How to Make a Lumpy Random Number Generator

Michael A. Covington

University of Georgia
and CORAID, Inc.

International Workshop on Plan 9
Athens, Georgia      October 23, 2009
“How would you make a random number generator with a preference for certain values?”
(This not a paper about Plan 9. It’s about some scientific computing that happens to have been done with Plan 9.)
Why would anybody want a “lumpy” random number generator?

- Simulation

- To equalize wear on machinery, load on networks, etc.

- To compensate for nonlinearity elsewhere in the system

- Because it’s an interesting mathematical problem!
A very simple way to get non-uniform random numbers:

Use your random number generator $n$ times, and sum the results.
$n = 1$
$n = 2$
$n = 4$
But how do you control the nonuniformity?

Synthesize any histogram you want, by combining bell curves!
Key idea:
A bell curve is a lot like a wavelet.
Any (finite) curve can be synthesized by adding wavelets together.
**Difference:** Wavelets go below 0, average to 0, so adding wavelets doesn’t change the height of the curve you’re building.
For us, that doesn’t matter, because the height of the histogram is constrained by the fact that it’s a histogram!
Code to make a bell curve with controlled width and position

```c
int genrand(int bmin, int bmax, int rmin, int rmax, int n)
{
    int i, u, sum;
    do {
        sum = 0;
        for (i=0; i<n; i++) sum += bmin + (rand() % (bmax - bmin));
        if (sum < 0) sum -= n-1; // prevent pileup at 0
        u = sum / n;
    } while ( !(rmin <= u && u < rmax) );
    return u;
}
```
Code to stack several bell curves for a custom shape

```c
int customrand(void)
{
    switch (rand() % 10)
    {
        case 0:
        case 1:
        case 2:
        case 3:
            return genrand(0,1000,0,1000,1);  // flat baseline
        case 4:
        case 5:
        case 6:
            return genrand(-400,300,0,300,3);  // peak beyond left edge
        case 7:
        case 8:
            return genrand(600,900,600,900,3);  // peak at 750
        default:
            return genrand(0,700,0,700,3);  // low, broad peak at 350
    }
}
```
The result
Why do this rather than transform with a polynomial?

- No floating-point math
- Not much code (Good for compact embedded systems)