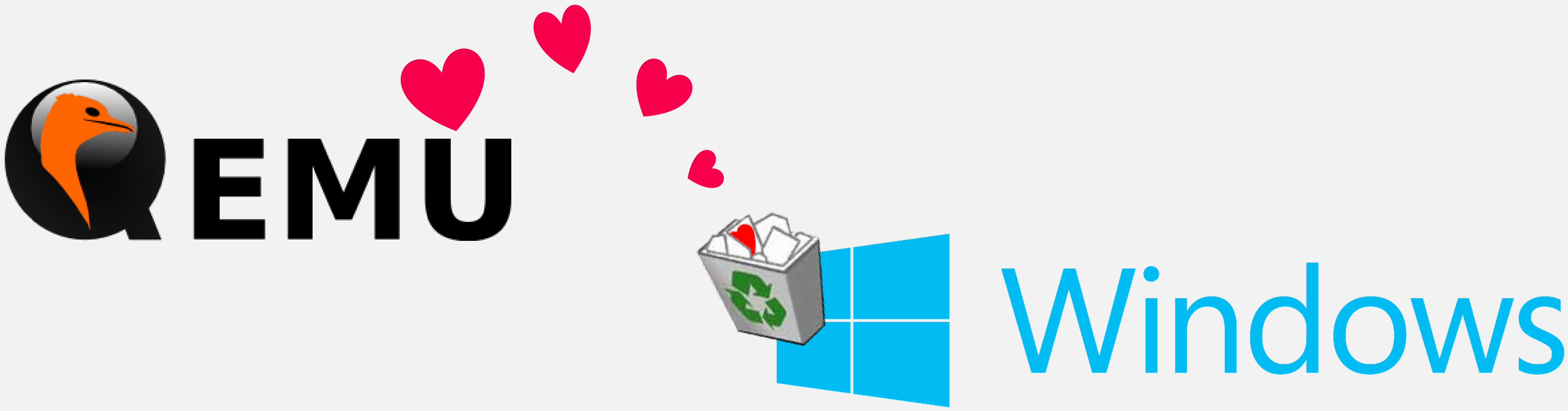


3D acceleration on Windows with Virgl3D

Nathan Gauër | 2017

- Windows guests miss 3D acceleration
- Virgl3D is stable and will help us
- Fedora has it ‘out of the box’

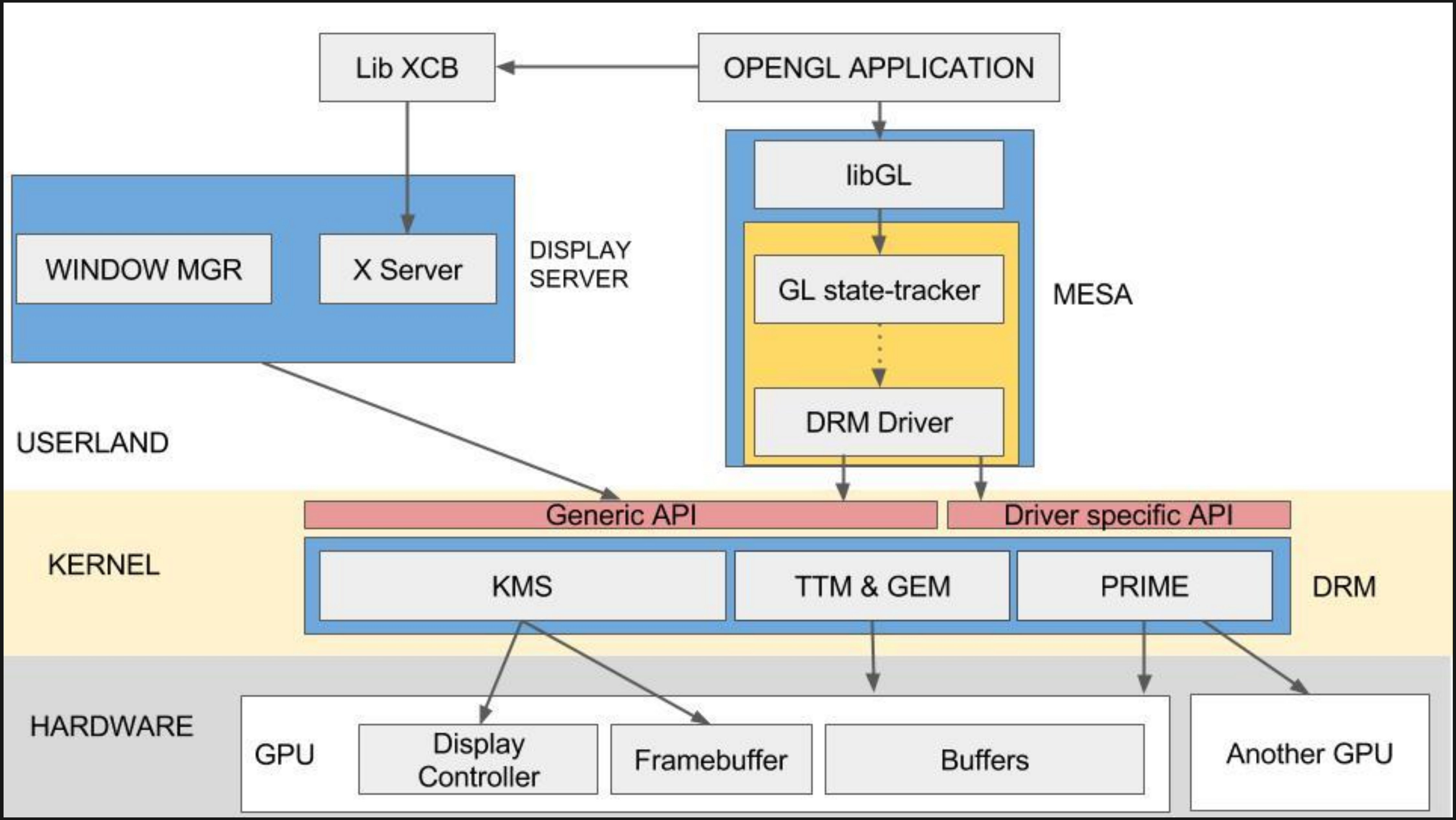
- How Virgl3D works
- How Virgl3D and Windows’ graphic stack behave
- How we can implement this

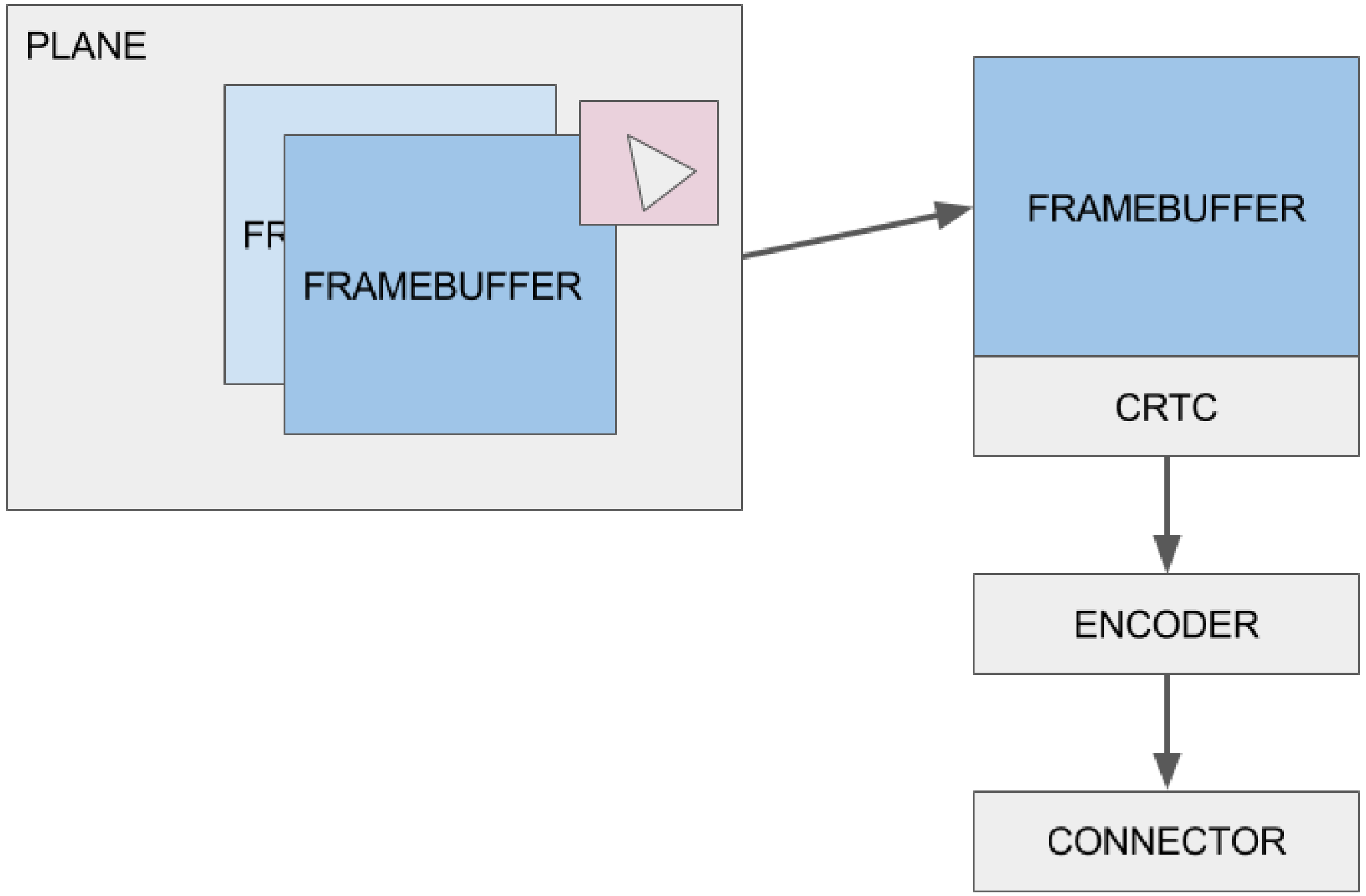
What this presentation is NOT about

- Linux graphic stack (in depth)
- DirectX (No, you do not want to touch that)
- 2D acceleration on Windows

Linux graphic stack

Linux graphic stack

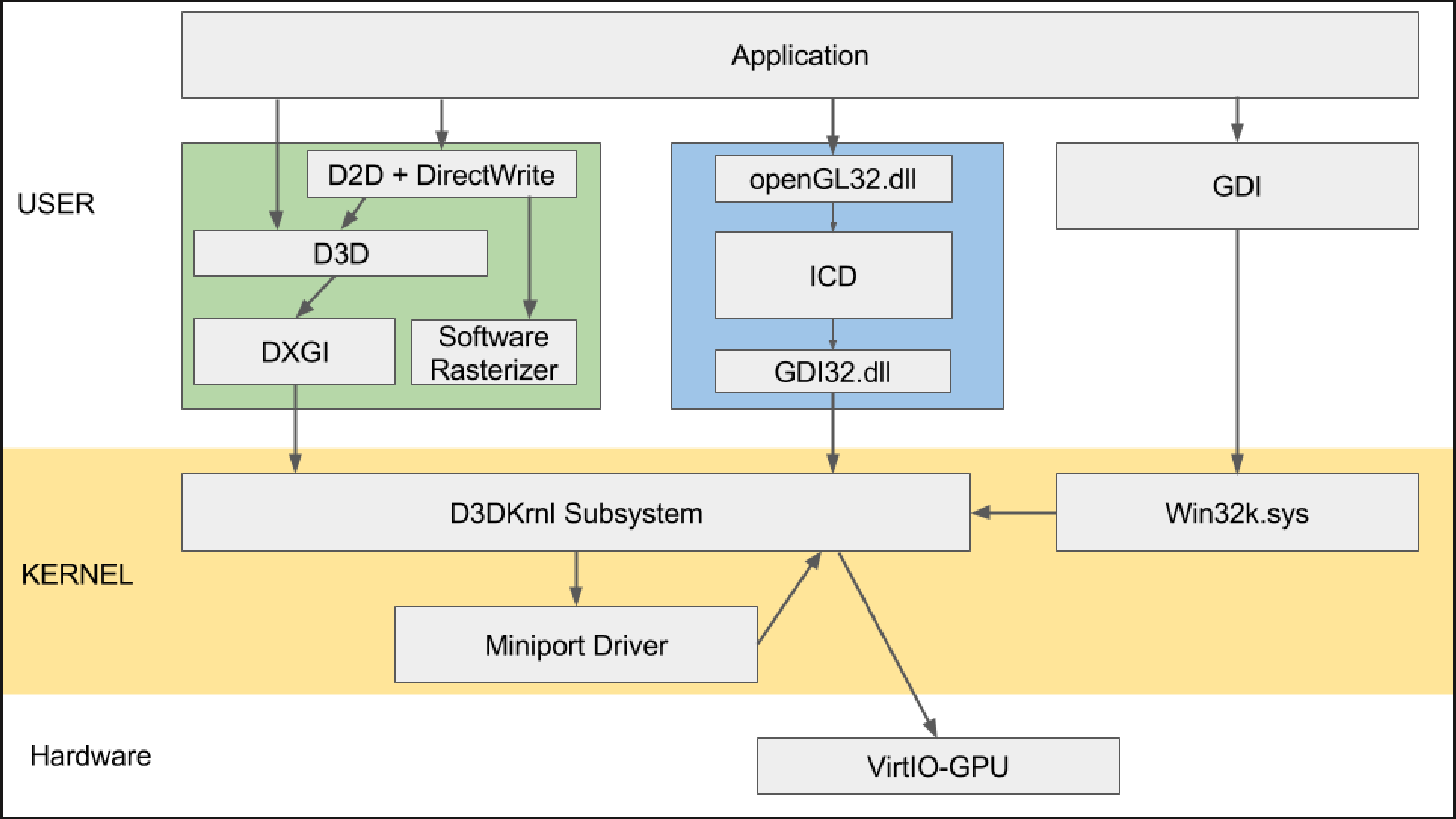



PLANE
FR
FRAMEBUFFER
FRAMEBUFFER
CRTC
ENCODER
CONNECTOR

- Mesa speaks GLSL/OpenGL
- Back-ends has to speak TGSI/Gallium

Windows graphic stack

Windows graphic stack



What do we want ?

- Userland application using OpenGL
- Have some 3D acceleration

What do we have ?

- Poor background
- poorer documentation
- Closed-source OS

What do we have ?

- Poor background
- poorer documentation
- Closed-source OS

Step 1: DOD & API-Forwarding

- Hook OpenGL calls
- Forward them to QEMU/KVM
- Run them on the host

- Our driver has register callbacks (simple DLL)
- And need to use GDI.dll to call kernel driver

```
D3DKMT_ESCAPE escape = { 0 };
escape.hAdapter = info.adapter;
escape.hDevice = info.device;
escape.type = D3DKMT_ESCAPE_DRIVERPRIVATE;
escape.flags.Value = 1;
escape.hContext = info.context;
escape.privateDriverData = command;
escape.privateDriverDataSize = commandSize * sizeof(BYTE);
PFND3DKMT_ESCAPE func = getGDIFunction<PFND3DKMT_ESCAPE>("D3DKMTEscape");
```

NTSTATUS DxgkInitializeDisplayOnlyDriver(_In_ PDRIVER_OBJECT DriverObject, _In_ PUNICODE_STRING RegistryPath, _In_ PKMDDOD_INITIALIZATION_DATA KmdDo

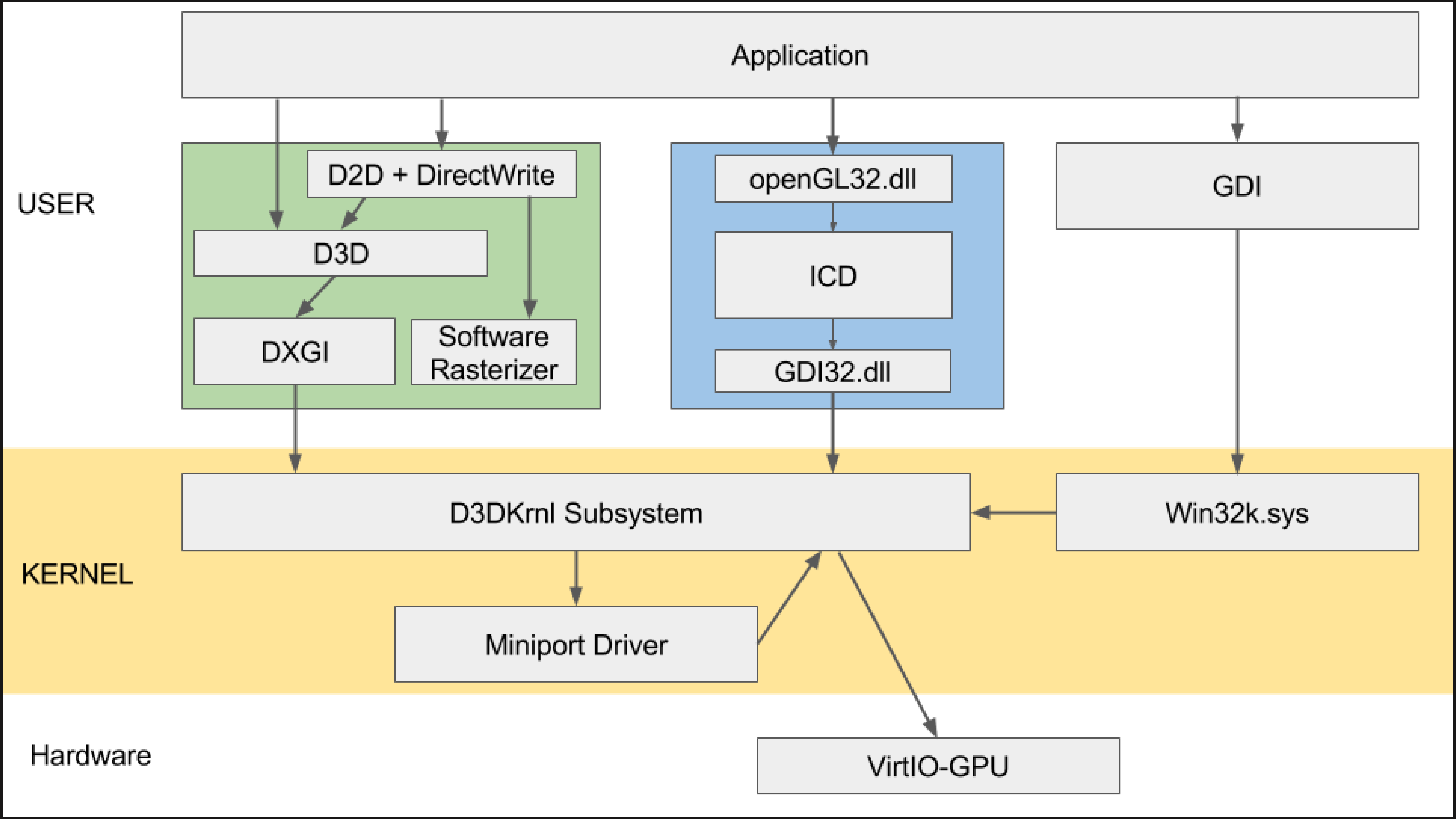
DOD & API-Forwarding

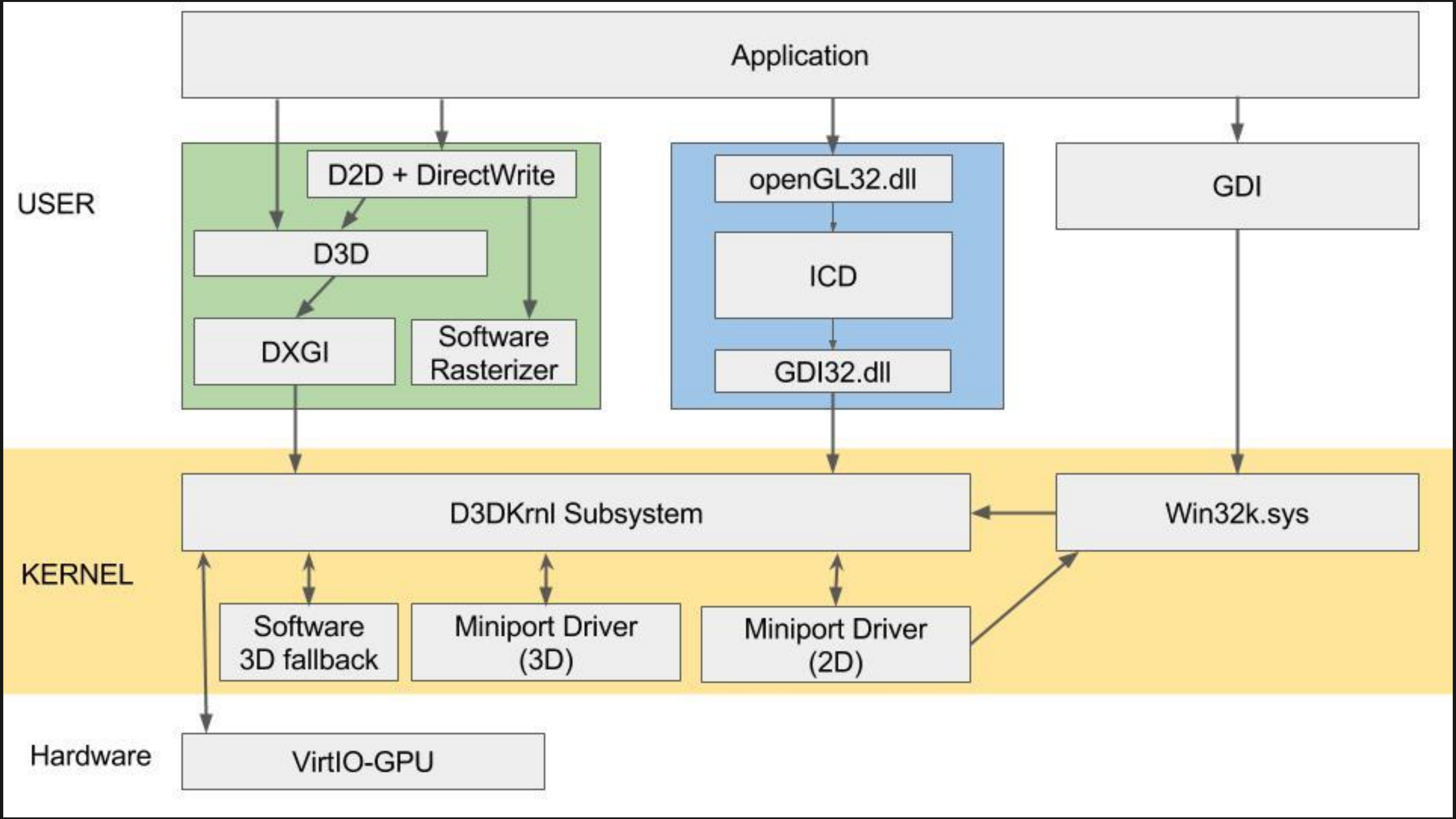
```
struct _KMDDOD_INITIALIZATION_DATA {
    ULONG                       Version;
    PDXGKDDI_ADD_DEVICE         DxgkDdiAddDevice;
    PDXGKDDI_START_DEVICE       DxgkDdiStartDevice;
    …
    PDXGKDDI_ESCAPE             DxgkDdiEscape;
    …
};
```

```
NTSTATUS DxgkInitializeDisplayOnlyDriver(
  _In_ PDRIVER_OBJECT             DriverObject,
  _In_ PUNICODE_STRING            RegistryPath,
  _In_ PKMDDOD_INITIALIZATION_DATA KmdDodInitializationData
);
```

- Driver does not receive anything
- Userland receive bad answer or something else.
- ??

Edit 17/07/2017:
Userland can communicate with our DOD driver. I failed to do so because I wanted to instantiate my device, and thus, spoke with a hypothetical fallback. Without instantiating, I can speak to DOD driver.


Application
USER
D2D + DirectWrite
D3D
DXGI
Software Rasterizer
openGL32.dll
ICD
GDI32.dll
GDI
KERNEL
D3DKrnl Subsystem
Win32k.sys
Miniport Driver
Hardware
VirtIO-GPU


Application
USER
D2D + DirectWrite
D3D
DXGI
Software Rasterizer
openGL32.dll
ICD
GDI32.dll
GDI
KERNEL
D3DKrnl Subsystem
Win32k.sys
Software 3D fallback
Miniport Driver (3D)
Miniport Driver (2D)
Hardware
VirtIO-GPU

Step 2: 3D Driver

- Same system, but with DxgkInitialize
- More callbacks
- Shady constraints

Strategy ?

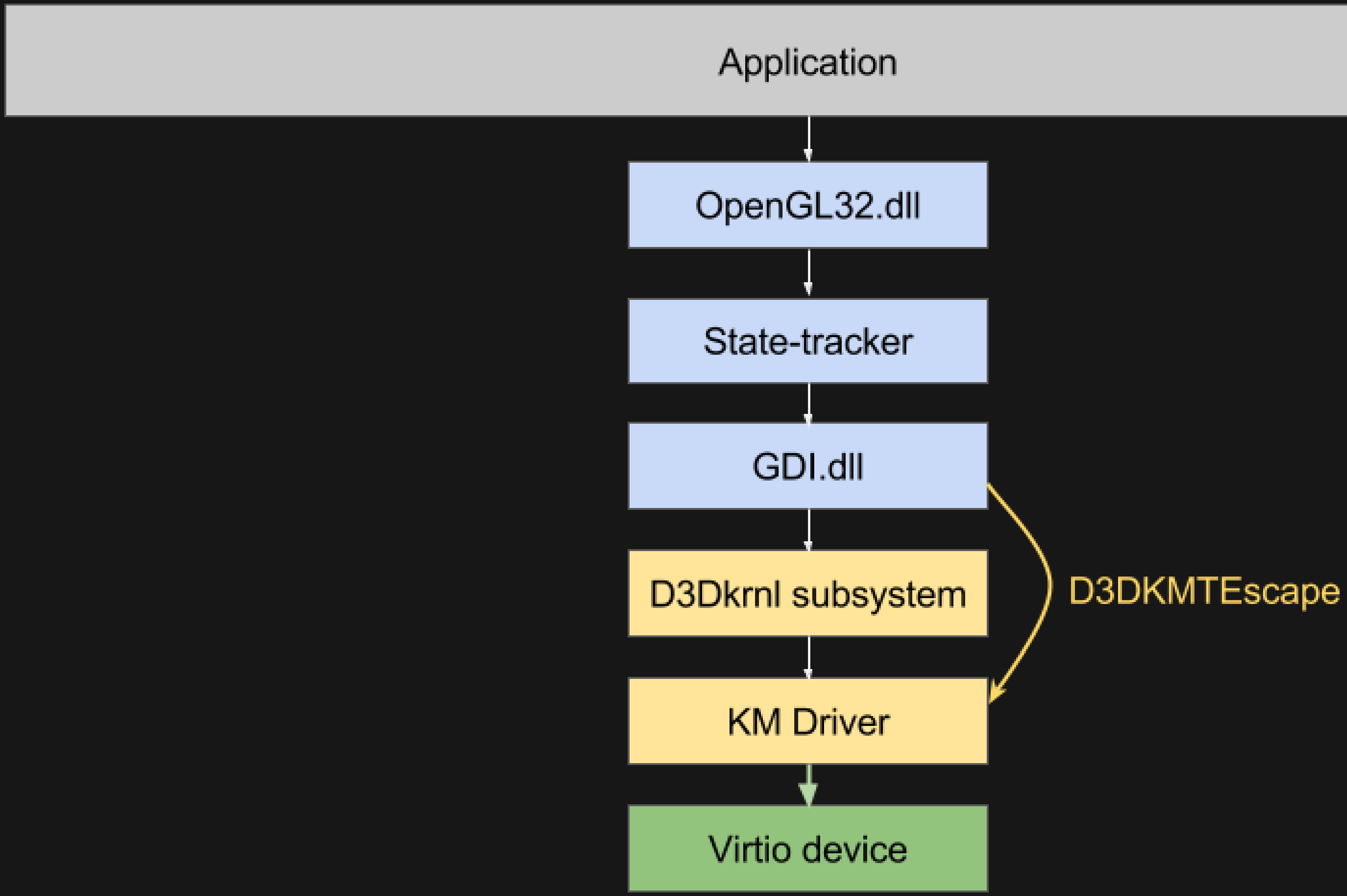
Poke the bear until it works

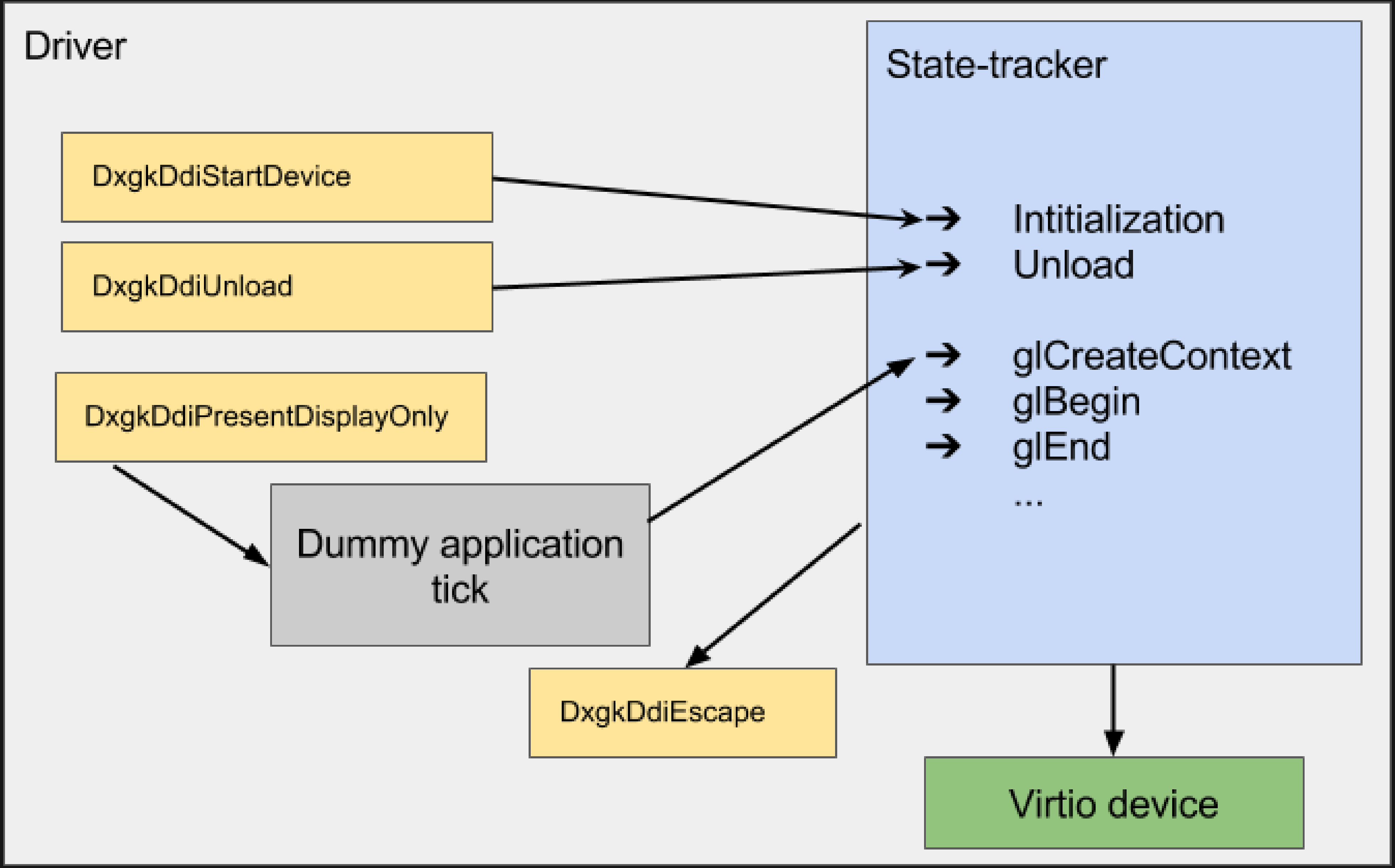
- Results ? None.
- Windows loads the driver, talks, then unloads it.
- No error code, nothing

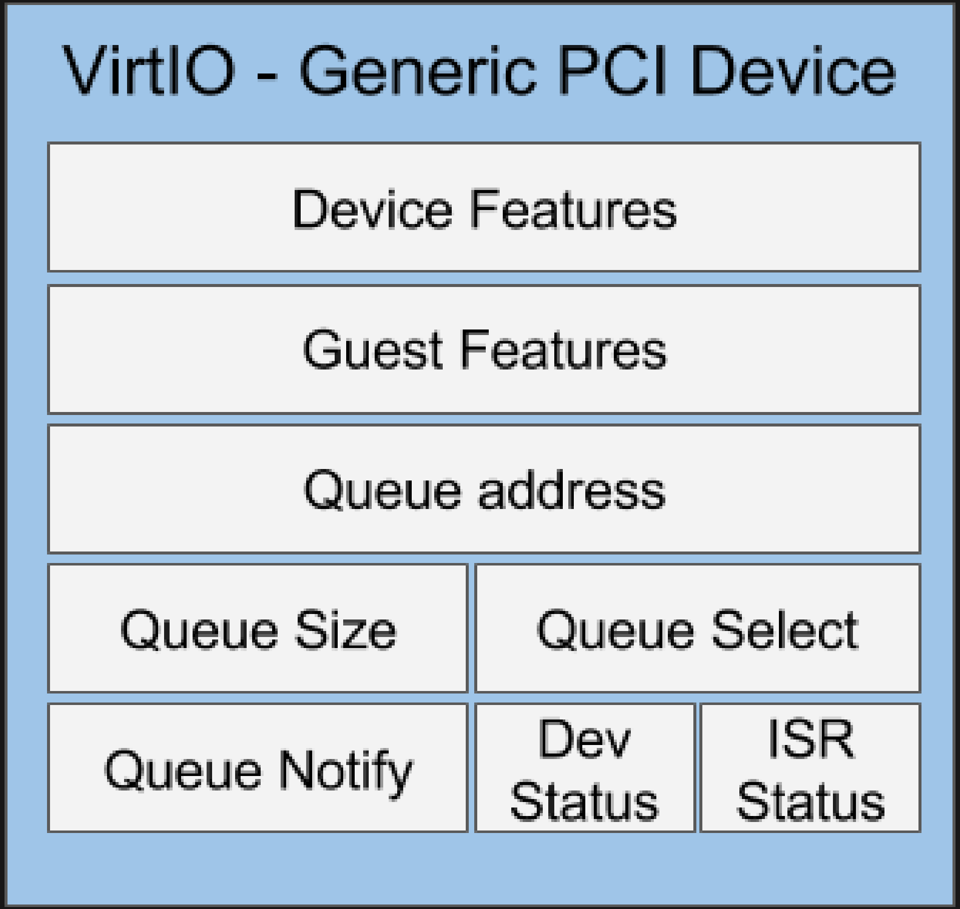
Step 2: ~~3D Driver~~ In-kernel OpenGL !

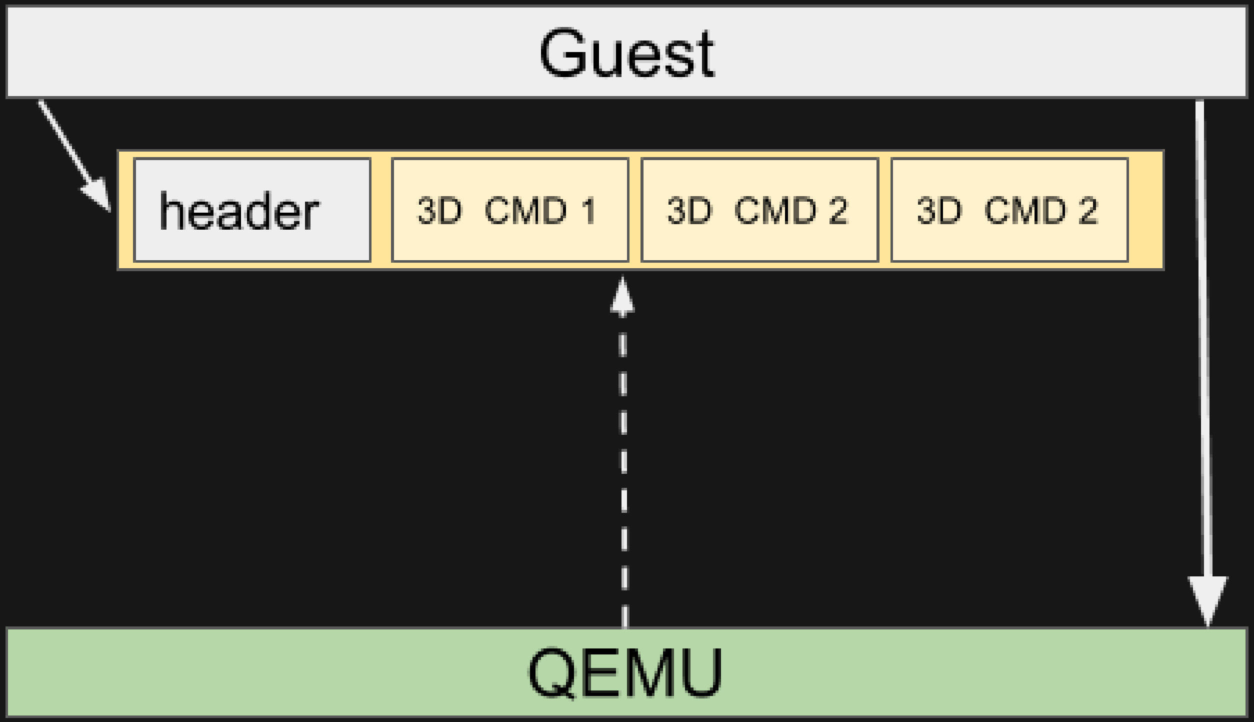
- Simulate an OpenGL application in our driver
- Be able to test Virgil3D commands and a state tracker
- Once 3D driver works, just move some code to userland.

In-kernel OpenGL




Driver
DxgkDdiStartDevice
DxgkDdiUnload
DxgkDdiPresentDisplayOnly
Dummy application tick
DxgkDdiEscape
State-tracker
→ Intitialization
→ Unload
→ glCreateContext
→ glBegin
→ glEnd
...
Virtio device


VirtIO - Generic PCI Device
Device Features
Guest Features
Queue address
Queue Size
Queue Select
Queue Notify
Dev Status
ISR Status


Guest
header
3D CMD 1
3D CMD 2
3D CMD 2
QEMU

- VIRGL commands are sent through a queue
- IRQs are used for fences, cursor… notifs !

VIRGL_CMD_CREATE_CTX
VIRGL_CMD_DESTROY_CTX
VIRGL_CMD_CREATE_RESOURCE_2D
VIRGL_CMD_CREATE_RESOURCE_3D
VIRGL_CMD_SUBMIT_3D
…

$\searrow$

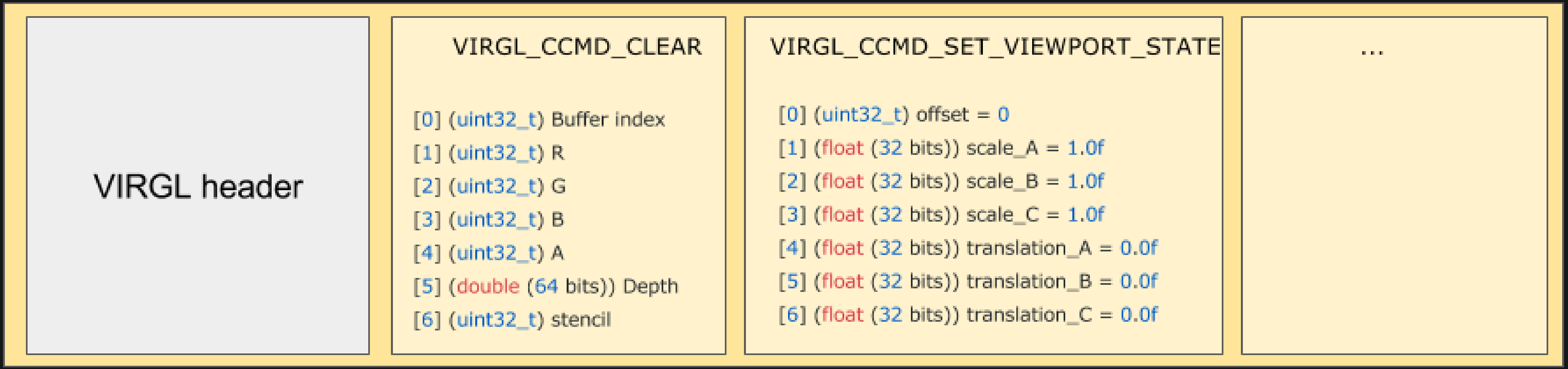
VIRGL_CCMD_CREATE_SUB_CTX
VIRGL_CCMD_MAKE_CURRENT_SUB_CTX
VIRGL_CCMD_ATTACH_RESOURCE_CTX
…

```
struct virtio_gpu_ctrl_hdr
{
    uint32_t type;
    uint32_t flags;
    uint64_t fence_id;
    uint32_t ctx_id;
    uint32_t padding;
};
```

VIRTIO_GPU_CMD_SUBMIT_3D

```
struct virtio_3d_cmd
{
    uint16_t header;
    uint16_t opt;
    uint32_t length;
};
```

VIRTIO_GPU_CMD_SUBMIT_3D



VIRGL header

VIRGL_CCMD_CLEAR

[0] (uint32_t) Buffer index
[1] (uint32_t) R
[2] (uint32_t) G
[3] (uint32_t) B
[4] (uint32_t) A
[5] (double (64 bits)) Depth
[6] (uint32_t) stencil

VIRGL_CCMD_SET_VIEWPORT_STATE

[0] (uint32_t) offset = 0
[1] (float (32 bits)) scale_A = 1.0f
[2] (float (32 bits)) scale_B = 1.0f
[3] (float (32 bits)) scale_C = 1.0f
[4] (float (32 bits)) translation_A = 0.0f
[5] (float (32 bits)) translation_B = 0.0f
[6] (float (32 bits)) translation_C = 0.0f

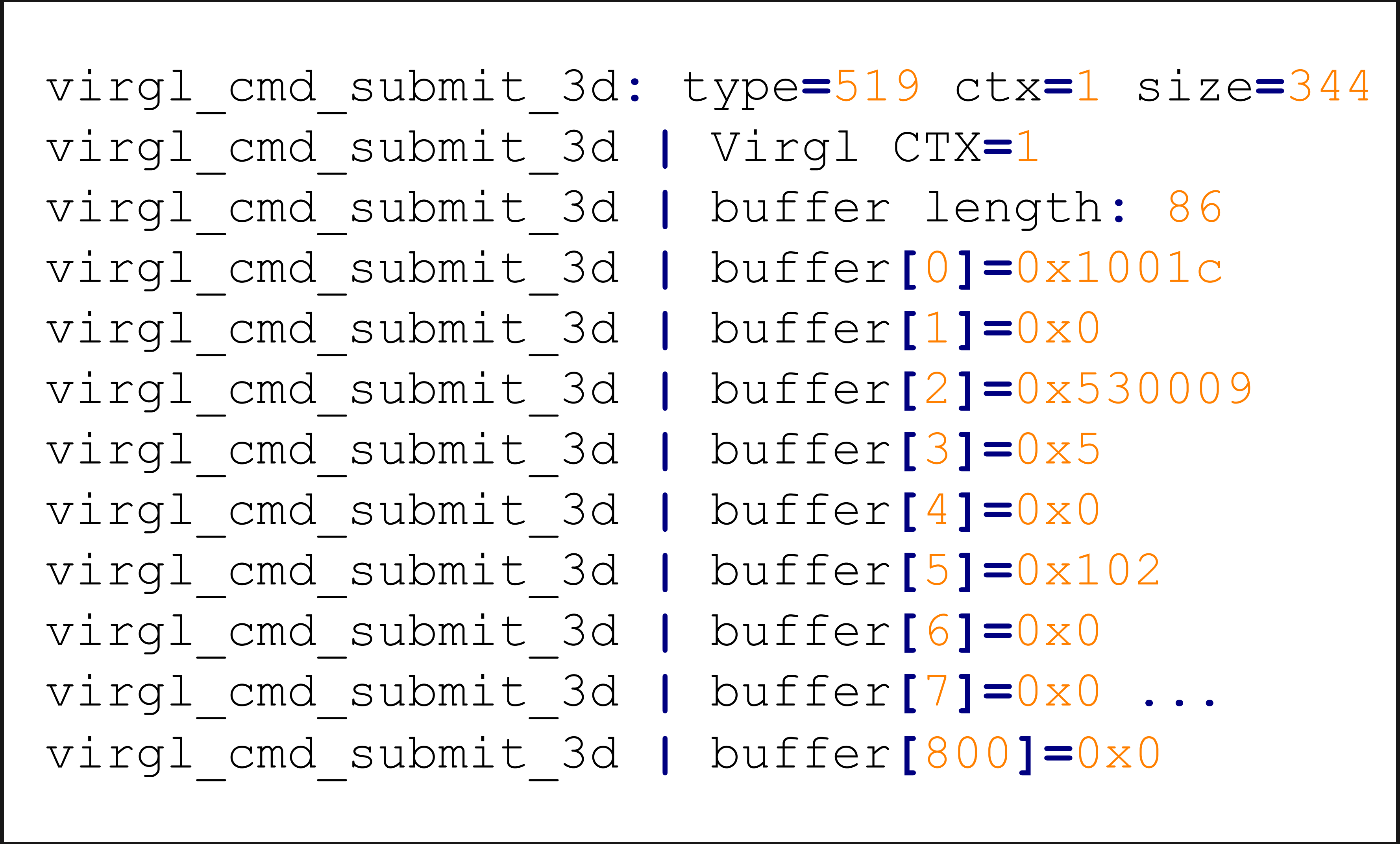
...

Sanity check

- Send something
- Add debug everywhere
- **Check** return values

Learn how to ICD

- Fedora has a working driver
- X server generates a lot of noise
- KMS application can improve noise/signal ratio



```
virgl_cmd_submit_3d: type=519 ctx=1 size=344
virgl_cmd_submit_3d | Virgl CTX=1
virgl_cmd_submit_3d | buffer length: 86
virgl_cmd_submit_3d | buffer[0]=0x1001c
virgl_cmd_submit_3d | buffer[1]=0x0
virgl_cmd_submit_3d | buffer[2]=0x530009
virgl_cmd_submit_3d | buffer[3]=0x5
virgl_cmd_submit_3d | buffer[4]=0x0
virgl_cmd_submit_3d | buffer[5]=0x102
virgl_cmd_submit_3d | buffer[6]=0x0
virgl_cmd_submit_3d | buffer[7]=0x0 ...
virgl_cmd_submit_3d | buffer[800]=0x0
```

In-kernel OpenGL

```
[&] SET SUB_CTX H=0
[*] INLINE WRITE H=5
[+] NEW_OBJECT H=2 TYPE=DSA
[-] BIND_OBJECT H=2 TYPE=DSA
[+] NEW_OBJECT H=3 TYPE=SHADER
[-] BIND SHADER H=3 TYPE=FRAGMENT_SHADER
[+] NEW_OBJECT H=5 TYPE=RASTERIZER
[-] BIND_OBJECT H=5 TYPE=RASTERIZER
[-] SET POLYGON_STIPPLE
…
[*] CLEAR H=4
[$] END CMDBUFFER OBJECTS

[1] FRAMEBUFFER (1024x768x1)
[0] ZBUFFER (-1x-1x-1)
[2] DSA (32x32x1)
[3] FRAGMENT_SHADER (65536x1x1)
[4] VERTEX_SHADER (4096x2160x1)
[5] VERTEX_BUFFER (864x1x1)
[6] BLEND (8x1x1)
==========
```

Learn how to ICD

Example: Shader creation

- Create a 3D resource: type shader, proper size
- Attach guest buffer to the resource (backing attachment)
- Attach resource to the correct context
- Bind shader to the correct sub-context

Shaders ? Wow wow wow !

I love this project

- OpenGL speaks GLSL
- Mesa takes GLSL, translates it to TGSI
- Backend speaks TGSI

- [GUEST] OpenGL speaks GLSL
- [GUEST] Mesa takes GLSL, translates it to TGSI
- [GUEST] Virtio-gpu speaks TGSI

- [GUEST] OpenGL speaks GLSL
- [GUEST] Mesa takes GLSL, translates it to TGSI
- [GUEST] Virtio-gpu speaks TGSI

- [HOST] Mesa takes GLSL, translates it to TGSI
- [HOST] Backend speaks TGSI

I love this project

- [GUEST] OpenGL speaks GLSL
- [GUEST] Mesa takes GLSL, translates it to TGSI
- [GUEST] Virtio-gpu speaks TGSI
- [HOST] Virtio-gpu translates TGSI to GLSL
- [HOST] Mesa takes GLSL, translates it to TGSI
- [HOST] Backend speaks TGSI

I love this project

- We can find both TGSI and ASCII GLSL on the V-GPU
- Can dump it, and use it on Windows

Basic state tracker

- glContext
- glViewport
- glClear
- glBegin
- glEnd
- glVertex3i
- glColor2i
- glFlush

- Context creation
  - Create sub-context
  - Set it as active
  - Setup inner state on guest

- Vertex creation
  - Allocate buffer on guest
  - Create vertex buffer resource
  - Bind it to the proper sub context

- Rendering
  - Setup default DSA, rasterizer, shaders
  - Bind vertex and uniform buffer to the proper sub-context
  - Call draw VBO command

What’s next ?

What’s next ?

- Make it works ?
- TGSI <-> GLSL translation on host
- Documentation ++

What to improve

- Switch to Vulkan ?
- Add GPGPU features

Question ?

Links

https://virgil3d.github.io
https://github.com/vrozenfe/virtio-gpu-win
https://www.github.com/Keenuts/virtio-gpu-documentation
https://www.github.com/Keenuts/virtio-gpu-win
https://github.com/Keenuts/virtio-gpu-win-icd

Thank you !

Links

https://virgil3d.github.io
https://github.com/vrozenfe/virtio-gpu-win
https://www.github.com/Keenuts/virtio-gpu-documentation
https://www.github.com/Keenuts/virtio-gpu-win
https://github.com/Keenuts/virtio-gpu-win-icd