

## Re: rand() pattern in texture?

**Source:** <http://sci.tech-archive.net/Archive/sci.math/2004-10/4183.html>

---

**From:** Phil Carmody (*thefatphil\_demunged\_at\_yahoo.co.uk*)

**Date:** 10/14/04

Date: 14 Oct 2004 20:13:32 +0300

Carlos Moreno <moreno\_at\_mochima\_dot\_com@xx.xxx> writes:

> *Phil Carmody wrote:*

>

> >>> *This is an urban legend -- it's false, period! (from what I've*

> >>> *read, I think there was, at some point in history, a buggy*

> >>> *implementation of rand() somewhere, and that one had less*

> >>> *uncertainty in the LSbits than in the MSbits).*

> > *No, what you've read has been incorrect. There were \_many\_ PRNGs*

> > *which were \_designed\_ to be crap (usually LC ones), and were*

> > *implemented correctly.*

>

> *I meant implementation of the LC concept, as opposed to the*

> *programming implementation of a given algorithm. The*

> *implementation of a LC includes the choice of the multiplier*

> *and the modulo -- and that's most likely what was "wrong".*

>

> > *If you think about it, even briefly, you'll see that the low order n*

> > *bits of any LC PRNG \_must\_ have period  $\leq 2^n$ , if you think about it.*

>

> *This is completely false.*

I was assuming the usual use thereof, the situation of M being a power of 2, I should have said so, sorry.

If M is not a power of 2, and the range you return is a power of two, then you either have to throw away samples, or have a non-uniform distribution.

> *Here's a linear congruential generator:*

>

>  $X_n = (X_{n-1} * 11 + 8) \text{ mod } 17$

>

> *Here's the sequence it produces when seeded with 1*

> *(actually, here's part of the sequence it produces when*

> *seeded with anything):*

>

> 1 2 13 15 3 7 0 8 11 10 16 14 9 5 12 4 1 2 13 15

sci.math: Re: rand() pattern in texture?

$$P(i : i \neq 6) = 1/16$$

$$P(6) = 0.$$

That's not a uniform distribution.

Now imagine what happens if you seed it with 6.

- > > *I'm sure that Carlos can work out therefore what the period for the*
- > > *lowest single bit must be.*
- >
- > *I'm sure you're suggesting that the period of the single*
- > *lowest bit MUST be 2 or 1... Well, doesn't look like a*
- > *period of 2 to me. The single lowest bit of the above is:*
- >
- > *1 0 1 1 1 1 0 0 1 0 0 0 1 1 0 0 1 0 1 1 1 1 .....*
- >
- > *Looks to me like the period of the single lowest bit is 17,*
- > *as is the period of the generator itself.*

No and yes. You're non-maximal. If you'd have found a maximal one, it would have been 17, yes. The period of the lower bits always divides the order of a larger set of bits, obviously.

- > *But I think we have a winner!! I think we've just found the*
- > *origin of the urban legend, which I now must concede, may not*
- > *be 100% urban legend: Your argument is true IF the modulo*
- > *is a power of 2 -- which I think is almost always the case,*
- > *for efficiency reasons.*

That was the assumption, yup, your deduction is correct.

- > *OR, perhaps this is precisely what the "buggy implementations"*
- > *used to do, and perhaps nowadays they realized that it's better*
- > *to use a prime number as the modulo -- or perhaps they did what*
- > *you suggest below: never show the lower numbers; show only*
- > *the bits starting at bit 16 or bit 32 -- if that is the case,*
- > *then it is still true that the lower bits are "less random"*
- > *(i.e., have a shorter period), even if it is not too obvious*
- > *(a period of  $2^{32}$  instead of  $2^{64}$ , or  $2^{16}$  instead of  $2^{48}$ ,*
- > *etc.).*

The crazy thing is that by adding a tiny bit more state, you can expand the period to be so huge that you can entirely ignore all low bits, with their short periods. Given that word\*word%word calculations are relatively expensive, you can clock simple lagged  $M=2^{|word|}$  ones many times for lower computational cost than a single  $M=prime$  operation.

- > *But notice that this is not a property of LCG's per se; it*
- > *is a property of a particular class of implementations of*

Re: rand() pattern in texture?

sci.math: Re: rand() pattern in texture?

> LCG -- those that use a power of 2 as the module.

Yup. Knuth devotes a little section to this very situation, prefixing it with the dreaded words "the most common..." or similar, IIRC.

> > I have reason to believe, from reading various empirically-  
> > oriented papers, that Microsoft C, IBM fortran, Microsoft  
> > Visual Basic, and others also have weak generators, or at least did  
> > up until 2002.

>  
> I've been contesting your arguments so far, but as far as  
> the bottom line of the argument goes, I am with you -- I've  
> always thought to myself "why, oh why, do they use crappy  
> PRNG as the standard random number generator for most  
> compilers/languages?" -- whenever you use a built-in facility  
> of a language/compiler, you tend to think that it is a good  
> implementation; if you need to take a square root, your  
> first thought wouldn't be to code your own square root  
> algorithm because you have reason to believe that the  
> built-in one is crap -- no, you tend to assume that it  
> would be extremely hard to do better than the built-in one,  
> since that one was coded by serious people, presumably with  
> a lot more experience than one has. This general argument  
> breaks down so loudly when it comes to rand()... Why can't

Couldn't agree more.

> they use MT, or, if they want something simple (to guarantee  
> efficiency in the general case), something like a feedback  
> shift-register, or even a substitution-permutation network?

I can't approve the MT as standard, it's too much a heavyweight footprint-wise. Marsaglia's multiply with carry one's a pretty good compromise. (period  $2^{63}$  with a 32-bit word size, IIRC).

> Ok, I'll stop here... I guess it is getting close to that  
> time at which we should let this thread die...

Ooops! :-)

Phil

--

They no longer do my traditional winks tournament lunch - liver and bacon.  
It's just what you need during a winks tournament lunchtime to replace lost  
... liver. -- Anthony Horton, 2004/08/27 at the Cambridge 'Long Vac.'