CUDAでポストエフェクト

ぺりむ (@hi2p_perim),
originally from @luminance64

ポストエフェクト？

a.k.a. ポストプロセッシング (Post Processing)
リアルタイムCGでは...

画面全体にかかるエフェクト
full-screen post processing

Depth of Field (DoF)

Screen Space Ambient Occlusion (SSAO)

Fast Approximate Anti-Aliasing (FXAA)

CUDA

NVIDIAが開発したGPUを用いた汎用計算用のフレームワーク。CGにも使える。

目的:

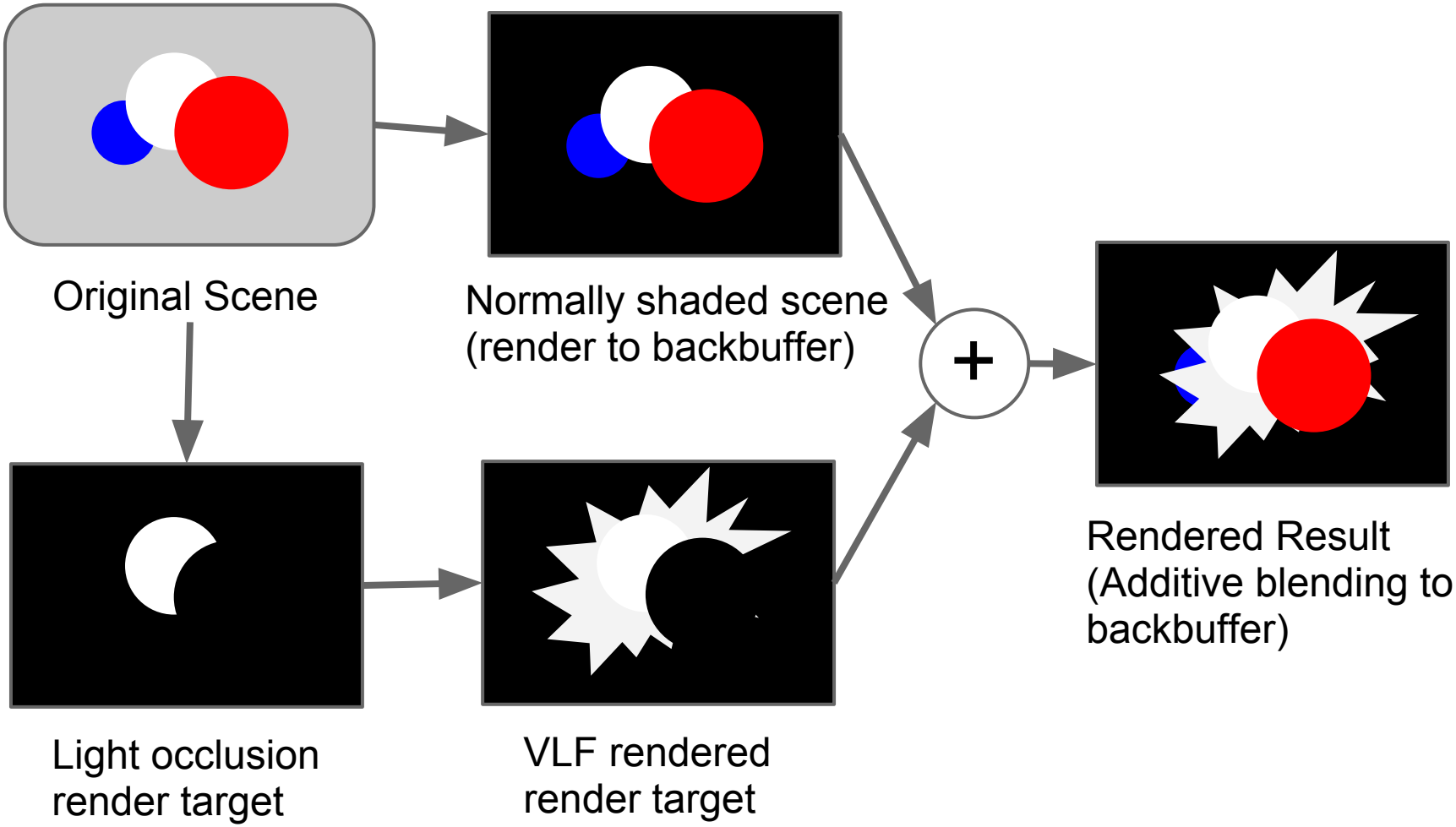
CUDAを用いて、ポストエフェクトを何か実装する

実装するポストエフェクト

Volumetric Light Scattering, God ray
(cf. GPU Gems 3, Chapter 3)

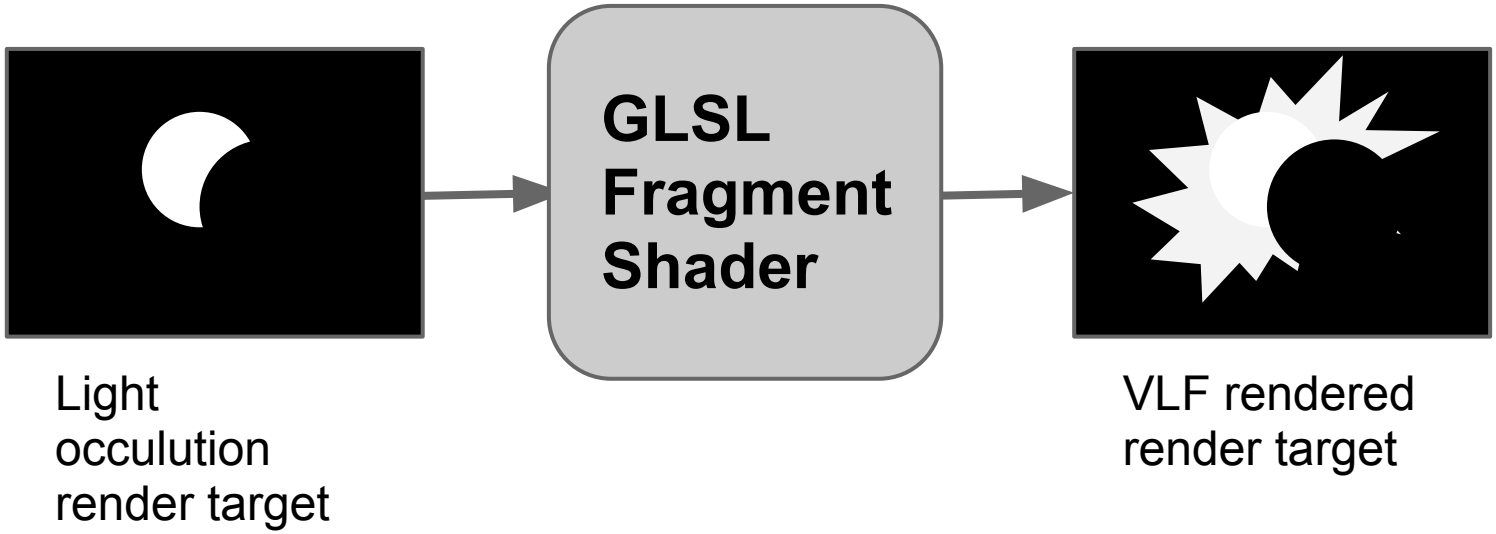
Actual picture

実装手法


Original Scene
Normally shaded scene
(render to backbuffer)
+
Rendered Result
(Additive blending to
backbuffer)
Light occlusion
render target
VLF rendered
render target

GLSL Fragment Shader Version

GLSLでも実装できる

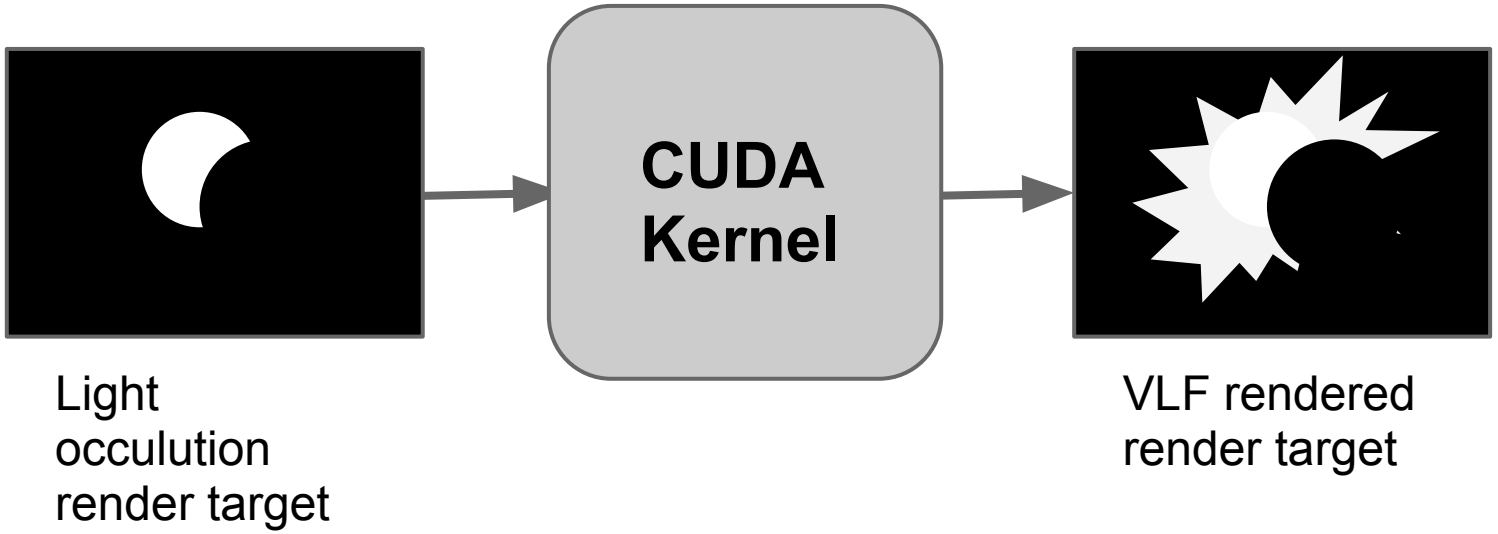


GLSL Fragment Shader Version

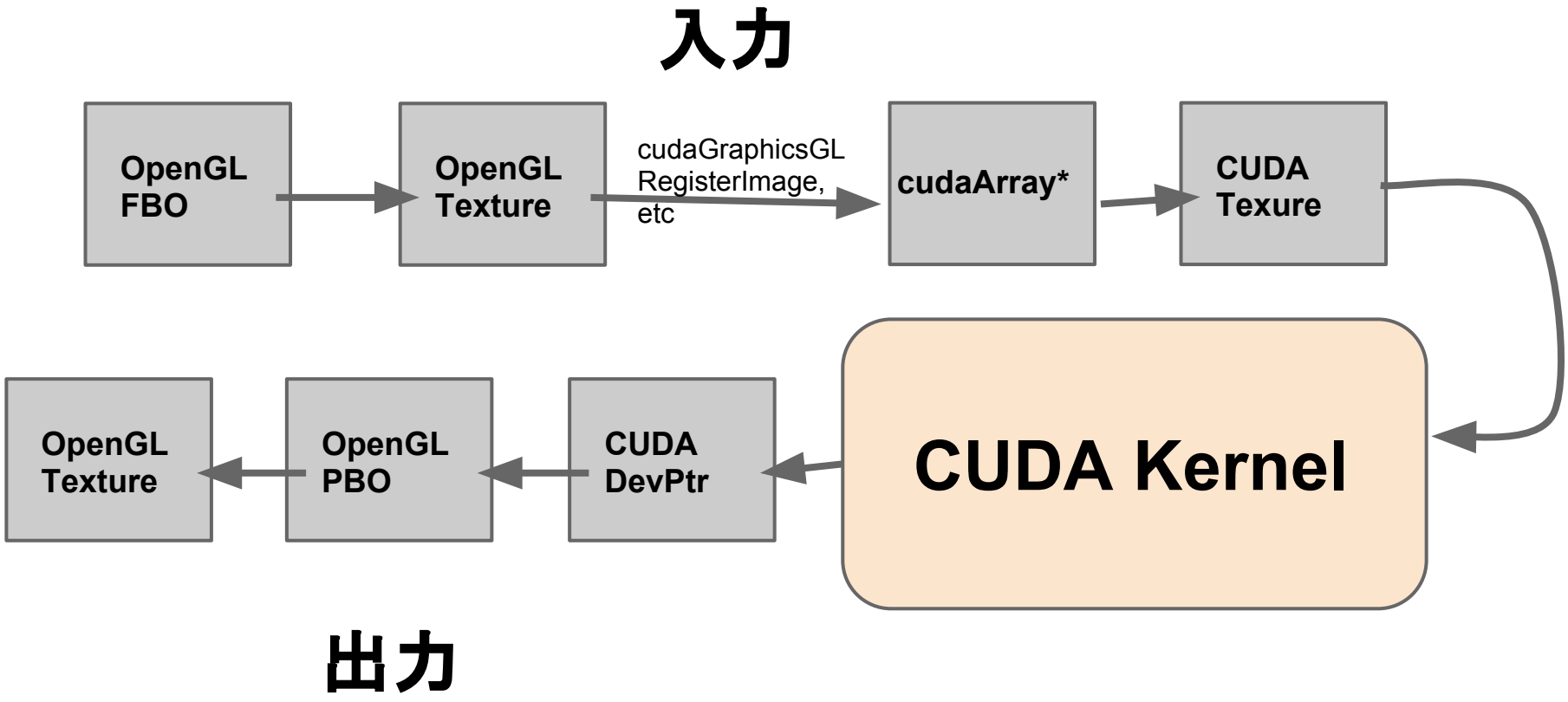
```
in vec2 vTexCoord;
out vec4 fragColor;

uniform vec2 screenLightPos;
uniform sampler2D RT;

const int numSamples = 100;
const float exposure = 0.003f;
const float decay = 1.0f;
const float density = 0.8f;
const float weight = 5.8f;
```

```
void main()
{
    vec2 deltaTexCoord = vec2(vTexCoord - screenLightPos);
    vec2 texCoord = vTexCoord;
    deltaTexCoord *= 1.0 /  float(numSamples) * density;
    float illuminationDecay = 1.0;

    for (int i = 0; i < numSamples; i++)
    {
        texCoord -= deltaTexCoord;
        vec4 sample = texture(RT, texCoord);
        sample *= illuminationDecay * weight;
        fragColor += sample;
        illuminationDecay *= decay;
    }

    fragColor *= exposure;
}
```

CUDA Version



OpenGLとCUDAの連携

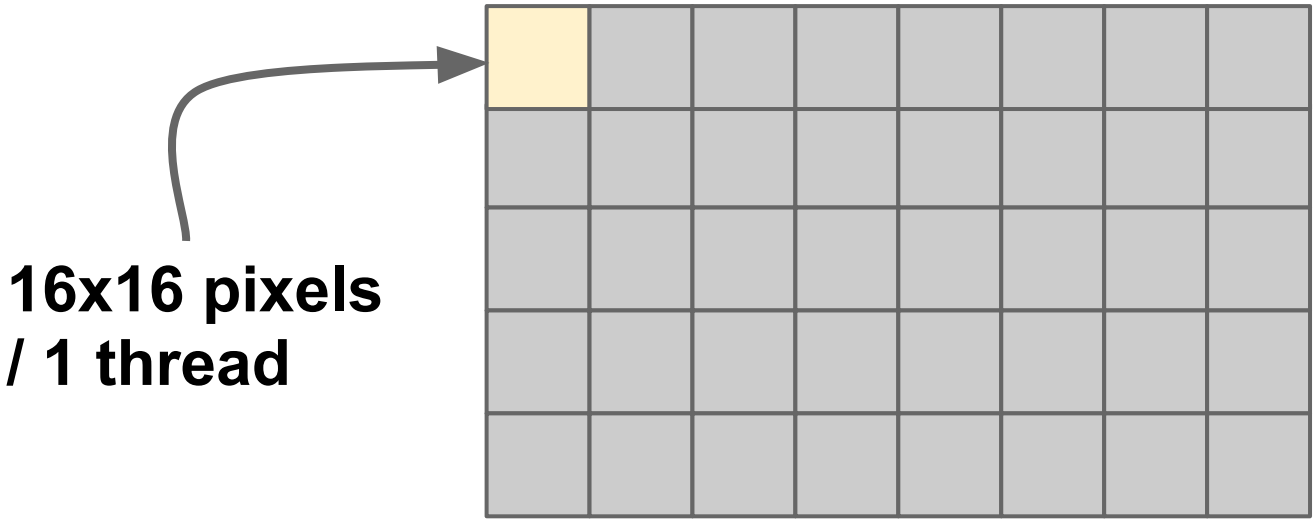


OpenGLとCUDAの連携

```
// Map resources
HandleCudaError(cudaGraphicsMapResources(1, &cudaPbo, NULL));
HandleCudaError(cudaGraphicsMapResources(1, &cudaLightOcclusionRt, NULL));

// Get device pointer
uchar4* cudaPboDevPtr;
size_t bufferSize;
cudaArray* cudaLightOcclusionRtDataArray;

// Get the device pointer of the output PBO
HandleCudaError(cudaGraphicsResourceGetMappedPointer((void**)&cudaPboDevPtr, &bufferSize, cudaPbo));

// Get the data array of the input texture
HandleCudaError(cudaGraphicsSubResourceGetMappedArray(&cudaLightOcclusionRtDataArray,
cudaLightOcclusionRt, 0, 0));

// Run the kernel
// ...

// Unmap resouces
HandleCudaError(cudaGraphicsUnmapResources(1, &cudaLightOcclusionRt, NULL));
HandleCudaError(cudaGraphicsUnmapResources(1, &cudaPbo, NULL));
```

Kernel関数の起動

- block size
  - Stream Multiprocessorの数だけ設定
  - GTX 570は15個
- thread size
  - 適当に2次元的に分割



Kernel関数

```
texture<float4, 2, cudaReadModeElementType> RT;
__device__ unsigned int blockCounter;
__global__ void VLSKernel(int width, int height, int gridWidth, int gridNum, float2 screenLightPos,
uchar4* dst)
{
    __shared__ unsigned int blockIndex;
    __shared__ unsigned int blockX;
    __shared__ unsigned int blockY;
    while (1)
    {
        (Blockを選択する)
        int ix = blockDim.x * blockX + threadIdx.x;
        int iy = blockDim.y * blockY + threadIdx.y;
        if (ix < width && iy < height)
        {
            float4 fragColor;
            (ロジック: 略)
            (色を出力する)
        }
    }
}
```

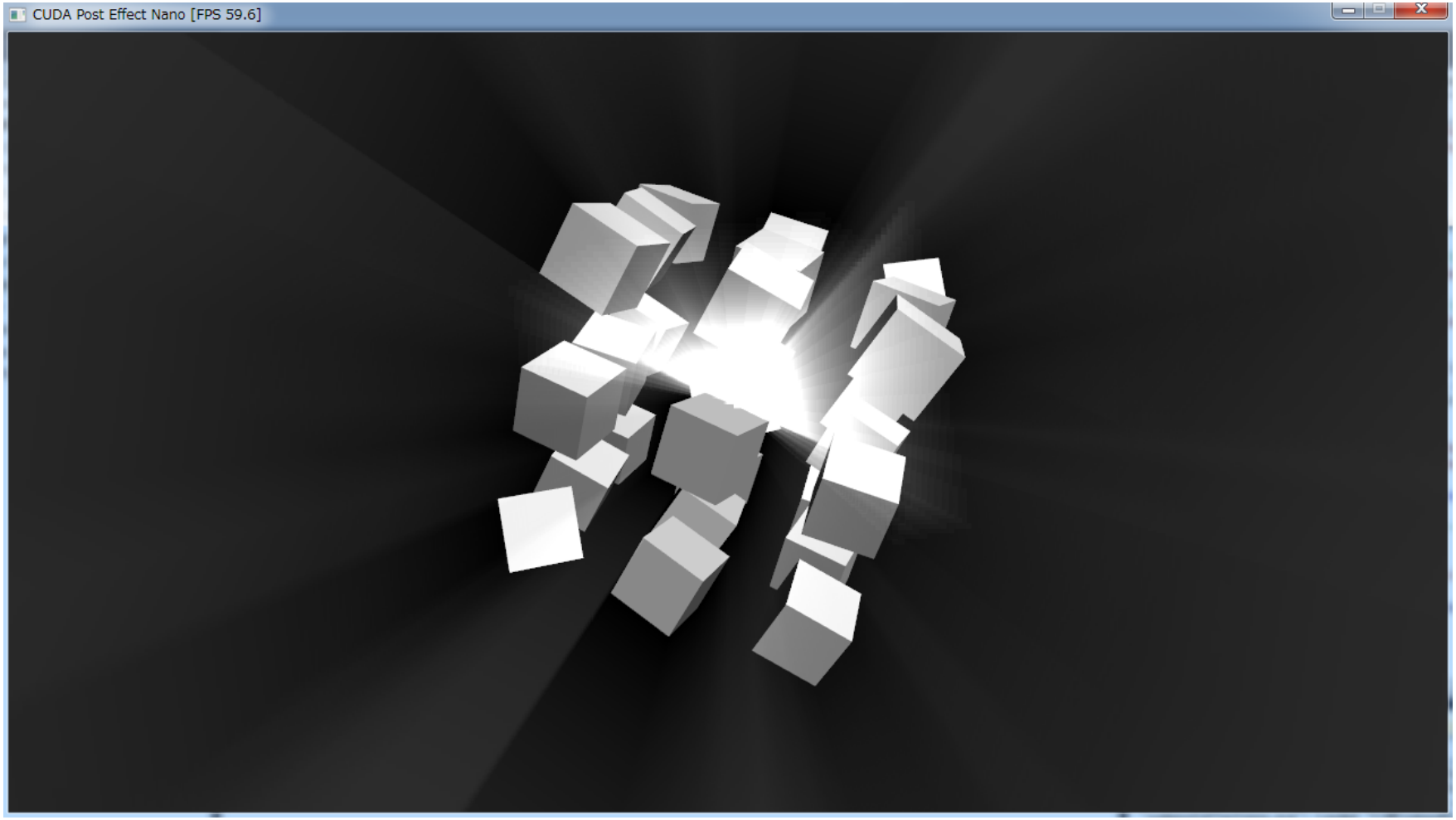
Kernel関数: blockの選択

```
if (threadIdx.x == 0 && threadIdx.y == 0)
{
    blockIndex = atomicAdd(&blockCounter, 1);
    blockX = blockIndex % gridWidth;
    blockY = blockIndex / gridWidth;
}

__syncthreads();

if (blockIndex >= gridNum)
{
    break;
}
```

Kernel関数: 色を出力する

```
uchar4 icolor;
icolor.x = clamp(255.0f * fragColor.x, 0.0f, 255.0f);
icolor.y = clamp(255.0f * fragColor.y, 0.0f, 255.0f);
icolor.z = clamp(255.0f * fragColor.z, 0.0f, 255.0f);
icolor.w = clamp(255.0f * fragColor.w, 0.0f, 255.0f);

int offset = width * iy + ix;
dst[offset] = icolor;
```

結果



結果

- パフォーマンスはGLSLより遅い
- あくまでも実験