VKMS: Virtual Kernel Modesetting

Haneen Mohammed, Rodrigo Siqueira

About Us and Our Mentors

Haneen Mohammed

Rodrigo Siqueira

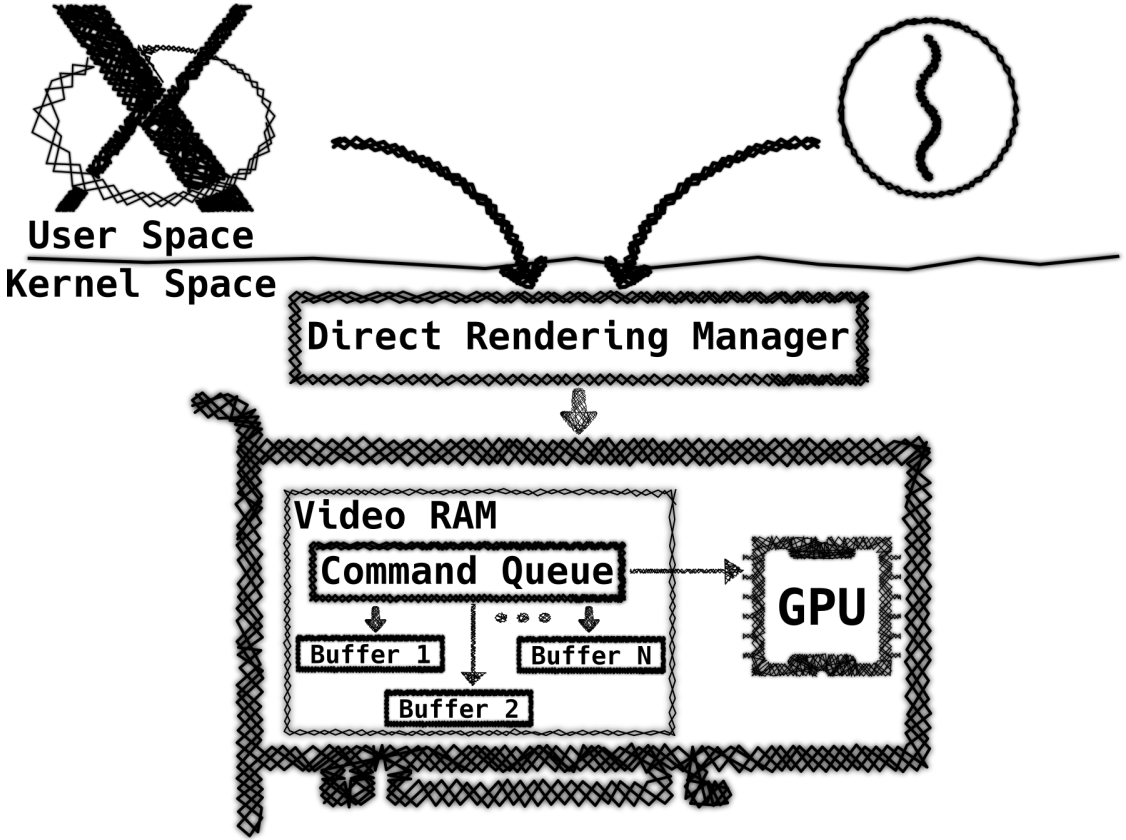
Google
Summer of Code

About Us and Our Mentors

Mentors:

- Daniel Vetter
- Gustavo Padovan
- Sean Paul
- All the dri-devel community (❤)

Kernel Mode-Setting



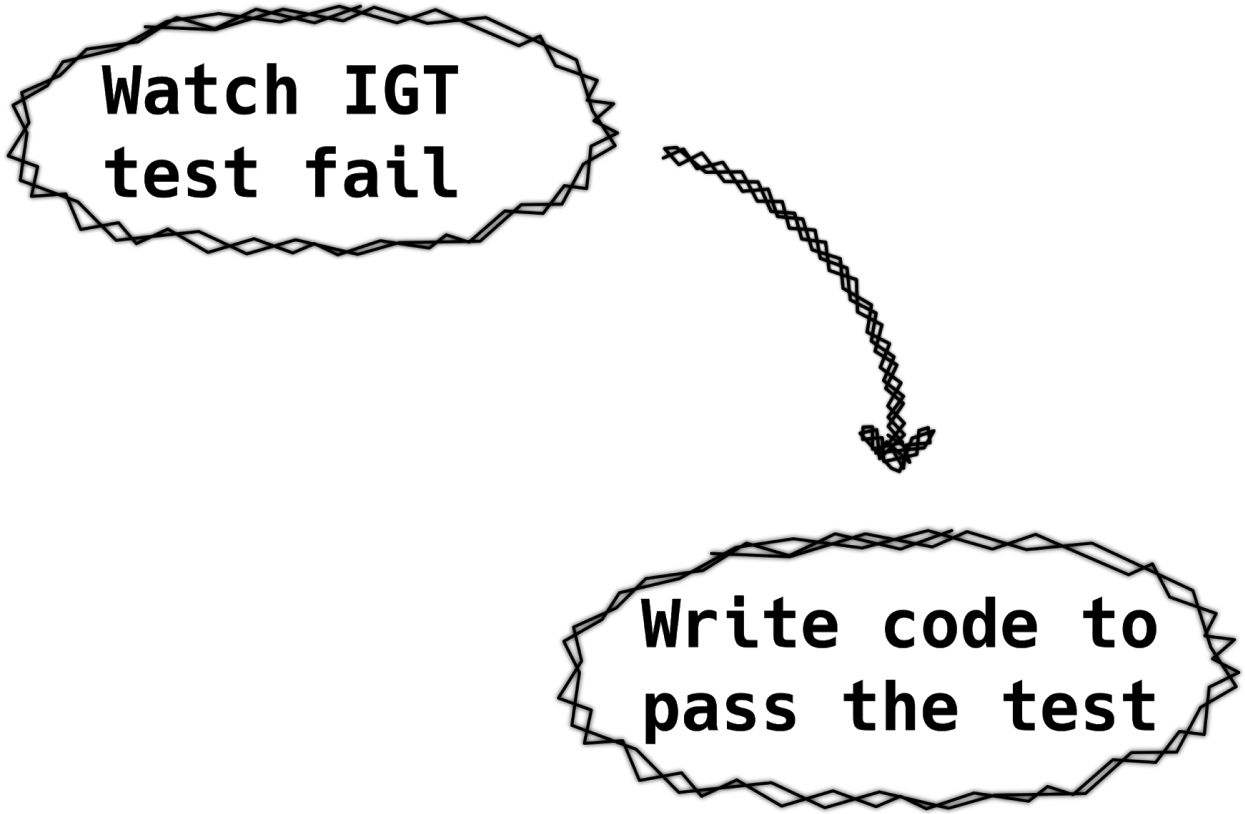
Why do we need VKMS?

- Increase DRM test coverage
- It may work as a tool to help graphics developers

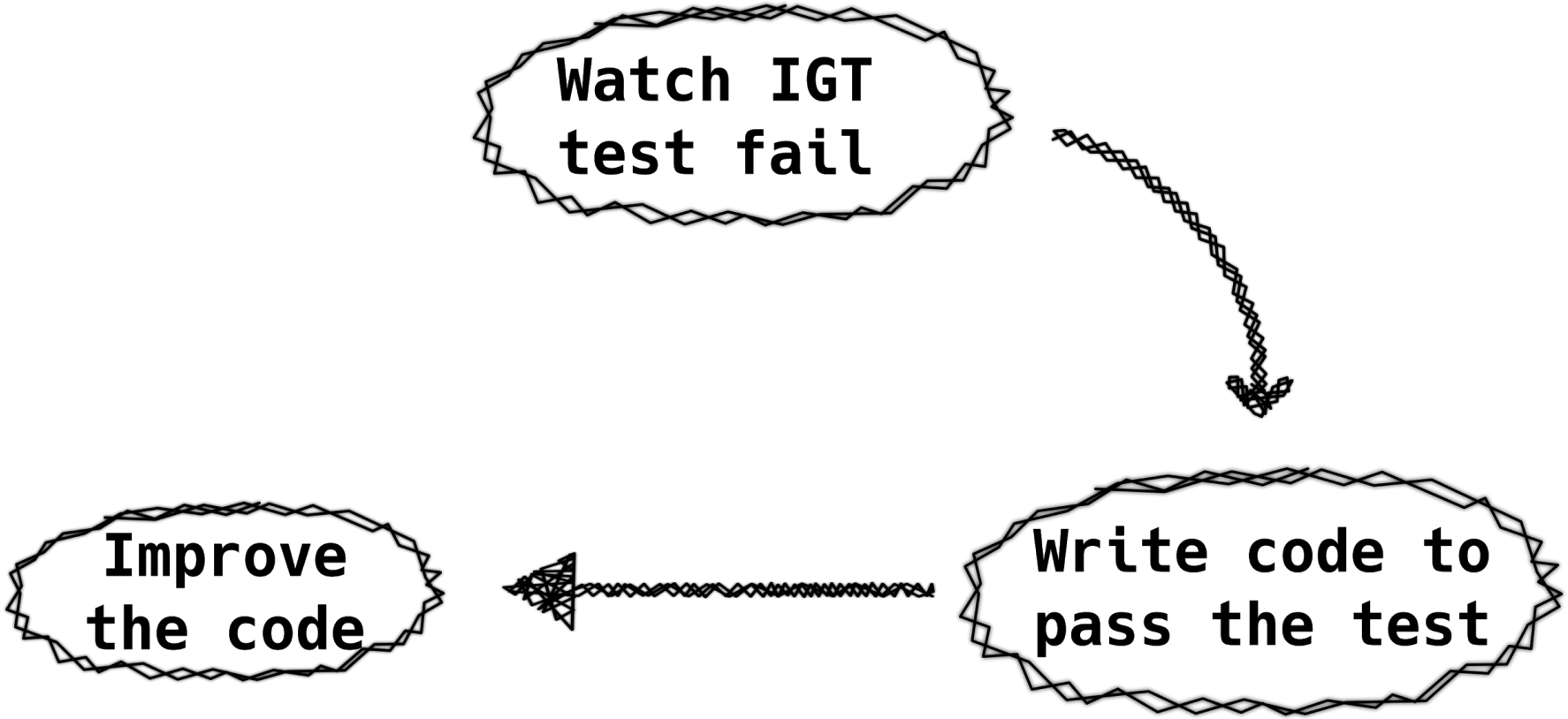
Development Cycle

Development Cycle

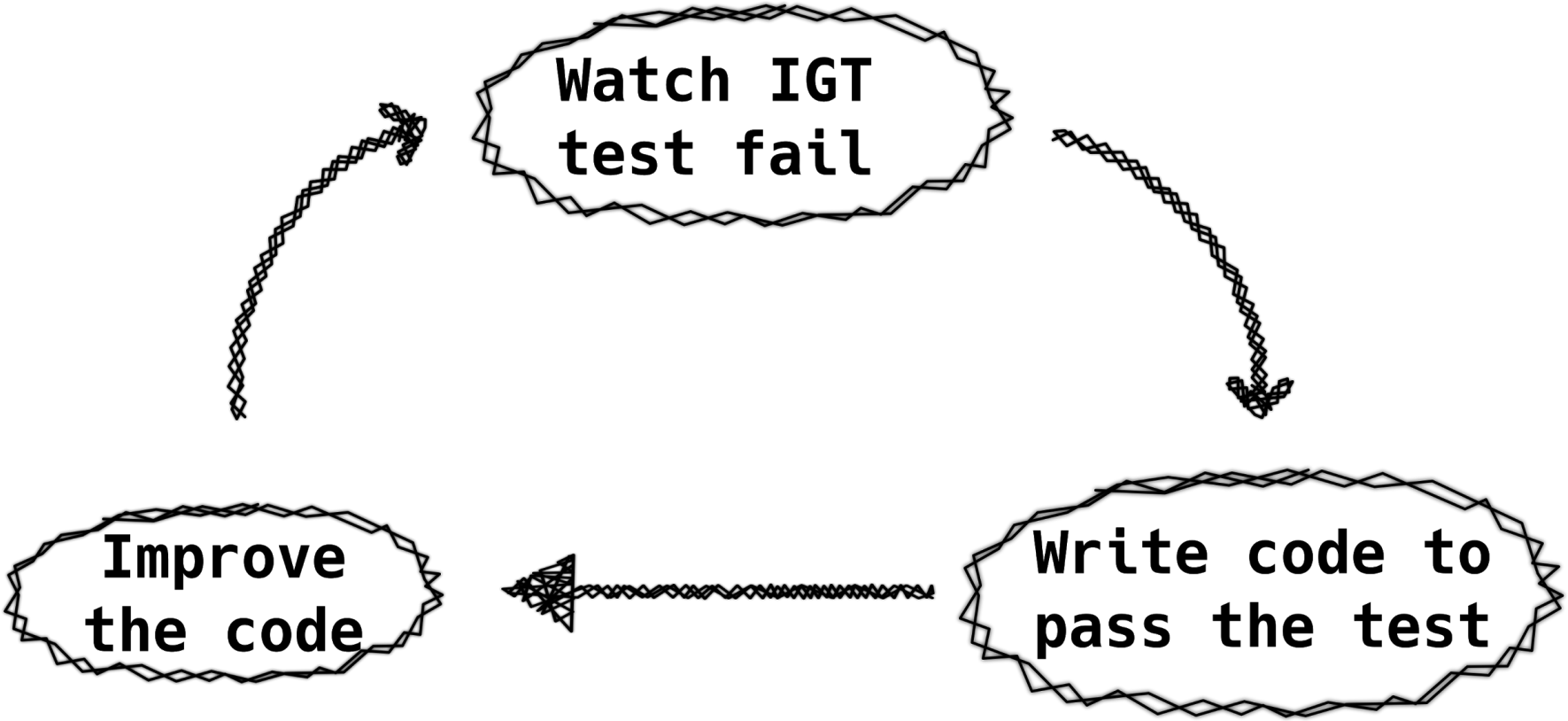
Development Cycle



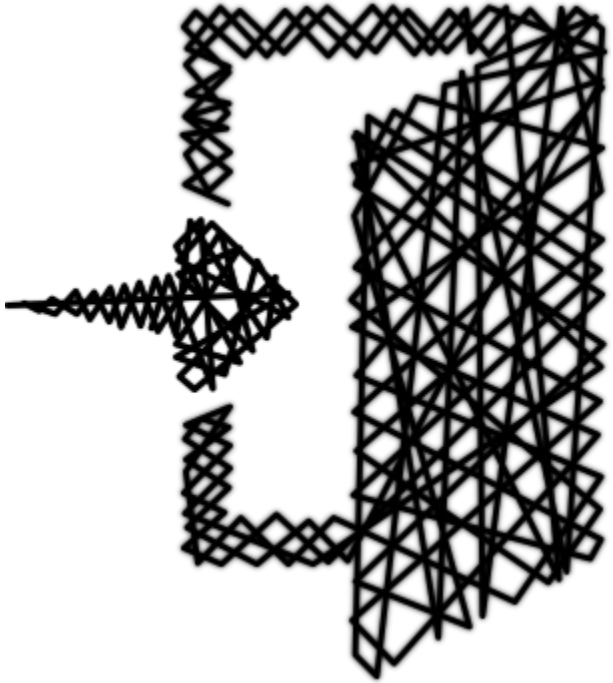
Development Cycle



Development Cycle

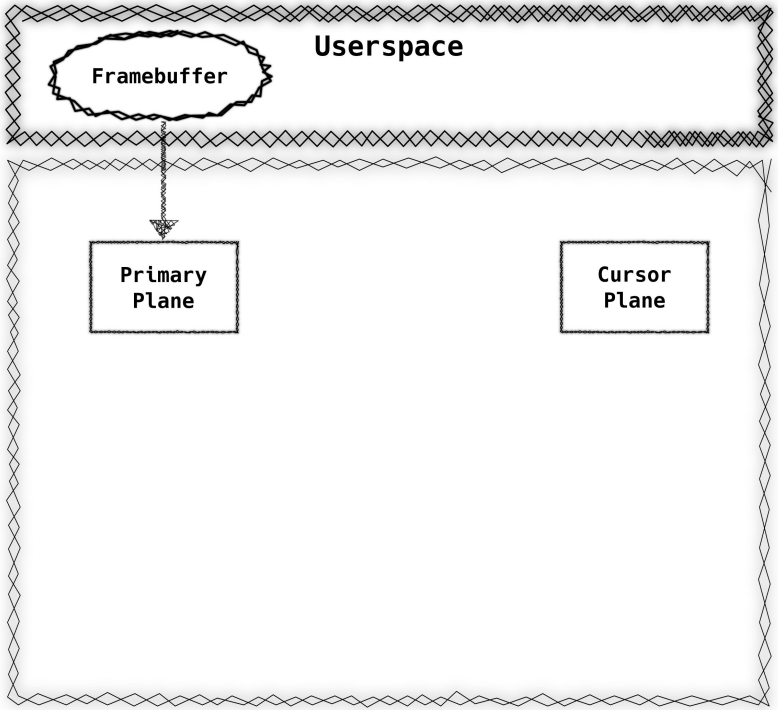


Inside VKMS

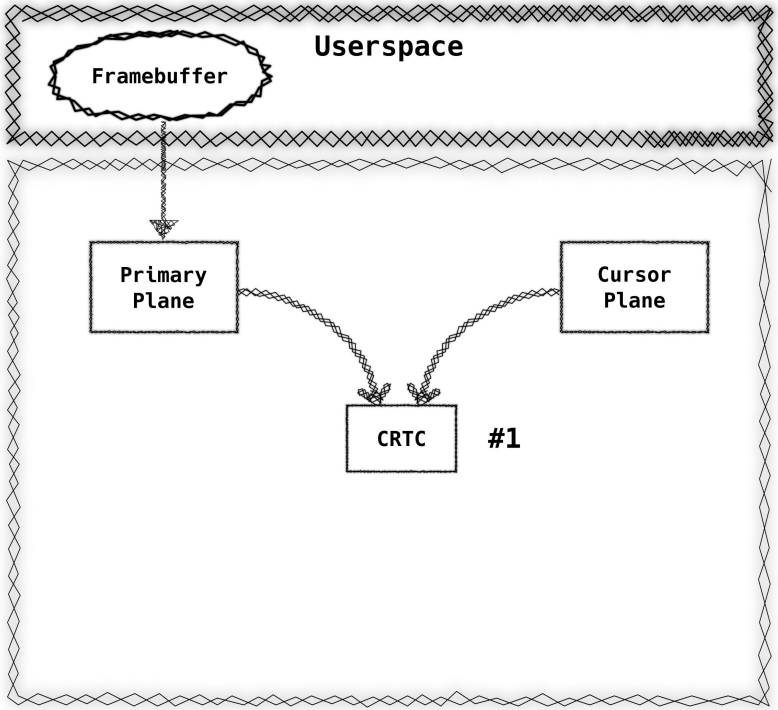


Basic Features

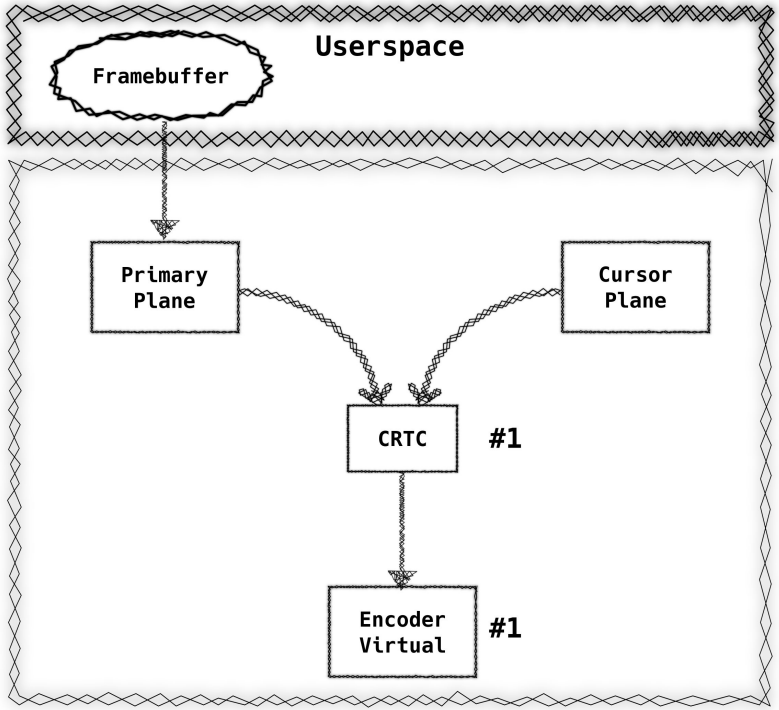
Basic Features



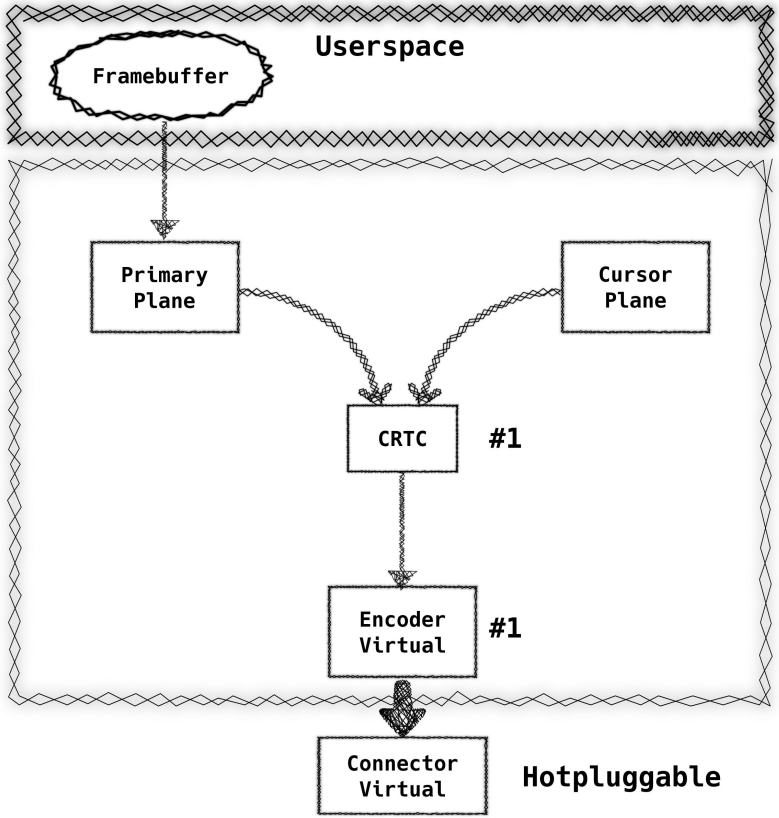
Basic Features



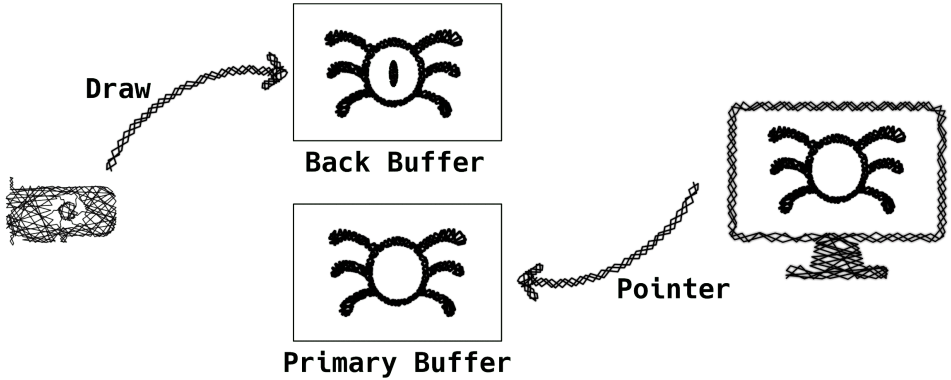
Basic Features



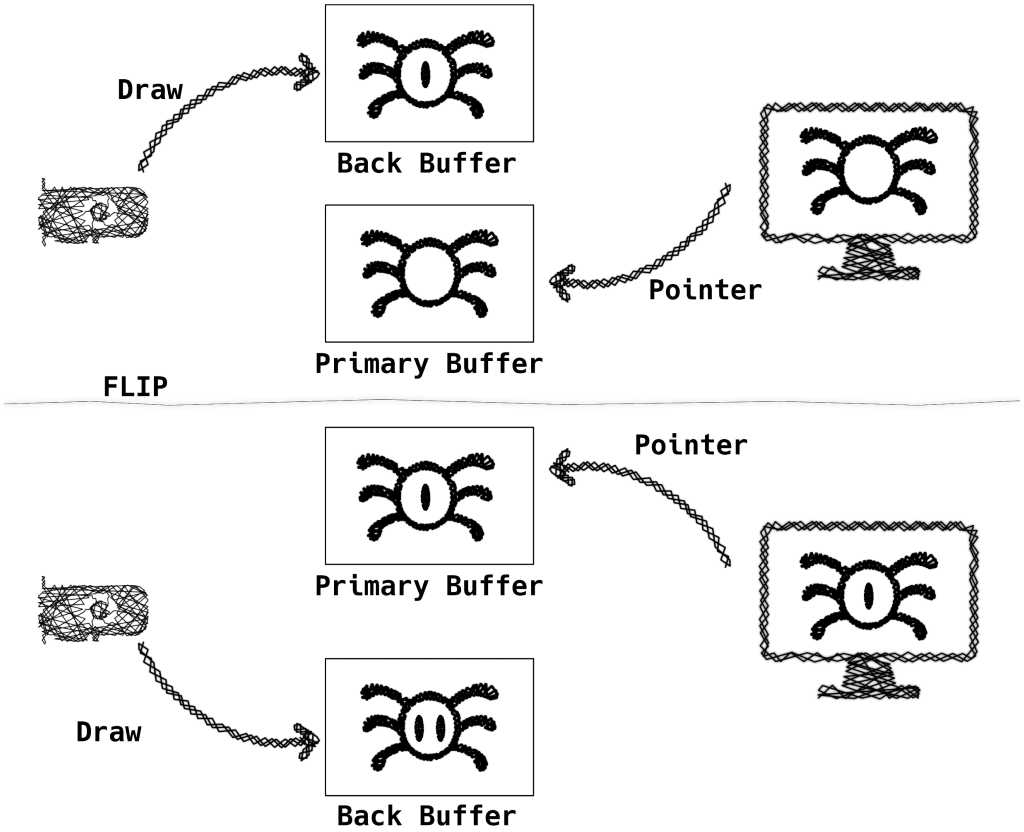
Basic Features



Page Flip and Vblank



Page Flip and Vblank

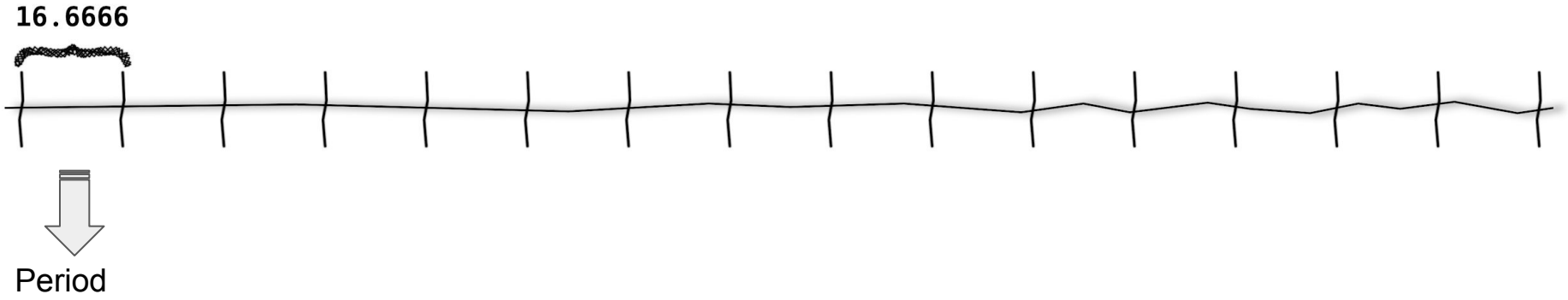


Vblank on VKMS

Simulating Vblank with Hrtimers

Vblank on VKMS

Simulating Vblank with Hrtimers

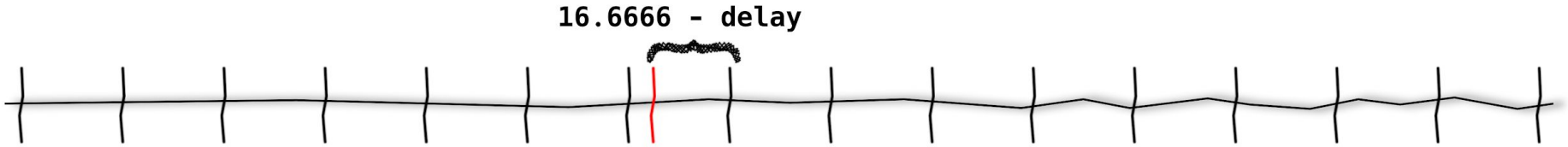


Vblank on VKMS

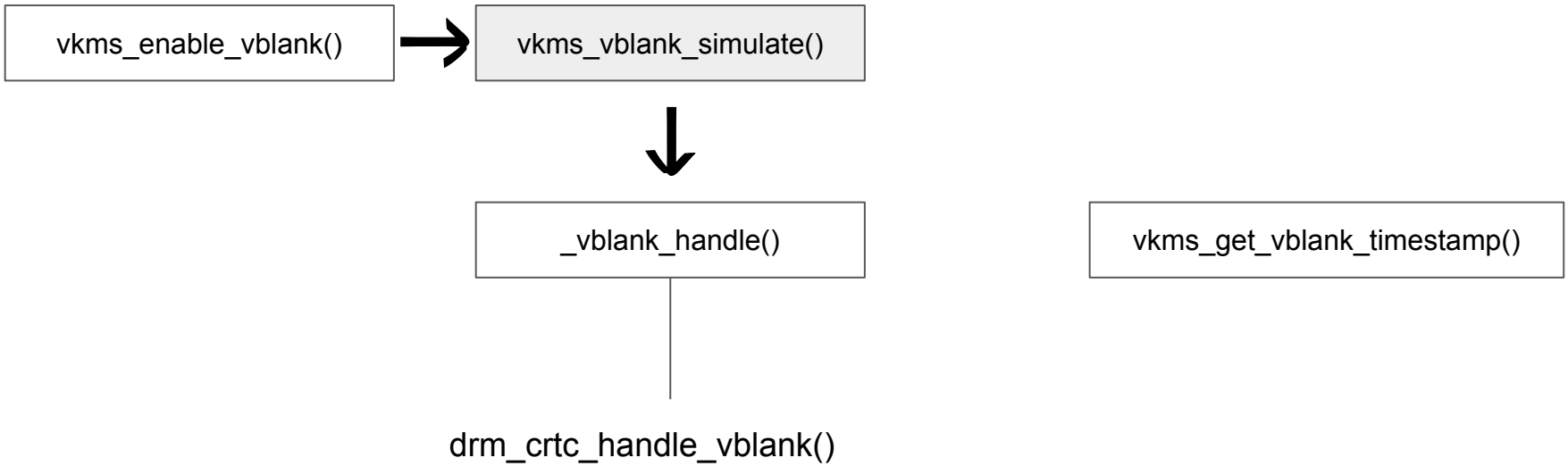
Simulating Vblank with Hrtimers

Vblank on VKMS

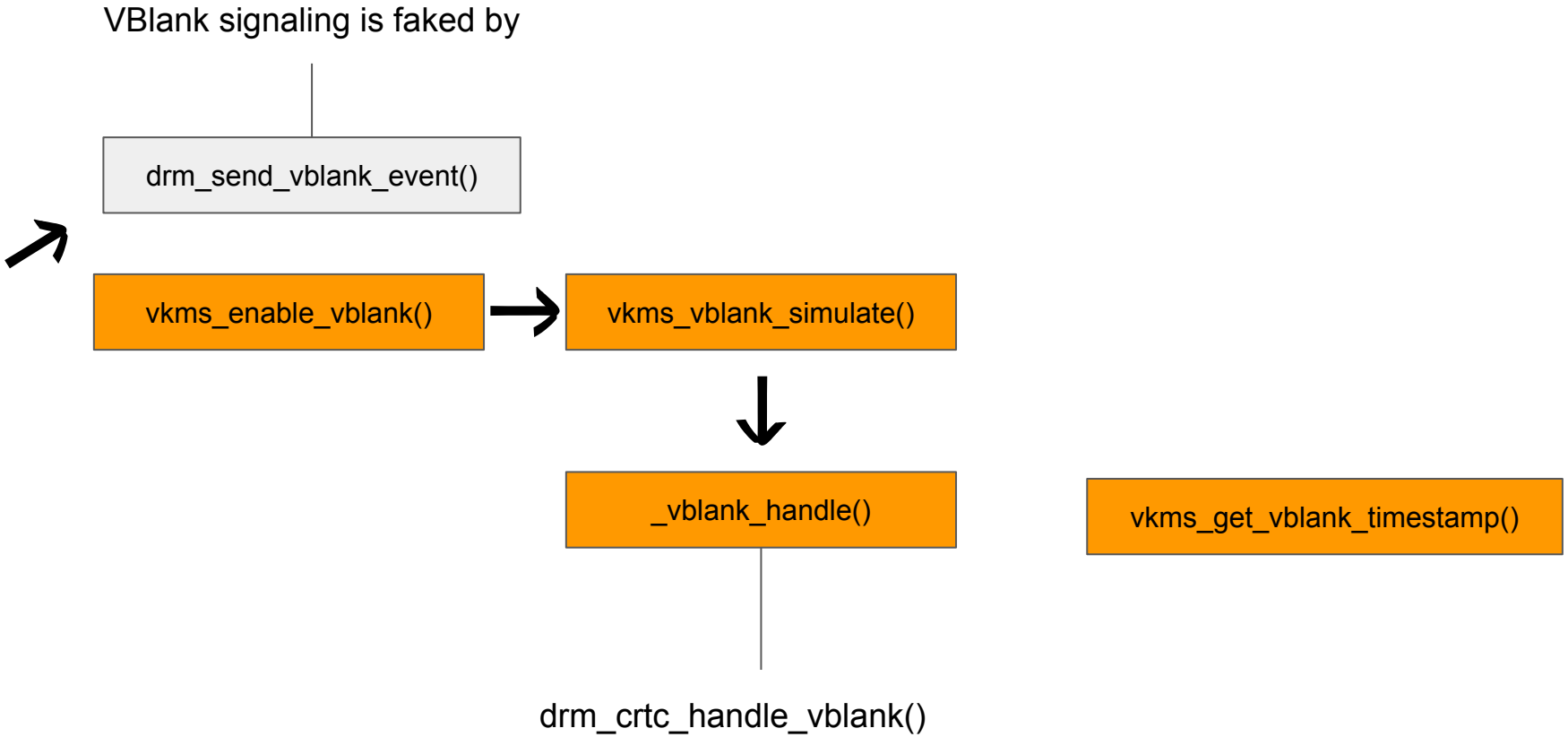
Simulating Vblank with Hrtimers



Vblank on VKMS



VKMS without VBlank (Patch)



Vblank on VKMS

```
[siqueira@atma igt-host]$ sudo ./tests/kms_flip  --run-subtest basic-plain-flip
IGT-Version: 1.22-ge29bd428 (x86_64) (Linux: 4.18.0-rc3-VKMS-RULES+ x86_64)
Using monotonic timestamps
DRM_IOCTL_I915_GEM_APERTURE failed: Invalid argument
Assuming 131072kB available aperture size.
May lead to reduced performance or incorrect rendering.
get chip id failed: -1 [22]
param: 4, val: 0
Beginning basic-plain-flip on pipe A, connector Virtual-1
  1024x768 60 1024 1048 1184 1344 768 771 777 806 0xa 0x48 65000
................................................................................
................................................................................
................................................................................
........................
basic-plain-flip on pipe A, connector Virtual-1: PASSED

Subtest basic-plain-flip: SUCCESS (10.185s)
[siqueira@atma igt-host]$
```

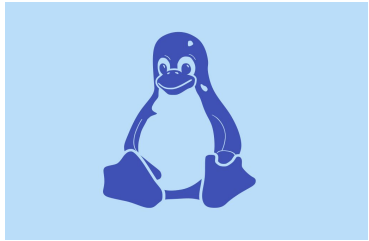
Vblank on VKMS

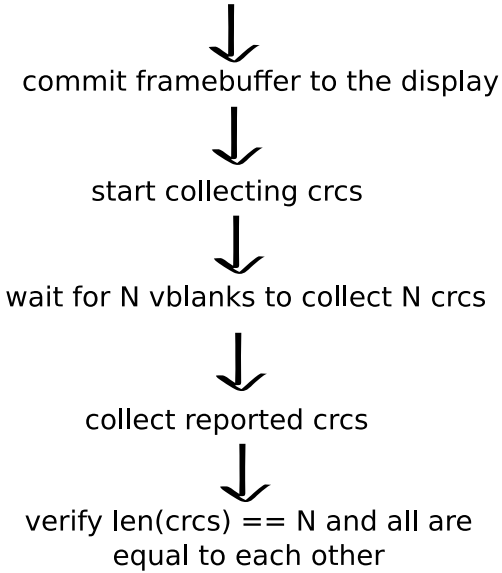
```
[siqueira@atma igt-host]$ sudo ./tests/kms_flip  --run-subtest wf_vblank-ts-chec
IGT-Version: 1.22-ge29bd428 (x86_64) (Linux: 4.18.0-rc3-VKMS-RULES+ x86_64)
Using monotonic timestamps
DRM_IOCTL_I915_GEM_APERTURE failed: Invalid argument
Assuming 131072kB available aperture size.
May lead to reduced performance or incorrect rendering.
get chip id failed: -1 [22]
param: 4, val: 0
Beginning wf_vblank-ts-check on pipe A, connector Virtual-1
  1024x768 60 1024 1048 1184 1344 768 771 777 806 0xa 0x48 65000
Expected frametime: 16666us; measured 16665.6us +- 0.500us accuracy 0.01%
................................................................................
................................................................................
..................
wf_vblank-ts-check on pipe A, connector Virtual-1: PASSED

Subtest wf_vblank-ts-check: SUCCESS (30.582s)
[siqueira@atma igt-host]$
```

CRC API Support With VKMS

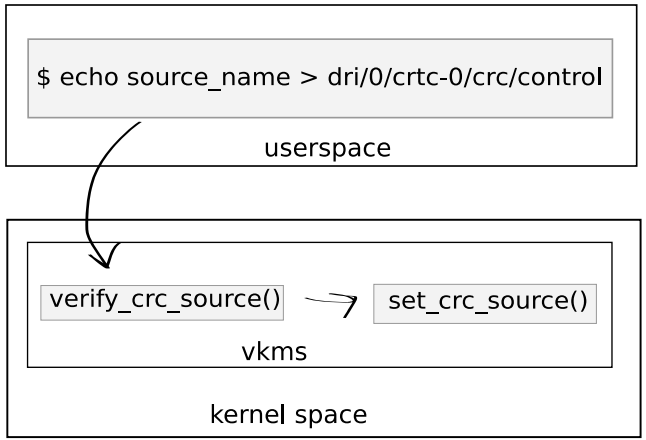
- value = **crc**(displayed frame)
- Goal: Pass the following tests from the IGT test suite
  - **kms_pipe_crc_basic**
  - **kms_cursor_crc**

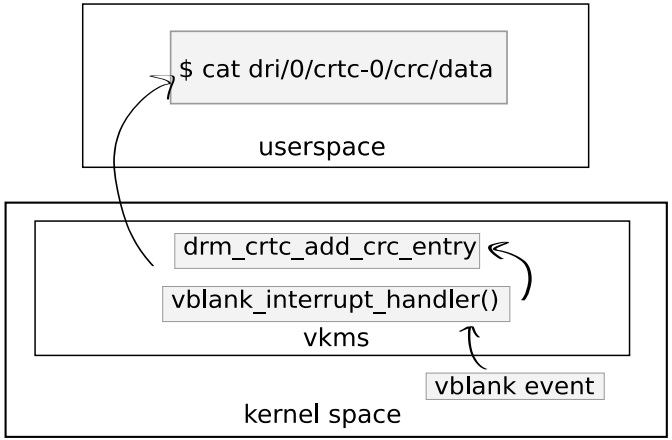




CRC API Support in DRM

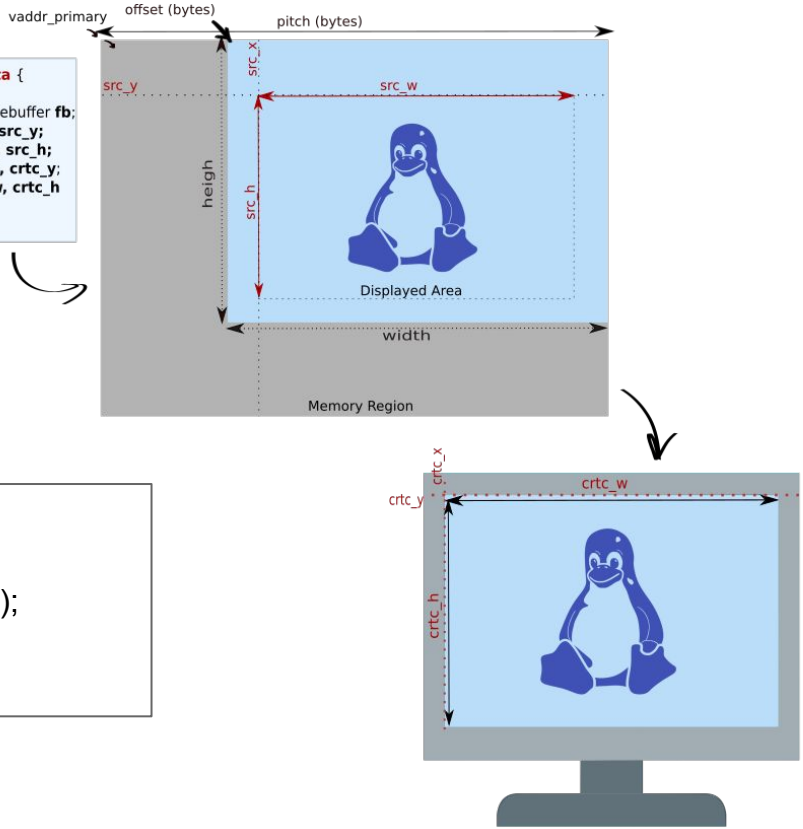
- Add the following to the drm_crtc vfuns table:
  - **verify_crc_source**()
  - **set_crc_source**()
- **drm_crtc_add_crc_entry**()
- CRC API exposed at **/sys/kernel/debug/dri/0/crtc-N/crc** -> **control** and **data** files





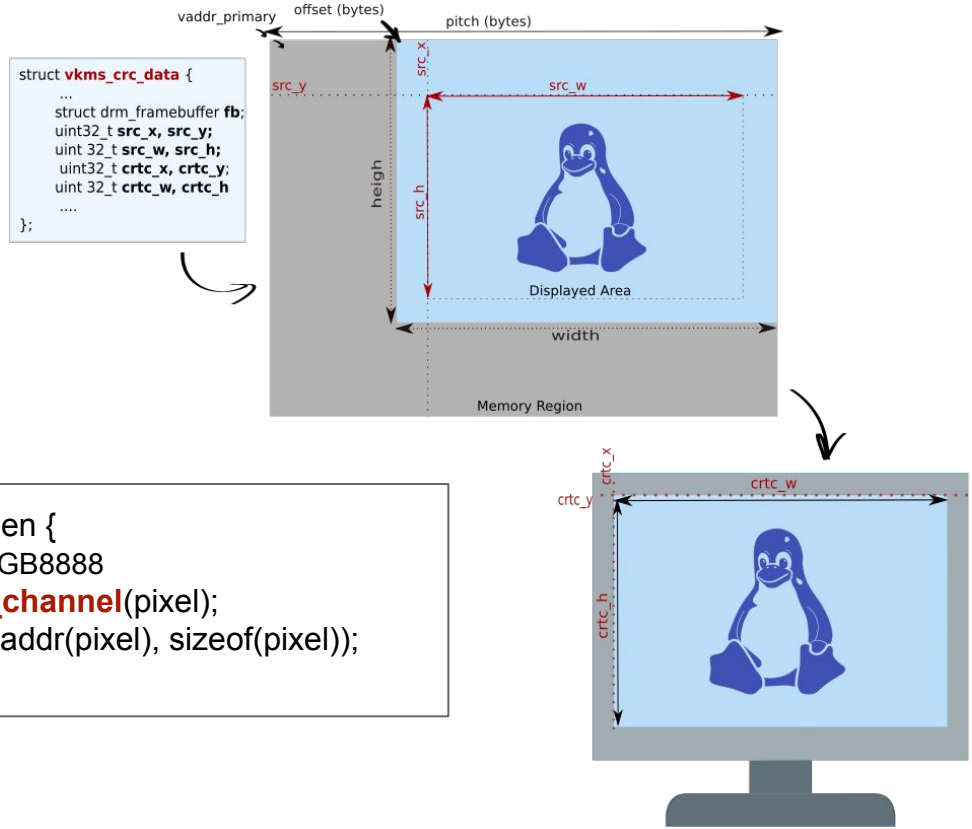
Computing CRC

```
struct vkms_crc_data {
        ...
        struct drm_framebuffer fb;
        uint32_t src_x, src_y;
        uint 32_t src_w, src_h;
         uint32_t crtc_x, crtc_y;
        uint 32_t crtc_w, crtc_h
         ....
};
```



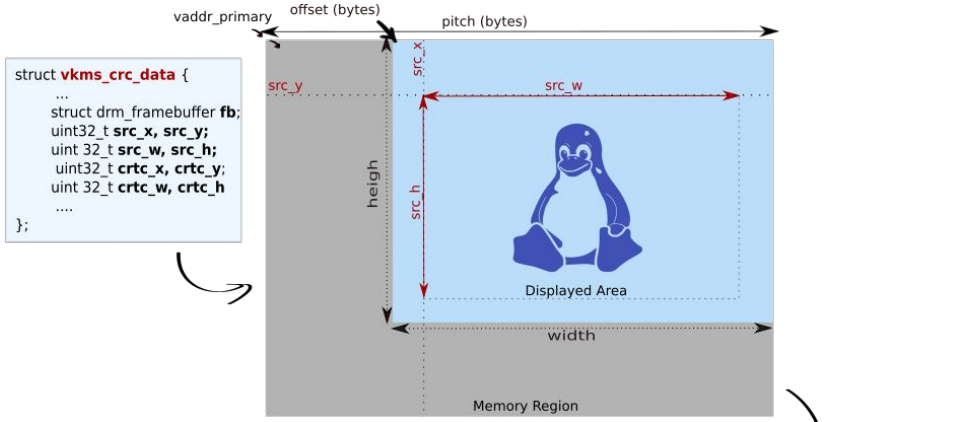

```
for each pixel visible in screen {
        // DRM_FORMAT_XRGB8888
        pixel = clear_alpha_channel(pixel);
        crc = compute_crc(crc, pixel);
}
```

Computing CRC

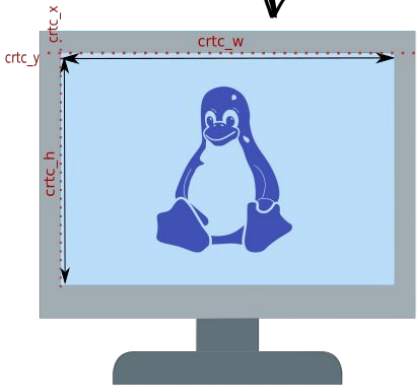



```
for each pixel visible in screen {
        // DRM_FORMAT_XRGB8888
        pixel = clear_alpha_channel(pixel);
        crc = crc32_le(crc, vaddr(pixel), sizeof(pixel));
}
```

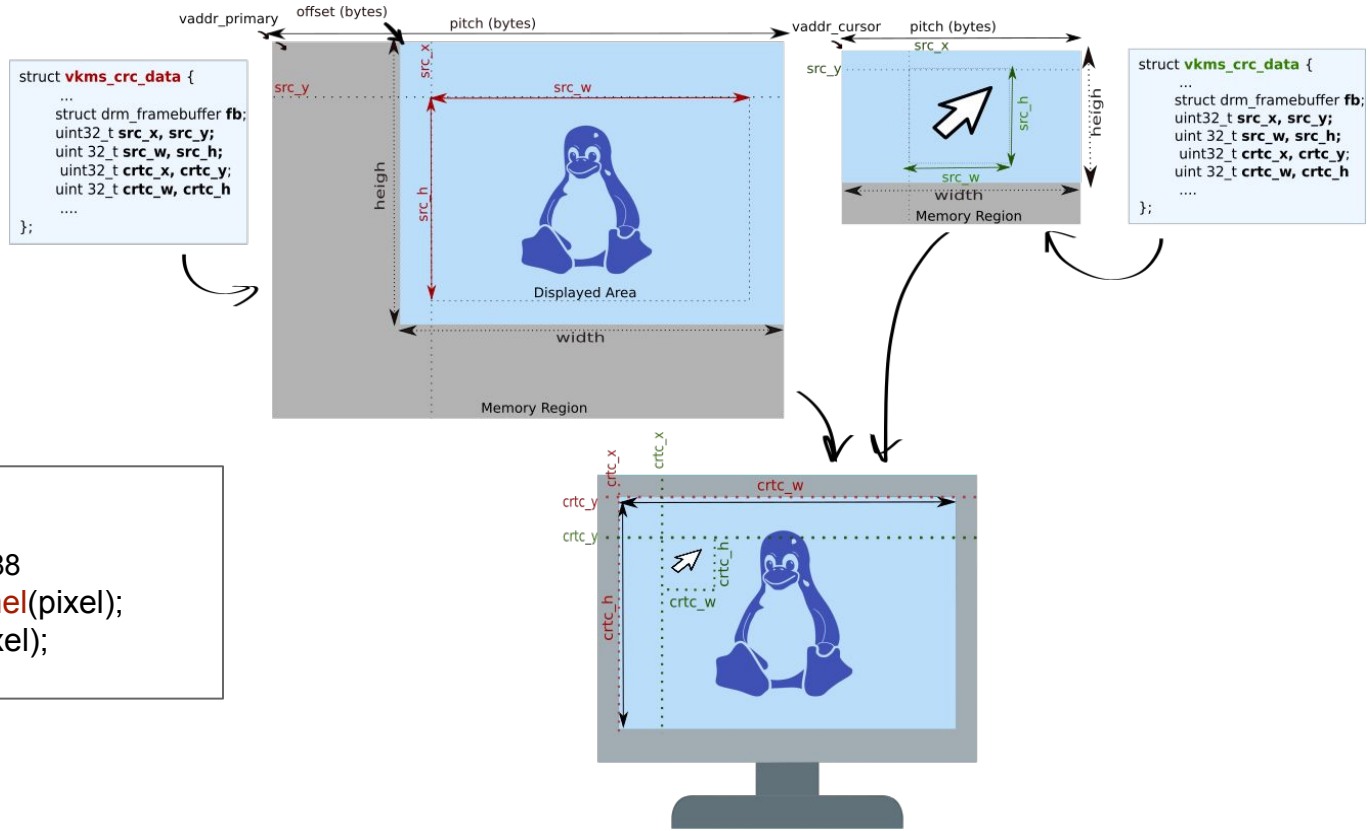
Computing CRC




```
for (i = src_y; i < src_y + src_h; ++i) {
        for (j = src_x; j < src_x + src_w; ++j) {
                v_offset = i * pitch;
                h_offset = j * cpp /* bytes per pixel */;
                src_offset = offset + v_offset + h_offset;
                memset(vaddr, src_offset + 24, 0, 8);
                crc = crc32_le(crc, vaddr + src_offset, sizeof(u32));
        }
}
```



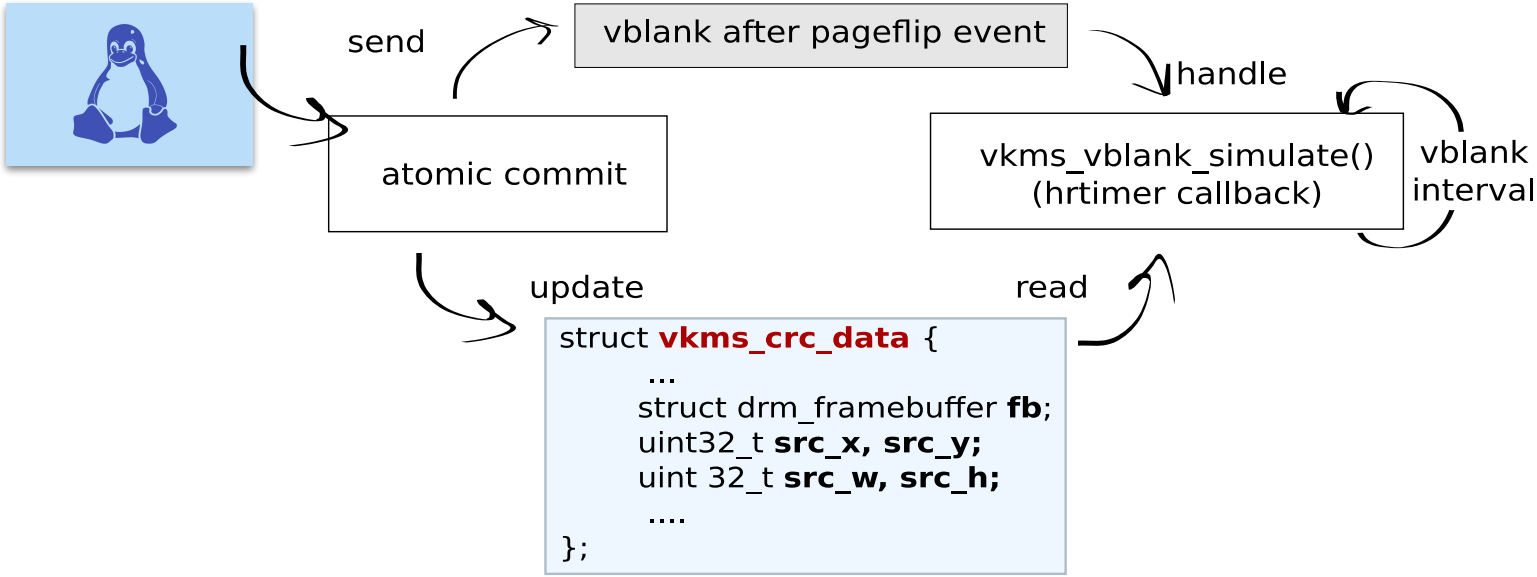
Computing CRC



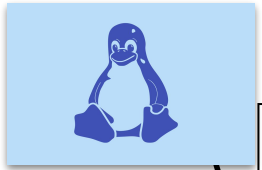

```
blend(primary, cursor);
for each pixel visible in screen {
        // DRM_FORMAT_XRGB8888
         pixel = clear_alpha_channel(pixel);
         crc = compute_crc(crc, pixel);
}
```

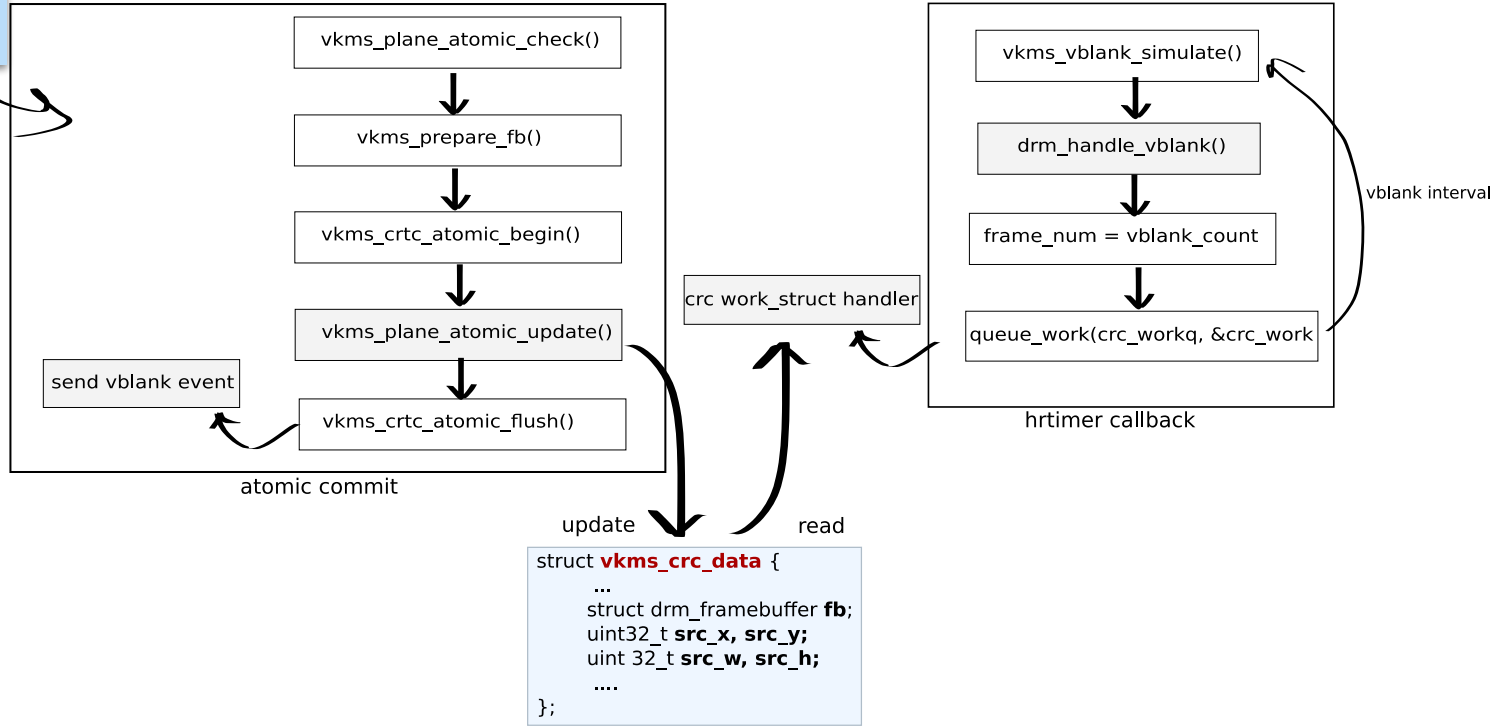
10000 Foot View

- How to synchronize framebuffer update with crc computations and flip event?

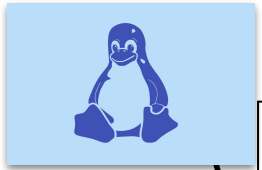


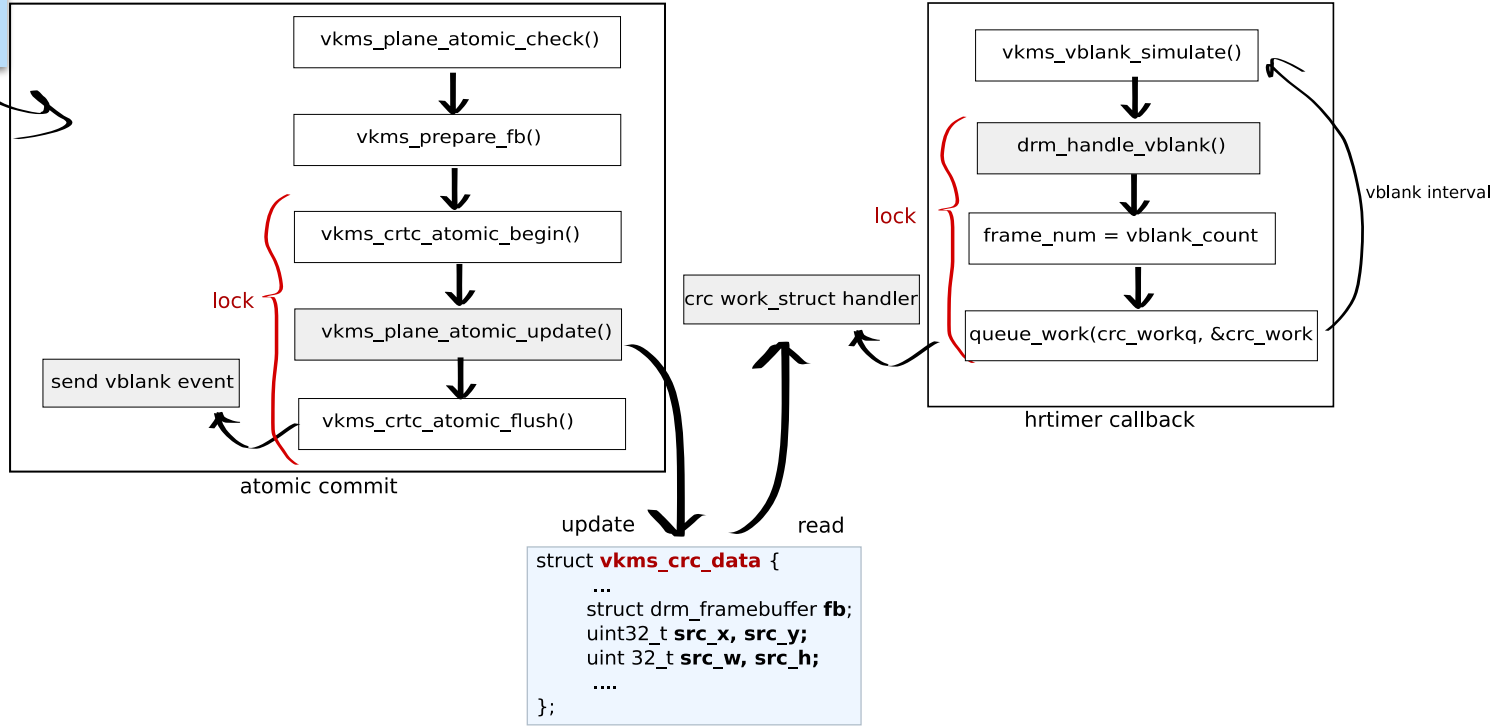
CRC API Support With VKMS (Challenges)





CRC API Support With VKMS (Solution)





CRC API Support With VKMS (results)

kms_pipe_crc_basic	kms_cursor_crc
1. bad-source 2. read-crc-pipe-A 3. read-crc-A-frame-sequence 4. nonblocking-crc-pipe-A 5. nonblocking-crc-pipe-A-frame-sequence	1. cursor-size-change 2. cursor-64x64-onscreen 3. cursor-64x64-offscreen 4. cursor-64x64-sliding 5. cursor-64x64-random 6. cursor-64x64-dpms

Conclusion and Future Works

VKMS is a working in progress project. We still have to improve:

- There are some tests related to kms_flip that fails
- There are some improvements to make at the CRC part
- Make Wayland run on top of VKMS
- Many other features

Future works:

- Probably I will get 5 extra months of work in VKMS

THANKS

Recab:

1. Development workflow
2. VBlank Support
3. CRC API Support

Questions?