GPU Random Numbers via the Tiny Encryption Algorithm

Fahad Zafar, Marc Olano and Aaron Curtis

University of Maryland Baltimore County
Outline

- Introduction
- Previous Work
- Analysis and Results
- Monte Carlo Shadow Algorithm
- Conclusion
Introduction

• GPUs need random numbers
  ◦ Real time graphics, Simulations, GPGPU

• Required characteristics
  • Repeatability
  • Random access
  • Multiple independent streams
  • Speed
  • Minimal Statistical bias
Gradient Noise

- Use pseudo random gradient vectors at each lattice point and use them to generate the noise value.

Visualizing 2D and 3D gradient noise neighboring vectors

(Left) 2D Fourier Transform
(Right) 2D Noise Tile
Tiny Encryption Algorithm for gradient noise

PRNG

Tiny Encryption Algorithm

2D/3D/…..
Outline

- Introduction
- Previous Work
- Analysis and Results
- Monte Carlo Shadow Algorithm
- Conclusion
Previous Work

- The Image Synthesizer Perlin [1985] and Improving Noise papers by Perlin [2002]

Required a memory lookup for a permutation table-based hash and a table of random gradients for each point on an integer grid surrounding the noise argument.
Previous Work

**Wavelet Noise** is an alternative to Perlin noise which reduces the problems of aliasing and detail loss, that are encountered when Perlin noise is summed into a fractal.

Figure 2: (a) Image $R$ of random noise, (b) Half-size image $R^{\downarrow}$, (c) Half-resolution image $R^{\uparrow\downarrow}$, (d) Noise band image $N = R - R^{\uparrow\downarrow}$.
• **Anisotropic Noise by Goldberg et al. [2008]**  
  ◦ Blend of directional noise stored in different channels of a texture to reduce detail loss

Images Reference: Anisotropic Noise Goldberg et al. 2008
**Modified Noise by Olano 2005**

- Used Blum Blum Shub to create permutations on the fly.
- No texture access
- Very Fast and Simple
- Bad quality output

\[ \text{Hash}(x) = x^2 \mod M, \; M = 61 \]
• White Noise Generation via Cryptographic Hash (Tzeng and Wei [2008])
  ◦ Use MD5 instead of Blum Blum Shub.
  ◦ Quality random numbers.
  ◦ Very Slow (64 rounds)
    • (adds to number of instructions)
  ◦ 4 Types of rounds

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>DH Tests Passed</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5GPU</td>
<td>15 / 15</td>
</tr>
<tr>
<td>GPU CEICG</td>
<td>10 / 15</td>
</tr>
<tr>
<td>GPU BBS</td>
<td>2 / 15</td>
</tr>
<tr>
<td>GPU AES</td>
<td>7 / 15</td>
</tr>
<tr>
<td>Goulburn</td>
<td>9 / 15</td>
</tr>
<tr>
<td>rand</td>
<td>6 / 15</td>
</tr>
<tr>
<td>drand48</td>
<td>12 / 15</td>
</tr>
<tr>
<td>M. Twister</td>
<td>15 / 15</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Previous Work
- Analysis and Results
- Monte Carlo Shadow Algorithm
- Conclusion
Compute bound shader analysis

Sequential Texture Access

Random Texture Access

GPU: NVIDIA 8600M
Searching for a Faster Hash

- **RC4;** which requires a lookup table [Golic 1997]
  - Stream cipher
  - Key scheduling algorithm (initialize)

- **CAST-128;** which requires a large amount of data for initialization
  - is a block cipher
  - There are three alternating types of round function.

- **Blowfish;** which was fast but each new key required pre-processing equivalent to encrypting nearly 4 kilobytes of data [Schenier 1996]
Tiny Encryption Algorithm (TEA)

- Used extensively on legacy computers
- Analyzed by Reddy [2003] (Defeating TEA)
- Perfect match to our requirements
- XTEA and XXTEA [Needham et al. 1997]

**One Round**

```plaintext
sum += 0x9e3779b9;
v0 += ((v1 << 4) + 0xA341316C)^(v1 + sum)^(v1 >> 5) + 0xC8013EA4;
v1 += ((v0 << 4) + 0xAD90777D)^(v0 + sum)^(v0 >> 5) + 0x7E95761E;
```

<table>
<thead>
<tr>
<th>XOR</th>
<th>SHIFTS</th>
<th>ADDITIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>
Gradient Noise Fourier Transforms

(a) Perlin [Per02]  (b) BBS [Ola05]  (c) MD5_{64} [TW08]  (d) TEA_{2}  (e) 4×4 Kernel TEA_{2}
Timing Noise with different PRNGs in Mpxixels/sec

<table>
<thead>
<tr>
<th>RNG</th>
<th>2-D Noise HD 4870</th>
<th>2-D Noise 8600M</th>
<th>4 x 4 Kernel HD 4870</th>
<th>4 x 4 Kernel 8600M</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBS</td>
<td>3077</td>
<td>1628</td>
<td>27</td>
<td>10</td>
</tr>
<tr>
<td>MD5\textsubscript{64}</td>
<td>173</td>
<td>61</td>
<td>322</td>
<td>281</td>
</tr>
<tr>
<td>TEA\textsubscript{8}</td>
<td>1000</td>
<td>322</td>
<td>281</td>
<td>81</td>
</tr>
<tr>
<td>TEA\textsubscript{2}</td>
<td>3243</td>
<td>1305</td>
<td>1000</td>
<td>364</td>
</tr>
<tr>
<td>XTEA\textsubscript{16}</td>
<td>635</td>
<td>184</td>
<td>187</td>
<td>40</td>
</tr>
<tr>
<td>XTEA\textsubscript{2}</td>
<td>3529</td>
<td>1551</td>
<td>1143</td>
<td>419</td>
</tr>
</tbody>
</table>
Per round frequency domain analysis for noise

• Good quality noise can still be produced with a lower quality random number generator.
Per round time analysis for Encryption Algorithms as RNG

GPU: NVIDIA 9800 GT
DIEHARD Tests passed

TEA vs MD5_64
DIEHARD Tests passed

XTEA vs MD5_64
DIEHARD Tests passed

XXTEA vs MD5_64

[Graph showing comparisons between XXTEA and MD5_64]
NIST Tests passed

TEA vs MD5_64

NIST Tests Passed

No. of tests passed

No. of rounds

- TEA
- MD5
- XTEA
- XXTEA

N/A
Usability scaling for PRNGs in shaders

GPU: NVIDIA 9800 GT
Cuda analysis results

Time (in milliseconds) to complete all threads on NVIDIA 8600M. Some intervals were too short for the counter.
Outline

- Introduction
- Previous Work
- Analysis and Results
- Monte Carlo Shadow Algorithm
- Conclusion
Real-time Soft Shadows via Monte Carlo Sampling

- Shadow maps are used to represent discretized occluding geometry.
- More than one shadow maps.
- Ray tracing done through the depth maps.
- Depth sensitive Gaussian filter.

An area light source and corresponding shadows. In our approach, the depth map is rendered from a projection point placed behind the center of the light.
Monte Carlo Shadow Algorithm demo

1. Algorithm Demonstration. Sampling at the edge of the shadow region

2. Increase Penumbra region

3. TEA-1

4. TEA-2 and TEA-4

5. Add filtration
Outline

- Introduction
- Previous Work
- Analysis and Results
- Monte Carlo Shadow Algorithm
- Conclusion
Conclusion

- We have provided an extensive per round analysis for TEA and MD5

- TEA is a better alternative to MD5 for GPU noise and as a PRNG
Conclusion

- Easily adjustable Speed/Quality tradeoff. Our approach is more tunable and comparatively better.

- TEA is easy to scale and implement
  - Adjusts to both compute intensive and compute extensive applications
Questions