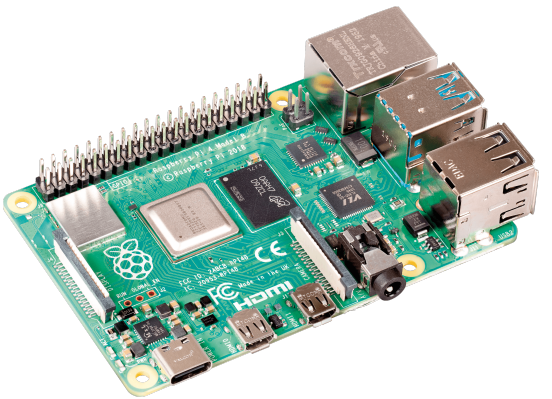
Writing a bare metal GPU driver for the Raspberry PI 4



Darius ENGLER

July 9, 2022

Introduction

Introduction

- What's a GPU driver?

Introduction

- What's a GPU driver?
  - A kernel-space driver (for communicating with the hardware) e.g. linux

Introduction

- What's a GPU driver?
  - A kernel-space driver (for communicating with the hardware) e.g. linux
  - A user-space driver (the opengl / vulkan implementation) e.g. mesa

Introduction

- What's a GPU driver?
  - A kernel-space driver (for communicating with the hardware) e.g. linux
  - A user-space driver (the opengl / vulkan implementation) e.g. mesa
- What's our goal?

Introduction

- What's a GPU driver?
  - A kernel-space driver (for communicating with the hardware) e.g. linux
  - A user-space driver (the opengl / vulkan implementation) e.g. mesa
- What's our goal?
  - Basic communication with the GPU

Introduction

- What's a GPU driver?
  - A kernel-space driver (for communicating with the hardware) e.g. linux
  - A user-space driver (the opengl / vulkan implementation) e.g. mesa
- What's our goal?
  - Basic communication with the GPU
  - Running (compute) shaders

Introduction

- What's a GPU driver?
  - A kernel-space driver (for communicating with the hardware) e.g. linux
  - A user-space driver (the opengl / vulkan implementation) e.g. mesa
- What's our goal?
  - Basic communication with the GPU
  - Running (compute) shaders
  - Rendering a 3D scene on the GPU

Motivation

Motivation

- Learn more about modern GPU

Motivation

- Learn more about modern GPU
- alternative for linux/mesa

Hardware overview

Hardware overview

- BCM2711 SoC

Hardware overview

- BCM2711 SoC
  - 4x Cortex-A72 Armv8 @ 1.5 GHz

Hardware overview

- BCM2711 SoC
  - 4x Cortex-A72 Armv8 @ 1.5 GHz
  - Broadcom VideoCore VI @ 500 MHz

Hardware overview

- BCM2711 SoC
  - 4x Cortex-A72 Armv8 @ 1.5 GHz
  - Broadcom VideoCore VI @ 500 MHz
    - VPU

Hardware overview

- BCM2711 SoC
  - 4x Cortex-A72 Armv8 @ 1.5 GHz
  - Broadcom VideoCore VI @ 500 MHz
    - VPU
    - QPU

Communicating with the GPU

Communicating with the GPU

- Mailbox interface managed by VCOS

Communicating with the GPU

- Mailbox interface managed by VCOS
- MMIO Registers

Mailbox interface

Mailbox interface

- Documented

Mailbox interface

- Documented
- Plenty of examples online

Mailbox interface

- Documented
- Plenty of examples online

Execute Code

- Tag: 0x00030010
- Request:
  - Length: 28
  - Value:
    - u32: function pointer
    - u32: r0
    - u32: r1
    - u32: r2
    - u32: r3
    - u32: r4
    - u32: r5
- Response:
  - Length: 4
  - Value:
    - u32: r0

Calls the function at given (bus) address and with arguments given. E.g. r0 = fn(r0, r1, r2, r3, r4, r5); It blocks until call completes. The (GPU) instruction cache is implicitly flushed. Setting the lsb of function pointer address will suppress the instruction cache flush if you know the buffer hasn't changed since last execution.

https://github.com/raspberrypi/firmware/wiki/Mailbox-property-interface#execute-code

MMIO Registers

MMIO Registers

arch/arm/boot/dts/bcm2711-rpi-ds.dtsi

```
compatible = "brcm,2711-v3d";
reg =
    <0x7ec00000  0x0 0x4000>,
    <0x7ec04000  0x0 0x4000>;
reg-names = "hub", "core0";
```

MMIO Registers

arch/arm/boot/dts/bcm2711-rpi-ds.dtsi

```
compatible = "brcm,2711-v3d";
reg =
    <0x7ec00000  0x0 0x4000>,
    <0x7ec04000  0x0 0x4000>;
reg-names = "hub", "core0";
```

drivers/gpu/drm/v3d/v3d_regs.h

```
#define V3D_HUB_AXICFG                          0x00000
# define V3D_HUB_AXICFG_MAX_LEN_MASK            V3D_MASK(3, 0)
# define V3D_HUB_AXICFG_MAX_LEN_SHIFT           0
#define V3D_HUB_UIFCFG                          0x00004
#define V3D_HUB_IDENT0                          0x00008

#define V3D_HUB_IDENT1                          0x0000c
# define V3D_HUB_IDENT1_WITH_MSO                BIT(19)
# define V3D_HUB_IDENT1_WITH_TSY                BIT(18)
# define V3D_HUB_IDENT1_WITH_TFU                BIT(17)
# define V3D_HUB_IDENT1_WITH_L3C                BIT(16)
# define V3D_HUB_IDENT1_NHOSTS_MASK             V3D_MASK(15, 12)
# define V3D_HUB_IDENT1_NHOSTS_SHIFT            12
# define V3D_HUB_IDENT1_NCORES_MASK             V3D_MASK(11, 8)
# define V3D_HUB_IDENT1_NCORES_SHIFT            8
# define V3D_HUB_IDENT1_REV_MASK                V3D_MASK(7, 4)
# define V3D_HUB_IDENT1_REV_SHIFT               4
# define V3D_HUB_IDENT1_TVER_MASK               V3D_MASK(3, 0)
# define V3D_HUB_IDENT1_TVER_SHIFT              0
```

MMIO Registers

MMIO Registers

V3D_HUB_IDENT0 : 0x42554856 ("VHUB")

MMIO Registers

V3D_HUB_IDENT0 : 0x42554856 ("VHUB")

*V3D_HUB_IDENT0 =

MMIO Registers

V3D_HUB_IDENT0 : 0x42554856 ("VHUB")

*V3D_HUB_IDENT0 = 0xDEADBEEF

MMIO Registers

V3D_HUB_IDENT0 : 0x42554856 ("VHUB")

*V3D_HUB_IDENT0 = 0xDEADBEEF

https://forums.raspberrypi.com/viewtopic.php?f=72&t=35966

3. Back on the ARM side I started to trawl thru the Linux sources to see how they turn on the V3D, after a day of work I had managed to crash the BUS of the V3D enough so that instead of showing me **0xDEADBEEF**, it crashed the R-Pi with any read from V3D peripheral region of memory only, possibly making it into the same state as my bootcode.bin demo.

4. I had noticed from my **Tags Channel** demo https://github.com/PeterLemon/Raspberry ... agsChannel
that in bare metal the V3D is in an "Enabled" state, but the frequency is set to zero, so I forced the clock frequency of the V3D to 250MHz using (250 * 1000 * 1000) which I had found in Linux sources. I then tried reading the V3D registers but still got **0xDEADBEEF**...

5. Finally I remembered that **Herman H Hermitage** had done a Fast Fourier Transform demo using the QPU and I started to look at the sources... I found that he used 2 new totally undocumented **Mailbox Property Interface** tags, namely **0x30011 = Execute_QPU**, & **0x30012 = Enable_QPU**. So I ran the tag "Enable_QPU" using 1 as my data, to set it to enable, and then tried reading the V3D registers and magically it kicks it into place!

This is really neat as my main V3D initialization is 2 simple concatenated tags of: (A) Set V3D Clock To 250Mhz, (B) Enable The QPU =D
A massive thankyou to **Herman H Hermitage**, without your help I would not have got it to work!!

I hope you guys like this, as it is a stepping stone to fully alpha blended textured shaded 2D & 3D GFX in bare metal on the Raspberry Pi, which will make the homebrew scene on this wonderful HW explode!

For my next V3D demo, I will try to show you guys howto run **Control Lists**, so we can use OpenGLES style functions to draw HW GFX to the screen =D

P.S We need to put those 2 new QPU Mailbox Property Tags onto some wiki or document, as they were so hard to find, and they are very important.

Logging register accesses

Logging register accesses

```
1  static inline u32 v3d_readl_wrapper(void* addr, const char* name, const char* file, int line)
2  {
3      printk("v3d-test: V3D_READ() : %s(0x%016llX) in %s:%d", name, (u64)addr, file, line);
4      return readl(addr);
5  }
6
7  static inline void v3d_writel_wrapper(u32 val, void* addr, const char* name, const char* file, int line)
8  {
9      printk("v3d-test: V3D_WRITE(0x%X) : %s(0x%016llX) in %s:%d", val, name, (u64)addr, file, line);
10     writel(val, addr);
11 }
12
13 #define V3D_READ(offset) v3d_readl_wrapper(v3d->hub_regs + offset, #offset, __FILE__, __LINE__)
14 #define V3D_WRITE(offset, val) v3d_writel_wrapper(val, v3d->hub_regs + offset, #offset, __FILE__, __LINE__)
15
16 #define V3D_BRIDGE_READ(offset) v3d_readl_wrapper(v3d->bridge_regs + offset, #offset, __FILE__, __LINE__)
17 #define V3D_BRIDGE_WRITE(offset, val) v3d_writel_wrapper(val, v3d->bridge_regs + offset, #offset, __FILE__, __LINE__)
18
19 #define V3D_GCA_READ(offset) v3d_readl_wrapper(v3d->gca_regs + offset, #offset, __FILE__, __LINE__)
20 #define V3D_GCA_WRITE(offset, val) v3d_writel_wrapper(val, v3d->gca_regs + offset, #offset, __FILE__, __LINE__)
21
22 #define V3D_CORE_READ(core, offset) v3d_readl_wrapper(v3d->core_regs[core] + offset, #offset, __FILE__, __LINE__)
23 #define V3D_CORE_WRITE(core, offset, val) v3d_writel_wrapper(val, v3d->core_regs[core] + offset, #offset, __FILE__, __LINE__)
```

Logging register accesses

```
1  static inline u32 v3d_readl_wrapper(void* addr, const char* name, const char* file, int line)
2  {
3      printk("v3d-test: V3D_READ() : %s(0x%016llX) in %s:%d", name, (u64)addr, file, line);
4      return readl(addr);
5  }
6
7  static inline void v3d_writel_wrapper(u32 val, void* addr, const char* name, const char* file, int line)
8  {
9      printk("v3d-test: V3D_WRITE(0x%X) : %s(0x%016llX) in %s:%d", val, name, (u64)addr, file, line);
10     writel(val, addr);
11 }
12
13 #define V3D_READ(offset) v3d_readl_wrapper(v3d->hub_regs + offset, #offset, __FILE__, __LINE__)
14 #define V3D_WRITE(offset, val) v3d_writel_wrapper(val, v3d->hub_regs + offset, #offset, __FILE__, __LINE__)
15
16 #define V3D_BRIDGE_READ(offset) v3d_readl_wrapper(v3d->bridge_regs + offset, #offset, __FILE__, __LINE__)
17 #define V3D_BRIDGE_WRITE(offset, val) v3d_writel_wrapper(val, v3d->bridge_regs + offset, #offset, __FILE__, __LINE__)
18
19 #define V3D_GCA_READ(offset) v3d_readl_wrapper(v3d->gca_regs + offset, #offset, __FILE__, __LINE__)
20 #define V3D_GCA_WRITE(offset, val) v3d_writel_wrapper(val, v3d->gca_regs + offset, #offset, __FILE__, __LINE__)
21
22 #define V3D_CORE_READ(core, offset) v3d_readl_wrapper(v3d->core_regs[core] + offset, #offset, __FILE__, __LINE__)
23 #define V3D_CORE_WRITE(core, offset, val) v3d_writel_wrapper(val, v3d->core_regs[core] + offset, #offset, __FILE__, __LINE__)
```

$ dmesg | grep "v3d"

```
...
[    1.873515] v3d-mbox: tag=RPI_FIRMWARE_SET_GPIO_STATE(0x00038041), size=0x8, buf=8400000000000000
[    1.881719] v3d-mbox: tag=RPI_FIRMWARE_GET_CLOCK_STATE(0x00030001), size=0x12, buf=0E000000000000000000000000
[    1.881789] v3d-mbox: tag=RPI_FIRMWARE_GET_CLOCK_STATE(0x00030001), size=0x12, buf=0D000000000000000000000000
[    1.884571] v3d-mbox: tag=RPI_FIRMWARE_GET_CLOCK_STATE(0x00030001), size=0x12, buf=0B000000000000000000000000
[    1.887295] v3d-mbox: tag=RPI_FIRMWARE_GET_CLOCK_STATE(0x00030001), size=0x12, buf=050000000000000000000000
[    1.889991] v3d-mbox: tag=RPI_FIRMWARE_GET_CLOCK_STATE(0x00030001), size=0x12, buf=040000000000000000000000
[    1.892633] v3d-mbox: tag=RPI_FIRMWARE_GET_CLOCK_STATE(0x00030001), size=0x12, buf=030000000000000000000000
[    1.943423] v3d-mbox: tag=RPI_FIRMWARE_SET_GPIO_STATE(0x00038041), size=0x8, buf=8400000001000000
[   19.801280] v3d-test: V3D_READ() : V3D_MMU_DEBUG_INFO(0xFFFFFFC011C99238) in drivers/gpu/drm/v3d/v3d_drv.c:266
[   19.801304] v3d-test: V3D_READ() : V3D_HUB_IDENT1(0xFFFFFFC011C9800C) in drivers/gpu/drm/v3d/v3d_drv.c:271
[   19.801378] v3d-mbox: tag=RPI_FIRMWARE_GET_CLOCK_RATE(0x00030002), size=0x12, buf=050000000000000000000000
[   19.806450] v3d-test: V3D_WRITE(0x0) : V3D_CTL_L2TFLSTA(0xFFFFFFC011CA8034) in drivers/gpu/drm/v3d/v3d_gem.c:78
[   19.806472] v3d-test: V3D_WRITE(0xFFFFFFFF) : V3D_CTL_L2TFLEND(0xFFFFFFC011CA8038) in drivers/gpu/drm/v3d/v3d_gem.c:79
[   19.806489] v3d-test: V3D_WRITE(0x1B600) : V3D_MMU_PT_PA_BASE(0xFFFFFFC011C99204) in drivers/gpu/drm/v3d/v3d_mmu.c:71
[   19.806504] v3d-test: V3D_WRITE(0x60D0C01) : V3D_MMU_CTL(0xFFFFFFC011C99200) in drivers/gpu/drm/v3d/v3d_mmu.c:72
[   19.806518] v3d-test: V3D_WRITE(0x8006373C) : V3D_MMU_ILLEGAL_ADDR(0xFFFFFFC011C99230) in drivers/gpu/drm/v3d/v3d_mmu.c:81
...
```

MMIO Registers

MMIO Registers

```
1 void v3d_test_ident0(void)
2 {
3         u32* addr = ioremap(0xFEC00000, 0x4000);
4         if (addr)
5                 printk("v3d-ident0: 0x%08X", addr[2]);
6         else
7                 printk("v3d-error: ioremap failed");
8 }
```

MMIO Registers

```
1 void v3d_test_ident0(void)
2 {
3         u32* addr = ioremap(0xFEC00000, 0x4000);
4         if (addr)
5                 printk("v3d-ident0: 0x%08X", addr[2]);
6         else
7                 printk("v3d-error: ioremap failed");
8 }
```

drivers/soc/bcm/bcm2835-power.c

```
 1 printk("bcm2835_asb_power_on 6");
 2 v3d_test_ident0();
 3
 4 ret = bcm2835_asb_enable(power, asb_s_reg);
 5 if (ret) {
 6     dev_err(power->dev, "Failed to enable ASB slave for %s\n",
 7         pd->base.name);
 8     goto err_disable_asb_master;
 9 }
10
11 printk("bcm2835_asb_power_on 7");
12 v3d_test_ident0();
```

MMIO Registers

```
1 void v3d_test_ident0(void)
2 {
3         u32* addr = ioremap(0xFEC00000, 0x4000);
4         if (addr)
5                 printk("v3d-ident0: 0x%08X", addr[2]);
6         else
7                 printk("v3d-error: ioremap failed");
8 }
```

drivers/soc/bcm/bcm2835-power.c

```
 1 printk("bcm2835_asb_power_on 6");
 2 v3d_test_ident0();
 3
 4 ret = bcm2835_asb_enable(power, asb_s_reg);
 5 if (ret) {
 6     dev_err(power->dev, "Failed to enable ASB slave for %s\n",
 7         pd->base.name);
 8     goto err_disable_asb_master;
 9 }
10
11 printk("bcm2835_asb_power_on 7");
12 v3d_test_ident0();
```

```
[    8.803038] bcm2835_asb_power_on 6
[    8.803059] v3d-ident0: 0xDEADBEEF
[    8.803069] bcm2835_asb_power_on 7
[    8.803105] v3d-ident0: 0x42554856
```

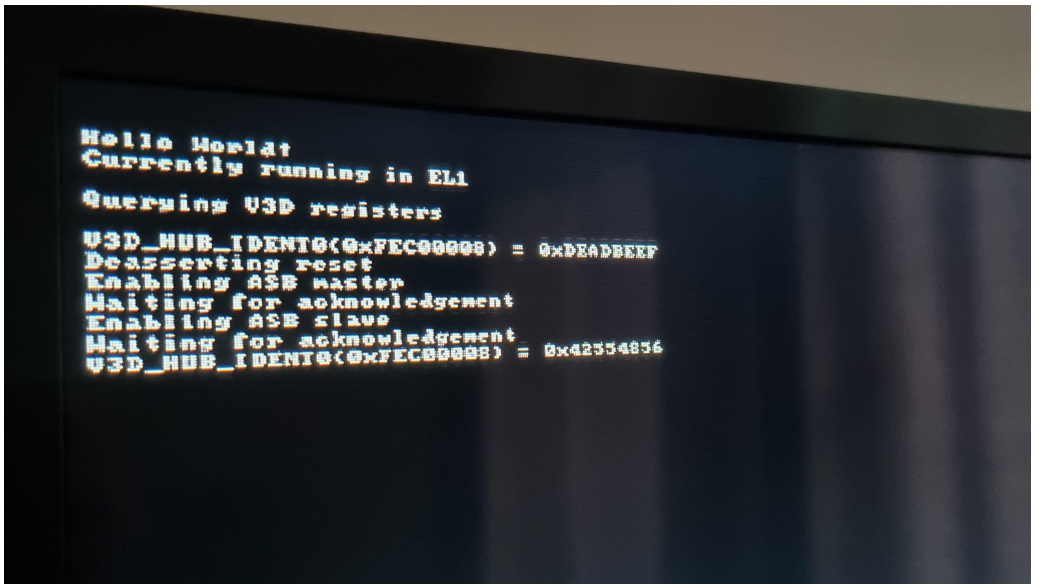
MMIO Registers

MMIO Registers

```
 1 static constexpr u32 PM_V3DRSTN     = bit(6);
 2 static constexpr u32 PM_PASSWORD    = 0x5A000000;
 3 static constexpr u32 ASB_REQ_STOP   = bit(0);
 4 static constexpr u32 ASB_ACK        = bit(1);
 5 static constexpr u32 ASB_EMPTY      = bit(2);
 6 static constexpr u32 ASB_FULL       = bit(3);
 7
 8 // Deasserting reset
 9 MMIO::write(MMIO::PM_GRAFX, MMIO::read(MMIO::PM_GRAFX) | PM_PASSWORD | PM_V3DRSTN);
10
11 // Enabling ASB master
12 MMIO::write(MMIO::ASB_V3D_M_CTRL, PM_PASSWORD | (MMIO::read(MMIO::ASB_V3D_M_CTRL) & ~ASB_REQ_STOP));
13 // Waiting for acknowledgement
14 while (MMIO::read(MMIO::ASB_V3D_M_CTRL) & ASB_ACK);
15
16 // Enabling ASB slave
17 MMIO::write(MMIO::ASB_V3D_S_CTRL, PM_PASSWORD | (MMIO::read(MMIO::ASB_V3D_S_CTRL) & ~ASB_REQ_STOP));
18 // Waiting for acknowledgement
19 while (MMIO::read(MMIO::ASB_V3D_S_CTRL) & ASB_ACK);
```

MMIO Registers

```
static constexpr u32 PM_V3DRSTN     = bit(6);
static constexpr u32 PM_PASSWORD    = 0x5A000000;
static constexpr u32 ASB_REQ_STOP   = bit(0);
static constexpr u32 ASB_ACK        = bit(1);
static constexpr u32 ASB_EMPTY      = bit(2);
static constexpr u32 ASB_FULL       = bit(3);

// Deasserting reset
MMIO::write(MMIO::PM_GRAFX, MMIO::read(MMIO::PM_GRAFX) | PM_PASSWORD | PM_V3DRSTN);

// Enabling ASB master
MMIO::write(MMIO::ASB_V3D_M_CTRL, PM_PASSWORD | (MMIO::read(MMIO::ASB_V3D_M_CTRL) & ~ASB_REQ_STOP));
// Waiting for acknowledgement
while (MMIO::read(MMIO::ASB_V3D_M_CTRL) & ASB_ACK);

// Enabling ASB slave
MMIO::write(MMIO::ASB_V3D_S_CTRL, PM_PASSWORD | (MMIO::read(MMIO::ASB_V3D_S_CTRL) & ~ASB_REQ_STOP));
// Waiting for acknowledgement
while (MMIO::read(MMIO::ASB_V3D_S_CTRL) & ASB_ACK);
```



GPU Tasks

GPU Tasks

- CSD (compute shader dispatch)

GPU Tasks

- CSD (compute shader dispatch)
  - takes code buffer, uniform buffer

GPU Tasks

- CSD (compute shader dispatch)
  - takes code buffer, uniform buffer
  - executes code

GPU Tasks

- CSD (compute shader dispatch)
  - takes code buffer, uniform buffer
  - executes code
- TFU (Texture Formatter Unit)

GPU Tasks

- CSD (compute shader dispatch)
  - takes code buffer, uniform buffer
  - executes code
- TFU (Texture Formatter Unit)
  - takes source/destination image & format

GPU Tasks

- CSD (compute shader dispatch)
  - takes code buffer, uniform buffer
  - executes code
- TFU (Texture Formatter Unit)
  - takes source/destination image & format
  - Texture blitting

GPU Tasks

- CSD (compute shader dispatch)
  - takes code buffer, uniform buffer
  - executes code
- TFU (Texture Formatter Unit)
  - takes source/destination image & format
  - Texture blitting
- Binner/Render

GPU Tasks

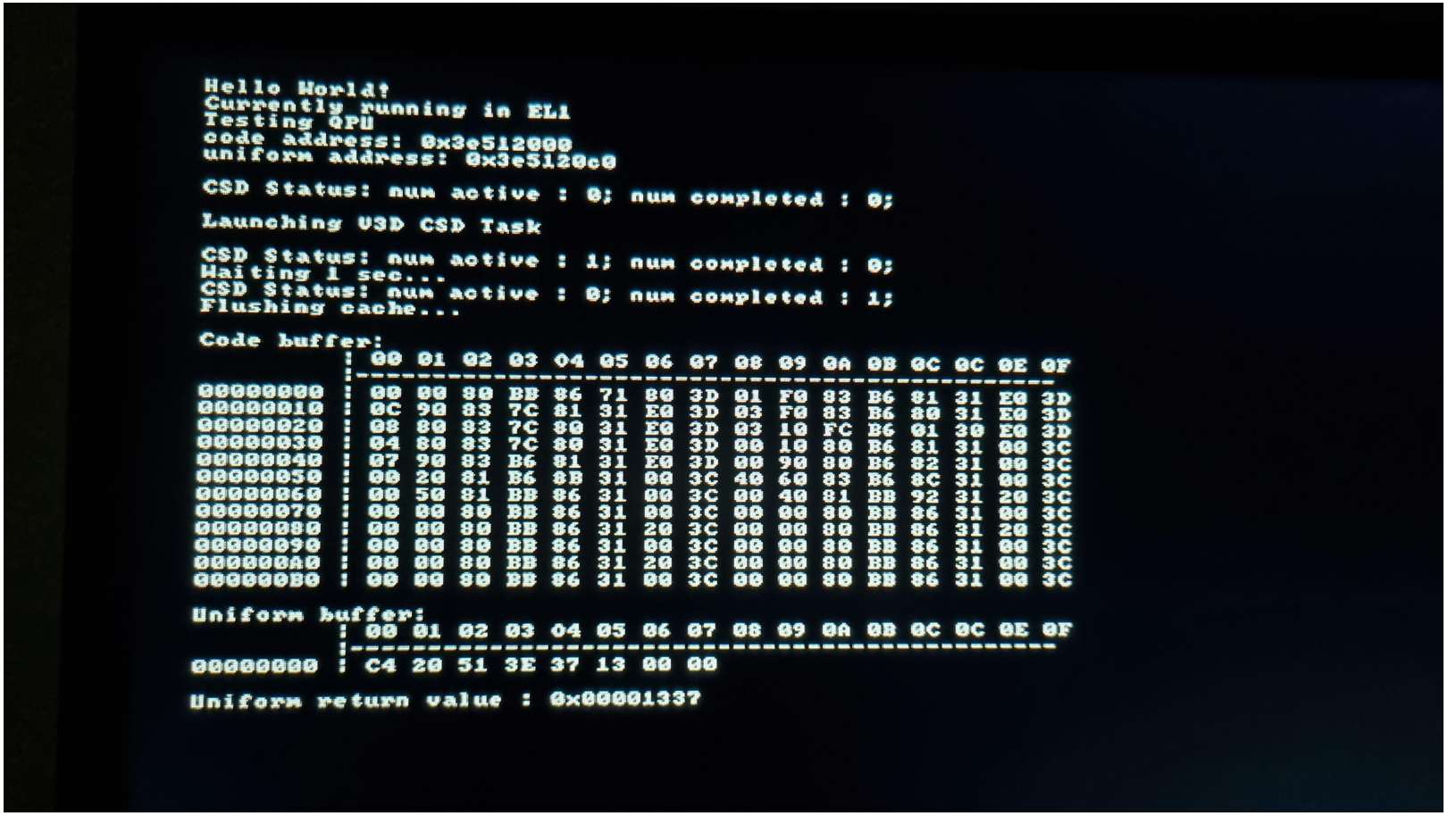
- CSD (compute shader dispatch)
  - takes code buffer, uniform buffer
  - executes code
- TFU (Texture Formatter Unit)
  - takes source/destination image & format
  - Texture blitting
- Binner/Render
  - Takes command lists

GPU Tasks

- CSD (compute shader dispatch)
  - takes code buffer, uniform buffer
  - executes code
- TFU (Texture Formatter Unit)
  - takes source/destination image & format
  - Texture blitting
- Binner/Render
  - Takes command lists
  - Renders commands

CSD Task

CSD Task


Hello World!
Currently running in EL1
Testing QPU
code address: 0x3e512000
uniform address: 0x3e5120c0
CSD Status: num active : 0; num completed : 0;
Launching V3D CSD Task
CSD Status: num active : 1; num completed : 0;
Waiting 1 sec...
CSD Status: num active : 0; num completed : 1;
Flushing cache...
Code buffer:
Uniform buffer:
Uniform return value : 0x00001337

Binning? Rendering?

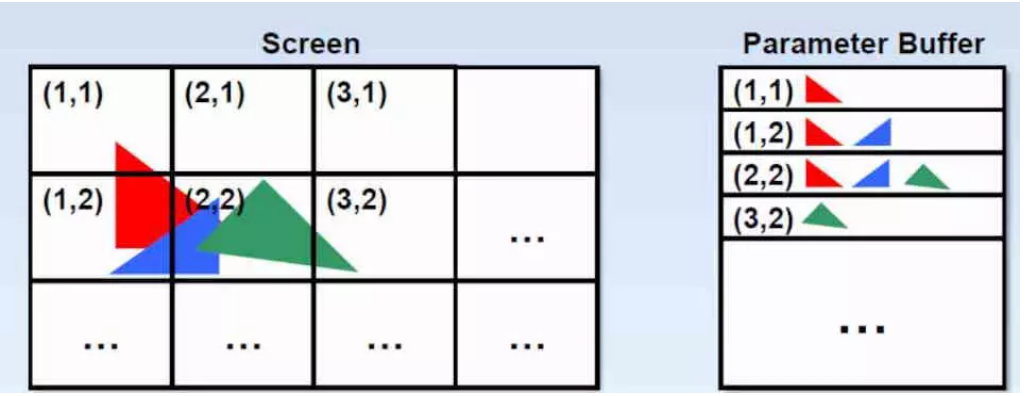
Binning? Rendering?

- The VideoCore VI is a Tile Renderer!

Binning? Rendering?

- The VideoCore VI is a Tile Renderer!

Binning

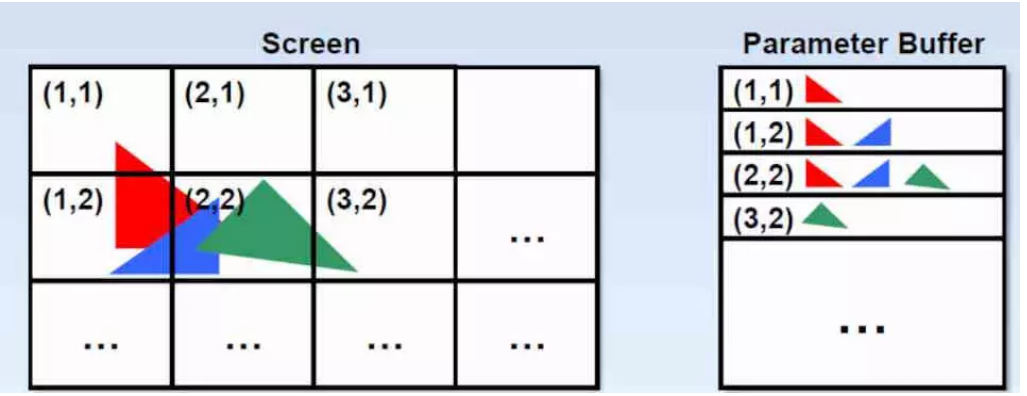


https://itigic.com/tile-rendering-architecture-of-very-low-consumption-gpus/

Binning? Rendering?

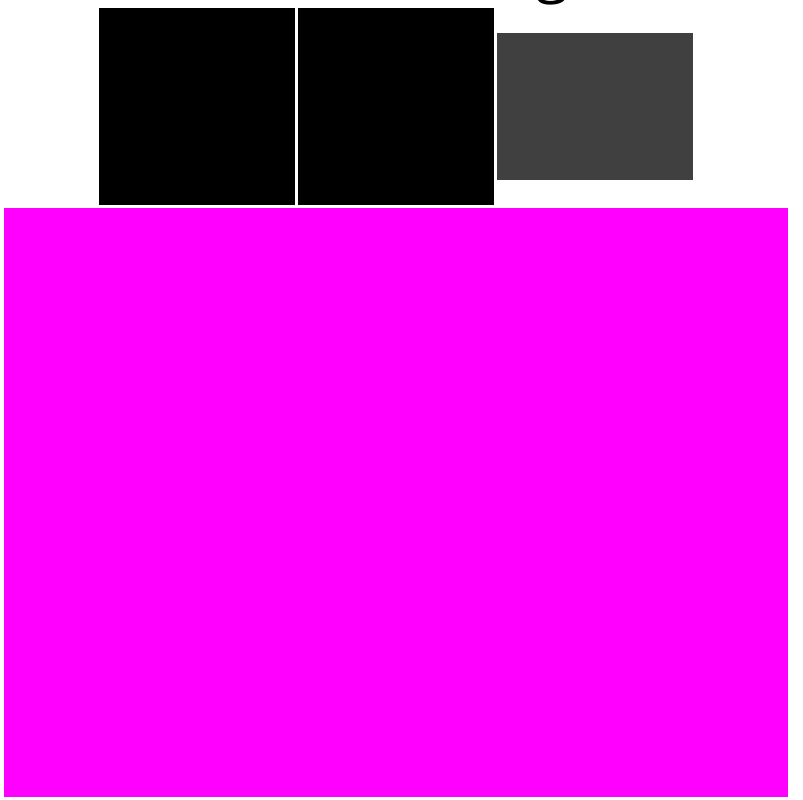
- The VideoCore VI is a Tile Renderer!

Binning



https://itigic.com/tile-rendering-architecture-of-very-low-consumption-gpus/

Rendering



https://developer.samsung.com/galaxy-gamedev/resources/articles/gpu-framebuffer.html

Figuring out the command list format

Figuring out the command list format

```
<struct name="GL Shader State Record" min_ver="41">
<field name="Point size in shaded vertex data" size="1" start="0" type="bool"/>
<field name="Enable clipping" size="1" start="1" type="bool"/>

<field name="Vertex ID read by coordinate shader" size="1" start="2" type="bool"/>
<field name="Instance ID read by coordinate shader" size="1" start="3" type="bool"/>
<field name="Base Instance ID read by coordinate shader" size="1" start="4" type="bool"/>
<field name="Vertex ID read by vertex shader" size="1" start="5" type="bool"/>
<field name="Instance ID read by vertex shader" size="1" start="6" type="bool"/>
<field name="Base Instance ID read by vertex shader" size="1" start="7" type="bool"/>

<field name="Fragment shader does Z writes" size="1" start="8" type="bool"/>
<field name="Turn off early-z test" size="1" start="9" type="bool"/>
<field name="Coordinate shader has separate input and output VPM blocks" size="1" start="10" type="bool"/>
<field name="Vertex shader has separate input and output VPM blocks" size="1" start="11" type="bool"/>
<field name="Fragment shader uses real pixel centre W in addition to centroid W2" size="1" start="12" type="bool"/>
<field name="Enable Sample Rate Shading" size="1" start="13" type="bool"/>
<field name="Any shader reads hardware-written Primitive ID" size="1" start="14" type="bool"/>
<field name="Insert Primitive ID as first varying to fragment shader" size="1" start="15" type="bool"/>
<field name="Turn off scoreboard" size="1" start="16" type="bool"/>
<field name="Do scoreboard wait on first thread switch" size="1" start="17" type="bool"/>
<field name="Disable implicit point/line varyings" size="1" start="18" type="bool"/>
<field name="No prim pack" size="1" start="19" type="bool"/>

<field name="Number of varyings in Fragment Shader" size="8" start="3b" type="uint"/>

<field name="Coordinate Shader output VPM segment size" size="4" start="4b" type="uint"/>
<field name="Min Coord Shader output segments required in play in addition to VCM cache size" size="4" start="36" type="uint"/>

<field name="Coordinate Shader input VPM segment size" size="4" start="5b" type="uint"/>
<field name="Min Coord Shader input segments required in play" size="4" start="44" type="uint" minus_one="true"/>

<field name="Vertex Shader output VPM segment size" size="4" start="6b" type="uint"/>
<field name="Min Vertex Shader output segments required in play in addition to VCM cache size" size="4" start="52" type="uint"/>

<field name="Vertex Shader input VPM segment size" size="4" start="7b" type="uint"/>
<field name="Min Vertex Shader input segments required in play" size="4" start="60" type="uint" minus_one="true"/>
```

https://gitlab.freedesktop.org/mesa/mesa/-/blob/main/src/broadcom/cle/v3d_packet_v33.xml

Tracing GPU communication

Tracing GPU communication

- wrap MMIO registers access macros

Tracing GPU communication

- wrap MMIO registers access macros
- log every buffer object before sending tasks

Tracing GPU communication

- wrap MMIO registers access macros
- log every buffer object before sending tasks
- https://github.com/matusnovak/rpi-opengl-without-x

Tracing GPU communication

- wrap MMIO registers access macros
- log every buffer object before sending tasks
- https://github.com/matusnovak/rpi-opengl-without-x

```
[   42.222817] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[   42.222832] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[   42.222849] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[   42.222857] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[   42.222866] BO 00C80000-00D0A000(0x8A000) : tile_alloc
[   42.222871] BO 00060000-00849000(0x7E9000) : framebuffer
[   42.222875] BO 00860000-00C55000(0x3F5000) : zbuf
[   42.222879] BO 00D20000-00D40000(0x20000)
[   42.222886] v3d-submit: bcl=0x00c60000(0x0000007a) rcl=0x00d80000(0x000001fc)
[   42.222890] BCL:
[   42.222913] 78 00 00 00 00 7F 07 37 04 13 5C 00 00 00 00 06 6B 00 00 00 00 80 07 38 04 60 07 70 02 68 00 00 80 3F 69 00 00 80 3F 6E
[   42.222931] 00 00 70 48 00 00 07 C8 6F 00 00 00 3F 00 00 00 3F 6D 00 00 00 00 00 00 80 3F 6C 00 C0 03 00 00 1C 02 00 57 00 00 00 00
[   42.222947] 56 00 00 00 00 00 00 00 00 61 63 58 4A 00 5C 00 00 00 00 5B 0F 00 80 3F 47 44 40 41 00 D4 00 24 04 03 00 00 00 00 00 00
[   42.222952] 00 04
[   42.222958] RCL:
[   42.222978] 79 00 80 07 38 04 00 01 00 79 03 FF 00 00 FF 00 00 00 79 81 00 00 00 00 00 00 00 79 02 00 00 00 80 3F 00 00 7E 04 7B 00
[   42.222996] 00 C8 00 7A 01 01 0F 09 1E 10 01 00 7C 00 00 00 1A 1D 08 00 00 00 00 00 00 00 00 00 00 19 03 1B 7C 00 00 00 1A 1D 08
[   42.223015] 00 00 00 00 00 00 00 00 00 00 00 00 1B 13 14 74 00 D4 00 98 00 D4 00 17 00 00 17 01 00 17 02 00 17 03 00 17 04 00 17 05 00
[   42.223032] 17 06 00 17 07 00 17 08 00 17 09 00 17 0A 00 17 0B 00 17 0C 00 17 0D 00 17 0E 00 17 00 01 17 01 01 17 02 01 17 03 01 17
[   42.223049] 04 01 17 05 01 17 06 01 17 07 01 17 08 01 17 09 01 17 0A 01 17 0B 01 17 0C 01 17 0D 01 17 0E 01 17 00 02 17 01 02 17 02
[   42.223067] 02 17 03 02 17 04 02 17 05 02 17 06 02 17 07 02 17 08 02 17 09 02 17 0A 02 17 0B 02 17 0C 02 17 0D 02 17 0E 02 17 00 03
[   42.223084] 17 01 03 17 02 03 17 03 03 17 04 03 17 05 03 17 06 03 17 07 03 17 08 03 17 09 03 17 0A 03 17 0B 03 17 0C 03 17 0D 03 17
[   42.223101] 0E 03 17 00 04 17 01 04 17 02 04 17 03 04 17 04 04 17 05 04 17 06 04 17 07 04 17 08 04 17 09 04 17 0A 04 17 0B 04 17 0C
[   42.223118] 04 17 0D 04 17 0E 04 17 00 05 17 01 05 17 02 05 17 03 05 17 04 05 17 05 05 17 06 05 17 07 05 17 08 05 17 09 05 17 0A 05
[   42.223134] 17 0B 05 17 0C 05 17 0D 05 17 0E 05 17 00 06 17 01 06 17 02 06 17 03 06 17 04 06 17 05 06 17 06 06 17 07 06 17 08 06 17
[   42.223152] 09 06 17 0A 06 17 0B 06 17 0C 06 17 0D 06 17 0E 06 17 00 07 17 01 07 17 02 07 17 03 07 17 04 07 17 05 07 17 06 07 17 07
[   42.223168] 07 17 08 07 17 09 07 17 0A 07 17 0B 07 17 0C 07 17 0D 07 17 0E 07 17 00 08 17 01 08 17 02 08 17 03 08 17 04 08 17 05 08
[   42.223181] 17 06 08 17 07 08 17 08 08 17 09 08 17 0A 08 17 0B 08 17 0C 08 17 0D 08 17 0E 08 0D
```

Buffer Objects

Buffer Objects

- BCL (Binning) / RCL (Rendering) command lists

Buffer Objects

- BCL (Binning) / RCL (Rendering) command lists
- Vertex attributes

Buffer Objects

- BCL (Binning) / RCL (Rendering) command lists
- Vertex attributes
- Shader code buffer

Buffer Objects

- BCL (Binning) / RCL (Rendering) command lists
- Vertex attributes
- Shader code buffer
- Shader uniform buffer

Buffer Objects

- BCL (Binning) / RCL (Rendering) command lists
- Vertex attributes
- Shader code buffer
- Shader uniform buffer
- Framebuffer / Z Buffer / Stencil Buffer

Buffer Objects

- BCL (Binning) / RCL (Rendering) command lists
- Vertex attributes
- Shader code buffer
- Shader uniform buffer
- Framebuffer / Z Buffer / Stencil Buffer

```
*************************** BCL ****************************

78 00 00 00 00 7F 07 37 04 TILE_BINNING_MODE_CFG
13 FLUSH_VCD_CACHE
5C 00 00 00 00 OCCLUSION_QUERY_COUNTER
06 START_TILE_BINNING
04 FLUSH
```

```
************************** INDIRECT ****************************

7D TILE_COORDINATES_IMPLICIT
1A END_OF_LOADS
38 02 PRIM_LIST_FORMAT
15 00 BRANCH_TO_IMPLICIT_TILE_LIST
1D 00 B0 11 00 E0 01 00 00 00 00 06 00 STORE_TILE_BUFFER_GENERAL
19 03 CLEAR_TILE_BUFFERS
1B END_OF_TILE_MARKER
12 RETURN_FROM_SUB_LIST
```

```
**************************** RCL ****************************

79 00 80 07 38 04 00 01 00 TILE_RENDERING_MODE_CFG_COMMON
79 03 FF 00 00 FF 00 00 00 TILE_RENDERING_MODE_CFG_CLEAR_COLORS_PART1
79 81 00 00 00 00 00 00 00 TILE_RENDERING_MODE_CFG_COLOR
79 02 00 00 00 00 00 00 00 TILE_RENDERING_MODE_CFG_ZS_CLEAR_VALUES
7E 04 TILE_LIST_INITIAL_BLOCK_SIZE
7B 00 00 C8 00 MULTICORE_RENDERING_TILE_LIST_SET_BASE
7A 01 01 0F 09 1E 10 01 00 MULTICORE_RENDERING_SUPERTILE_CFG

7C 00 00 00 TILE_COORDINATES
1A END_OF_LOADS
1D 08 00 00 00 00 00 00 00 00 00 00 STORE_TILE_BUFFER_GENERAL
19 03 CLEAR_TILE_BUFFERS
1B END_OF_TILE_MARKER
7C 00 00 00 TILE_COORDINATES
1A END_OF_LOADS
1D 08 00 00 00 00 00 00 00 00 00 00 STORE_TILE_BUFFER_GENERAL
1B END_OF_TILE_MARKER

13 FLUSH_VCD_CACHE
14 00 00 D6 00 17 00 D6 00 START_ADDRESS_OF_GENERIC_TILE_LIST(00D60000-00D60017)
17 00 00 SUPERTILE_COORDINATES
17 01 00 SUPERTILE_COORDINATES
...
17 0D 08 SUPERTILE_COORDINATES
17 0E 08 SUPERTILE_COORDINATES
0D END_OF_RENDERING
```

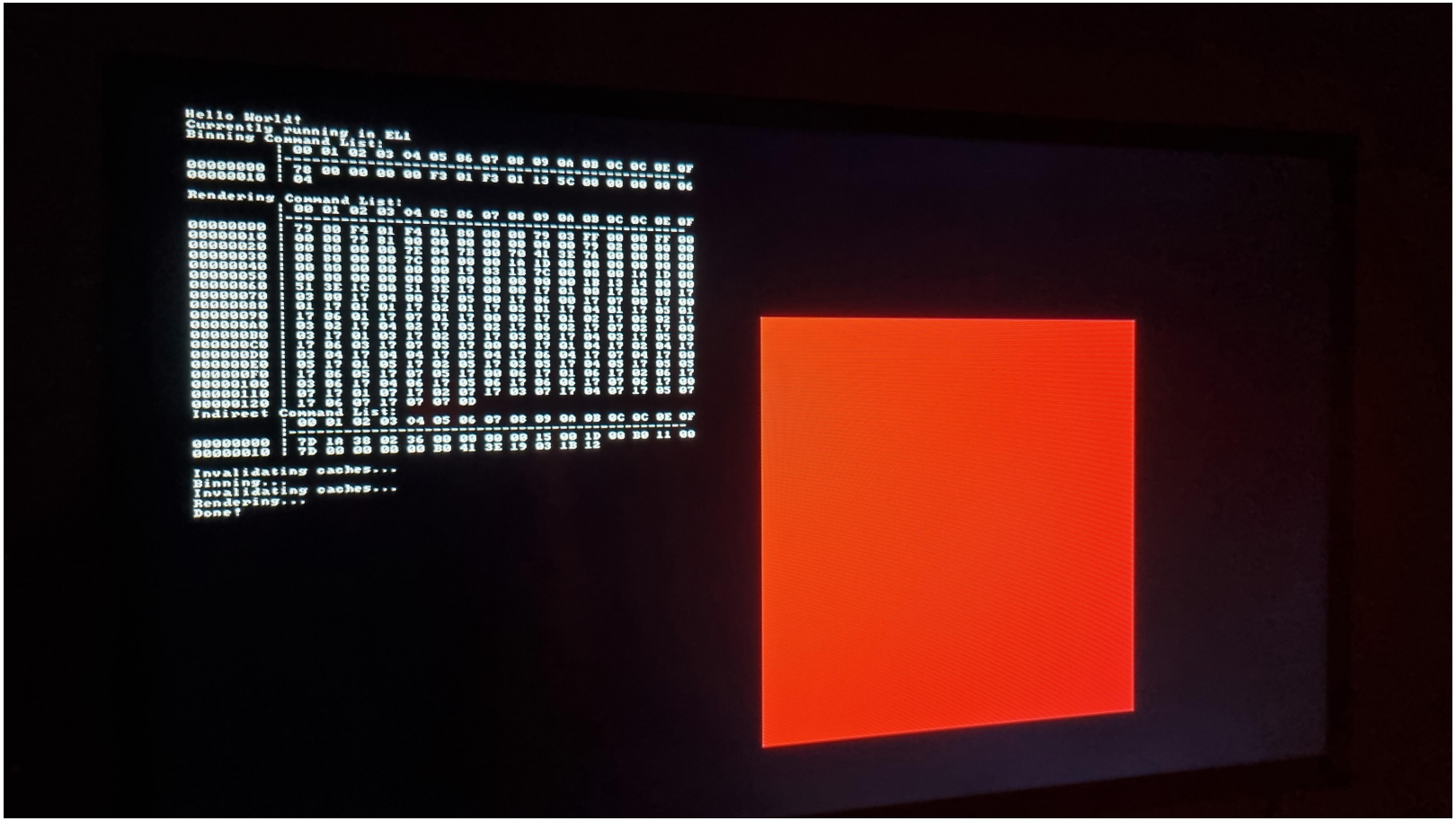
Putting everything together

Putting everything together

```
/* =============== BCL ============== */
    {
        bcl << TileBinningModeCfg(0, 0, 1, 0, false, false, width, height);
        bcl << OP_FLUSH_VCD_CACHE;
        bcl << OcclusionQueryCounter(0);
        bcl << OP_START_TILE_BINNING;

        bcl << ClipWindow(0, 0, width, height);
        bcl << CfgBits(true, true, true, false, 0, 0, false, 7, false, false, true, false, false, false, false);
        bcl << PointSize(1.0f);
        bcl << LineWidth(1.0f);
        bcl << ClipperXYScaling((width / 2) * 256.f, (height / 2) * -256.f);
        bcl << ClipperZScaleAndOffset(.5f, .5f);
        bcl << CLipperZMinMaxClippingPlanes(0.f, 1.f);
        bcl << ViewportOffset(width / 2, height / 2, 0, 0);
        bcl << ColorWriteMasks(0);
        bcl << BlendConstantColor(0, 0, 0, 0);
        bcl << OP_ZERO_ALL_FLAT_SHADE_FLAGS;
        bcl << OP_ZERO_ALL_NON_PERSPECTIVE_FLAGS;
        bcl << OP_ZERO_ALL_CENTROID_FLAGS;
        bcl << TransformFeedbackSpecs(0, false);
        bcl << OcclusionQueryCounter(0);
        bcl << SampleState(0xF, 1.0f);
        bcl << VcmCacheSize(4, 4);

        bcl << GlShaderState(shaderStateRecordAddr, attrCount);
        // mode=4 (triangle)
        bcl << VertexArrayPrims(4, vertexCount, 0);

        // bcl epilogue
        bcl << OP_FLUSH;
    }

    /* =============== RCL ============== */
    {
        // must be first
        rcl << TileRenderingModeCfgCommon(1, width, height, 0, false, false, 0, false, 2, false);
        rcl << TileRenderingModeCfgClearColorsPart1(0, 0xFF101010, 0x000000);
        rcl << TileRenderingModeCfgColor(0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
        // must be last
        rcl << TileRenderingModeCfgZSClearValues(0, 1.0f);
        rcl << TileListInitialBlockSize(0, true);

        u32 superTileW;
        u32 superTileH;
        u32 frameWSupertile;
        u32 frameHSupertile;
        getSuperTiles(superTileW, superTileH, frameWSupertile, frameHSupertile, tilesX, tilesY);
```

First Render!

First Render!



Actual Geometry

We need shaders

```
GLShaderStateRecord shader{};
shader.enable_clipping = true;
shader.fragment_shader_uses_real_pixel_centre_w_in_addition_to_centroid_w2 = true;
shader.disable_implicit_point_line_varyings = true;
shader.number_of_varyings_in_fragment_shader = 4; // 4 component color
shader.coordinate_shader_output_vpm_segment_size = 1;
shader.coordinate_shader_input_vpm_segment_size = 1;
shader.vertex_shader_output_vpm_segment_size = 1;
shader.vertex_shader_input_vpm_segment_size = 1;
shader.address_of_default_attribute_values = defaultAttrAddr;

shader.fragment_shader_code_address = fragShaderAddr >> 3;
shader.fragment_shader_uniforms_address = fragUnifAddr;
shader.fragment_shader_4_way_threadable = true;
shader.fragment_shader_start_in_final_thread_section = false;
shader.fragment_shader_propagate_nans = true;

shader.vertex_shader_code_address = vtxShaderAddr >> 3;
shader.vertex_shader_uniforms_address = vtxUnifAddr;
shader.vertex_shader_4_way_threadable = true;
shader.vertex_shader_start_in_final_thread_section = true;
shader.vertex_shader_propagate_nans = true;

shader.coordinate_shader_code_address = coordShaderAddr >> 3;
shader.coordinate_shader_uniforms_address = coordUnifAddr;
shader.coordinate_shader_4_way_threadable = true;
shader.coordinate_shader_start_in_final_thread_section = true;
shader.coordinate_shader_propagate_nans = true;
indirect << shader;
```

and attributes

```
// positions
GlShaderStateAttributeRecord posAttr{};
posAttr.address = posAttrAddr;
posAttr.number_of_values_read_by_vertex_shader = 3;
posAttr.number_of_values_read_by_coordinate_shader = 3;
posAttr.instance_divisor = 0;
posAttr.stride = 3 * sizeof(f32);
posAttr.maximum_index = 0xFFFFFF;
posAttr.vec_size = 3;
posAttr.type = 2; // ATTRIBUTE_FLOAT

// colors
GlShaderStateAttributeRecord colorAttr;
colorAttr.address = colorAttrAddr;
colorAttr.number_of_values_read_by_vertex_shader = 4;
colorAttr.number_of_values_read_by_coordinate_shader = 0;
colorAttr.instance_divisor = 0;
colorAttr.stride = 4 * sizeof(u8);
colorAttr.maximum_index = 0xFFFFFF;
colorAttr.type = 4; // ATTRIBUTE_BYTE
colorAttr.normalized_int_type = true;
indirect << colorAttr;
```

Coordinate shaders?

```
#version 300 es
layout (location = 0) in vec3 position;
layout (location = 1) in vec3 normal;
layout (location = 2) in vec4 color;

uniform mat4 u_model;
uniform mat4 u_view;
uniform mat4 u_proj;

out vec4 v_color;
out vec3 v_normal;

void main()
{
    v_color = color;
    v_normal = mat3(transpose(inverse(u_model))) * normal;
    gl_Position = u_proj * u_view * u_model * vec4(position, 1.0);
}
```

Coordinate shaders?

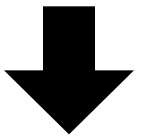
```
#version 300 es
layout (location = 0) in vec3 position;
layout (location = 1) in vec3 normal;
layout (location = 2) in vec4 color;

uniform mat4 u_model;
uniform mat4 u_view;
uniform mat4 u_proj;

out vec4 v_color;
out vec3 v_normal;

void main()
{
    v_color = color;
    v_normal = mat3(transpose(inverse(u_model))) * normal;
    gl_Position = u_proj * u_view * u_model * vec4(position, 1.0);
}
```

Coordinate shaders?

```
#version 300 es
layout (location = 0) in vec3 position;
layout (location = 1) in vec3 normal;
layout (location = 2) in vec4 color;

uniform mat4 u_model;
uniform mat4 u_view;
uniform mat4 u_proj;

out vec4 v_color;
out vec3 v_normal;

void main()
{
    v_color = color;
    v_normal = mat3(transpose(inverse(u_model))) * normal;
    gl_Position = u_proj * u_view * u_model * vec4(position, 1.0);
}
```

Coordinate shaders?

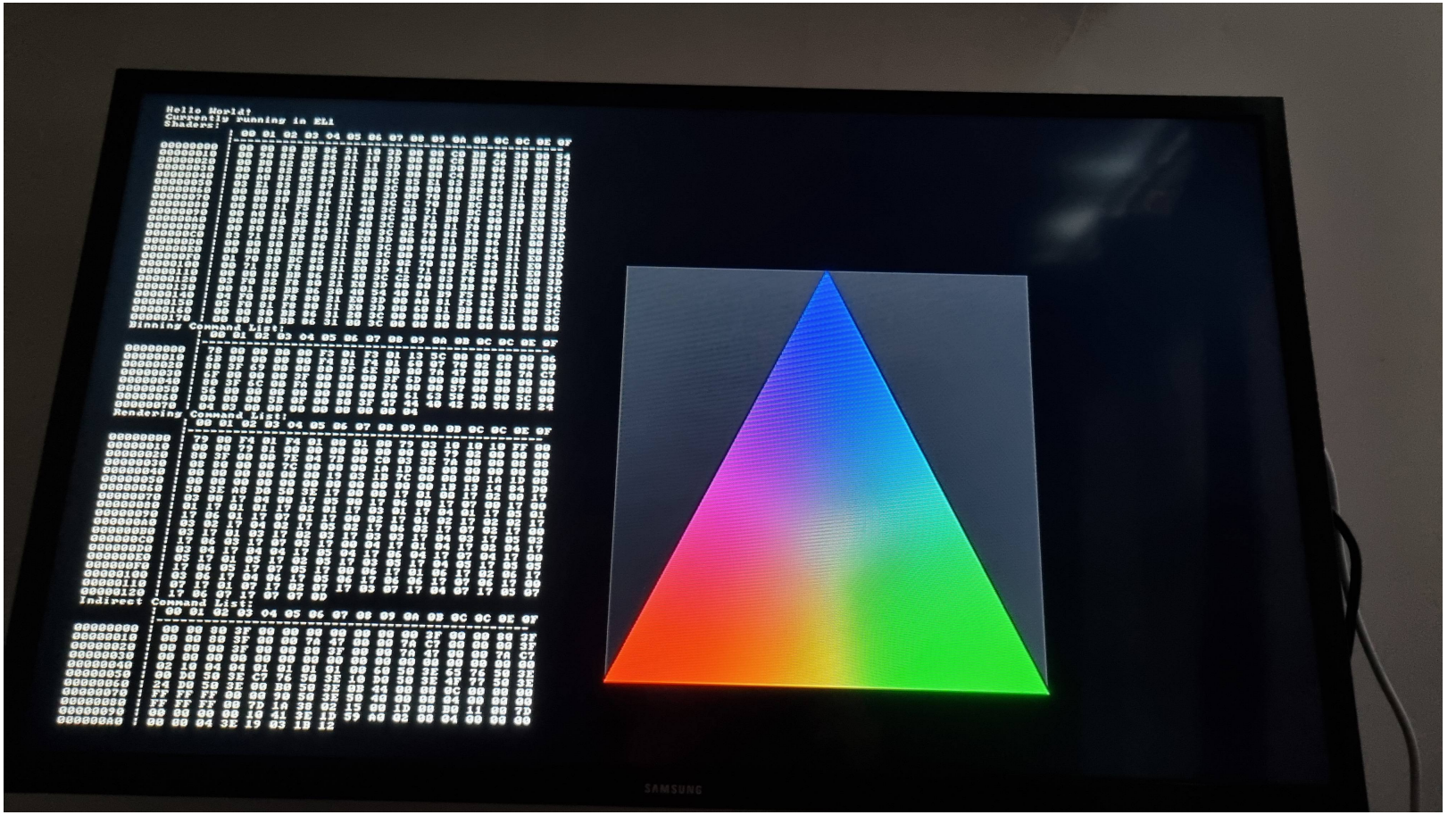
```
#version 300 es
layout (location = 0) in vec3 position;
layout (location = 1) in vec3 normal;
layout (location = 2) in vec4 color;

uniform mat4 u_model;
uniform mat4 u_view;
uniform mat4 u_proj;

out vec4 v_color;
out vec3 v_normal;

void main()
{
    v_color = color;
    v_normal = mat3(transpose(inverse(u_model))) * normal;
    gl_Position = u_proj * u_view * u_model * vec4(position, 1.0);
}
```



```
#version 300 es
layout (location = 0) in vec3 position;

uniform mat4 u_model;
uniform mat4 u_view;
uniform mat4 u_proj;

void main()
{
    gl_Position = u_proj * u_view * u_model * vec4(position, 1.0);
}
```

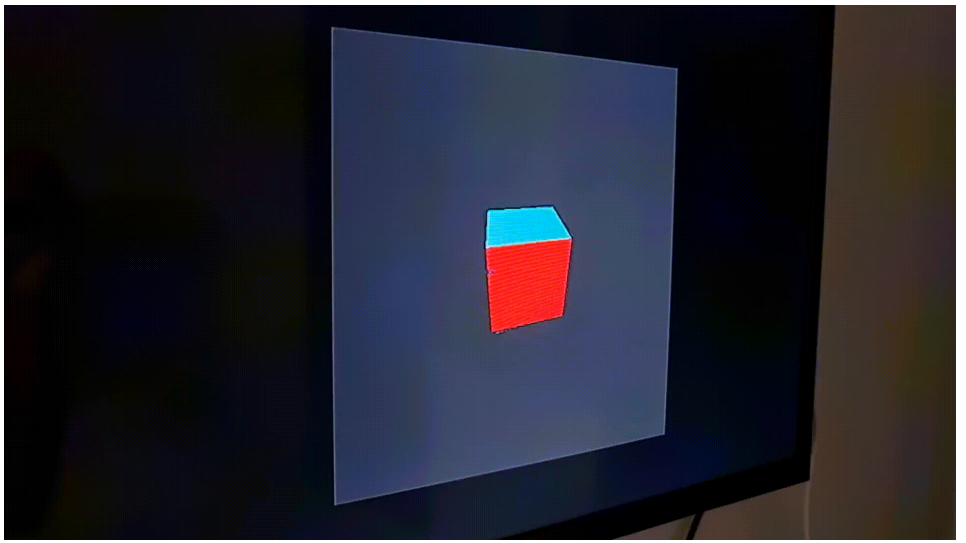
First Triangle!

First Triangle!

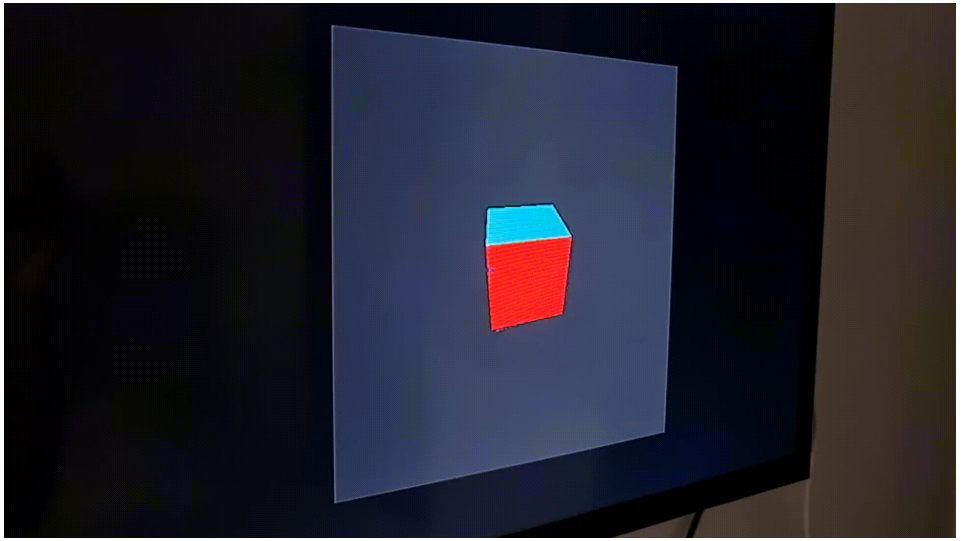


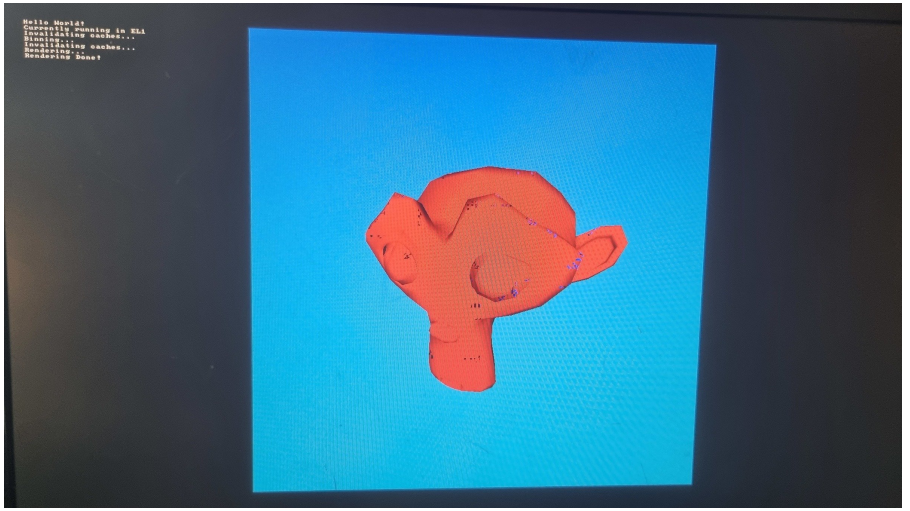
More complex scenes

More complex scenes

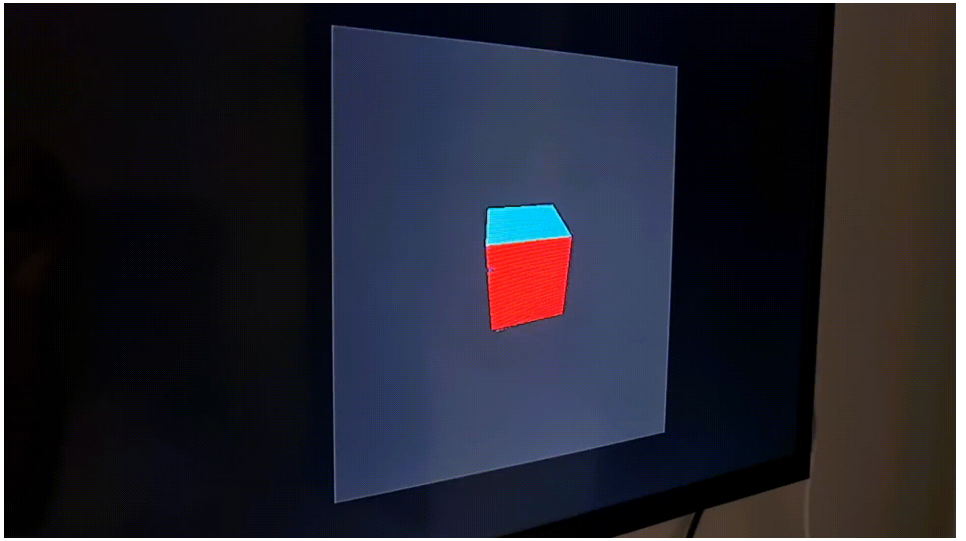


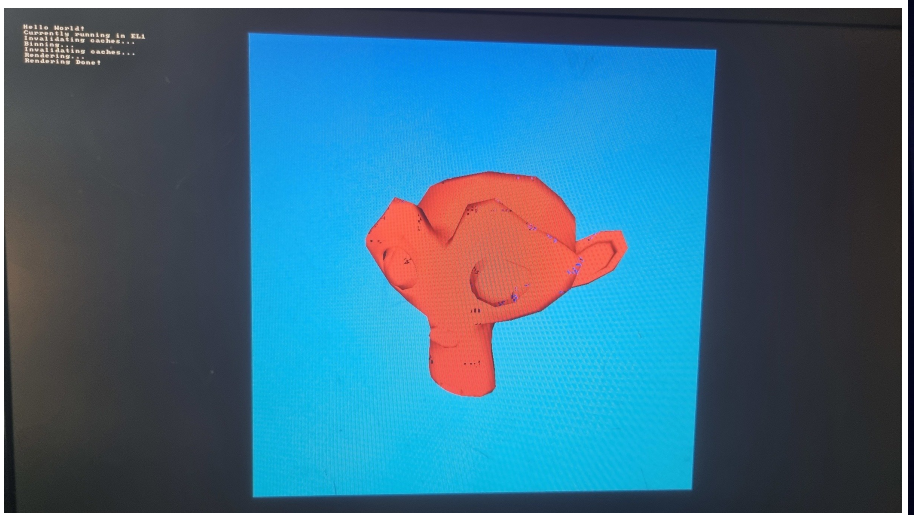
More complex scenes

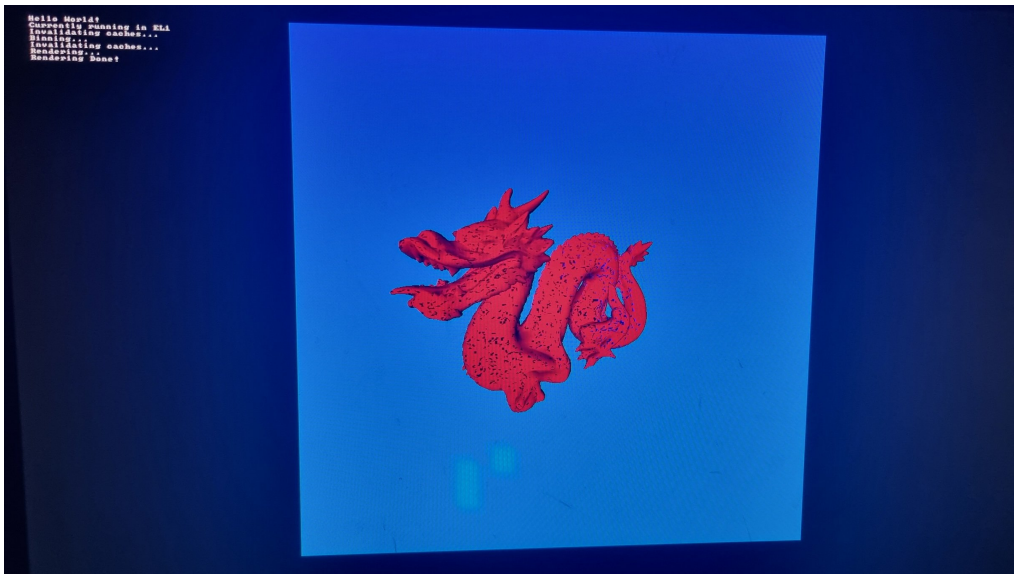




More complex scenes



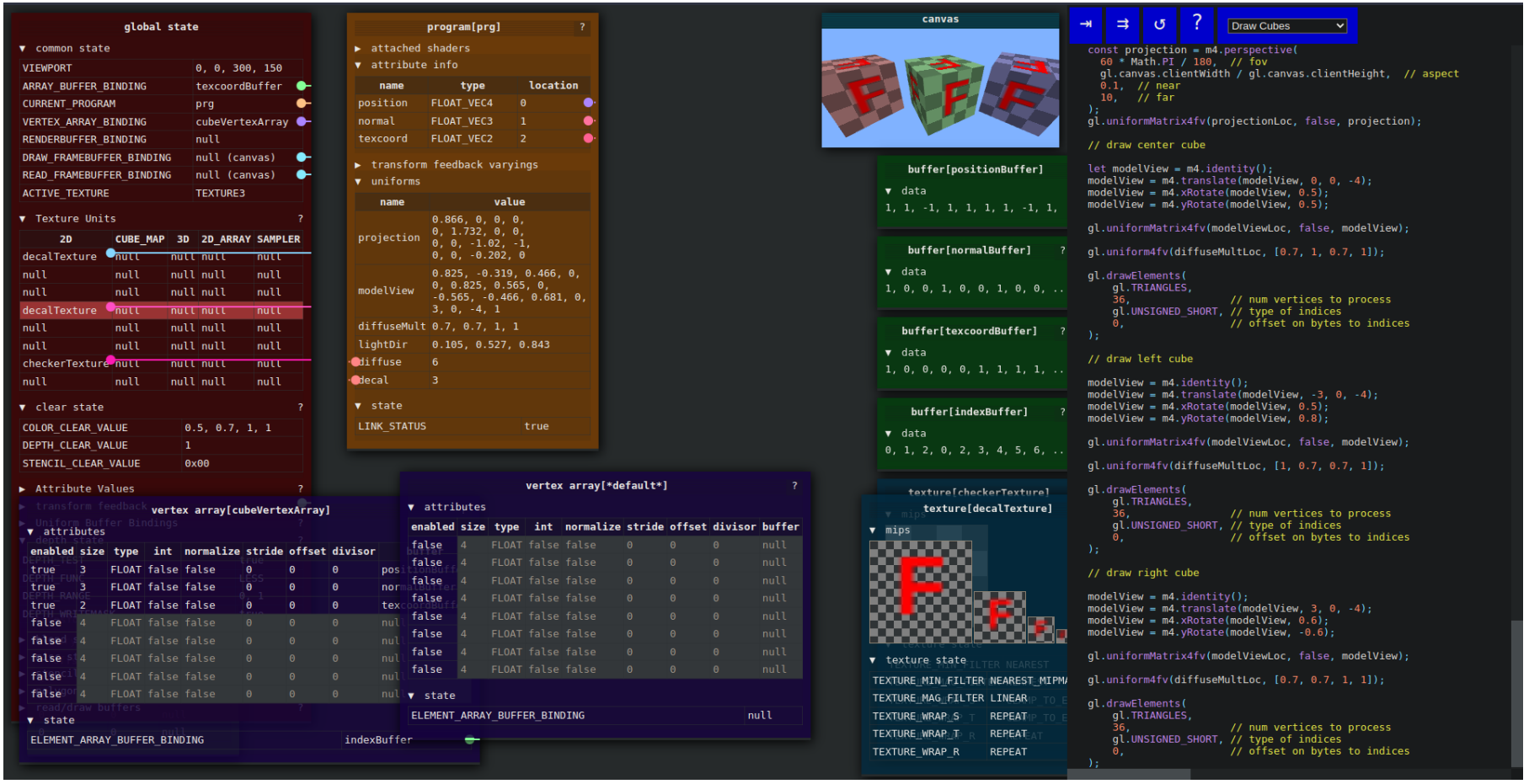




Mini OpenGL API

Mini OpenGL API

https://webgl2fundamentals.org/webgl/lessons/resources/webgl-state-diagram.html



Mini OpenGL API

```
gl.clearColor(0.06f, 0.06f, 0.06f, 1.0f);
gl.clearStencil(0.0f);
gl.clearDepth(1.0f);

u32 positionLoc = gl.getAttribLocation(prg, "position");
u32 colorLoc = gl.getAttribLocation(prg, "color");

u32 triangleVAO = gl.createVertexArray();
gl.bindVertexArray(triangleVAO);

u32 positionBuffer = gl.createBuffer();
gl.bindBuffer(GL_ARRAY_BUFFER, positionBuffer);
gl.bufferData(GL_ARRAY_BUFFER, sizeof(g_PosAttrBuff), g_PosAttrBuff, GL_STATIC_DRAW);

u32 colorBuffer = gl.createBuffer();
gl.bindBuffer(GL_ARRAY_BUFFER, colorBuffer);
gl.bufferData(GL_ARRAY_BUFFER, sizeof(g_ColorAttrBuff), g_ColorAttrBuff, GL_STATIC_DRAW);


gl.enableVertexAttribArray(positionLoc);
gl.bindBuffer(GL_ARRAY_BUFFER, positionBuffer);
gl.vertexAttribPointer(
    positionLoc,
    3,            // 2 values per vertex shader iteration
    GL_FLOAT,     // data is 32bit floats
    false,        // don't normalize
    0,            // stride (0 = auto)
    0            // offset into buffer
);

gl.bindBuffer(GL_ARRAY_BUFFER, colorBuffer);
gl.enableVertexAttribArray(colorLoc);
gl.vertexAttribPointer(
    colorLoc,
    4,                // 4 values per vertex shader iteration
    GL_UNSIGNED_BYTE, // data is 8bit unsigned bytes
    true,             // do normalize
    0,                // stride (0 = auto)
    0                // offset into buffer
);

gl.useProgram(prg);
gl.drawArrays(GL_TRIANGLES, 0, 3);
```

Conclusion

Conclusion

- Managed to write a driver that renders basic scenes

Conclusion

- Managed to write a driver that renders basic scenes
- Progress was non-linear

Conclusion

- Managed to write a driver that renders basic scenes
- Progress was non-linear
- Command lists are closer to graphics API than expected

TODO:

TODO:

- Fix GPU artifacts occurring with complex geometry

TODO:

- Fix GPU artifacts occurring with complex geometry
- Support texture samplers

TODO:

- Fix GPU artifacts occurring with complex geometry
- Support texture samplers
- Reverse engineer Tile allocation / TSDA (Tile State Data Array)

TODO:

- Fix GPU artifacts occurring with complex geometry
- Support texture samplers
- Reverse engineer Tile allocation / TSDA (Tile State Data Array)
- handle GPU MMU

TODO:

- Fix GPU artifacts occurring with complex geometry
- Support texture samplers
- Reverse engineer Tile allocation / TSDA (Tile State Data Array)
- handle GPU MMU
- handle GPU interrupt

TODO:

- Fix GPU artifacts occurring with complex geometry
- Support texture samplers
- Reverse engineer Tile allocation / TSDA (Tile State Data Array)
- handle GPU MMU
- handle GPU interrupt
- Write a SPIR-V => QPU compiler

Questions?