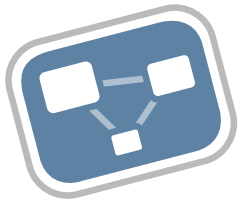
Improving Graphics Interactivity
It's all in the Timing

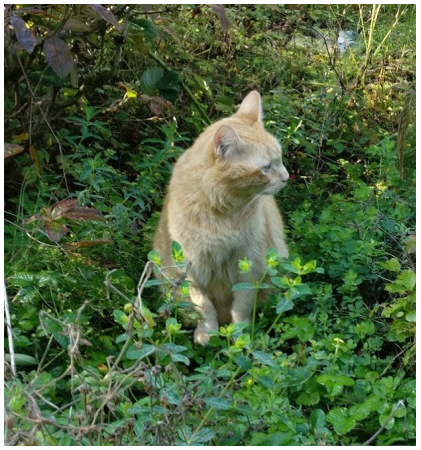
Keith Packard
keithp.com
Valve

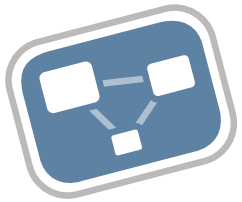


VALVE

Introduction

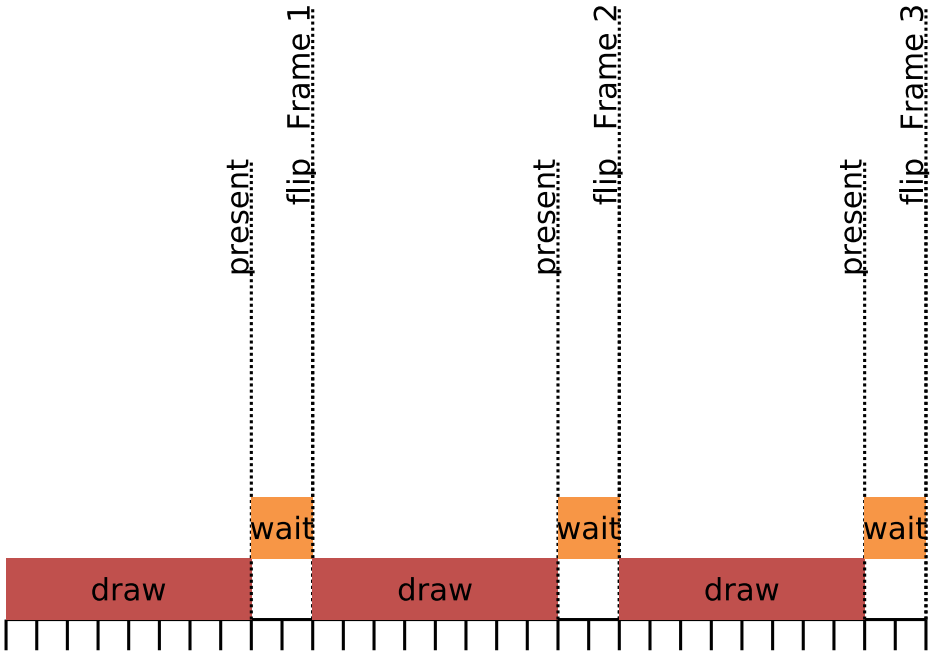
- What do we want?
  - Every frame displayed precisely when the application wants it.
  - Constant frame rate.
- Why is this hard?
  - Lots of moving parts:
    - application scene changes
    - compositing environment changes
    - power/thermal management
  - Asynchronous processing
    - Applications queue rendering to GPU
    - Display must wait for GPU completion

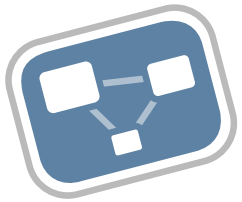




VALVE®

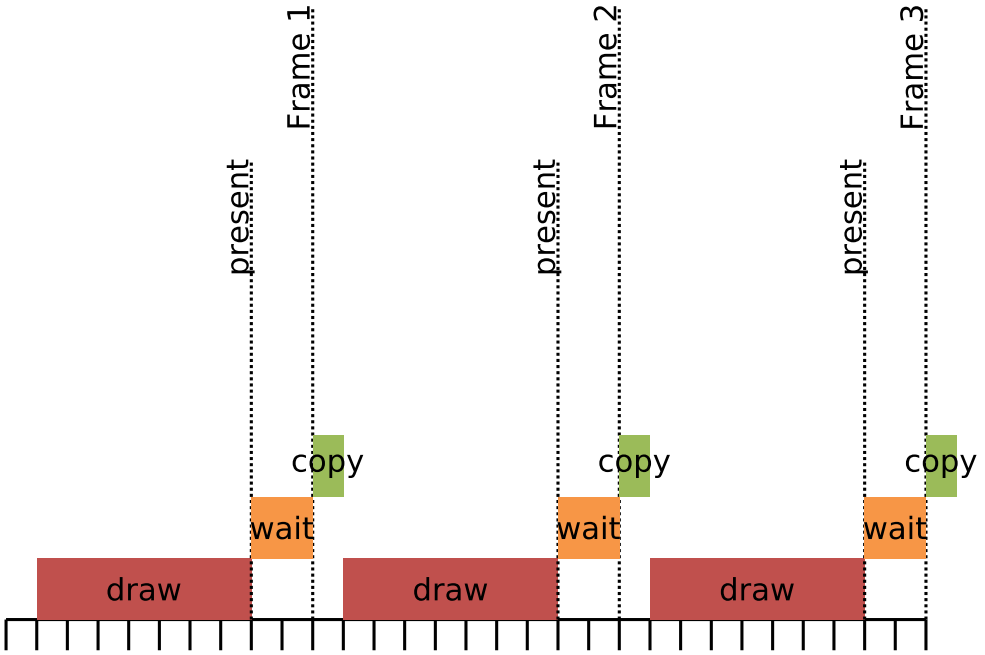
Direct with Flip

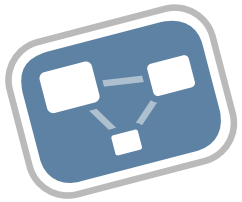




VALVE®

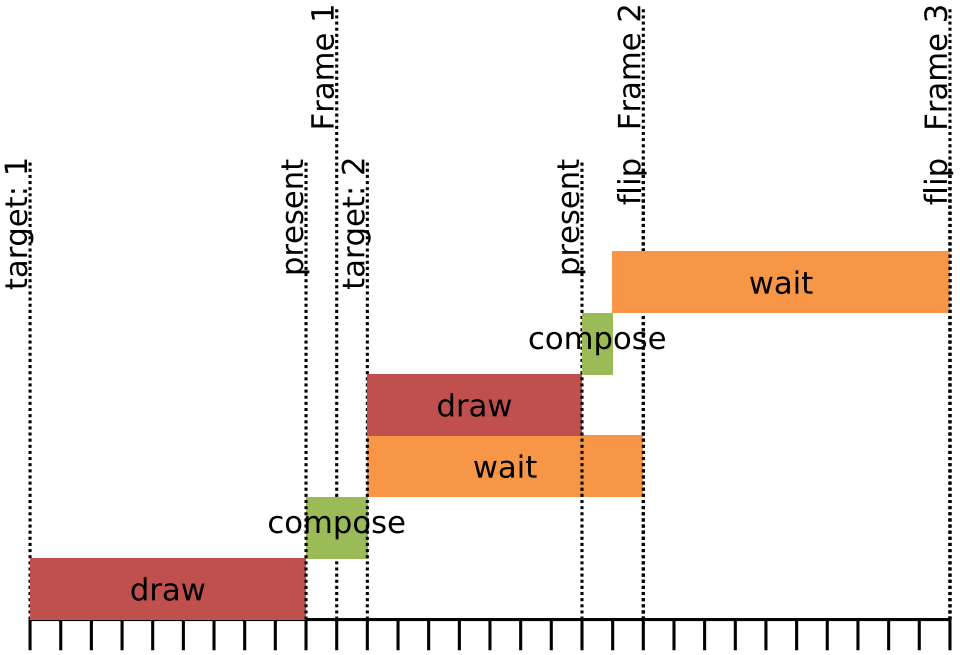
Direct with Copy

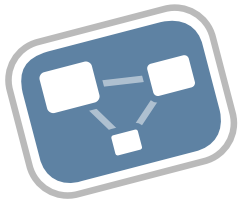




VALVE®

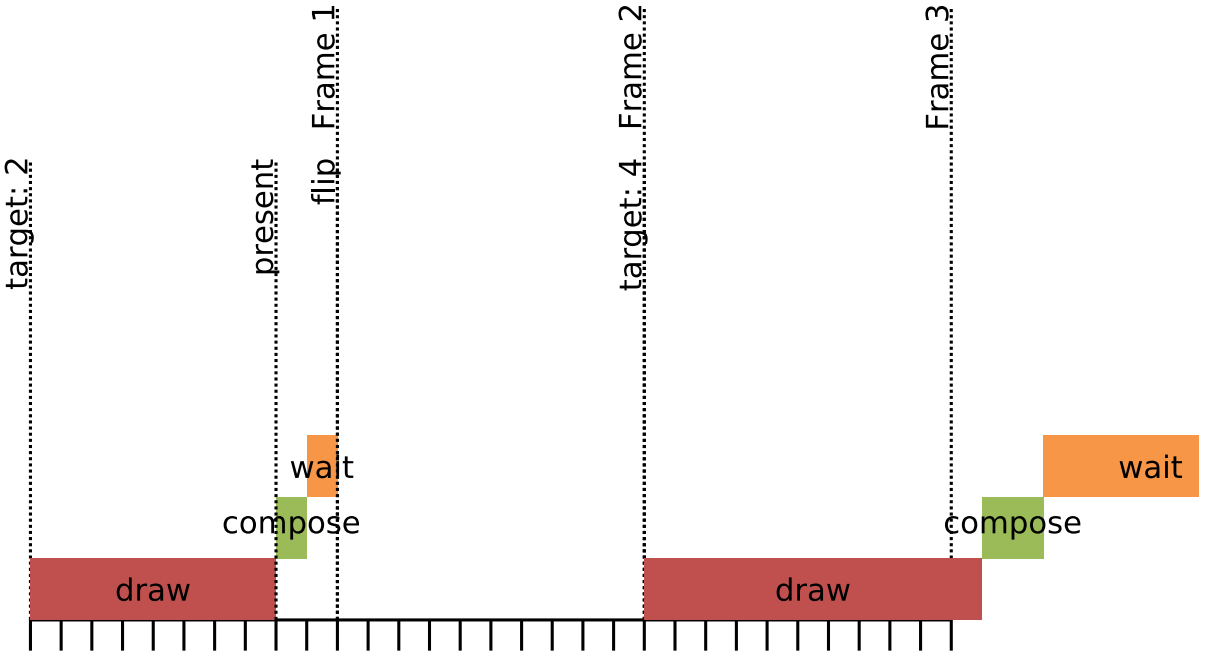
Missing a Frame

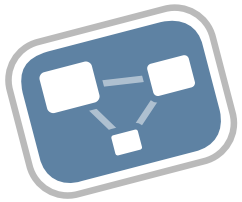




VALVE®

Displaying a Frame Early

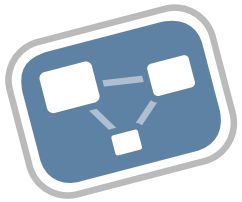




VALVE®

Requirements

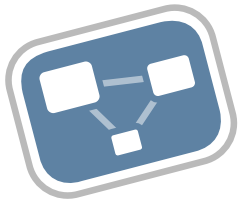
- Tell apps when vblank will be
- Allow apps to specify when frames should be displayed
- Get frames displayed on time
- Tell apps when frames were displayed
  - And when rendering was complete, in the same time domain



VALVE®

OpenGL

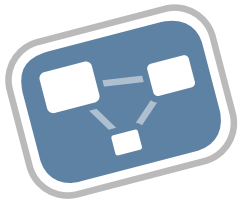
- GLX_OML_sync_control
  - Specify target present frame count
  - Avoids early frame presentation
- But, no feedback about when frames were actually presented
  - Many kludges required to guess
- GLX_EXT_swap_control
  - Sets (min) number of frames per presentation
  - No feedback on actual presentation time.



VALVE®

Vulkan

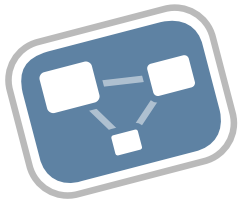
- GOOGLE_display_timing
  - Specify absolute (CLOCK_MONOTONIC) time for frame
  - Feedback about when frames were presented
    - May be delayed by a long time (but not with Mesa).
- Clocking application rendering
  - Best practice today uses vblank fences
    - Using EXT_display_control
    - Which only works for direct display
    - And doesn't say whether a present happened
  - Want something triggered by present
    - That turns out to be hard to specify
- EXT_calibrated_timestamps
  - Get GPU/OS clocks values for the “same time”
  - Allows conversion between GPU and OS time domains



VALVE®

Old Vulkan Loop

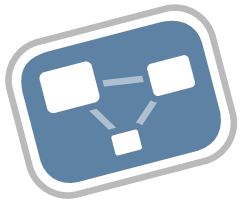
```
frame_step = 16.67 ms
current_time = 0
while(running) {
    RenderFrame(frame_step);
    current_time += frame_step;
    PresentFrame();
    frame_step = LengthOfThisFrame();
}
```



VALVE

New Vulkan Loop

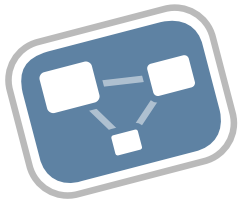
```
frame_step = 16.67 ms
current_time = 0
while(running) {
    RenderFrame(current_time);
    current_frame_id = PresentFrame(current_time);
    history = QueryFrameInfos();
    frame_step = FrameTimingHeuristics(history);
    current_time += frame_step;
}
```



VALVE

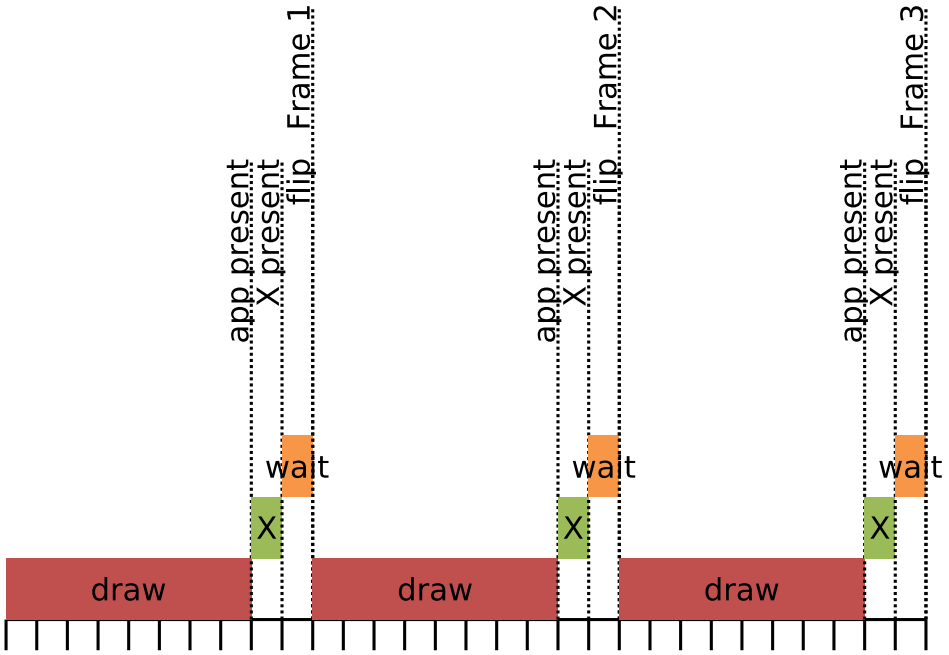
X

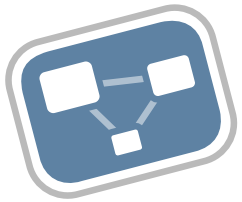
- Present extension spec is ready
  - Specify target frame for PresentPixmap
  - Provides feedback on when PresentPixmap was processed
- But the implementation lags
  - When the desktop is composited



VALVE

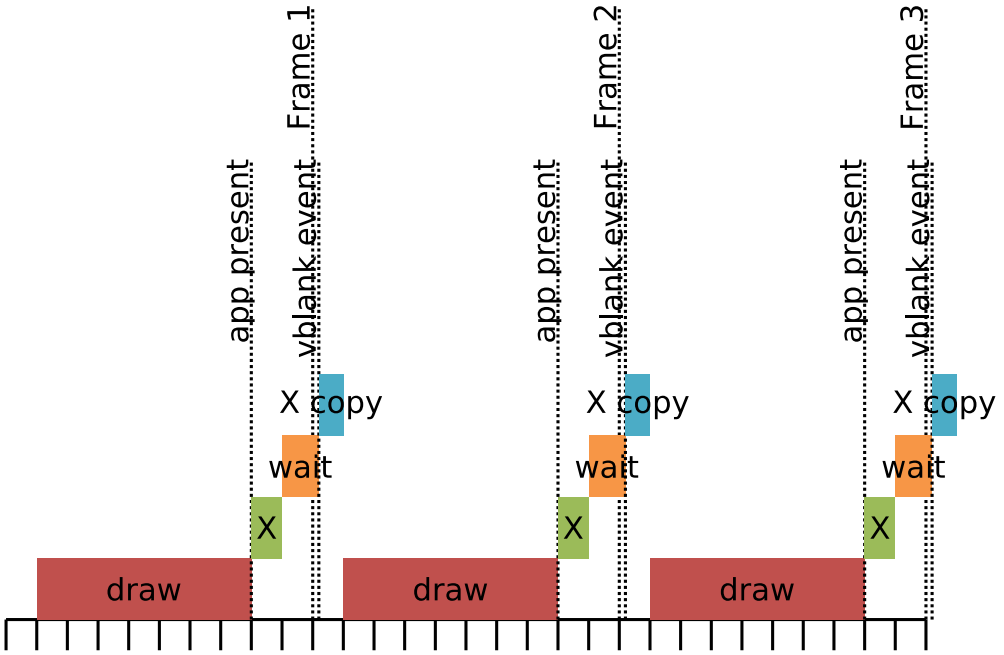
X with Flip

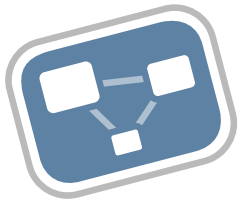




VALVE®

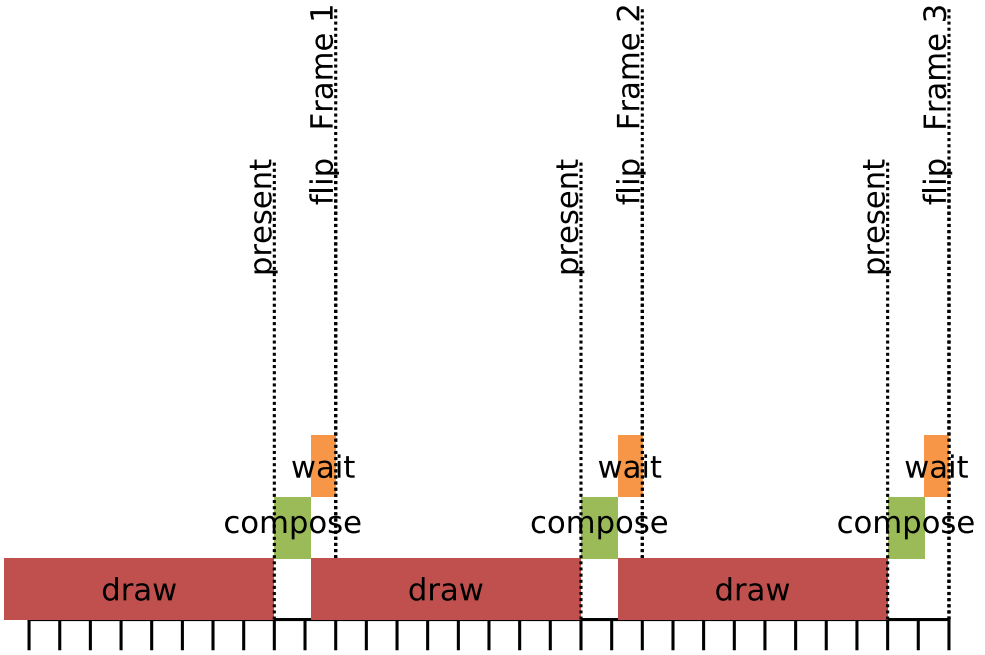
X with Copy

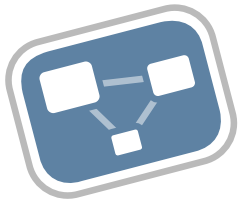




VALVE®

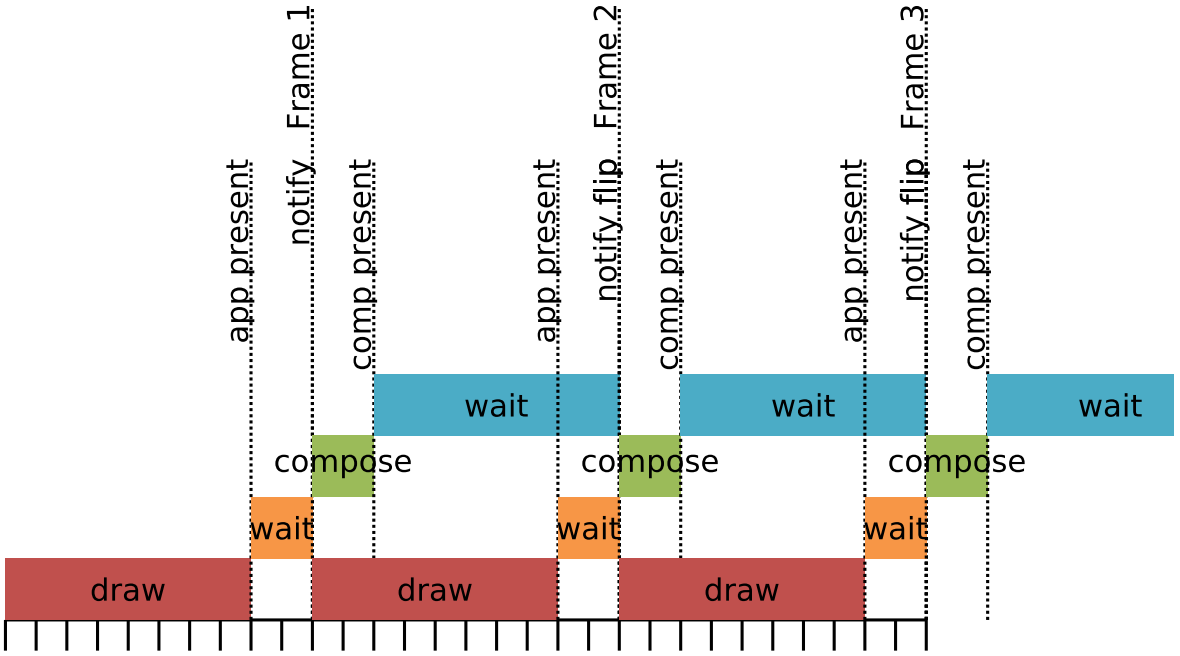
Ideal Composited

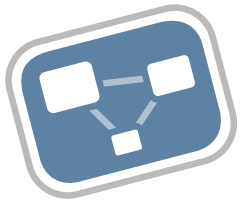




VALVE®

Current X Composited

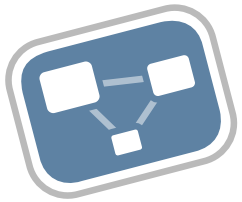




VALVE

Current X Compositing Process

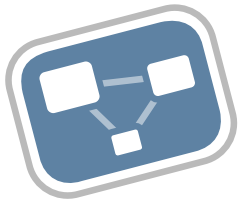
- Each app rendering request generates damage events to compositor
- Compositor collects damage
- At 'suitable time', compositor draws and calls PresentPixmap



VALVE®

Simple X Kludge

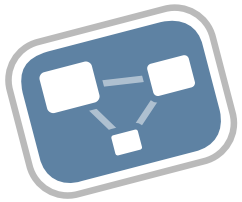
- Send damage immediately at PresentPixmap time
  - Compositor can start building the next frame immediately
- Send Present event at next vblank
  - Assumes that compositor succeeded
- This fixes the frame delay
  - Most of the time
  - When the system isn't busy
  - When the app doesn't ask for a delay



VALVE®

Slightly better X Kludge

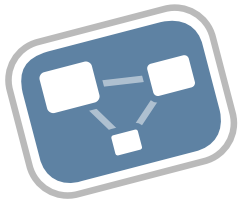
- Pend Damage in X server until 'the right time'
- Deliver damage to compositor. Remember which damage was sent.
- Send Present events to apps at the same time we send Present event to compositor
- No changes in compositor required



VALVE®

Principled X Fix

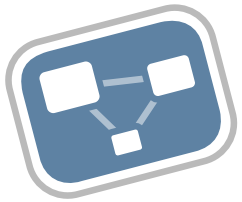
- Mark damage events with sequence numbers
- Change compositor to notify X server which damage sequence is processed by PresentPixmap
- X server can associate compositor PresentPixmap with app PresentPixmaps and deliver correct events.



VALVE®

When is The “Right Time”?

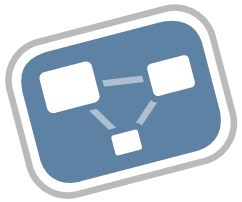
- At some fixed point in the frame?
  - But compositor operations may vary
- At the latest possible point in the frame?
  - Estimate compositor time based on previous frames and amount of change?
- When InputFocus app calls PresentPixmap?
  - Likely to be where the user is working
  - Have the app inform on plans?
  - Estimate based on previous app actions?
  - Fallback to other method, or drop frame?



VALVE®

Linux Flip API

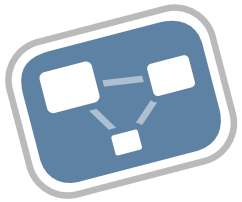
- Current API is awkward
  - Finite event limit in kernel mixes flips and vblank notifies
  - Applications must work-around in user space
    - Test for failure, attempt to empty pending events, retry
  - Times in µS instead of nS
    - Doesn't match Vulkan time precision
- Single queue spot
  - Queue other buffers in user space
- No 'unqueue'
  - Commit to planned frame up front
- Blocks waiting for rendering(?)
  - The non-atomic path does
  - And I think the atomic does as well.
- Cannot actually support “Mailbox” mode.



VALVE®

Queue without blocking

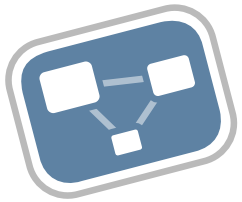
- Kernel can move to HW when rendering completes.
- Allow user space to continue.
- Alternative is to have user space take an event and delay queuing until then.



VALVE

Multiple flips queued

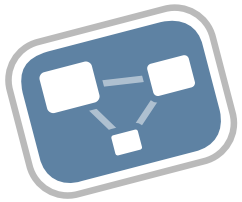
- For same frame
  - Kernel picks last one ready at vblank
  - Idles (and notifies) when possible
- For future frames
  - Allow user space to go idle for longer



VALVE

Cancel queued entries

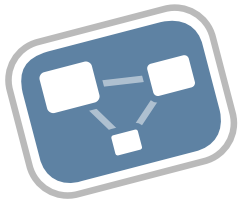
- Useful when queued for many future frames
  - avoid displaying from terminated apps
- Necessary if we don't get multi-queue
  - Handle all of that from user space



VALVE®

Summary

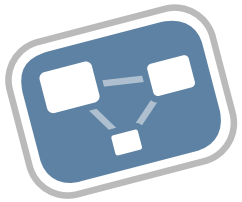
- Extend Vulkan to expose existing X capabilities
- Fix timing under composited X
- Enhance Linux flip API
  - Make flips more reliable
  - Support Mailbox mode
  - Provide ns resolution



VALVE®

Thanks!

Keith Packard
keithp@keithp.com
Valve



VALVE