

# PVRTune

## User Manual

Copyright © Imagination Technologies Ltd. All Rights Reserved.

This publication contains proprietary information which is subject to change without notice and is supplied 'as is' without warranty of any kind. Imagination Technologies and the Imagination Technologies logo are trademarks or registered trademarks of Imagination Technologies Limited. All other logos, products, trademarks and registered trademarks are the property of their respective owners.

Filename : PVRTune.User Manual.1.49f.DeveloperExternal.doc  
Version : 1.49f External Issue (Package: POWERVR SDK 2.09.29.0619)  
Issue Date : 19 Sep 2011  
Author : Imagination Technologies Ltd

## Contents

<b>1. Introduction</b>	<b>3</b>
1.1. Component Structure	4
<b>2. SGXPerfServer</b>	<b>5</b>
2.1. Installation	5
2.1.1. Android	5
2.1.2. Linux, Neutrino and Windows	5
2.2. Driver (DDK) Compatibility	6
2.2.1. DDK build options	6
2.3. Usage	7
2.3.1. Command-Line Options	9
2.3.2. Run-time options	9
<b>3. PVRTune</b>	<b>10</b>
3.1. Installation	10
3.2. Usage	11
3.2.1. Connecting to a Server	11
3.2.2. SGXPerfServer Remote Control	12
3.2.3. Available performance profiling counters	13
3.2.4. Counter properties	14
3.2.5. Graph views	15
3.2.6. Data Save and Load	17
3.2.7. Miscellany	17
<b>4. How to: analyse an application for performance bottlenecks</b>	<b>18</b>
4.1. Setup	18
4.2. CPU or SGX limited?	18
4.3. Where is my time going?	21
4.3.1. FAQ: why is no counter showing 100% load?	21
<b>5. Related Materials</b>	<b>22</b>
<b>6. Contact Details</b>	<b>23</b>
<b>Appendix A. Counter list</b>	<b>24</b>

## List of Figures

Figure 1-1 PVRTune Structure	4
Figure 2-1 SGXPerfServer – Awaiting Connection	8
Figure 3-1 PVRTune Connection Screen	11
Figure 3-2 PVRTune Connected	12
Figure 3-3 Select Columns	13
Figure 3-4 Graph Views	15
Figure 3-5 Timing Data	16
Figure 4-1 CPU Limited	19
Figure 4-2 Vertex Limited	20
Figure 4-3 Fragment Limited	20

## 1. Introduction

PVRTune, together with SGXPerfServer, is used to graphically view two types of data from the SGX hardware and drivers:

1. Hardware and software counters, rendered as graphs in PVRTune.
2. Timing data from the tile accelerator (TA) and 3D processing cores in SGX, rendered as blocks of activity.

As the communications protocol may change between releases, it is typically necessary to use PVRTune and SGXPerfServer from the same package – it is not recommended to update one to a new build without also updating the other as they may use different versions of the protocol. PVRTune will test and prevent connections to an incompatible build of SGXPerfServer. Saved files are similarly tied to a particular build of PVRTune and SGXPerfServer.

Note: in situations where the device's CPU is being stressed SGXPerfServer may drop data. The loss of data is represented in PVRTune by a vertical green line. If you experience this then you could try reducing SGXPerfServer's sampling rate as this may improve the situation. Also, you should refrain from physically interacting with your device as doing so may be causing events to be processed in the background that would affect the CPU load. Two examples of 'unexpected' sources of CPU usage can be touch-screen drivers and motion-detection drivers.

## 1.1. Component Structure

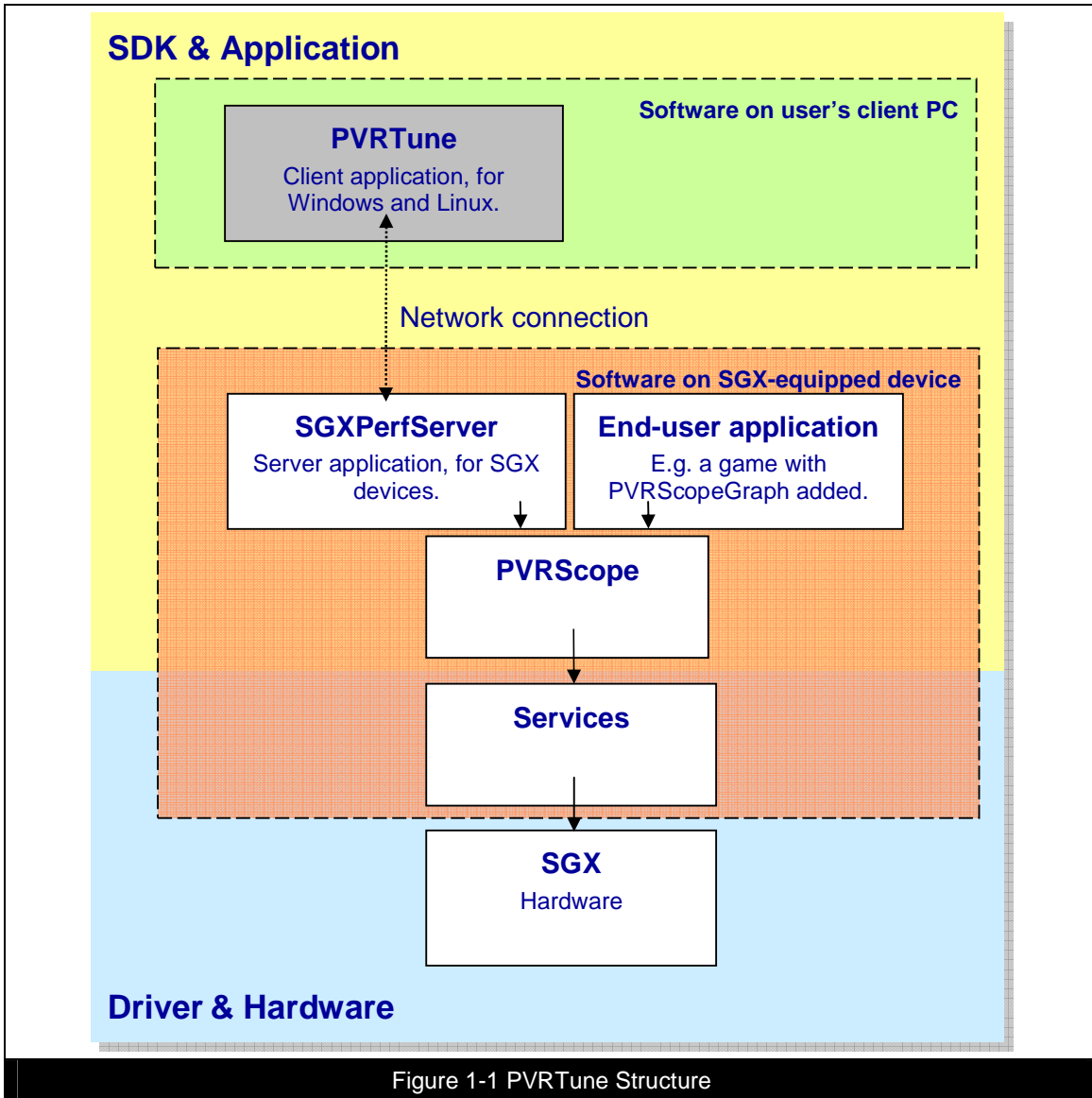


Figure 1-1 PVRTune Structure

Note: only one process, e.g. SGXPerfServer, may use the PVRScope library to access performance data at any given time.

## 2. SGXPerfServer

SGXPerfServer is typically a console application that runs as a server and waits for a client (PVRTune) to connect across the network. Alternatively, it can record data to a file for later loading by PVRTune. There are numerous builds including various Android, Linux, Symbian, Windows and Windows CE platforms.

### 2.1. Installation

#### 2.1.1. Android

Install the “SGXPerfServer.apk” file on the device.

#### 2.1.2. Linux, Neutrino and Windows

The SGXPerfServer executable does not need to be installed to any particular location; it simply needs to be on or available to the computer containing the SGX.

## 2.2. Driver (DDK) Compatibility

Early builds of PVRScope went through a number of variants until the modern interface was settled on. SDK release 2.4 was the first release to support the 1.3c interface.

Interface variant	Compatible driver ranges
"1.3c"	1.3.13.1589 onwards; 1.4.14.593 onwards

### 2.2.1. DDK build options

Support for hardware performance profiling in the drivers is an option that must be enabled by the platform supplier/vendor. Typically, if has been enabled, there will be a driver file called something like "PVRScopeServices.dll" or "libPVRScopeServices.so". If SGXPerfServer fails to initialise, it is possible that support for it may have been removed from the drivers on your platform.

## 2.3. Usage

1. If necessary, start/initialise the drivers; see the DDK documentation for further information.
  - 1.1. On Windows XP, run “init.exe” from the driver package or “net start pvrpdp” as per normal.
  - 1.2. On Linux, run “/etc/init.d/rc.pvr start”
2. Run SGXPerfServer.
  - 2.1. On Android, click on the SGXPerfServer icon located in the Android application menu.
  - 2.2. On Linux, run “SGXPerfServer”.
  - 2.3. On Neutrino, run “LD\_LIBRARY\_PATH=/usr/lib/graphics/omap4430 SGXPerfServer”
  - 2.4. On Windows Mobile:
    - 2.4.1. On some platforms, a single Ethernet connection can be used for both a KITL connection to Platform Builder and for generic networking use on the platform, i.e. it can also be used by SGXPerfServer.
    - 2.4.2. If this does not work, it may be necessary to set up a second IP address for the device over a USB connection (the same one used for ActiveSync).
 

On the desktop PC open the ActiveSync application.

Inside the “File” menu open the “Connection Settings...” menu.

Uncheck the “Allow USB connections” option (this is to avoid connection latencies due to accidental ActiveSync Synchronization activities between the PC and the development platform) then click Ok and close the ActiveSync application window.

Connect the USB cable between the development platform and the PC.

Click the “start” menu on the development platform.

Click on the “settings” icon.

Click on the “connections” icon.

Click on the “USB to PC” icon.

Make sure that the option “Enable advanced network functionality” is checked.

Then back on the “connections” menu click the “Network Cards” option.

Click on the “Remote-NDIS Host” option.

Make sure the option “Use server-assigned IP address” is selected and on that same window write down the IP address that is displayed on the textbox below the option selection (this is the IP address that will be provided to the desktop application).

The IP address, for use when connecting with PVRTune, will likely be 169.254.2.1 but this will depend on your settings.
    - 2.4.3. Run the SGXPerfServer.exe application on the development platform by tapping on the application icon.
    - 2.4.4. Windows Mobile will ask if the user trusts the “SGXPerfServer.exe” application, click yes.
  - 2.5. On Windows, run “SGXPerfServer.exe”. Run the application from a command prompt, so that the window does not close instantly in the case of an error and SGXPerfServer exiting.
3. If successful, you will see console-window output resembling Figure 2-1 SGXPerfServer – Awaiting Connection; GUI builds of the application, for example Android, and Windows Mobile, will have different user interfaces. The eighth line of the output shown in the figure gives you the name and IP address of the server, which may be of use when connecting from PVRTune.

```
SGXPerfServerComplete v1.93 - Build 2.09.29.0602
Copyright (C) Imagination Technologies Ltd. All rights reserved.
* Support:          DevTech@imgtec.com
* OS:              Linux version 2.6.32.9 (root@sgxdevtec) (gcc version 4.3.2)
* Driver build:    1.7.17.4817
* Device:          SGX540 1.0.1
* Processor count: 2
This server is sgxdevtec:6520 (lo:127.0.0.1,eth0:192.168.15.60)...
Waiting for connection (press q to quit)...
196.5ps 3t 0h : 00/00,Tu
```

Figure 2-1 SGXPerfServer – Awaiting Connection

4. When it is not connected, SGXPerfServer will wait for a connection.
5. No more than one client may be attached simultaneously.
6. When SGXPerfServer is connected to a client, it will repeatedly read/sample the performance data from the drivers and send them over the network to PVRTune.
7. Press the 'q' key to quit.

### 2.3.1. Command-Line Options

SGXPerfServer supports several command-line options.

Option	Effect
-h	Show help text.
/?	Show help text.
--disable-user-input	Disable user input. Not available on all operating systems. Use to run SGXPerfServer in the background on Linux consoles.
--disable-hwperf	Disables the use of PVRScope' hardware performance functionality.
-group=N	On start-up, switch the hardware to the specified group number.
-port=N	Network port to use; default is 6520.
-sendto="N"	Instead of using the network, record data directly to file "N"
-t=N	Time, in milliseconds, between counter updates; the default value is 2. This value can be increased to reduce the CPU usage of SGXPerfServer.
-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).

### 2.3.2. Run-time options

SGXPerfServer supports several key-presses at run-time.

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 SGXPerfServer.

## 3. PVRTune

PVRTune is a GUI application that connects as a client to SGXPerfServer across a network, or loads files written by SGXPerfServer; it allows counter and timing data to be viewed remotely and, in the case of a live connection, commands can be sent to the SGX drivers. There are three builds: one each for Linux, Mac and Windows.

The OS that PVRTune is running on does not have to match the OS that SGXPerfServer is running on, they are fully independent.

Multiple copies of PVRTune may be run on the same computer, each attached to a different server, where a “server” is an SGX-equipped computer successfully running SGXPerfServer.

### 3.1. Installation

The PVRTune executable does not need to be installed to any particular location; it simply needs to be on or available to the computer on which it is to be run.

On Windows, if PVRTune is installed using an SDK installer there will be a link in the Start menu. Alternatively the executable can be taken from the ZIP file.

Desktop Linux and Windows installations allow PVRTune to be ran on the same computer as the SGX and SGXPerfServer; your platform may or may not support this. Performance profiling an application is generally simplified by running PVRTune on a second computer, particularly when running full-screen applications on the SGX computer.

### 3.2. Usage

Note that while these instructions utilise the menu options, most menu options are also available elsewhere in the application.

#### 3.2.1. Connecting to a Server

1. Launch PVRTune.
2. Once PVRTune is running, it presents a welcome screen as shown in Figure 3-1 PVRTune Connection Screen.

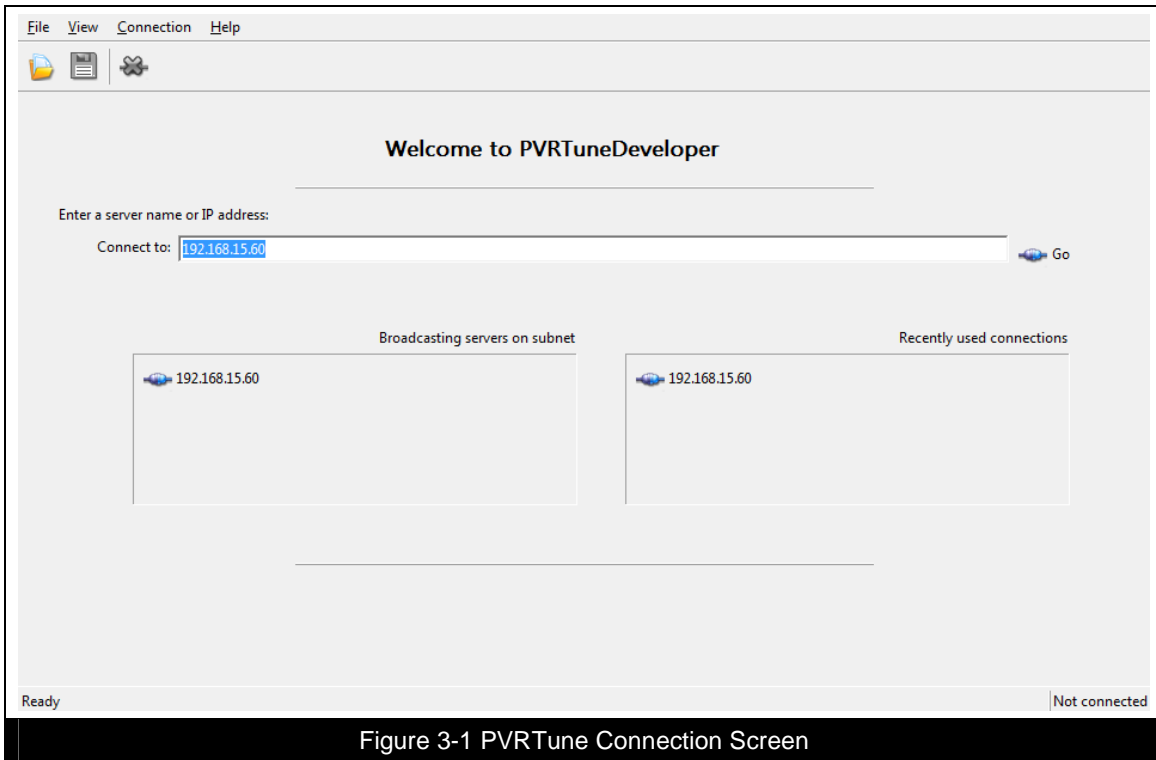


Figure 3-1 PVRTune Connection Screen

3. Connect to SGXPerfServer, using one of the following options:
  - 3.1. “Connection\New...” pops up a dialog box in which the name or IP address of the server can be entered.
  - 3.2. “Connection\localhost” should be used when SGXPerfServer and PVRTune are both running on the same, SGX equipped, computer.
  - 3.3. “Connection\Recent Connections”, or the ‘Recently used connections’ pane, offers a list of previously entered server names and IP addresses. This list will initially be empty.
  - 3.4. “Connect To” can be used to enter a server name or IP address which can be connected to with the “Go” button.
  - 3.5. The “Broadcasting servers on subnet” list can be used to connect to an existing server.
4. On Windows, it may be necessary to disable the built-in Windows Firewall on the server or client computers. This can be done using the “Security Centre” in the Control Panel.
5. Once a connection is successfully made, the PVRTune user interface switches to the graph view, as shown in Figure 3-2 PVRTune Connected.

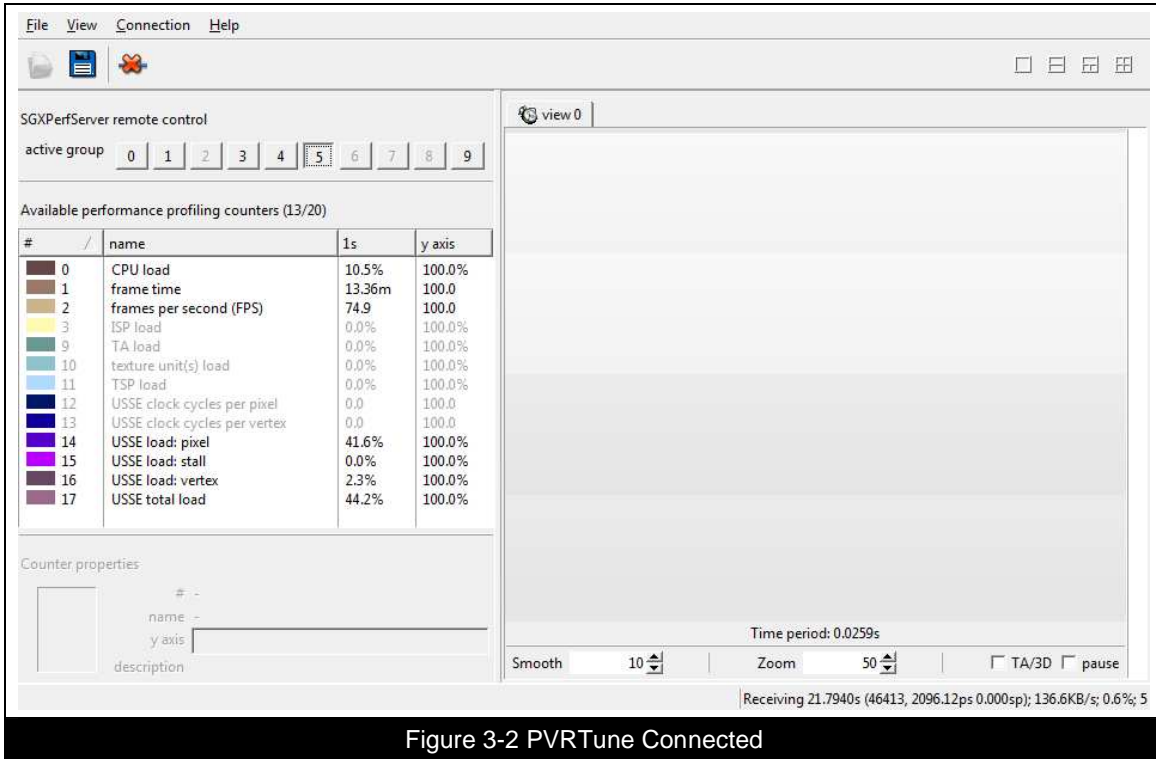


Figure 3-2 PVRTune Connected

### 3.2.2. SGXPerfServer Remote Control

1. The active hardware counter group can be changed using the buttons labelled “0” through “9”; group-buttons with no available counters will be unavailable.
  - 1.1. If connected via the network, changing this sends a command to the SGX to switch the active counter group, which changes the list of available hardware counters.

### 3.2.3. Available performance profiling counters

1. This list shows all the counters that exist.
  - 1.1. Counters are classified as “advanced” or not; right click and toggle “Hide advanced counters” to show/hide these.
2. Counters are greyed out when they are not currently available.
  - 2.1. Right click and select “Hide inactive counters” to toggle the display of unavailable counters.
3. The displayed columns can be changed by right-clicking on the column names and choosing “Select columns...” Figure 3-3 Select Columns illustrates the dialog box that appears.

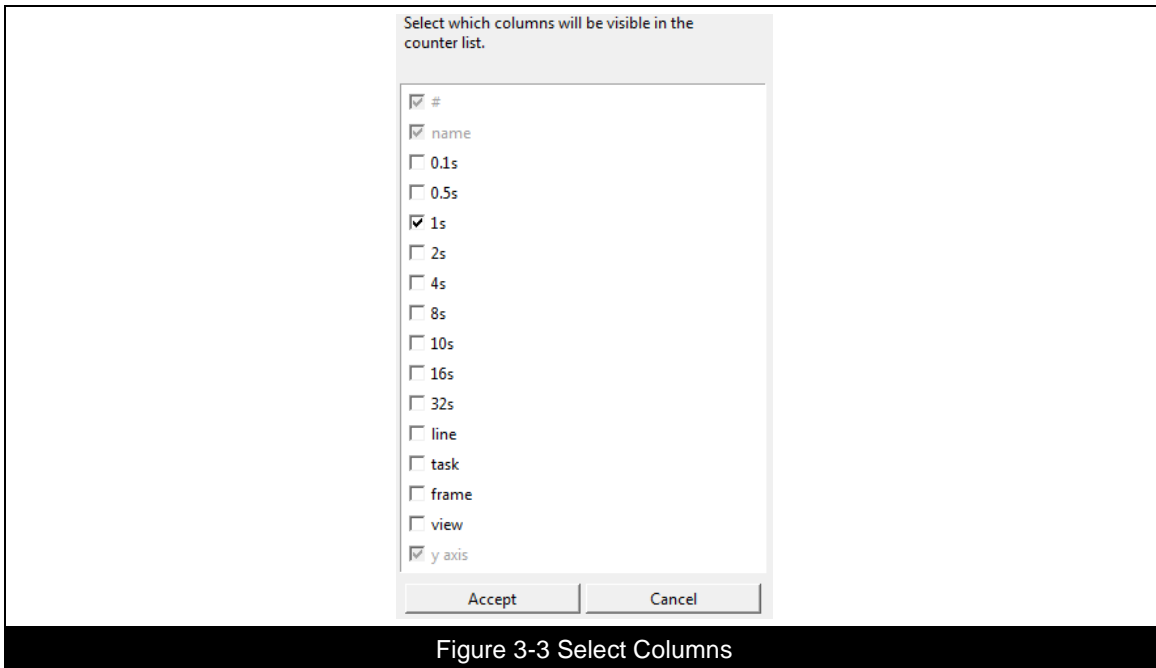


Figure 3-3 Select Columns

- 3.1. The time columns (those with a number followed by an “s” for “seconds”) show the most recent counter values as they arrive over the network, averaged over the specified time period.
- 3.2. The “line” column responds to the mouse pointer when it is moved over a graph; it shows the counter values averaged over the one-pixel wide time period that is under the mouse pointer (i.e. under the vertical line that follows the mouse).
- 3.3. The “task” and “frame” columns respond to the mouse pointer when it is moved over a graph in which TA/3D timing data is enabled.
  - 3.3.1. The “task” column uses the time range defined by the single task under the pointer.
  - 3.3.2. The “frame” column uses the time range defined by the frame number under the pointer; from the start of the first task to the end of the final task of whatever frame number the pointer is over.
- 3.4. The “view” column shows the average counter values across the entire visible range of the graph view that’s under the mouse pointer.
4. The “y axis” column controls the counter scale when it is graphed; the specified value will be at the top of the view.

### 3.2.4. Counter properties

1. When a counter is clicked, the counter properties box, below the counter list, changes to represent that counter.
2. A value for the y-axis scale when the counter is graphed can be typed into the "y-axis" box.
  - 2.1. Note that it will be necessary to enter enough zeroes if the value is very large and the table is showing a shortened form of the value, for example six zeroes for millions (M).
  - 2.2. Note that a fractional number, e.g. 0.01, can be entered for graphs with a very low range, such as frame time.
3. The counter colour can be changed by clicking on the box which shows the current colour.

### 3.2.5. Graph views

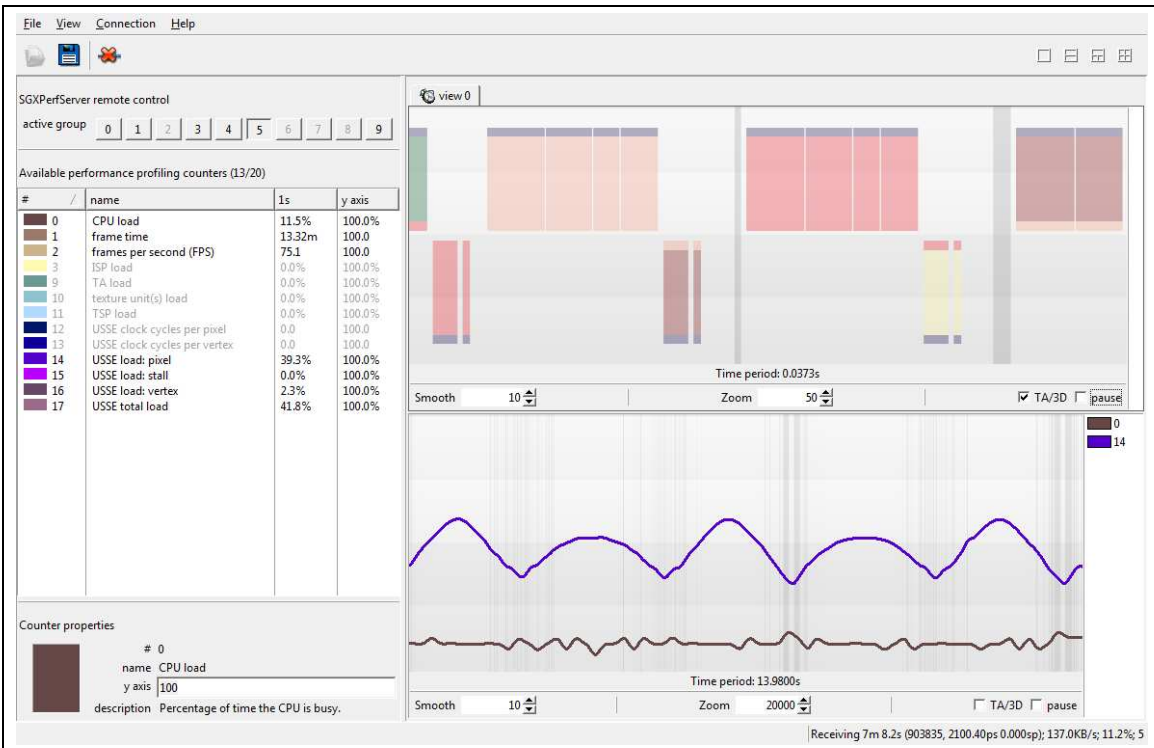


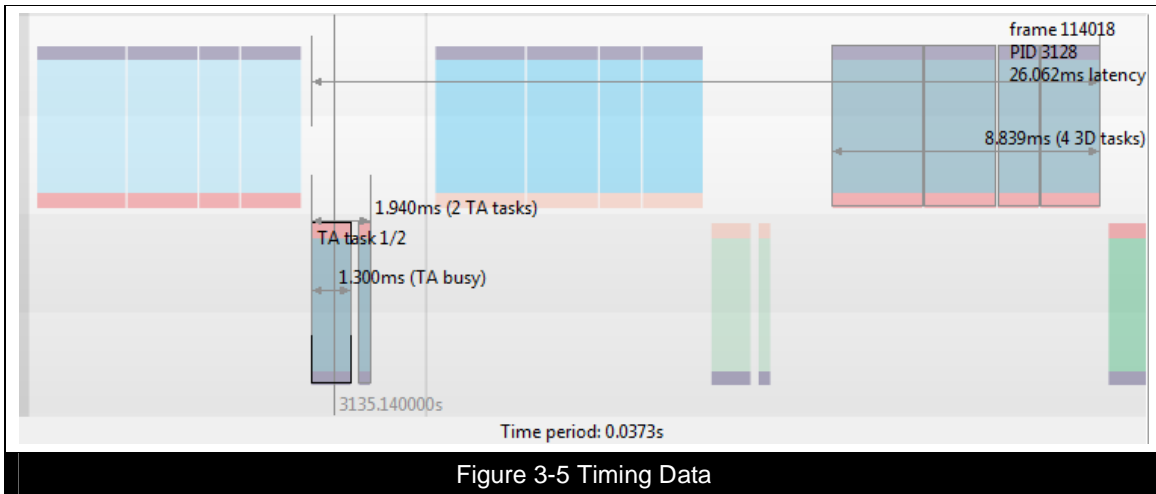
Figure 3-4 Graph Views

#### User interaction

1. On the right hand side of the PVRTune window there can be several tab-pages of graphs.
  - 1.1. Tab-pages can be added, removed and re-named using the options in the File menu.
  - 1.2. Each page can contain some number of graphs. The number of graph windows available can be changed using the View menu or the four icons in the top-right of the PVRTune window.
2. The “pause” tick-box will halt the graph; it will no longer track the live data as it arrives (the data is still received and stored, however).
3. A graph can be panned by clicking and dragging in the graph view; this automatically enables “pause” for that graph.
4. The “Zoom” control changes the number of micro-seconds per pixel; higher numbers effectively zoom out the graph.
  - 4.1. A graph can be more easily zoomed by placing the mouse cursor over the graph and using the mouse-wheel.
5. The “Smooth” control alters the amount of smoothing applied to the counter data. Too-low values give noisy graphs; too-high values smooth out all the detail.
  - 5.1. The smoothing values for a graph can be more easily changed by holding the “Alt” key and using the mouse-wheel.
6. The “time period” value indicates the range of time currently shown in the graph; this is calculated using the current zoom value and the width of the window.
7. Right-clicking on a graph produces a context menu allowing:
  - 7.1. Render of TA/3D data to be toggled on or off.
  - 7.2. Render of “Marks” to be toggled on or off.
  - 7.3. The view to be paused, or tracking most recent data.
  - 7.4. The graph to be saved as a PNG file.

## Timing data

1. The “TA/3D” tick-box (located in the right-click context menu) enables the rendering of timing data for the TA and 3D processing cores. Two rows of coloured blocks will be drawn in the background; the top row is 3D and the bottom row is TA. Colour indicates that the core is active; gaps are left where the core is inactive. TA and 3D tasks belonging to the same frame number are rendered in the same colour, and colours are recycled every sixteen frames.
  - 1.1. If there are idle gaps between activity blocks on both the 3D and TA activity lines, the app is not SGX limited and performance improvement will come from optimising the application.
  - 1.2. If there are idle gaps in the TA line, but very small or no gaps in the 3D line, the application frame-rate is being limited by the render.
  - 1.3. If there are idle gaps in the 3D line, but very small or no gaps in the TA line, the application frame-rate is being limited by the vertex transforms and triangle clipping.



2. If the mouse pointer is held over the timing data more information will be presented, as shown in Figure 3-5 Timing Data. Four rulers are drawn.
  - 2.1. Time-length, type of, ordinal of and number of activity blocks of the same type for the same frame, for the activity block under the mouse.
  - 2.2. Time-length and count of all the 3D activities for the frame. If HW limited, this may be slightly less than the reciprocal of the frame rate.
  - 2.3. Time-length and count of all the TA activities for the frame. If HW limited, this may be slightly less than the reciprocal of the frame rate.
  - 2.4. Frame number and time-length from the first TA activity, to the end of the 3D activity. Note that this is not the reciprocal of the frame rate.

### Counter data

1. All the existing counters are listed on the left hand side of the PVRTune window; they are both software and hardware counters.
  - 1.1. Each counter in the list shows a name, the current value over a range of available time periods, an “axis” value. When a counter is graphed, it is scaled such that the y-axis value will be at the top of the graph window.
  - 1.2. The right-click context menu can be used to hide counters if they are currently inactive, or are “advanced.”
2. Each graph has a list of counters on its right-hand side.
3. Counters can be copied from the list on the left, or moved from any graph to any other graph, using the mouse and “drag and drop”.
4. Counters can be deleted from a graph by selecting them in the attached list and pressing “delete” on the keyboard, or by right-clicking the list and using the context menu.
5. Counters that are on the graph have their name and value shown numerically at the point where the graph crosses the vertical line under the mouse pointer.
6. When the mouse point is hovering over a counter, the PVRTune status bar shows the name of that counter.

### 3.2.6. Data Save and Load

1. “File\Save” saves all the data received so far and closes the file; it does not continue saving data as it is received.
2. “File\Load” can load binary files written from PVRTune or from SGXPerfServer.

### 3.2.7. Miscellany

1. PVRTune will automatically disconnect when approximately 1GB of data has been received.
2. When PVRTune is connected to a server, “Connection\Close” disconnects from the server, and keeps the received data on screen for the user to study.
3. When PVRTune is no longer connected to a server, “Connection\Close” discards the data from memory and returns to the welcome screen.
4. The bottom-right of the window displays some status information.
  - 4.1. These displays are for diagnostics and should be ignored.
  - 4.2. The values are: what PVRTune is currently doing, the time period for which it has been doing it, the number of times it has done it, the average number of times it is doing it per second, and the average time period between each occasion that it is done; the latter two values are the reciprocal of each other. The final three values are: the data-receive rate, the progress towards auto-disconnection due to memory usage as a percentage, and the active group.

## 4. How to: analyse an application for performance bottlenecks

PVRTune can be used to help achieve many possible aims; three are listed here.

- Improving frame rate to increase user enjoyment and improve feedback.
- Reducing render time, but not increase frame-rate, in order to have more idle time and thereby to save power.
- Increasing the visual quality without decreasing frame rate.

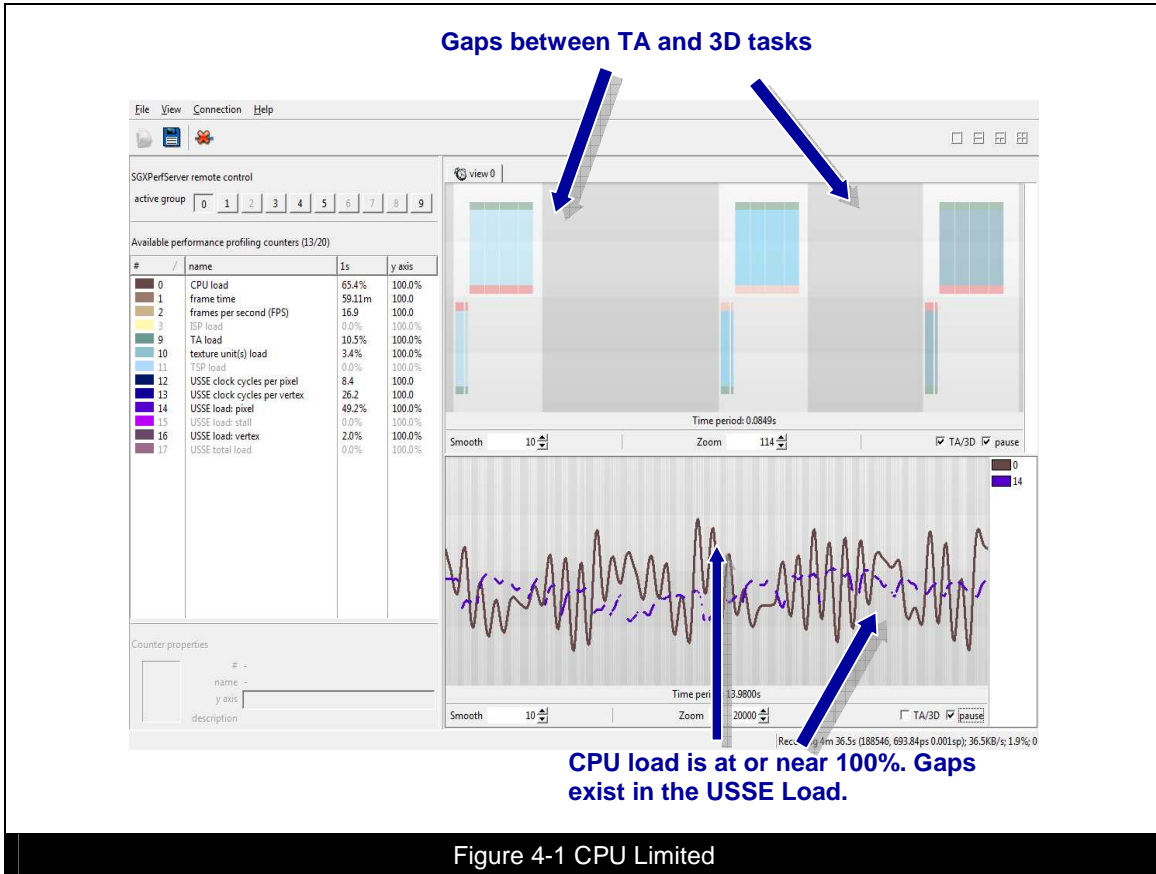
The following guidelines are written with the aim of satisfying the first of those possible goals.

### 4.1. Setup

1. Set up your device, plug it into the network, install SGX drivers and ensure that applications are working.
2. On the device, start/initialise the drivers if necessary then run SGXPerfServer. This is the server application that will send data to your development PC.
3. Next, run the application of interest on the device. This application must use the SGX for rendering, for example by using one of the following API's: OpenGL ES 2, OpenGL or OpenVG.
4. On your development PC, run PVRTune. Use the "File\Connect" menu, and then enter the name or IP address of the device that's running both SGXPerfServer and the program of interest. (Most SGXPerfServer builds will have automatically output the device name and IP address.)
5. Once the connection is made, PVRTune will switch to the graph view.

### 4.2. CPU or SGX limited?

1. The first thing to check is where the application is limited. Enable the TA/3D data and check for idle gaps between the activity blocks.
2. If both of the TA and 3D timing data rows have significant gaps, the application is CPU limited; this should most likely be addressed before further analysis of SGX behaviour.
  - 2.1. Most builds of SGXPerfServer include a "CPU Usage" counter; this is useful for helping diagnose CPU-limited scenarios.
  - 2.2. It might mean that v-syncs are enabled, limiting application frame rate to the refresh-rate of the display. V-syncs can be disabled using, for example, `eglSwapInterval()`.
  - 2.3. It might mean that the application is causing a "stall" – the application has done something that obliges the CPU to wait for SGX to finish a task, with a knock-on effect of later causing SGX to idle, waiting for the CPU to finish submitting draw commands.
  - 2.4. The application might be explicitly sleeping for some time period, or waiting for some other device, such as networking or a file-system.



3. If either activity line is solidly filled, without idle gaps, the application performance is likely limited by SGX.
4. If there are large gaps between the TA or the 3D activity blocks, but not in the other line, performance will be improved by reducing the work-load in whichever line is fully busy
  - 4.1. Alternatively, it may be possible to improve visual quality in the application by utilising the idle time in the partially-idle core.
  - 4.2. Note that some parts of the system are shared by both cores, for example system memory bandwidth and the unified shader engine; changing the workload for these shared resources will affect both cores.

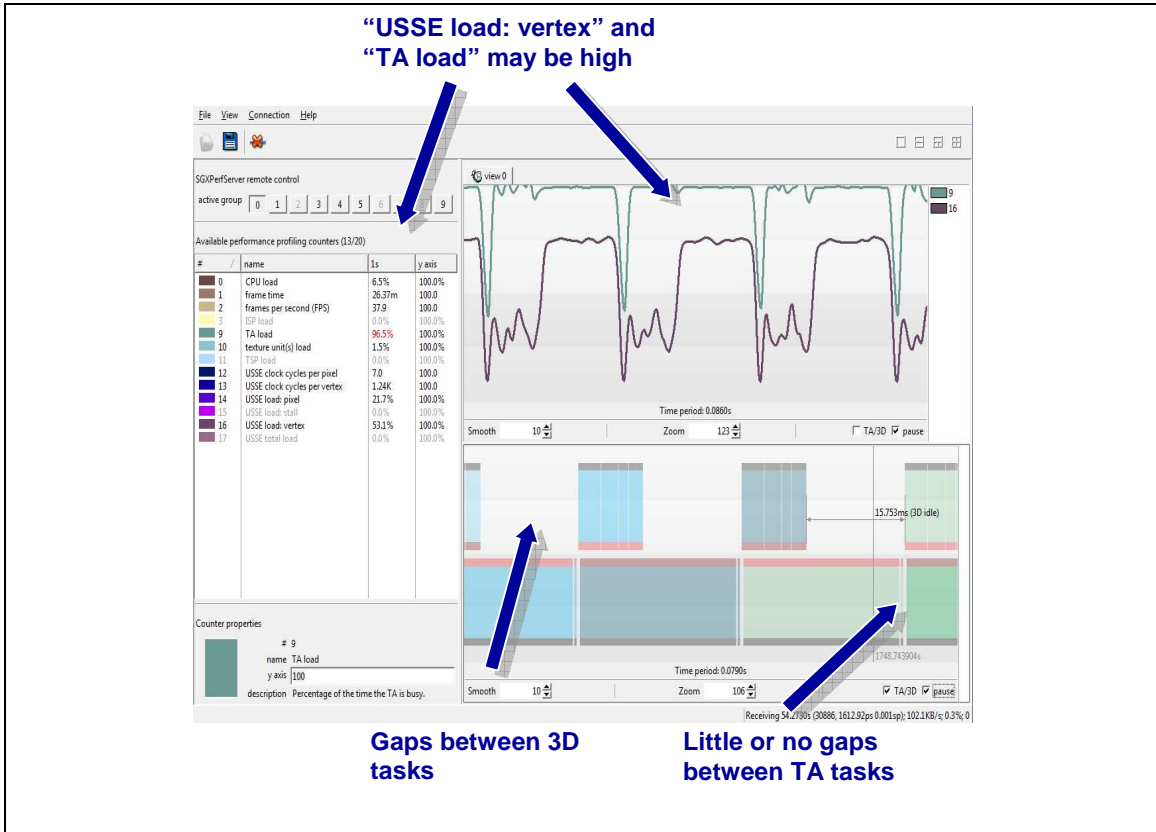


Figure 4-2 Vertex Limited

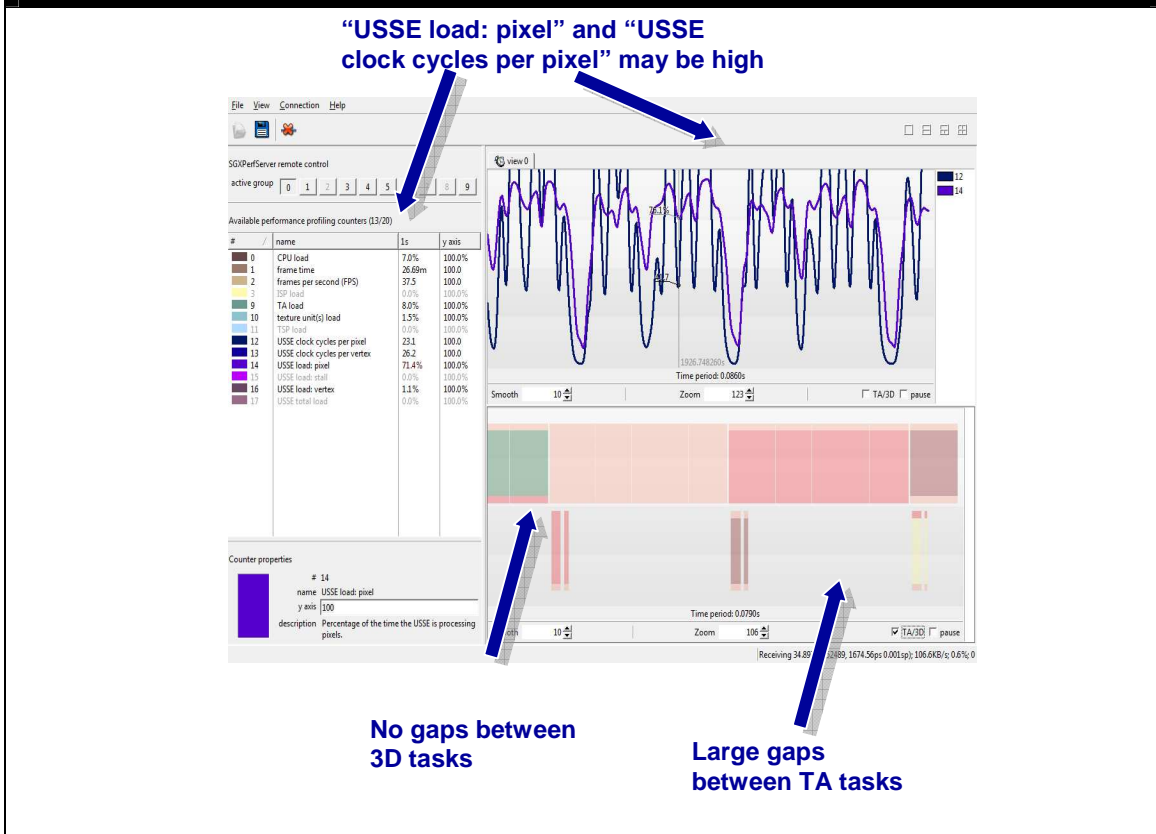


Figure 4-3 Fragment Limited

### 4.3. Where is my time going?

If it is concluded that one or other core in SGX is fully loaded, it is typically then desirable to find out where the time is being spent. That is the purpose of the counter data.

All the available counters are listed on the left-hand side of the PVRTune window. There are software counters and hardware counters; the latter are split into multiple groups, only one group of which can be active at any point in time. Switching between the groups (using the “0” through “9” buttons in the “SGXPerfServer Remote Control” part of the PVRTune window) sends a command over the network to switch the active hardware counter group.

The list of counters on the left-hand side shows numerically the current value of the graph, averaged over a specified time period; if a historical view is required, drag-and-drop the counter of interest onto a graph and if necessary set a value for the counter’s y-axis scale.

1. Group 0 and group 9 are “overview” groups; group 9 is only available in some revisions of SGX. Typically you should start with these groups to see which hardware modules are under high load. If something is identified, or even if it isn’t, the other counter groups can be used to see more details for various parts of the SGX hardware.
2. Group 1 is the primitive clipping counter group. It contains counters covering the clipper, and the numbers of triangles/vertices written from the TA.
3. Group 2 is the scene buffer counter group, with tiling and parameter buffer statistics.
4. Group 3 contains statistics covering the ISP.
5. Group 4 contains statistics covering the TSP.
6. Group 5 shows the activity on the USSE, broken down into vertex, pixel and stall cycles.
7. Group 6 is the texturing counter group, including the [shared] cache miss rate and the activity of the texture unit(s).

#### 4.3.1. FAQ: why is no counter showing 100% load?

In point tests that stress a single component of SGX, the sustained load on individual parts of the chip can approach 100%. However in normal behaviour, for fractions of a second various units are at 100% load and in the relatively low-resolution graph presented by PVRTune this will be “smoothed out” and therefore seen as lower utilisations. Also, if many units are accessing memory simultaneously then available system memory bandwidth limits can slow all operations on SGX; this is device specific and therefore there is no SGX counter that covers it.

The recommended approach is to use PVRTune counters to look for parts of the chip that have very low loads and use this to improve quality, or look for those parts that have high loads and change the application to reduce this to improve performance. There will not in all cases be a counter that approaches 100%.

## 5. Related Materials

### Software

- PVRTrace

### Documentation

- PVRTrace User Manual

## 6. Contact Details

For further support contact:

[devtech@imgtec.com](mailto:devtech@imgtec.com)

PowerVR Developer Technology  
Imagination Technologies Ltd.  
Home Park Estate  
Kings Langley  
Herts, WD4 8LZ  
United Kingdom

Tel: +44 (0) 1923 260511

Fax: +44 (0) 1923 277463

Alternatively, you can use the PowerVR Insider forums:

[www.imgtec.com/forum](http://www.imgtec.com/forum)

For more information about PowerVR or Imagination Technologies Ltd. visit our web pages at:

[www.imgtec.com](http://www.imgtec.com)

## Appendix A. Counter list

Note that hardware counters are exactly that – the information they supply is about hardware activity. If there are multiple applications using the hardware (or a single application and an OS doing hardware-accelerated composition), the stats will not be per- process/thread/context/program.

counter	description
TA load	Percentage of the time the TA is busy.
ISP load	Percentage of the time the ISP is busy. (Maximum of tri- and pixel-loads.)
TSP load	Percentage of the time the TSP is busy. (Maximum of tri- and pixel-loads.)
USSE load: vertex	Percentage of the time the USSE is processing vertices.
USSE load: pixel	Percentage of the time the USSE is processing pixels.
USSE load: stall	Percentage of the time the USSE has instructions to process but it stalled.
USSE total load	Percentage of the time the USSE is processing instructions (does not include stall cycles).
texture unit(s) load	Percentage of the time the texture units are busy.
CPU load	Percentage of time the CPU is busy.
frames per second (FPS)	Average number of frames rendered and presented per second.
frame time	Time to render frame.
USSE clock cycles per vertex	Average number of USSE clock cycles spent processing each vertex.
vertex transforms per second	Number of vertices transformed by the USSE per second.
vertex transforms per frame	Number of vertices transformed by the USSE per frame.
USSE clock cycles per pixel	Average number of USSE clock cycles spent processing each pixel.
on-screen primitives per second	Number of primitives that are on-screen and written to the parameter buffer, per second.
on-screen vertices per second	Number of vertices that are on-screen and written to the parameter buffer, per second.
on-screen primitives per frame	Number of primitives that are on-screen and written to the parameter buffer, per frame.
on-screen vertices per frame	Number of vertices that are on-screen and written to the parameter buffer, per frame.
on-screen vertices per primitive	Measures transform efficiency; polygon sorting can improve (reduce) this number. Also affects amount of parameter buffer memory required.