EE 4702-1, GPU Programming

When / Where

- Here (1218 Patrick F. Taylor Hall), MWF 11:30-12:20 Fall 2017
- http://www.ece.lsu.edu/koppel/gpup/

Offered By David M. Koppelman

- Room 3316R Patrick F. Taylor Hall
- Office Hours: Monday-Friday 15:00-16:00

Prerequisites by topic:

- C++.

GPU Definition

GPU: Graphics Processing Unit

- Runs 3D graphics in place of CPU...

  ... because it's much better at it.
- Also runs scientific-style computation in place of CPU.

GPU is Main Component of Video Cards

Major Companies and Brands

- NVIDIA
- ATI (Compaq/HP)
- Intel

This Course

Focus is on GPU Programming

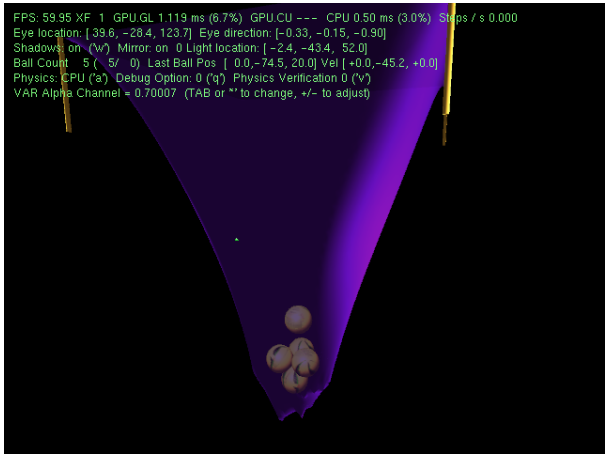
- *Shader* programming with *OpenGL Shader Language* (*OGSL)*.
- *GPGPU* programming with *CUDA*.

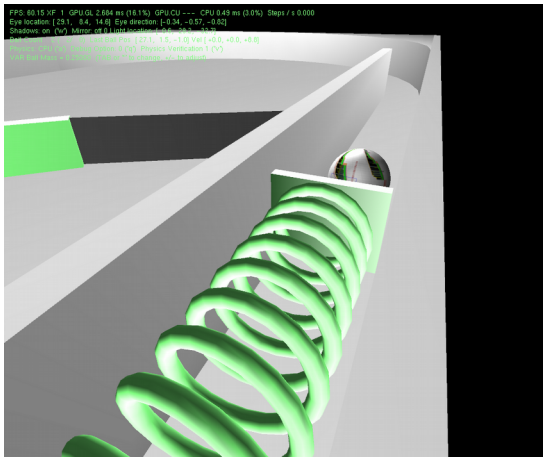
Also Some 3D Graphics, Game Physics

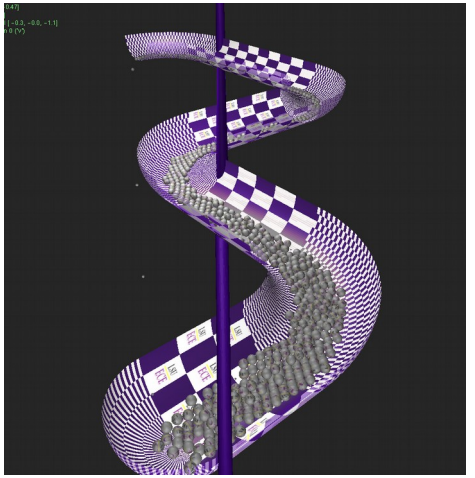
- Will cover enough graphics, OpenGL, and CUDA to do fun stuff.

Game Physics Term Project

Past Student Project Screenshots:





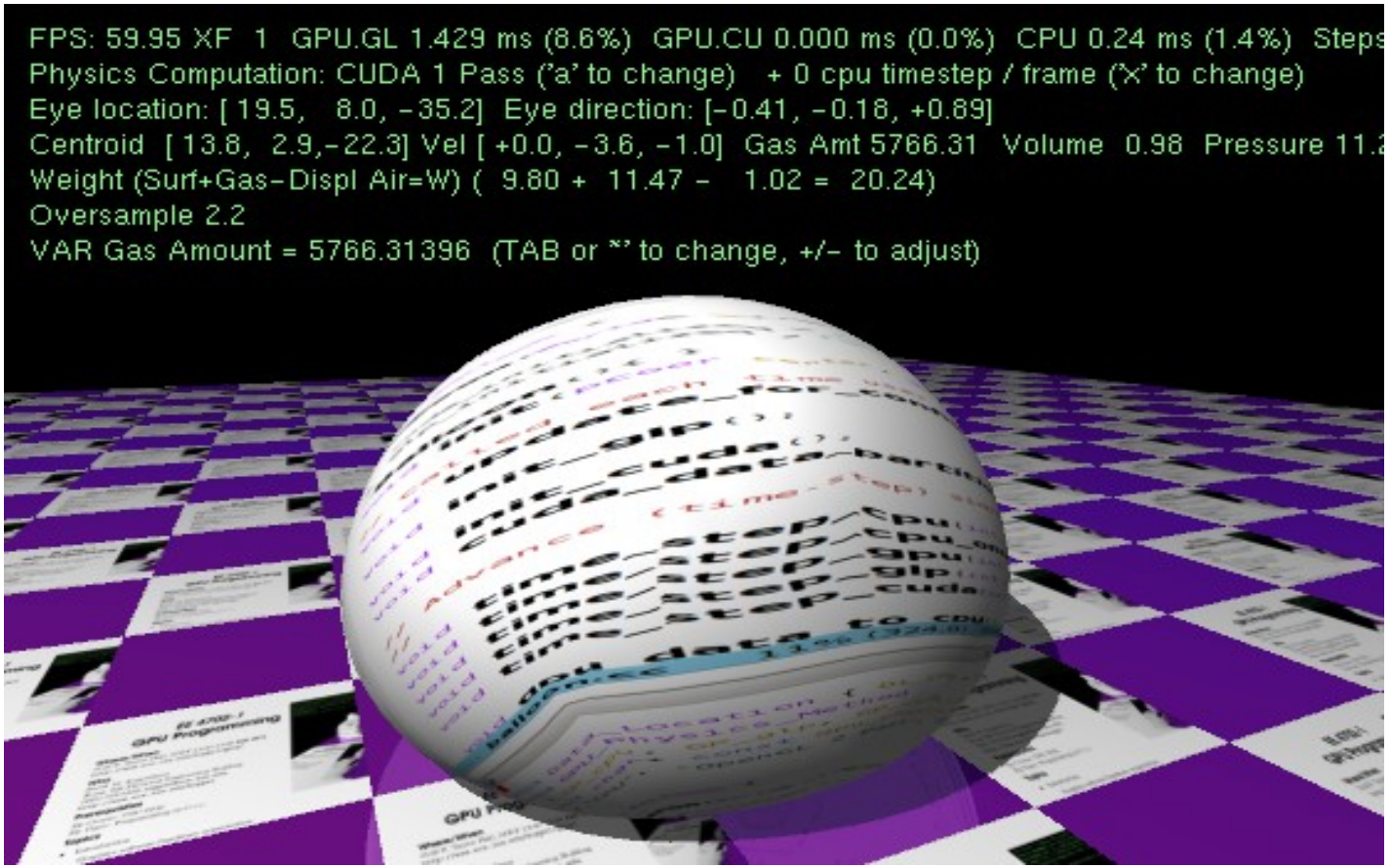


Balloon Demo

Simulation of a balloon.

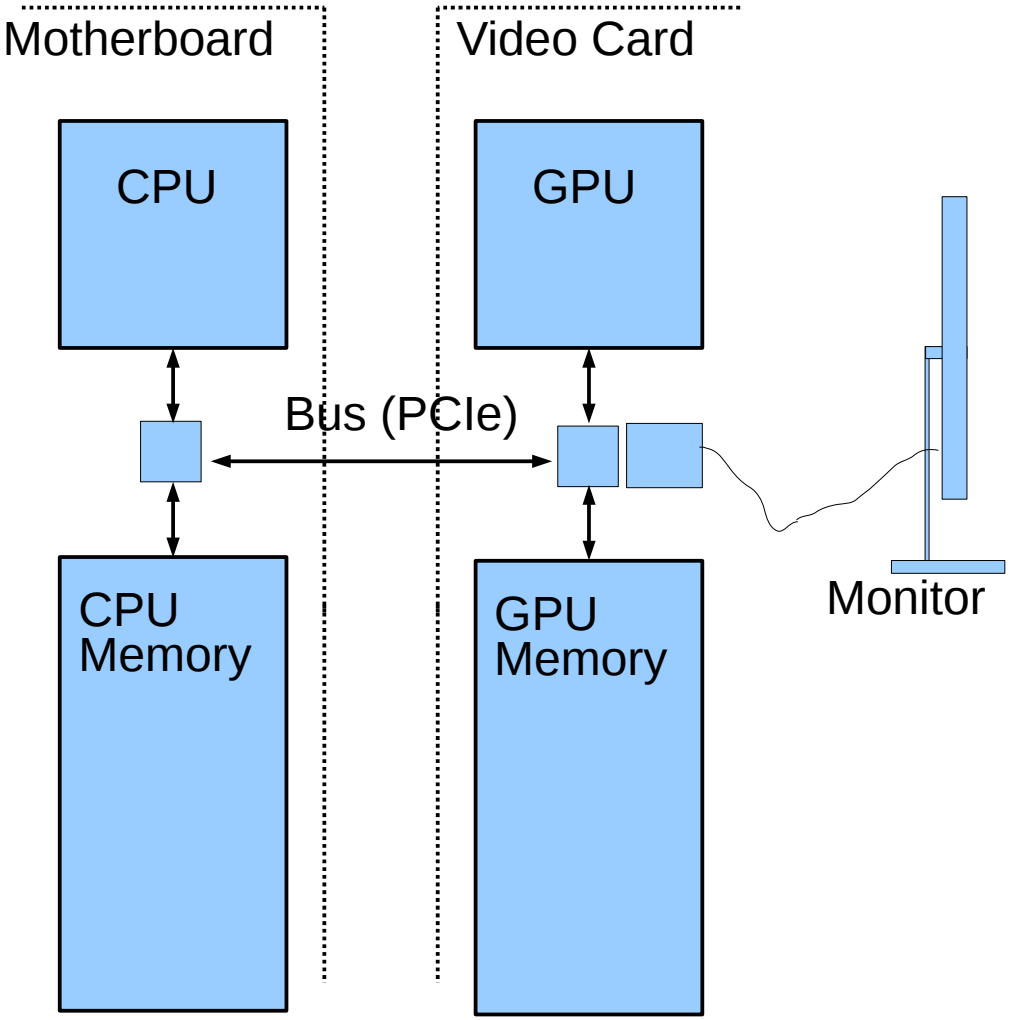
GPU always runs 3D graphics.

Code can switch between CPU-only and CPU/GPU physics.



System Overview

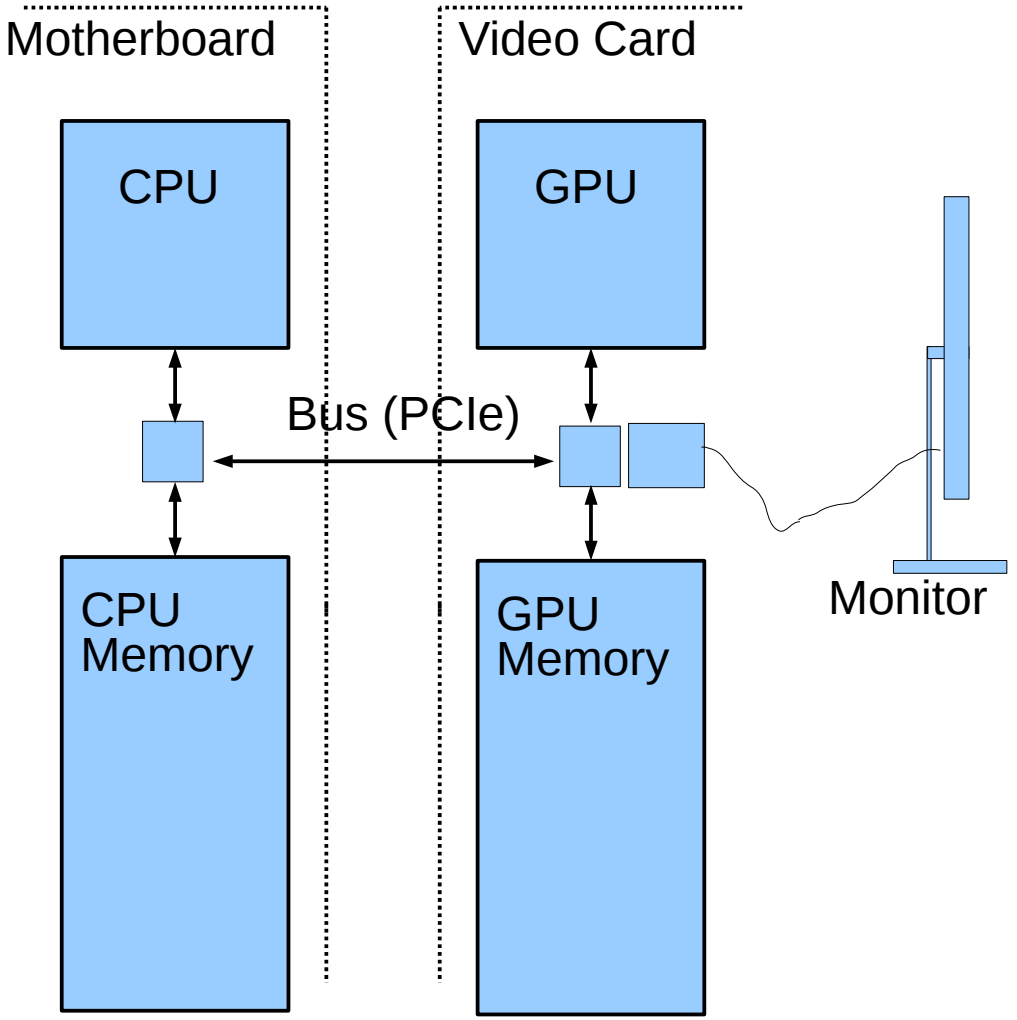
Quick look at how the GPU fits into the larger system...



System Overview: Hardware

Typical Hardware

- On Computer motherboard: CPU, CPU Memory
- On Video Card GPU, GPU Memory
- Connection between CPU/GPU via Bus, e.g., PCI Express (PCIe).
- Connection from video card to monitor.



System Overview: Frame Buffer

Frame Buffer

- Area of memory continuously converted to a video signal.
- Simple mapping from memory address to pixel coordinate.

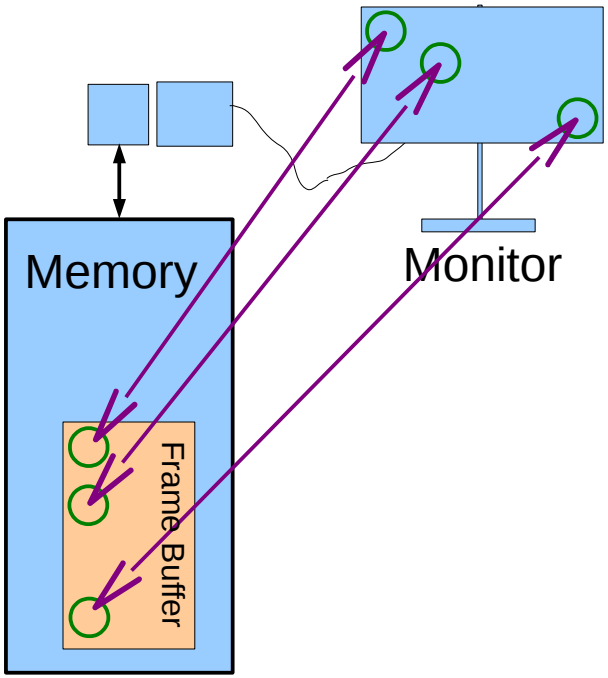
Older Systems

- Frame buffer in CPU memory.
- Application programs wrote frame buffer directly.

Typical Current Systems

- Frame buffer in GPU memory.
- Frame buffer written by GPU hardware (typical)...

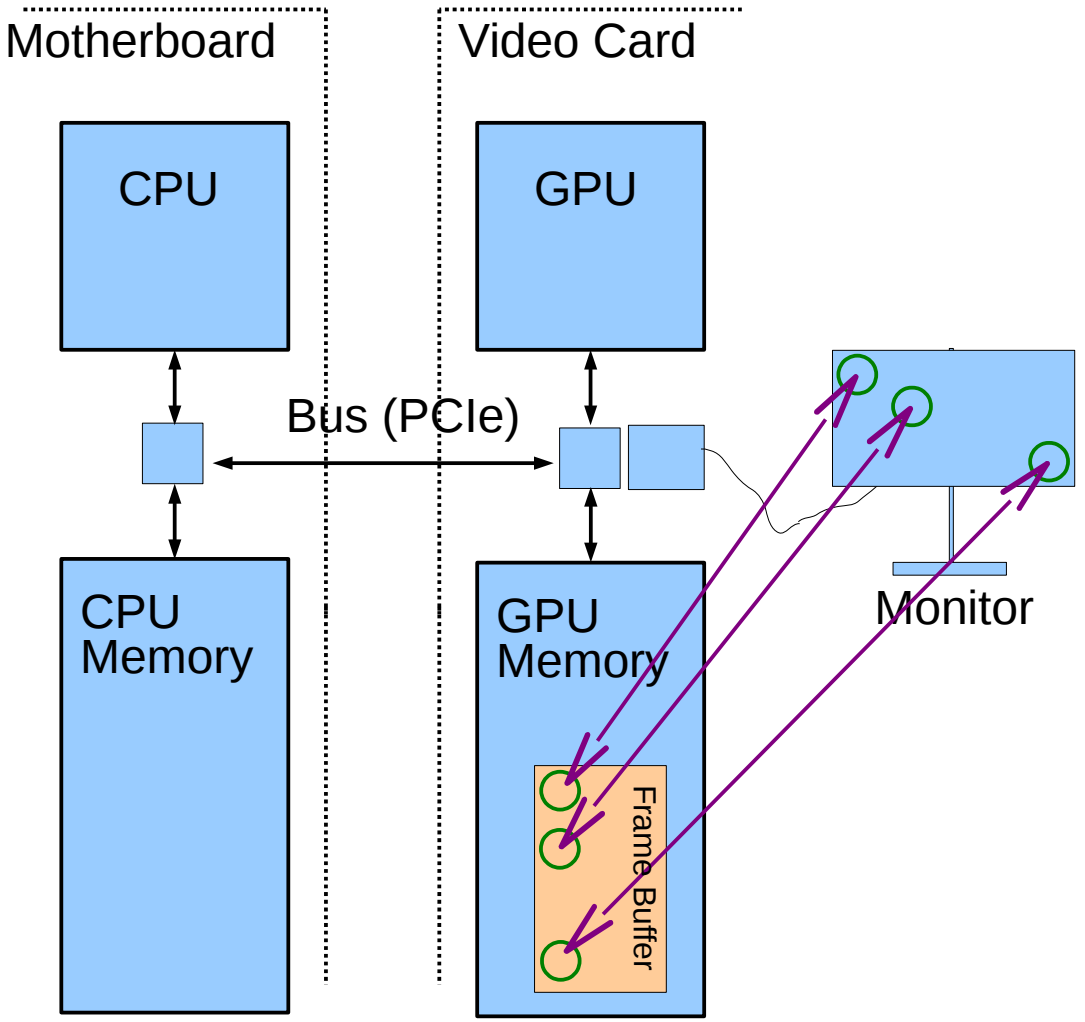
  … in response to code on CPU.



System Overview: Frame Buffer

Frame Buffer Contents

- Position in FB corresponds to particular pixel on display.
- In illustration, first FB element is upper-left pixel.
- A common FB element size is 32 bits.
- Frame buffer format varies with video mode and of course system.



Simple Frame Buffer Code Example

Consider Code

- frame_buffer[10][20] = 0;

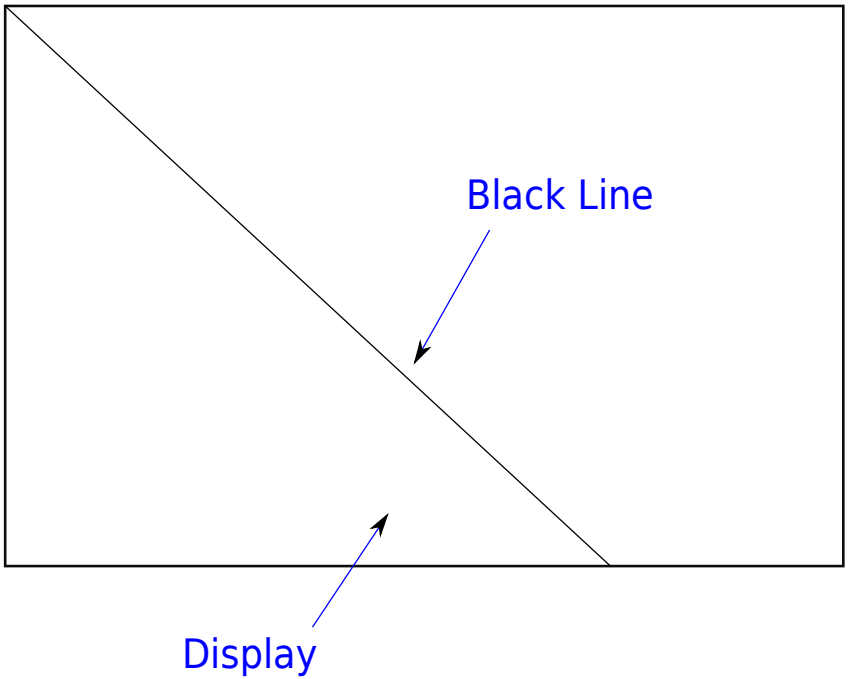
For Simple Code Example Assume

- The frame buffer is in CPU memory.
- Array frame_buffer points to the frame buffer location.
- Writing a “1” to the array makes a pixel white.
- Writing a “0” to the array makes a pixel black.

Simple Frame Buffer Example

```
// Make screen all white. (Assume 1 is white.)
//
for ( int x = 0;  x < width;  x++ )
  for ( int y = 0; y < height; y++ )
    frame_buffer[x][y] = 1;

// Draw a black diagonal line.
//
for ( int x = 0; x < height; x++ )
  frame_buffer[x][x] = 0;
```



GPU Rationale

Code on prior slide looked simple.

How much more complex would balloon code be?

Alot, of course!

Motivation for a GPU:

- Graphics (3D animated, especially) requires alot of computation.
- CPU is less suited for that kind of computation.

System Overview: Code Organization

Simplified View of Code on Typical System

Who Wrote Code?

- User. (Call it the Ap.)
- Vendor of GPU. (Call it the driver.)

Where Does It Run?

- CPU
- GPU

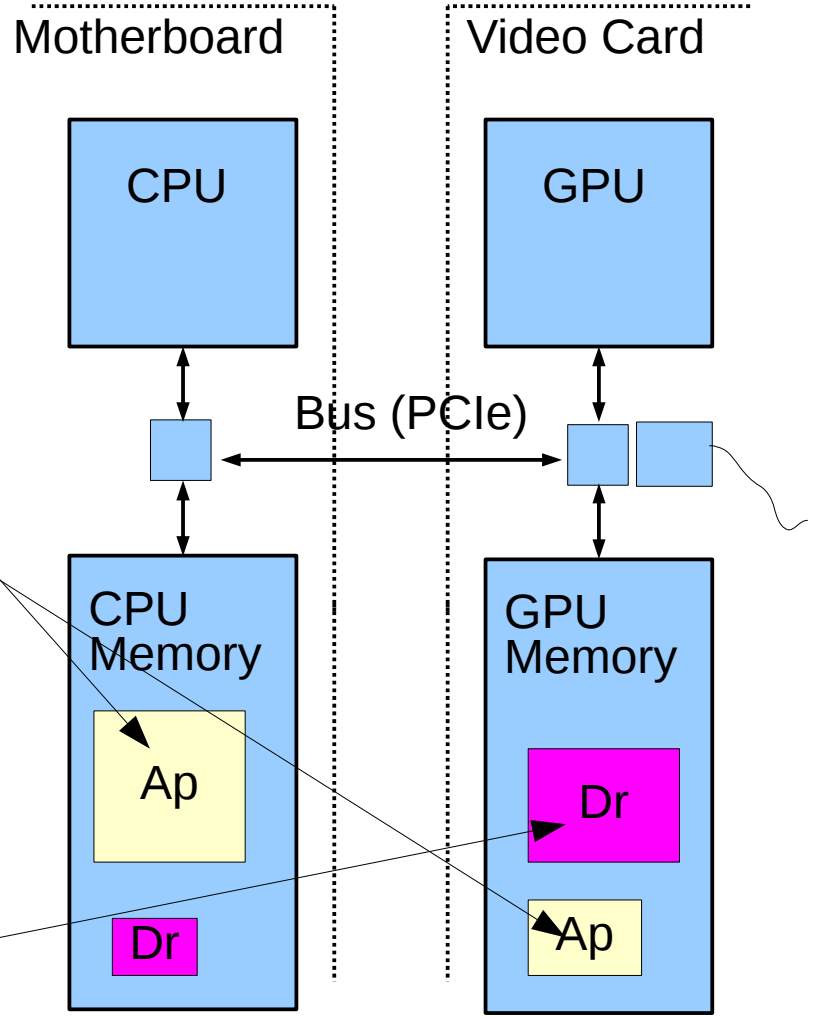
All Four Possible:

- User/CPU, User/GPU, Vendor/CPU, Vendor/GPU

System Overview: Software

Just For Today, Oversimplify to Two Kinds of Software

- *Application Program* (Ap for short*)
  - Written by ap. programmer.
  - E.g., Balloon Demo
  - Most of Ap runs on CPU.
  - Part of Ap may also run on GPU.
- *GPU Driver* (Driver or Dr for short)
  - Written by GPU manufacturer.
  - E.g., NVIDIA 384.59
  - Driver code runs on both CPU and GPU.
  - Most work done by driver code that runs on GPU.



*Don't confuse Ap with App.

System Overview: Running of Application

Typical Execution

- Application, running on CPU, ready to emit next frame.
- App. calls driver on CPU...

  ...driver on CPU starts more driver code on GPU...

  ...application resumes on CPU (while GPU driver code still running).
- Process above repeated many times for a frame.
- Driver code ultimately will write frame buffer.

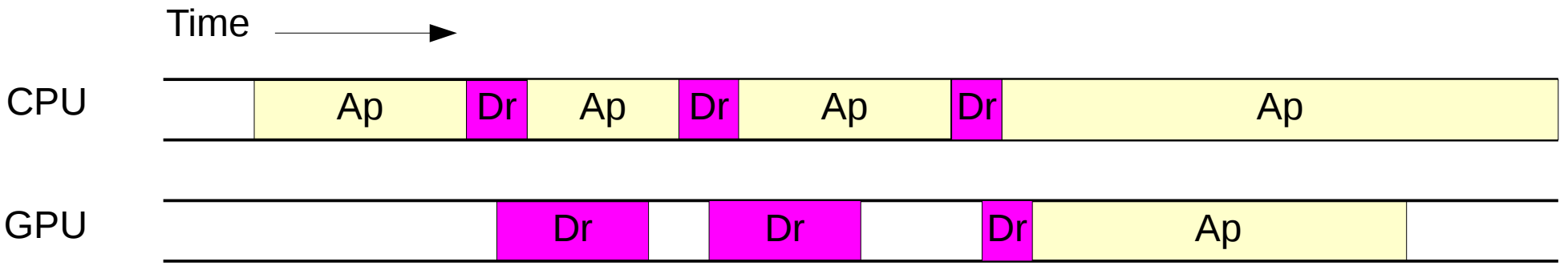
Important Points

- CPU and GPU can run code at same time.

System Overview: Execution Activities

Example Below

- Ap calls driver three times.
- First two times, driver does all work, mostly on GPU.
- Third time that ap calls driver, driver starts some ap code on GPU.



Course Coverage

Emphasis: GPU coding for high performance.

Topics Needed For Term Project (a dynamic simulation)

Topics

- Rudiments of Animation by Dynamic Simulation
- Term Project
- 3D Graphics basics: coordinates, transforms, primitives, colors, textures.
- Data movement and staging, efficiency.
- Coding with GPU *shader* model, CPU/GPU load balancing.
- Coding with CUDA, GPU physics.

APIs, Languages, Standards Used

OpenGL (Version 4.5)

- An API for controlling GPUs from CPU.

OpenGL Shader Language (Version 4.50)

- A language for code (usually graphical) that runs on GPUs.

CUDA (Version 8.0)

- An NVIDIA language for code (usually not graphical) that runs on GPUs.

C++11

- A common programming language.
- Students are assumed to be familiar with C++.

Toolchain

Operating Systems

- Red Hat Enterprise Linux 7 (or Scientific Linux 7)

Compiler

- gcc (GNU C Compiler)

Debugger

- gdb (GNU Debugger)
- Students expected to learn **and use** gdb.

Text Editor

- EMACS (recommended)

Code Repository

- Git. (Classroom examples, homework assignments, etc.)

Graphics Equipment

For assignments, use equipment in Workstation Lab

Workstation Lab

- Live Status Updates: http://www.ece.lsu.edu/koppel/gpup/sys-status.html
- Room 126 EE Building
- Several kinds of computers.

Graphics Workstations

- Mix of recent high-end graphics GPUs.
  - NVIDIA GTX 1080s (Fall 2017)
- Some machines have scientific-computing-grade accelerators:
  - NVIDIA K20c, GP100
  - Intel Xeon Phi