Developing a video simulation platform using multiple GPGPU Technologies

Freek Nossin

Introduction

- Personal background
- TASK24
- NXP

Contents

- Introduction application domain
- Gpu Challenges
- Shaders & Cuda

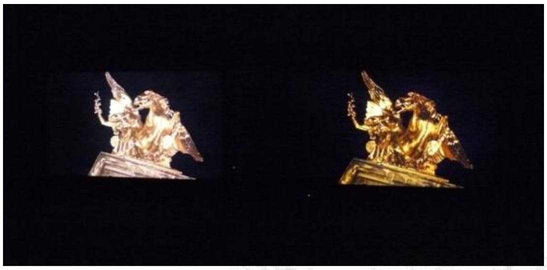
APPLICATION DOMAIN

Display Solutions

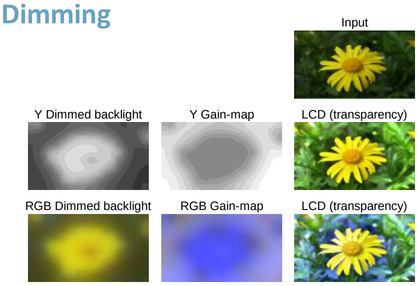
- All kinds of displays
  - Mobile phones
  - Flat panels
- Image/Video enhancement
  - Improve image quality
  - Lower power usage

Display technology

Local Dimming



Local Dimming



Ref [1,2]

Other examples

- 3D Images

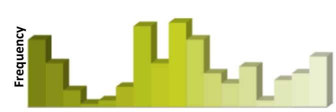
- Natural Motion
- ...

GPU COMPUTING CHALLENGES

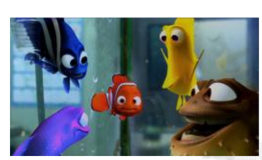
Video Simulation

- Huge amount of data
- Computational intensive algorithms
- Flexible development environment
- Optimization is last development step

Histogram



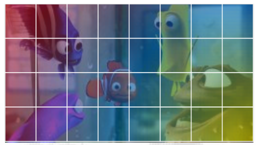
256 “buckets”



~2.000.000 pixels

Example dimming

- Histogram analysis
- Backlight dependent
  - 0D
  - 1D
  - 2D (local)
  - or “3D”



Histogram analysis

- Basic Algorithm
  - #bins can vary

```
for(int i = 0; i < datalen; i++)
{
   bin = data[ i ];
   histogram[bin]++;
}
```

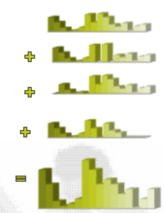
- Parallelization seems trivial

Histogram problem

- Memory access pattern
  - Access to image data is sequential
  - Result data is accessed randomly
- Multithreaded histogram incrementation
  1) read current count for bin
  2) increment count
  3) write updated count to memory

Histogram Solution

- Split up computation
  - Allows use of fast local memory
  - Less collisions



- Use atomic operations
  - Problem: atomic operations are not supported on every architecture

Ref [3, 4]

Final solution (without atomics)

```
#define TAG_MASK ( (1U << (UINT_BITS - LOG2_WARP_SIZE)) - 1U )
const uint threadTag = threadIdx.x << (UINT_BITS - LOG2_WARP_SIZE);

inline __device__ void addByte(volatile uint *s_WarpHist, uint data)
{
      uint count;
      do
      {
         count = s_WarpHist[data];
         count = count & TAG_MASK;
         count = count + 1;
         count = count | threadTag;
         s_WarpHist[data] = count;
      } while(s_WarpHist[data] != count);
}
```

Ref [5]

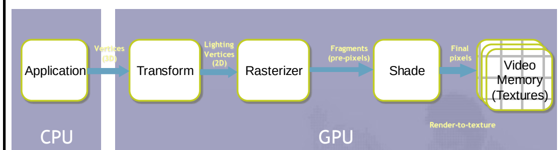
SHADERS VS CUDA

Graphics Cards api's

- Shader Languages, primarily focused on Games
  - GLSL (OpenGL)
  - HLSL (DirectX)
  - CG (DirectX and OpenGL)
- Gpgpu Languages, focus on scientific computations
  - SH
  - Brook
  - CUDA
  - AMD FireStream (based on Brook+)
  - OpenCL

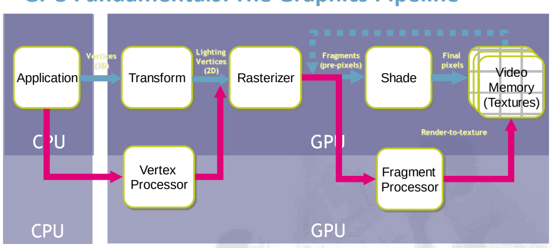
1 September 2010 19 Gpu symposium Tu/e - Developing a video simulation platform using Gpgpu

GPU Fundamentals: The Graphics Pipeline



1 September 2010 20 Gpu symposium Tu/e - Developing a video simulation platform using Gpgpu

GPU Fundamentals: The Graphics Pipeline



1 September 2010 21 Gpu symposium Tu/e - Developing a video simulation platform using Gpgpu

Shaders

- Pro
  - Easy to write
  - Designed for pixel based processing
  - Integrated in DirectX / OpenGL
  - Tooling and documentation available
- Con
  - No flexible input
  - Hard to debug

1 September 2010 22 Gpu symposium Tu/e - Developing a video simulation platform using Gpgpu

CG Tutorial – shader code

```
struct FragOut4
{
    float4 color : COLOR;  // output color
};

FragOut4 fpShaderExample
    (
        in float2 coords : TEXCOORD0,  // input texture coordinates
        uniform sampler2D inputTex,    // input texture
        uniform float f                // simulated "alpha" value
    )
{
    // output variable
    FragOut4 OUT;

    // retrieve the current pixel
    OUT.color = tex2D (inputTex, coords);

    // compute the new output pixel
    OUT.color = OUT.color * f;

    return OUT;
}
```

1 September 2010 23 Gpu symposium Tu/e - Developing a video simulation platform using Gpgpu

Cuda Example – The kernel

Loop over every pixel

```
__global__ void FilterKernel(unsigned int * dpVideoData,
                             unsigned int * dpOutputVideoData, int2 videoSize, float f)
{
    // compute global thread id
    const unsigned int globalTid = threadIdx.x + blockIdx.x * blockDim.x;

    // computer number of threads in this block
    const unsigned int num_threads = blockDim.x * gridDim.x;

    // multiply every pixel component with f
    for(int iPixel = globalTid; iPixel < (videoSize.x * videoSize.y); iPixel += num_threads)
    {
        unsigned int pixel = dpVideoData[iPixel];
        unsigned int outpixel = 0;
        outpixel = outpixel + ((unsigned int) ((pixel & 0x000000FF) * f) & 0x000000FF);
        outpixel = outpixel + ((unsigned int) ((pixel & 0x0000FF00) * f) & 0x0000FF00);
        outpixel = outpixel + ((unsigned int) ((pixel & 0x00FF0000) * f) & 0x00FF0000);
        outpixel = outpixel + ((unsigned int) ((pixel & 0xFF000000) * f) & 0xFF000000);
        dpOutputVideoData[iPixel] = outpixel;
    }
}
```

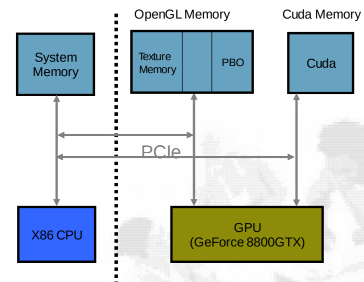
Pixel Operations have to be coded by hand.

1 September 2010 24 Gpu symposium Tu/e - Developing a video simulation platform using Gpgpu

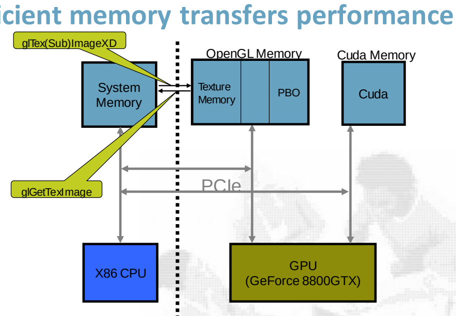
Combining shader and cuda programs

- OpenGL
  - PBO's (Pixel Buffer Objects)
    Can be used as a target or source for OpenGL operations
- Cuda OpenGL (and DirectX) Interoperability
  - Map on PBOs

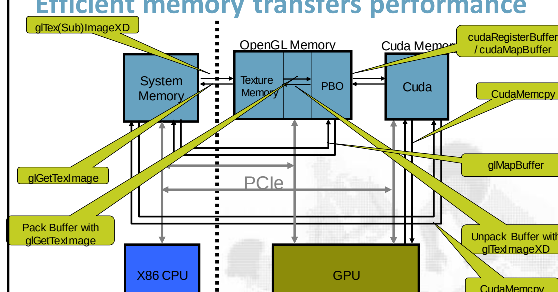
Efficient memory transfers performance



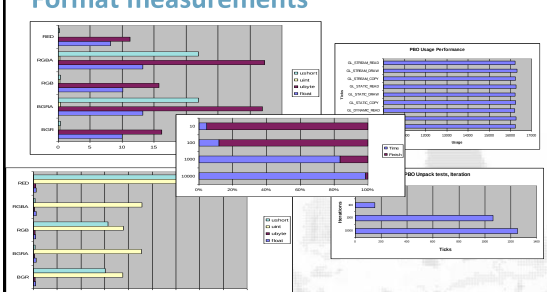
Efficient memory transfers performance



Efficient memory transfers performance



Format measurements



Example NBody (Gpu computing SDK)

Ref [5]

Conclusion

- Gpus are getting more versatile
  - More algorithms can run on the Gpu
- Gpus still excel at 3d rendering and video processing
  - Applications can combine Cuda kernels and render capabilities
- Gpu Computing requires
  - hardware knowledge
  - out-of-the-box thinking

References

1. Adaptive dimming and adaptive boosting backlight technologies for LCD-TV systems *by Hendriek Groot Hulze and Pierre de Greef* http://www.eetimes.com/design/power-management-design/4013430/Adaptive-dimming-and-adaptive-boosting-backlight-technologies-for-LCD-TV-systems
2. LCD backlighting using 2D LED dimming boosts TV performance and saves power *by Pierre de Greef, Hendriek Groot Hulze and Harm van der Heijden* http://www.eetimes.com/design/power-management-design/4013461/LCD-backlighting-using-2-D-LED-dimming-boosts-TV-performance-and-saves-power
3. Histogram calculation in CUDA *by Victor Podlozhnyuk* http://developer.download.nvidia.com/compute/cuda/1_1/Website/projects/histogram256/doc/histogram.pdf
4. Histogram computation with Cuda *by* Dr. *Ramtin Shams* http://users.cecs.anu.edu.au/~ramtin/cuda.htm
5. Cuda Zone: http://www.nvidia.com/object/cuda_home_new.html

Thanks for your time!

www.task24.nl