMax` value. How this value translates into a counter source can be found in the technical reference manual for the appropriate chip.

B.4. Marks

The `Marks` file contains three values: a microsecond accurate time based on the host clock; the process ID of the application which sent the mark; and the value passed to PVRTune from PVRScope in the form of a string.

B.5. User Counters

This option creates a file containing the user defined counters in CSV format. A description of the output format for counters is present in Appendix B.3.

B.6. User Timing Data

This option creates a file containing the user defined timeline data in CSV format. END{{Complete}}