Slide 1

Administrivia

- Due date for Homework 2 is not soon, but it's long, so I say start now/soon.

Slide 2

Minute Essay From Last Lecture

- (How to decide whether a RNG is "good" . . .)
- Some people mentioned just trying it in the intended application and observing whether the results seem good. Might actually be the best way, since a RNG that's "good" for one application might be less so for another.
- Others mentioned checking whether the sequence is evenly distributed, or has no repeats.
- In principle, what you want is something that seems unpredictable — but keeping in mind that for at least some applications you do want results to be reproducible.

Slide 3

Sidebar: `gnuplot`

- A tool I like for both quick interactive plots and nice-looking ones to use in papers is `gnuplot`. Available on most UNIX-like systems and (I think!) optionally for other operating systems. Home page at `gnuplot.sourceforce.net`. Can do 2D and 3D plots, the former with Cartesian or polar coordinates.

  (Interestingly(?) enough, the name has nothing to do with the GNU project!)
- To start it, `gnuplot`. Brings up a command-line interface. Online help available with `help`.

Slide 4

`gnuplot`, Continued

- Useful commands include `plot` to plot function(s) or data from file(s), `set` to set various things (e.g., x and y ranges).
- Default output to terminal, but with `set terminal` and `set output` you can instead store to a file in various formats.
- Can also put commands (`plot` etc.) in a file and execute batch-style, or with `load`. Useful if you want to regenerate plots when data changes.
- (Examples.)

Slide 5

Homework 2 — Implementing LCG

- Implementing a 48-bit LCG function is doable in both C (with `int64_t` and Java `Long`). Note, however, that the multiplication required to generate the next element can overflow — which is no problem since we only want the value mod 2^{48}, *but* consider what happens if the overflow produces a negative result. Hence my suggestion to compute this with bitwise "and" (`&`) rather than with `%`.
- Implementing the described leapfrog scheme is trickier. I decided to try it in Scala first and discovered that while the modified constants a' and b' only need to be 48 bits, computing them correctly — well, my current approach is to use `BigInt` to do the computation and then convert the result back to a `Long`. This should work in Java too (with suitable changes of names), and in C . . . There's a library called GMP that provides support for arbitrary-precision arithmetic that looks promising. "Stay tuned"?

Slide 6

GPGPU

- Recall from overview/introduction that the SIMD (Single Instruction, Multiple Data) model was popular in the relatively early days of parallel programming, fell of favor, and is now making a comeback as "GPGPU" (General-Purpose computing on Graphics Processing Units).
- Typically SIMD is a good fit for GPU hardware — but it's worth noting that they usually(?) have their own memory, not shared with "host" CPU, which makes programming more complicated and has implications for performance.

Slide 7

OpenCL

- Early work on shared-memory and message-passing programming resulted in many competing programming environments — but eventually, OpenMP and MPI emerged as standards.
- Similarly, initially many different programming environments for GPGPU, but OpenCL might be emerging as a standard.
- In both cases, idea was to come up with a single standard, then allow many implementations. For MPI, standard defines concepts and library. For OpenMP, standard defines concepts, library, and compiler directives. For OpenCL, standard defines concepts and library.
- First release 2008; evolving fairly rapidly. Meant to address not just GPGPU but more-general problem of "heterogeneous computing" (computing using mix of computational resources).

Slide 8

What's an OpenCL Program Like?

- Source code in C/C++, with calls to OpenCL functions.
- Typically includes source to be compiled at runtime for whatever device is to be used. "Device"? yes, many new terms/concepts . . . (And in context here it means something not exactly like what it has come to mean in popular usage!)

Slide 9

OpenCL Terms and Concepts

- *Compute device* — something capable of doing computations (CPU, GPU, etc.).
- *Kernel* — computation to execute on device.
- *Index-space* — range of indices (1D or more) on which to execute kernel.
- *Work-item* — one execution of kernel. Grouped into *work-groups*.
- *Compute context*, *program object*, *command queue* — various aspects of setting up environment and assigning work to devices.
- Several *memory regions* — host memory, local memory, etc.

Slide 10

OpenCL on Department Machines

- OpenCL implementations available for various platforms.
- The one for NVIDIA cards is part of the company's own toolkit for GPGPU, called CUDA. Installed on machines with NVIDIA graphics cards — Atlas machines, Deimos.
- The one for AMD cards is also part of the company's own toolkit, called AMD APP SDK. Hooked into device driver for GPU, and — well, this year we weren't able to get it installed on the machines with AMD graphics cards.

Slide 11

Simple(?) Examples

- Maybe worth noting that you can't really write a "hello world" program, since compute device doesn't necessarily have access to standard output!
- So as a first example — vector addition, briefly today and in more detail next time.

Slide 12

Minute Essay

- None really — just sign in, unless questions?