SGX Services

Unit Test Notes

Filename	:	SGX Services.Unit Test Notes.doc
Version	:	4.0.20 External Issue
Issue Date	:	08 Feb 2011
Author	:	PowerVR

Contents

List of Figures

1. Introduction

This document provides technical notes for each of the Services unit tests supplied with the DDK.

2. services_test

The test is designed to illustrate the use of API calls to the services device driver. The Test first establishes a connection with Services and then, via an array of API calls, obtains details of the services available and what devices are available. This test enables and initialises an interface to services and attached devices but hardware operations are not submitted to the devices themselves.

The unit tests can only run if they have exclusive access to the framebuffer. Where there is a Services Manager or Compositor managing the framebuffer, the test might fail or produce corrupt results.

To execute the test, use the following command line:

```
> sevices_test
```

2.1. Test log

```
---------------------- Start ----------------------
Try calling PVRSRVConnect with an invalid argument:
 OK
Call PVRSRVConnect with a valid argument:
 OK
Try calling PVRSRVEnumerateDevices with invalid puiNumDevices:
 OK
Get number of devices from PVRSRVEnumerateDevices:
 OK
Reported 1 devices
Device Number  | Device Type
           0000    | PVRSRV_DEVICE_ID_SGX
Attempt to acquire device 0:
 OK
Getting SGX Client info
 OK
ui32ProcessID:2616
Display Class API: enumerate devices
 OK
PVRSRVEnumerateDeviceClass() returns 1 display device(s)
 OK
Attempt to create memory context for SGX:
 OK
Display Class API: open device
 OK
Display Class API: Get display info
 OK
Name:PVRPDP
MaxSwapChains:1
MaxSwapChainBuffers:3
MinSwapInterval:0
MaxSwapInterval:10
Display Class API: enumerate display formats
 OK
 OK
Display format 0 - Pixelformat:20
Display Class API: enumerate display dimensions
 OK
 OK
Display dimensions 0 - ByteStride:2560 Width:640 Height:480
Display Class API: get the system (primary) buffer
 OK
Shared heap 0 - HeapID:0x7000000 DevVAddr:0x400000 Size:0x77800000 Attr:0x4014200
...
Shared heap 11 - HeapID:0x700000c DevVAddr:0xd0000000 Size:0x7c00000 Attr:0x4014200
Display Class API: map display surface to SGX
 OK
Attempt to create rendering context for SGX:
 OK
Attempt to add render target for SGX:
 OK
gather some mem stats:
 OK

Local Backing Store:

Arena 'TestChipDeviceMemory':
  allocCB=F74DED50 freeCB=00000000 handle=00000000 quantum=4096
span count              1
live segment count      30
free segment count      0
free resource count     228265984 (0xd9b1000)
total allocs            70
total frees             40
import count            0
export count            0
  segment Chain:
        base=0xa0000000 size=0x1000 type=live ref=00000000
        …
        base=0xa064f000 size=0xd9b1000 type=free ref=00000000
```

```
Device Type 8:

Device Type 7:

Kernel Context:

Arena '2D BS':
  allocCB=F74D3BC7 freeCB=F74D4173 handle=E1954628 quantum=4096
span count              0
live segment count      0
free segment count      0
free resource count     0 (0x0)
total allocs            0
total frees             0
import count            0
export count            0
  segment Chain:
...
Arena 'General':
  allocCB=F74DED50 freeCB=00000000 handle=E26EC110 quantum=4096
span count              1
live segment count      0
free segment count      0
free resource count     2004877312 (0x77800000)
total allocs            0
total frees             0
import count            0
export count            0
  segment Chain:
        base=0x400000 size=0x77800000 type=free ref=00000000

Attempt to remove render target:
 OK
Attempt to destroy render context:
 OK
Display Class API: unmap display surface from SGX
 OK
Attempt to destroy memory context for SGX:
 OK
Display Class API: close the device
 OK
SGXReleaseClientInfo:
 OK
PVRSRVDisconnect:
 OK
----------------------End loop 1----------------------
```

3. sgx_init_test

The objective of this test is similar to the `services_test` but with a reduced set of API calls, for example, a render context and a render target are not created.

The test queries the driver components (host driver, microkernel) for debugging purposes and reports the following information:

- SGX host driver DDK version (e.g. 1.6.16.204)
- SGX hardware core version and revision
- SGX microkernel DDK version (which should match that of the host driver)
- SGX microkernel core revision used for build (which should match the HW revision or 0 if the RTL head is used)
- List of enabled SGX microkernel build options (not exhaustive). A brief explanation of these options is provided in the SGX Services Porting Guide.

To execute the test, use the following command line:

```
> sgx_init_test
```

3.1. Test log

```
PVR:
PVR: Memory Stats
PVR: ------------
PVR:
PVR: High Water Mark = 0 bytes
----------------------- Start -----------------------
Try calling PVRSRVConnect with an invalid argument:
PVR:(Error): PVRSRVConnectServices: Invalid parameter [272,
/home/thomas.morris/img dev/eurasia/services4/srvclient/bridged/bridged pvr glue.c]
PVR:(Error): PVRSRVConnect: Unable to open connection. [318,
/home/thomas.morris/img dev/eurasia/services4/srvclient/bridged/bridged pvr glue.c]
 OK
Call PVRSRVConnect with a valid argument:
 OK
Try calling PVRSRVEnumerateDevices with invalid puiNumDevices:
PVR:(Error): PVRSRVEnumerateDevices: Invalid params [447,
/home/thomas.morris/img dev/eurasia/services4/srvclient/bridged/bridged pvr glue.c]
 OK
Get number of devices from PVRSRVEnumerateDevices:
 OK
.... Reported 1 devices
.... Device Number  | Device Type
          0000    | PVRSRV DEVICE ID SGX
Attempt to acquire device 0:
 OK
Getting SGX Client info
 OK
.... ui32ProcessID:2328
Display Class API: enumerate devices
 OK
PVRSRVEnumerateDeviceClass() returns 1 display device(s)
 OK
Attempt to create memory context for SGX:
 OK
Display Class API: open device
 OK
Display Class API: Get display info
 OK
.... Name:PVRPDP
.... MaxSwapChains:1
.... MaxSwapChainBuffers:3
.... MinSwapInterval:0
.... MaxSwapInterval:10
Display Class API: enumerate display formats
 OK
 OK
.... Display format 0 - Pixelformat:20
Display Class API: enumerate display dimensions
 OK
 OK
.... Display dimensions 0 - ByteStride:2560 Width:640 Height:480
Display Class API: get the system (primary) buffer
 OK
.... Shared heap 0 - HeapID:0x7000000 DevVAddr:0x1800000 Size:0x6fff000 Attr:0x4014200
.... Shared heap 1 - HeapID:0x7000001 DevVAddr:0xc800000 Size:0xfff000 Attr:0x4024200
.... Shared heap 2 - HeapID:0x7000002 DevVAddr:0xe400000 Size:0x7f000 Attr:0x4024200
.... Shared heap 3 - HeapID:0x7000003 DevVAddr:0xf000000 Size:0x3ff000 Attr:0x4024200
.... Shared heap 4 - HeapID:0x7000004 DevVAddr:0xf400000 Size:0x4ff000 Attr:0x4014200
.... Shared heap 5 - HeapID:0x7000005 DevVAddr:0xfc00000 Size:0x1ff000 Attr:0x4014200
.... Shared heap 6 - HeapID:0x7000006 DevVAddr:0xdc00000 Size:0x7ff000 Attr:0x4014200
.... Shared heap 7 - HeapID:0x7000007 DevVAddr:0xe800000 Size:0x7ff000 Attr:0x4014200
.... Shared heap 8 - HeapID:0x7000008 DevVAddr:0xd800000 Size:0x3ff000 Attr:0x4024200
.... Shared heap 9 - HeapID:0x7000009 DevVAddr:0x8800000 Size:0x3fff000 Attr:0x4014200
Display Class API: map display surface to SGX
 OK
Misc Info API: Query the SGX features from host driver
 OK
.... SGX Host driver DDK version: 1.6.16.204
Misc Info API: Query the SGX features from microkernel
 OK
.... Hardware core designer: 0, HW core revision: 1.0.1
```

```
.... Hardware core ID: 276, name: SGX 540
.... SGX microkernel DDK version: 1.6.16.204
.... SGX microkernel software core ID: SGX 540, revision: 65537
SGX microkernel build options
.... DEBUG: enabled
.... PDUMP: enabled
.... PVRSRV_USSE_EDM_STATUS_DEBUG: disabled
.... SUPPORT_HW_RECOVERY: enabled
.... PVR SECURE HANDLES: enabled
.... SGX BYPASS SYSTEM CACHE: enabled
.... SGX DMS AGE ENABLE: disabled
.... SGX FAST DPM INIT: disabled
.... SGX_FEATURE_DCU: disabled
.... SGX_FEATURE_MP: disabled
.... SGX FEATURE MP CORE COUNT: 1
.... SGX FEATURE MULTITHREADED UKERNEL: disabled
.... SGX FEATURE OVERLAPPED SPM: disabled
.... SGX FEATURE SYSTEM CACHE: enabled
.... SGX_SUPPORT_HWPROFILING: disabled
.... SUPPORT_ACTIVE_POWER_MANAGEMENT: enabled
.... SUPPORT DISPLAYCONTROLLER TILING: disabled
.... SUPPORT PERCONTEXT PB: enabled
.... SUPPORT SGX HWPERF: enabled
.... SUPPORT SGX MMU DUMMY PAGE: disabled
.... SUPPORT_SGX_PRIORITY_SCHEDULING: enabled
.... USE_SUPPORT_NO_TA3D_OVERLAP: disabled
Display Class API: unmap display surface from SGX
 OK
Attempt to destroy memory context for SGX:
 OK
Display Class API: close the device
 OK
SGXReleaseClientInfo:
 OK
PVRSRVDisconnect:
 OK
--------------------End loop 1---------------------
```

4. sgx_blit_test

4.1. Technical Overview

This test performs two source copy blits to the primary surface (one to the top-left and one to the bottom right quadrants of the primary surface).

The HW dithering capability in the PBE is tested by performing a colour downsampling blit to the bottom right quadrant of a red gradient to RGB565 colour space, which is restricted to 32 unique shades of red. Banding can be observed if dithering is disabled in the test.

4.2. Test execution and success confirmation

To execute the test, use the following command line:

```
> sgx_blit_test
```

The test should produce output as shown below :

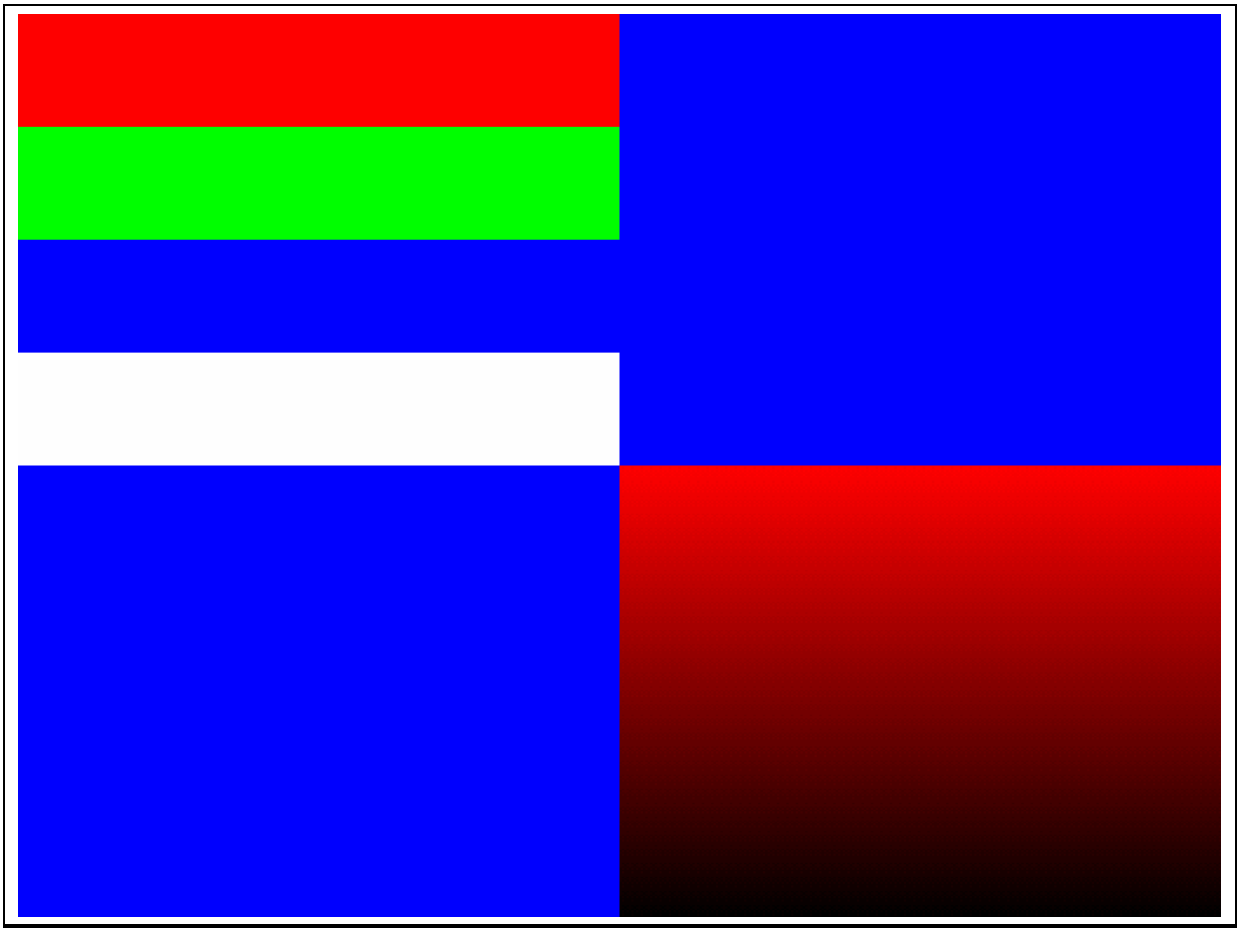

Figure 1 Blit Test Output Frame

4.3. Test Log

```
------------------ SGX 3D Blit test -----------------
----------------------- Start -----------------------
Call PVRSRVConnect with a valid argument:
 OK
Get number of devices from PVRSRVEnumerateDevices:
 OK
.... Reported 1 devices
.... Device Number  | Device Type
            0000    | PVRSRV DEVICE ID SGX
Attempt to acquire device 0:
 OK
Getting SGX Client info
 OK
.... ui32ProcessID:2574
Display Class API: enumerate devices
 OK
PVRSRVEnumerateDeviceClass() returns 1 display device(s)
 OK
Display Class API: open device
 OK
Display Class API: Get display info
 OK
.... Name:PowerVR Linux Framebuffer Driver
.... MaxSwapChains:0
.... MaxSwapChainBuffers:0
.... MinSwapInterval:0
.... MaxSwapInterval:0
Display Class API: enumerate display formats
 OK
 OK
.... Display format 0 - Pixelformat:20
Display Class API: enumerate display dimensions
 OK
 OK
.... Display dimensions 0 - ByteStride:4096 Width:1024 Height:768
Attempt to create memory context for SGX:
 OK
.... Shared heap 0 - HeapID:0x7000000 DevVAddr:0x10000000 Size:0xc1fff000 Attr:0x2014200
.... Shared heap 1 - HeapID:0x7000001 DevVAddr:0xe2000000 Size:0xcfff000 Attr:0x2024200
.... Shared heap 2 - HeapID:0x7000002 DevVAddr:0xf2000000 Size:0x7f000 Attr:0x2024200
.... Shared heap 3 - HeapID:0x7000003 DevVAddr:0xf4000000 Size:0x4fff000 Attr:0x2024200
.... Shared heap 4 - HeapID:0x7000004 DevVAddr:0xf9000000 Size:0x4ff000 Attr:0x2014200
.... Shared heap 5 - HeapID:0x7000005 DevVAddr:0xfe000000 Size:0x1ff000 Attr:0x2014200
.... Shared heap 6 - HeapID:0x7000006 DevVAddr:0xf0000000 Size:0x1fff000 Attr:0x2014200
.... Shared heap 7 - HeapID:0x7000007 DevVAddr:0xf2400000 Size:0x1bff000 Attr:0x2014200
.... Shared heap 8 - HeapID:0x7000008 DevVAddr:0xef000000 Size:0xfff000 Attr:0x2024200
.... Shared heap 9 - HeapID:0x7000009 DevVAddr:0xd2000000 Size:0xffff000 Attr:0x2014200
.... Shared heap 10 - HeapID:0x700000a DevVAddr:0x8000000 Size:0x7fff000 Attr:0x2024000
.... Shared heap 11 - HeapID:0x700000b DevVAddr:0x100000 Size:0x7eff000 Attr:0x2014200
Display Class API: get the system (primary) buffer
 OK
Display Class API: map display surface to SGX
 OK
Attempt to create transfer context for SGX:
 OK
 OK
Do a SRCCOPY blit via 2DCore to the centre of the display:
(centre should be Green on blue background):
 OK
Do a SRCCOPY blit to the bottom right quadrant of the display:
(bottom right quadrant should be red on blue background):
 OK
 OK
Do a SRCCOPY blit to the top left quadrant of the display:
(top left quadrant should be striped (r/g/b/w) on blue background):
 OK
 OK
Do a SRCCOPY blit with COLOUR DOWNSAMPLING from ARGB8888 to RGB565
and then present the RGB565 to the bottom right quadrant of the screen
(bottom right quadrant should be a red gradient):
 OK
 OK
 OK
 OK
```

```
Free the off screen surfaces:
 OK
 OK
 OK
 OK
 OK
Destroy the transfer context:
 OK
Display Class API: unmap display surface from SGX
 OK
Destroy Device Memory Context
Display Class API: close the device
 OK
Release SGX Client Info:
 OK
Disconnect from services:
 OK
------------------ SGX 3D Blit test -----------------
------------------------ End ------------------------
```

5. sgx_flip_test

5.1. Technical Overview

This test uses a swap chain to flip between coloured surfaces. Flips between surfaces should happen during the display's vertical sync period in order to prevent any visual artefacts.

5.2. Test execution and success confirmation

To execute the test, use the following command line:

```
> sgx_flip_test
```

When running correctly, the test will cycle between surfaces which are uniformly coloured red, green and blue. It should not be possible to observe any visual artefacts while running, specifically there should be no "tear lines" visible on the display.

5.3. Test Log

```
---------------------- Start -----------------------
Call PVRSRVConnect with a valid argument:
 OK
Get number of devices from PVRSRVEnumerateDevices:
 OK
Reported 1 devices
Device Number  | Device Type
            0000    | PVRSRV DEVICE ID SGX
Attempt to acquire device 0:
 OK
Getting SGX Client info
 OK
ui32ProcessID:3032
Display Class API: enumerate devices
 OK
PVRSRVEnumerateDeviceClass() returns 1 display device(s)
 OK
Attempt to create memory context for SGX:
 OK
Display Class API: open device
 OK
Display Class API: Get display info
 OK
Name:PVRPDP
MaxSwapChains:1
MaxSwapChainBuffers:3
MinSwapInterval:0
MaxSwapInterval:10
Display Class API: enumerate display formats
 OK
 OK
Display format 0 - Pixelformat:20
Display Class API: enumerate display dimensions
 OK
 OK
Display dimensions 0 - ByteStride:2560 Width:640 Height:480
Display Class API: get the system (primary) buffer
 OK
Shared heap 0 - HeapID:0x7000000 DevVAddr:0x400000 Size:0x77800000  Attr:0x4014200
…
Shared heap 11 - HeapID:0x700000c DevVAddr:0xd0000000 Size:0x7c00000 Attr:0x4014200
Display Class API: map display surface to SGX
 OK
 OK
 OK
 OK
 OK
 OK
waiting for flips to complete...
Surface 0, read ops complete: 59, pending 67
Surface 0, read ops complete: 67, pending 67
Surface 1, read ops complete: 66, pending 66
Surface 1, read ops complete: 66, pending 66
Surface 2, read ops complete: 66, pending 66
Surface 2, read ops complete: 66, pending 66
Display Class API: unmap swapchain display surfaces display surface from SGX
 OK
 OK
 OK
 OK
Display Class API: unmap display surface from SGX
 OK
Attempt to destroy memory context for SGX:
 OK
Display Class API: close the device
 OK
SGXReleaseClientInfo:
 OK
PVRSRVDisconnect:
 OK
```

6. sgx_render_flip_test

6.1. Technical Overview

This test renders a set of overlapping equilateral triangles rotating around their centre, with flipping between successive frames happening during the display's vertical sync period in order to prevent any visual artefacts.

6.2. Test execution and success confirmation

To execute the test, use the following command line:

```
> sgx_render_flip_test
```

The test should produce output as shown in the following 2 frames:

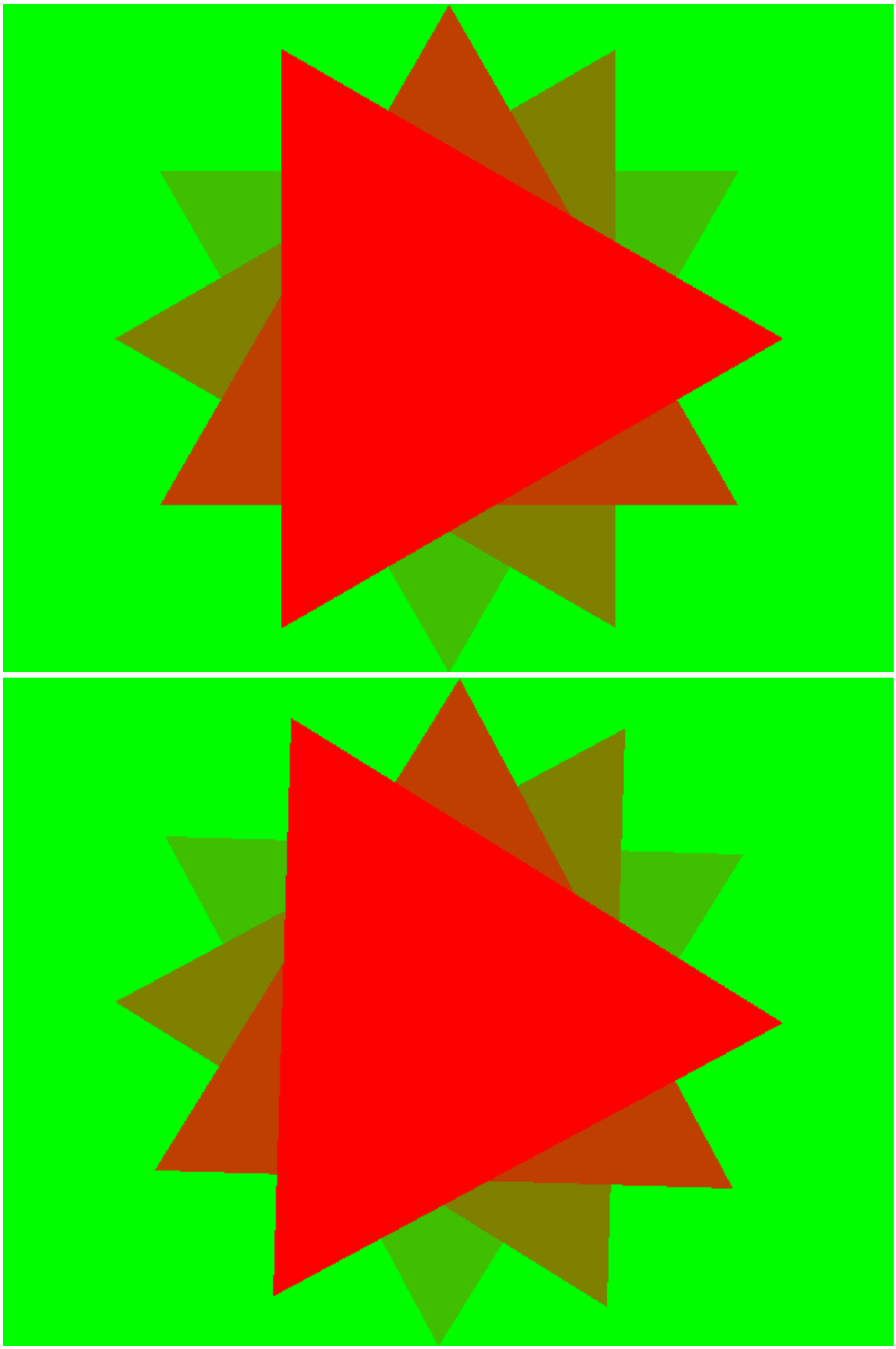

Figure 2 Render Flip Test Output Frames

It should not be possible to observe any visual artefacts while running, specifically there should be no “tear lines” visible on the display.

6.3. Test Log

```
---------------------- Start -----------------------
Call PVRSRVConnect with a valid argument:
 OK
Get number of devices from PVRSRVEnumerateDevices:
 OK
       Reported 1 devices
       Device Number  | Device Type
            0000    | PVRSRV DEVICE ID SGX
Attempt to acquire device 0:
 OK
Getting SGX Client info
 OK
       ui32ProcessID:2508
Display Class API: enumerate devices
 OK
PVRSRVEnumerateDeviceClass() returns 1 display device(s)
 OK
Display Class API: open device
 OK
Display Class API: Get display info
 OK
       Name:PVRPDP
       MaxSwapChains:1
       MaxSwapChainBuffers:3
       MinSwapInterval:0
       MaxSwapInterval:10
Display Class API: enumerate display formats
 OK
 OK
       Display format 0 - Pixelformat:20
Display Class API: enumerate display dimensions
 OK
 OK
       Display dimensions 0 - ByteStride:2560 Width:640 Height:480
Display Class API: get the system (primary) buffer
 OK
Display Class API: map display surface to SGX
 OK
 OK
 OK
 OK
 OK
 OK
Attempt to create memory context for SGX:
 OK
       Shared heap 0 - HeapID:0x7000000 DevVAddr:0x400000 Size:0x77800000 Attr:0x4014200
       ...
       Shared heap 11 - HeapID:0x700000c DevVAddr:0xd0000000 Size:0x7c00000 Attr:0x4014200
 OK (times 84)
Creating render context via SGXCreateRenderContext
 OK
Adding render target via SGXAddRenderTarget
 OK
waiting for renders to complete...
Surface 0, write ops complete: 1999, pending 2000
Surface 0, write ops complete: 2000, pending 2000
Surface 1, write ops complete: 2000, pending 2000
Surface 1, write ops complete: 2000, pending 2000
Surface 2, write ops complete: 2000, pending 2000
Surface 2, write ops complete: 2000, pending 2000
Attempt to remove render target:
 OK
Attempt to destroy render context:
 OK
Display Class API: unmap swapchain display surfaces display surface from SGX
 OK
 OK
 OK
Display Class API: unmap display surface from SGX
 OK
Display Class API: close the device
 OK
SGXReleaseClientInfo:
 OK
```

```
PVRSRVDisconnect:
 OK

Total time: 81416ms
Mean time per frame: 13ms
```

6.4. Variants

It is also possible to run this test with flipping disabled, causing it to render directly to the system buffer. This is done by passing “-nf” as a command-line option.

In this case the background colour will be grey instead of green.

Note that some visual artefacts may be observed in this mode due to front-buffer rendering.