GPGPU in Commercial Software: Lessons From Three Cycles of the Adobe Creative Suite

Kevin Goldsmith | Senior Engineering Manager

Why this talk?

Taking advantage of the GPU for things other than rendering to the screen feels like jumping into the future of technology, like you should be wearing a lab coat to work. It totally is, but if you want to do it for real software that people pay for, there are some things to watch out for.

Image from nasaimages.org

Overview

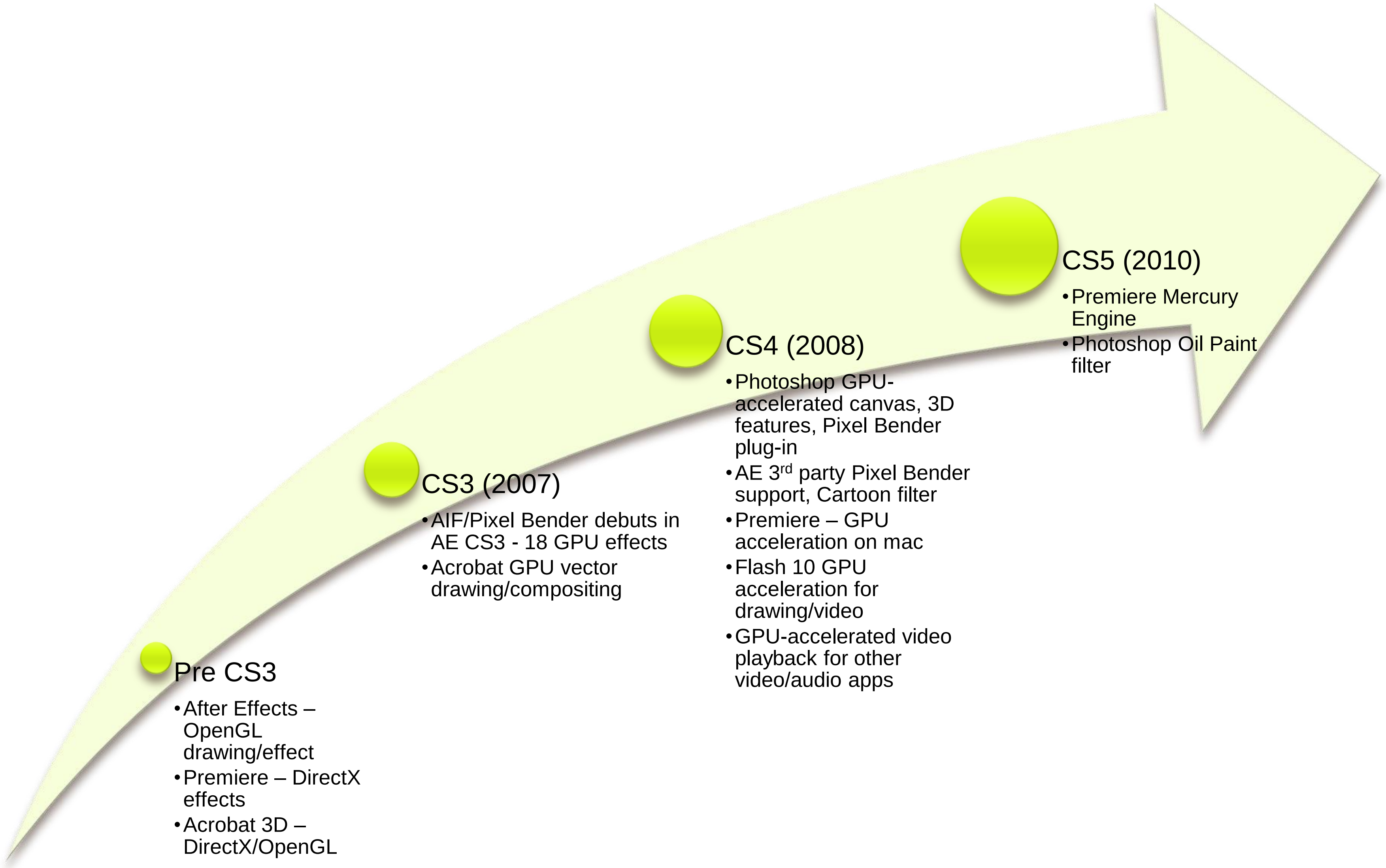
- Introduction
- Using GPUs for Commercial Software
- GPU Development Gotchas
- Questions

Introduction

My Team – Adobe Image Foundation

- In the Core Technologies Group at Adobe
- We produce libraries that make it easier for Adobe teams to use OpenGL, OpenCL, and CUDA
- We created the Pixel Bender language
- Framework for building accelerated GPU and multi-core image processing graphs
- Some of the products which use our technologies: Photoshop, After Effects, Premiere, Flash Player, Flash Pro, Flash Builder, Encore, Audition, Soundbooth, Premiere Elements, Photoshop Elements
- First release of our technology was in Adobe Creative Suite CS3

GPU features in Adobe Products



Pre CS3

- After Effects – OpenGL drawing/effect
- Premiere – DirectX effects
- Acrobat 3D – DirectX/OpenGL

CS3 (2007)

- AIF/Pixel Bender debuts in AE CS3 - 18 GPU effects
- Acrobat GPU vector drawing/compositing

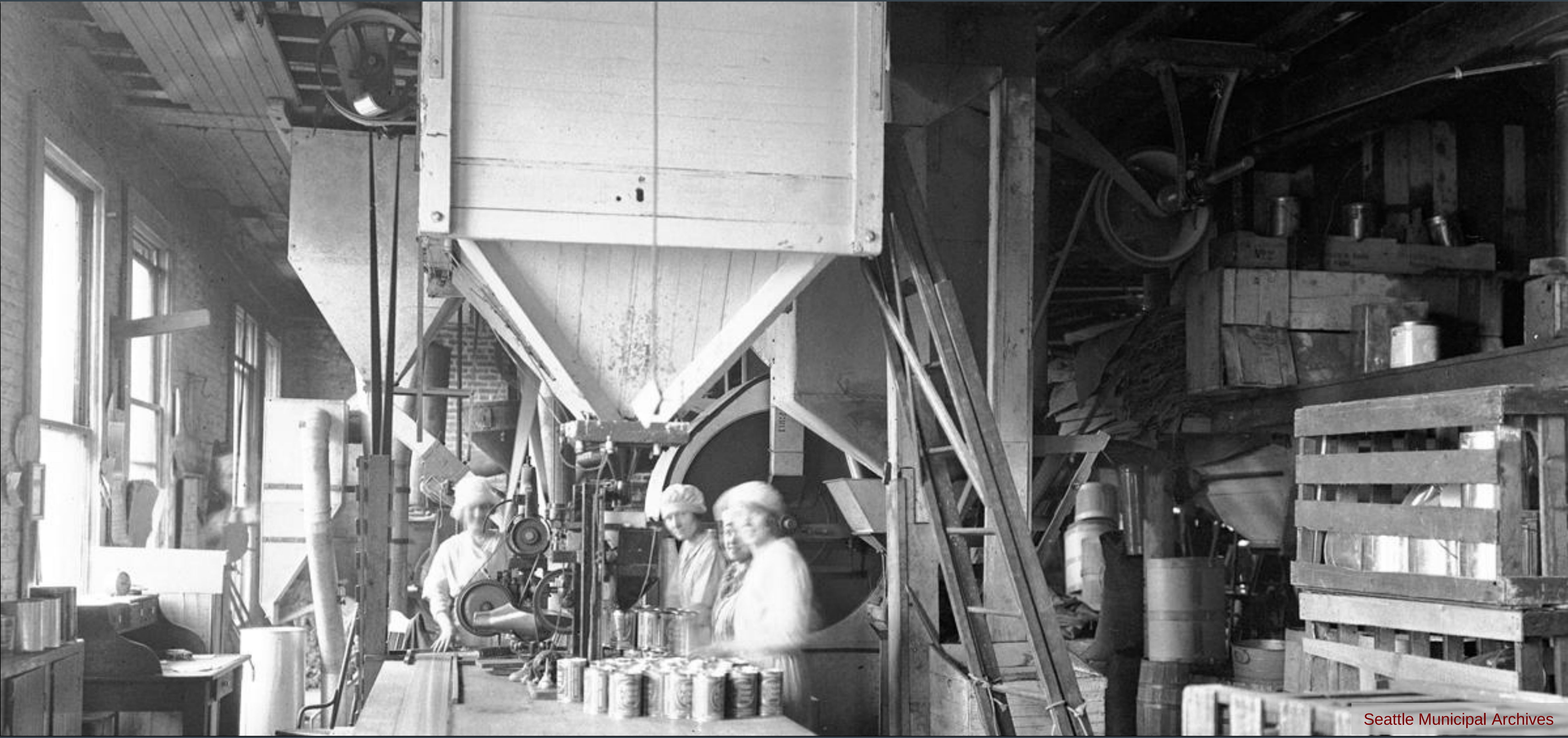
CS4 (2008)

- Photoshop GPU-accelerated canvas, 3D features, Pixel Bender plug-in
- AE 3rd party Pixel Bender support, Cartoon filter
- Premiere – GPU acceleration on mac
- Flash 10 GPU acceleration for drawing/video
- GPU-accelerated video playback for other video/audio apps

CS5 (2010)

- Premiere Mercury Engine
- Photoshop Oil Paint filter

Using GPUs in Commercial Software



Seattle Municipal Archives

Getting buy-in: Educating Management

- Performance
- Competition
- Demos

Getting going: Educating Engineering and QE

- Training
- Constraining supported cards
- Selecting Appropriate APIs
- Tooling
- Performance
- Demos

(most of what the rest of the talk is about)

Keeping it going: delivering and iterating

- Automation
- Fall-back plan
- Hot-patching

GPU development gotchas

Photo by Kevin Walsh

GPUs in the “real world”

- Intel has more than 50% of the GPU market (July 2010, Information Week)
- Real people don’t upgrade their hardware that often
  - Windows XP still has more than 60% of the OS market share (August 2010, Netmarketshare numbers)
  - Only 61% of Steam users have DX10-level hardware (August 2010, Steam Hardware & Software survey)

GPUs in the "real world"

- Intel has more than 50% of the GPU market (July 2010, Information Week)
- Real people don't upgrade their hardware that often
  - Windows XP still has more than 60% of the OS market share (August 2010, Netmarketshare numbers)
  - Only 61% of Steam users have DX10-level hardware (August 2010, Steam Hardware & Software survey)

What to do?

Strategies: Know your market!

- Professional users
  - *If I make my living using your software, I'll upgrade my equipment if it will make my life easier (and I can afford it)*
  - Premiere CS5: 64bit only, CUDA acceleration, 3 (high-end) cards supported
- Pro-sumers
  - Maybe more willing (and more able) to switch to new hardware or upgrade because their machines aren't their livelihood. Like the "cool" factor. Closer to gamers.
- Hobbyist/Occasional users
  - Won't upgrade for your software
  - Angry if it doesn't work with their machines
  - Premiere Elements: If it doesn't run near real-time on a Core 2 Duo with IIG the users will revolt

Strategies: Scale up

- Work on a wide range of hardware, but work better on better hardware
- This is what my team advocates within Adobe

- Pros
  - Users with high-end machines are happy
  - Users with low-end machines are (mostly) happy
  - You can sell to more users
- Cons
  - Need to implement the same feature in multiple ways
  - Much larger test matrix
  - May need to support multiple APIs
  - Significantly more development effort

It is a classic ROI decision.

Strategies: Be clear what hardware you work on

- Set clear minimum system configurations
- List specific cards or card families if possible
- Users want to know if their hardware will work with your software or which cards to buy
- Test for capabilities, not chipsets or drivers
- Fail gracefully
- Blacklist bad cards/drivers, don’t whitelist good ones.
- Consider allowing users to run a test program on their computer from your site to guarantee compliance

The system is only as fast as it’s slowest component

- Make sure that you are using the GPU for the right things
- Rather than try to integrate GPU processing into a hostile architecture, look for new features than can be developed using the GPU
- Measure performance constantly
- Beware “premature optimizations”
- Every trip over the bus is a bottleneck
- Can you replace CPU code with CUDA/OpenCL code to keep data on the card?
- Is slow GPU code in the middle of fast GPU code faster than moving data back to the CPU and then again to the card?

Strategies: Feel your user’s pain everyday

- Make sure that someone is someone is using the low-end of your supported hardware as their primary development / QE machine
  - If someone on the team isn’t having to live with that daily, how do you expect your users?
- Try to cycle cards through the development / QE team
  - So different members of the team notice the performance characteristics of each supported card

Strategies: Life with drivers

- Drivers change often
- Your code that worked perfectly can be busted by the time the user installs it
- You can’t expect a user will have the latest drivers installed

- Be prepared to work around driver issues
  - When you do, document them really well in the code
  - Be able to hot patch existing installs if necessary
- Work with the IHVs/OS Vendors to report bugs
  - Make sure you can send them code that reproduces the problem
- Someone on your team should always running the latest driver
- Blacklist known bad driver versions
- Test new features against old drivers periodically to make sure you aren’t turning up new driver bugs

Strategies: Choosing Your APIs

- Does your chosen API limit your supported hardware?
  - Not necessarily a bad thing
- Know when to use Proprietary vs Open APIs
  - Cross-platform development
  - Development/Support costs
  - Can always use both…
- Does an API limit mathematical precision for critical calculations?

Strategies: Automated testing

- Automation is key
- Test matrix is 3 dimensional
  - Operating System/Version
  - Graphics Card
  - Driver version
- Performance Tests need to be part of test runs
  - Getting tossed to the “slow path”
  - Need to know about it immediately
- Regression tests should include some low level HW verifications (really useful for finding driver issues)
- Good automated tests let you send them to the IHVs to be part of their driver validation suites

Strategies: Precision

- IEEE compliance is nice, but it is expensive
- Know when it is important and when it is ok to approximate
- No matter what level of compliance there is, results will vary from CPU to GPU and from GPU to GPU
- Understand how API choice and how runtime parameters affect precision
- Optionally, allow the user to specify their precision requirements

Questions?

Adobe

Image attribution

All images used in this presentation were used in accordance with the image creators' specified licensing. Their inclusion implies no endorsement of me, Adobe or this presentation. The URLs for the images are listed below.

- http://www.nasaimages.org/
- http://www.flickr.com/photos/seattlemunicipalarchives/3365336567/
- http://www.flickr.com/photos/86624586@N00/10176897/
- http://www.flickr.com/photos/walkn/3526522573/