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

Other examples
- 3D Images
- Natural Motion
- ...

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

Histogram
- ~2,000,000 pixels
- 256 "buckets"
Example dimming

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

Histogram analysis

- Basic Algorithm
  - #bins can vary
  
```c
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

Final solution (without atomics)

```c
#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 [3, 4]

SHADERS VS CUDA
Graphics Cards APIs

- **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

GPU Fundamentals: The Graphics Pipeline

- **Application**
- **Transform**
- **Rasterizer**
- **Shade**
- **Video Memory (Textures)**

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

CG Tutorial – shader code

```
struct Fragment
{
  float4 color : OSMAG; // oesrgb color
};

Fragment ModelExample
{
  float3 normal = normalize(position - worldPosition); // input normal
  float3 light = normalize(direction); // input light
  float3 view = normalize(-worldPosition); // input view
  float3 color = 0.1f * normal * light * view;
  return color;
};
```

Pixel Operations have to be coded by hand.

Loop over every pixel the kernel is supposed to process:

```
// Process global threadIdx.x on the global thread %blockIdx.x %blockId.x

// Process every pixel component with f

// Compute final pixel color in destination.x = videoFrame.y[iPixel] + u_iPixel == u_num_threads;

// Compute pixel position

uint uPixel = blockIdx.x * blockDim.x + threadIdx.x;
uint* uInputLocation = (uint*) &videoFrame.y[0];
uint* uOutputLocation = (uint*) &videoFrame.x[0];
uint* uOutputLocation2D = (uint*) &videoFrame.x[0];
```

1 September 2010
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

Example NBody (Gpu computing SDK)

Format measurements

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

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

3. Histogram calculation in CUDA by Victor Podlozhnyuk

4. Histogram computation with Cuda by Dr. Ramtin Shams


Thanks for your time!

www.task24.nl