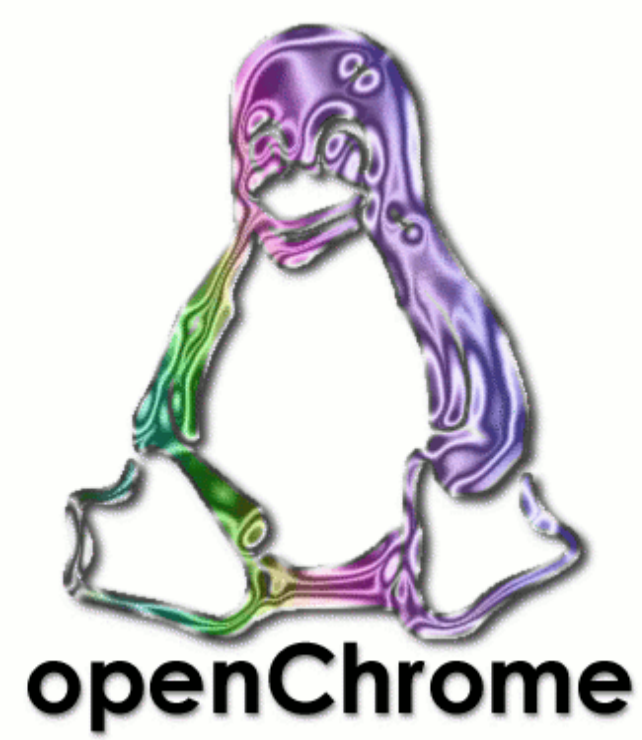
Update on OpenChrome Project



Kevin Brace
Brace Computer Laboratory
https://bracecomputerlab.com
XDC 2022

Development and Validation Environment Changes

- Fully migrated the project development environment from Xubuntu / Lubuntu to Gentoo Linux
  - Ubuntu based OSes completely dropped x86 32-bit OS support several years ago
  - The long held desire to increase code testing on CLE266 chipset
    - CLE266 chipset was almost always mated with VIA Technologies C3 processor, but early C3 processor models do not support 686 class processor features like PAE (Physical Address Extension) and CMOV (conditional move) instructions, so one needs to generate 586 class processor instructions to realistically test the code
    - Installed a 586 class processor compile Gentoo Linux image to a high endurance USB flash memory storage (i.e., SanDisk Extreme) recently
- Validation is mainly done on Gentoo Linux (32-bit / 64-bit) and Knoppix 8 and 9 (mainly 32-bit)

OpenChrome DDX Progress Since XDC2019

- Added the proper code support to handle an external DVI transmitter on CLE266 chipset's DIP0 (Digital Interface Port 0) and DIP1 (Digital Interface Port 1)
  - Validated on NeoWare CA10 thin client
- Various refactoring and bug fixes
- Updated via_drm.h to match changes made to OpenChrome DRM uAPI
- It appears that X.Org X Server Version 21 broke EXA support for OpenChrome (X Server boot time crash)
  - Turning off acceleration seems to workaround the issue (huge performance penalty)
  - It is recommended that users stay with X Server Version 1.20 for now

OpenChrome DRM Progress Since XDC2019

- Finally converted to atomic mode setting (November 2020)
  - Most of the code conversion was done a month after XDC2019 (November 2019)
  - Could not figure out how to handle adjusted mode for a while (March 2020)
  - One small code conversion error appears to be the reason why the code was not working (October 2020)
  - Still missing gamma correction / palette initialization code (quite different from “legacy” KMS)
- Backported CLE266 chipset external DVI transmitter support code proven on NeoWare CA10 (July 2022; still untested)

The Difficulties OpenChrome DRM Development Faced Since XDC2019

- Had 2 major upstream DRM subsystem code integration crises in 2021 that significantly impeded the code development throughout the year
  - TTM move callback related TTM code change took 3 months to resolve (January to March 2021)
  - The upstream DRM modules switching to using an embedded GEM object located inside ttm_buffer_object struct was missed (commit ce77038 and 8eb8833), and about 10 kernel development cycles later, when OpenChrome had to start using drm_gem_ttm_mmap() (based on commit 265ec0d), the code “snapped”
    - This one took 6 months to resolve (July 2021 to December 2021) because the 10 kernel development cycle depth made it hard to track down the issue
- Upstream DRM subsystem has not done major design changes for the past 7 to 8 Linux kernel development cycles, so I have been fairly successful in keeping up with the upstream changes lately

Getting OpenChrome DRM Pulled into the DRM Subsystem Tree - Preparation

- Started to think about getting the code pulled into the DRM subsystem tree (January 2022)
- Obviously, I was thinking of timing it against XDC2022
- Various code cleanups to prepare for Linux kernel DRM subsystem integration (May to June 2022)
- Changed module name from /drm/openchrome back to /drm/via (June 2022)
- Submitted the code for code review (mid to late June 2022)

Getting OpenChrome DRM Pulled into the DRM Subsystem Tree – Submitted Patches

- First code patches (v1) did not work with the upstream DRM repository code (worked only with the OpenChrome DRM repository code)
- Submitted second code patches (v2) to resolve the build issue with the upstream DRM repository code based on mailing list feedback
- Third code patches (v3) added the proper external DVI transmitter support for CLE266 chipset and minor code cleanups

Getting OpenChrome DRM Pulled into the DRM Subsystem Tree – What Happened?

- Originally expected DRM subsystem maintainers to give me feed back on the code, but got near zero attention
  - Is June a bad timing to get code review done?
- A developer who manages drm-misc repository was effectively the main code reviewer
  - Appeared to be receptive to pulling in the code, initially
  - This developer did not like the fact that OpenChrome DRM contained 3 uAPIs even though the DRM is not capable of any acceleration at this point
  - This developer wanted unaccelerated OpenChrome DRM to use DRM core's dumb buffer API
  - The point is that uAPI added becomes virtually permanent part of that DRM module (i.e., individual DRM device driver), so this developer does not want to see it in the code unless it is absolutely necessary

Getting OpenChrome DRM Pulled into the DRM Subsystem Tree – Outcome

- Since OpenChrome DDX is currently designed around using OpenChrome DRM uAPI, I did not wish to get rid of OpenChrome DRM uAPI since I plan to start implementing acceleration related features soon
- It appears that getting unaccelerated mode setting only DRM module pulled in is not worth the effort if the ultimate goal is to add uAPI for acceleration support later
- It was not a total waste of time since various people suggested small improvements here and there
- Legacy DRI1 era VIA DRM got compacted into one .c file (Linux 6.1), so it will be a little easier to insert OpenChrome DRM into the DRM subsystem tree eventually
- Will try again to get OpenChrome DRM pulled into the DRM subsystem tree after acceleration is working and uAPI is stable

Development Roadmap - OpenChrome DRM

- Implement CS (Command Submission) uAPI
  - Try to first get EXA working since unaccelerated mode is very painful to use
- Leverage legacy DRI1 era VIA DRM code written for UniChrome
  - Was able to compile drm/via/via_verifier.c and drm/via/via_dma.c with OpenChrome DRM recently (September 2022)
- Leverage Chrome9 DRM open source code provided by VIA Technologies
  - First one is DRI1 based code mainly for H5 graphics engine (i.e., K8M890 and P4M900 chipsets)
  - Second one is DRI2 based code for H6 graphics engine (i.e., VX800, VX855, and VX900 chipsets)
  - Copy command verifier code and integrate it into OpenChrome DRM
- No plans to maintain uAPI compatibility with VIA Technologies uAPI implementations

Development Roadmap - OpenChrome DDX

- Split the current OpenChorme DDX code, which has support for both KMS and UMS
  - It is a code maintenance nightmare
  - Difficult to support libdrm and maintain compilation backward compatibility since libdrm recently dropped Autotools support
- OpenChrome DDX active development branch (Version 2.x or later)
  - Try to make the code legacy free
  - OpenChrome DRM will now handle mode setting (KMS only)
  - Use libdrm (major reason for the code “split”)
  - EXA support
  - Probably no Xv or XvMC support

Development Roadmap - OpenChrome DDX

- OpenChrome DDX legacy branch (Version 1.x)
  - Keep DRI1 and no DRM operation mode
  - Keep Xv and XvMC code
  - Keep EXA code
  - Remove KMS support
- Committed to maintaining the code (i.e., refactoring, bug fix, stability fix, and mode setting code improvement) for the long haul, however, no new major feature will be added

Development Roadmap - OpenChrome Mesa

- UniChrome
  - Since DRI1 era Unichrome code from Mesa 7.11 exists, why reinvent the wheel?
  - Upgrade the code to support the new OpenChrome uAPI
  - Moderately high priority

Development Roadmap - OpenChrome Mesa

- Chrome9
  - No open source code exists
  - VIA Technologies did develop Chrome9 Mesa code in the past, but it was binary only distribution (IP related NDA issue?)
  - Probably designed around Chrome9 DRM uAPI
  - VIA Technologies tried to get the code into the DRM subsystem tree, but was rejected by then DRM maintainer
  - While hardware register programming manual exists for H6 graphics engine, Pixel Shader instruction format document ended up not getting released to the public
    - May be able to figure out the PS instruction format from VIA Technologies open source DDX source code
  - Low priority