CS 179: GPU Programming

Introduction
What We Will Cover

Programming GPUs, of course:
  • OpenGL Shader Language (GLSL)
  • Compute Unified Device Architecture (CUDA)

Other Interfaces:
  • OpenCL
  • DirectX/HLSL/DirectCompute
  • ATI Stream
  • Many others...
What we will cover

• What do we use the GPU for?
  • Graphics processing/rendering
  • Simulation
  • General purpose calculations
  • Basically anything parallelizable...
    • But why?
What we will cover

• Parallel Processing
  • Used everywhere
    (probably covered in CS2)
  • CPU: Most nowadays have 2-4 cores
    • Nice ones have 6, 8, 12, or even 16
• GPU:
  • NVIDIA GeForce GTX 560: 324 CUDA Cores
  • Radeon HD 7970: 2048 Processors
  • NVIDIA GeForce GTX 690: 3072 CUDA Cores (!)
What we will cover

- GPU for visualization
  - Mostly OpenGL, GLSL
  - Techniques used mainly for games
  - Can use GLSL for general computation, but this is tedious and difficult
What we will cover

- GPU for general computation
  - GLSL is one (undesirable) option
  - CUDA and OpenCL provide much cleaner, easier interfaces
    - We will use CUDA in this course
GLSL Topics

• Shading
  – Phong shading (per-pixel lighting)
    • Very easy; most common application of GPU programming (first homework)
  – Bump mapping, normal mapping, parallax mapping
    • Get (much) more detail without rendering more polygons
GLSL Topics

• Particle systems
  – The ‘old’ way of doing simulation on the GPU
    • Still relevant…
  – Ping-ponging textures (buffer flipping)
  – Vertex and pixel buffer objects

• Modern GPU pipeline
  – DirectX 11, OpenGL 4.0
    • GPU Tessellation
CUDA Topics

- GPU hardware and architecture
- CUDA architecture and its mapping to hardware
  - Threads, blocks, grids
  - Register, local, shared, global, texture, constant, and host memories
- Parallel programming basics
- Numerical considerations (floating point arithmetic, double-precision issues)
- Optimization and multi-GPU programming
  - Difficult to do general GPU optimization…
CUDA Applications

- Linear algebra
  - Can be up to 100,000 times faster than on the CPU!
- Digital signal processing
  - Audio, images, videos
- Fluid and/or particle simulation
- Fractal generation
- Raytracing
  - Not as impressive speedups; often divergent…
What we will NOT cover

• OpenGL
  - You can read and adapt the CPU code we provide, though
  - Or, take CS171 for a more thorough introduction to graphics

• C/C++
  - Take CS11 and/or CS24 first!
Administrative details…

• Assignments
  – 6 or 7 weekly homework assignments
    • 3 GLSL, 3 CUDA
    • Can adjust based on interest
  – Final Project:
    • Last weeks of class
    • Open project, do something interesting!
    • We have ideas if you need them
  – To pass, you’ll need an overall score of 60%, and a reasonable attempt on each assignment
  – No exams
Administrative details…

Assignments:
- Due Wednesdays (up for change)
- Office hours: Monday or Tuesdays (also up for change)
- Code must run on the CS cluster
- Turn in via e-mail to any TA
- Late policy:
  • Talk to us for extensions, or get a note from Deans
Administrative details...

● Note on assignments:
  – GLSL assignments will run on most relatively new graphics cards, including those in the Annenberg CS lab
  – CUDA assignments will only run on machines with newer NVIDIA graphics cards
    • You'll need to use the CUDA machines in the lab for this
  – I recommend you start early in case you run into problems...
Resources
(You can find these on the course website)

- OpenGL Red Book
  - General resource for OpenGL/graphics programming
- OpenGL Orange Book
  - GPU/GLSL version of the Red Book
  - Examples, documentation, drivers, etc.
  - Don't look up labs!
- Programming Massively Parallel Processors: A Hands-on Approach (Dave Kirk, Wen-mei W. Hwu)
Resources

• Course website:
  http://www.cs.caltech.edu/courses/cs101gpu/
  - Lectures, resources, and assignments will be posted here

• Contacts:
  - Connor DeFanti: cdefanti@caltech.edu
  - Tuan Anh Le: tle@caltech.edu
  - Prof. Al Barr: barr@cs.caltech.edu