



The GPGPU Programming Model

John Owens

University of California, Davis



Overview

- **Data-parallel programming basics**
- **The GPU as a data-parallel computer**
- **“Hello World” GPGPU Example**

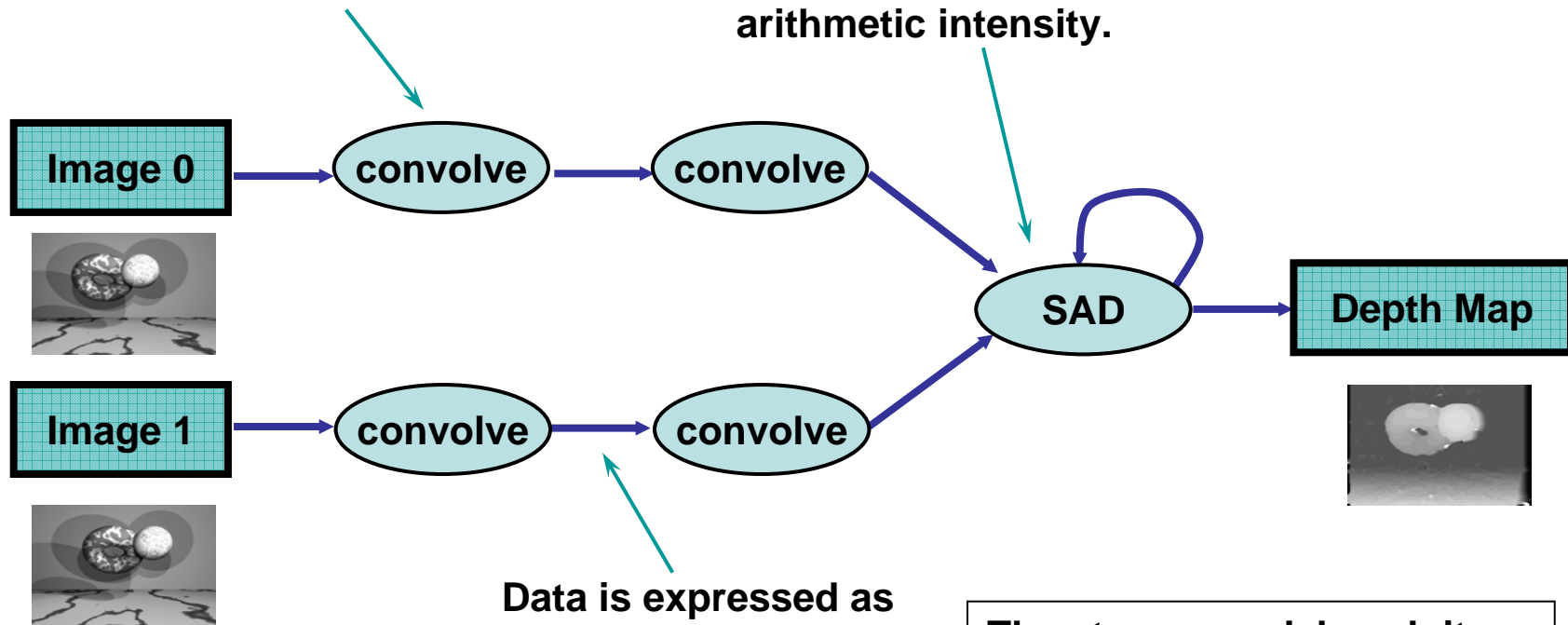
Data-Parallel Programming Basics

- **What is a data-parallel program?**
 - Exposes parallelism
 - Explicitly expresses data dependencies
- **Stream programming is data-parallel model**
 - Stream programs are dependency graphs
 - Kernels are graph nodes
 - Streams are edges flowing between kernels

Stream Programming Basics

Computation is expressed as kernels (ovals).

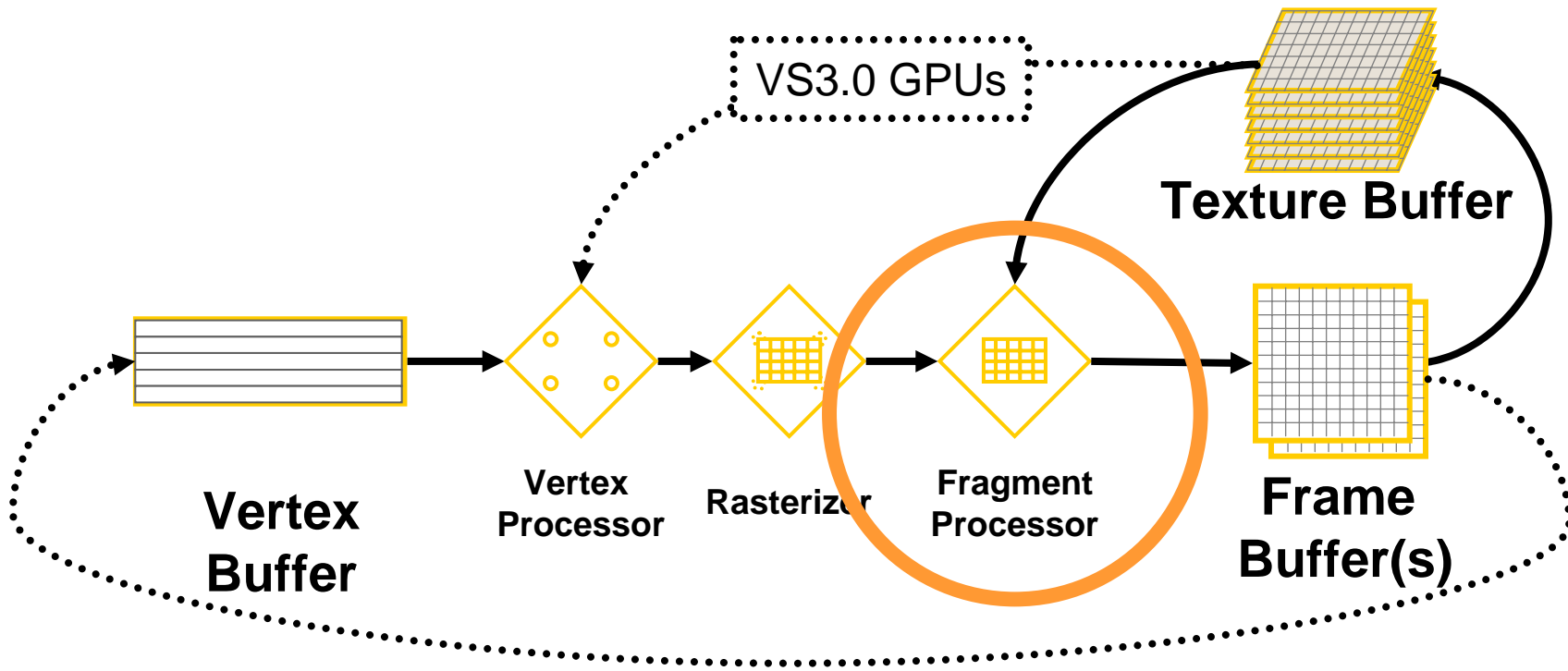
Kernels exploit both instruction (ILP) and data (SIMD) level parallelism. They should have simple control and high arithmetic intensity.



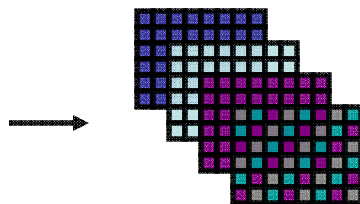
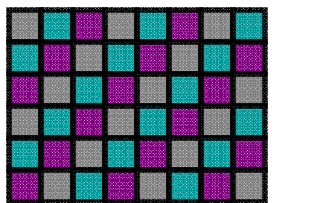
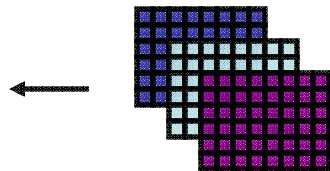
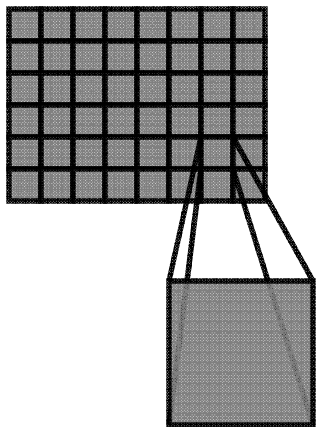
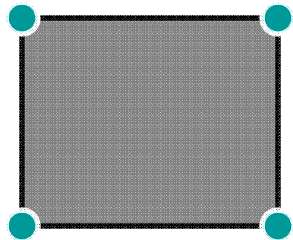
Data is expressed as streams (arrows).

The stream model exploits parallelism without the complexity of traditional parallel programming.

Modern Graphics Pipeline

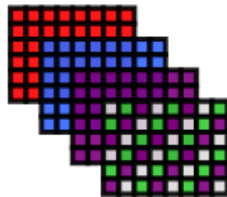
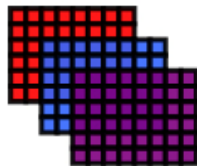
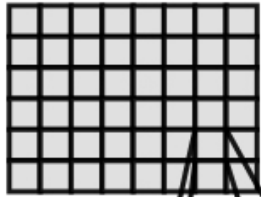
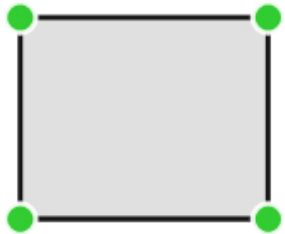


Programming a GPU for Graphics



- Application specifies geometry rasterized
- Each fragment is shaded w/ SIMD program
- Shading can use values from texture memory
- Image can be used as texture on future passes

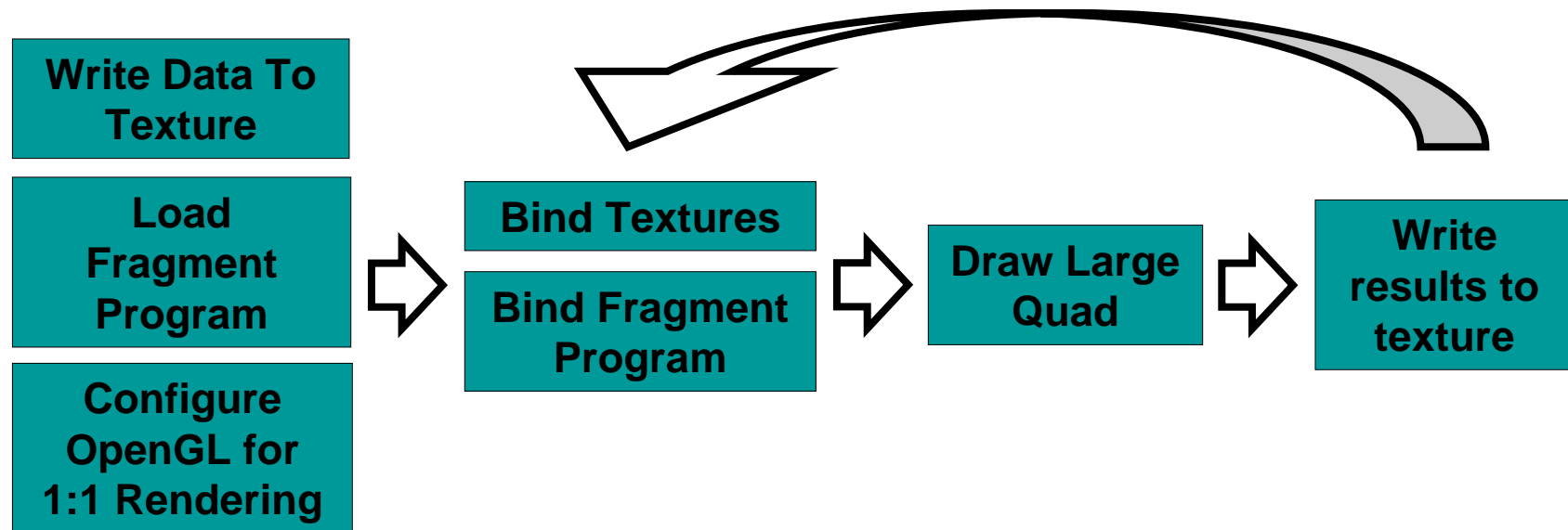
GPU as a Stream Processor



- Draw a screen-sized quad (creates stream of fragments)
- Run kernel (fragment program) over stream of fragments
- Read stream data from textures
- Write stream of results to frame buffer

GPU as a Stream Processor

- Streams → Textures
- Kernels → Fragment program
- `forEach` execution → Draw single large quad



“Hello World“ GPGPU Example

- **3 x 3 Image processing convolution**
 - Inputs: image, weights
 - Output: blurred image
- **CPU version**

```
image = loadImage( WIDTH, HEIGHT );
blurImage = allocZeros( WIDTH, HEIGHT );

for (x=0; x < WIDTH; x++)
    for (y=0; y < HEIGHT; y++)
        for (i=-1; i <= 1; i++)
            for (j=-1; j <= 1; j++)
                float w = computeWeight(i,j);
                blurImage[x][y] += w * image[x+i, y+j];
```

“Hello World“ GPGPU Example

- GPU Version

- 1) Load `image` into texture

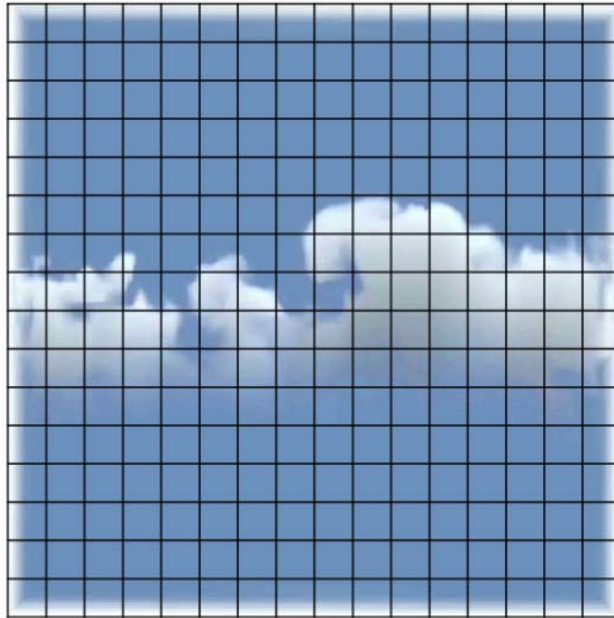


Figure courtesy of Mark Harris

- 2) Create `blurImage` texture to hold result

“Hello World“ GPGPU Example

- **GPU Version**

- 3) Load fragment program (kernel)

Example shown in Cg

```
float4 blurKernel( uniform samplerRECT image,
                  float2      winPos : WPOS,
                  out float4  blurImage )
{
    blurImage = float4(0,0,0,0);

    for (i=-1; i <= 1; i++) {
        for (j=-1; j <= 1; j++) {
            float2 texCoord = winPos + float2(i,j);
            float  w        = computeWeight(i,j);
            blurImage += w * texRECT( image, texCoord );
        }
    }
}
```

“Hello World“ GPGPU Example

- **GPU Version**

- 4) Configure OpenGL to draw 1:1
No projection or rescaling

```
glMatrixMode( GL_PROJECTION );  
glLoadIdentity();  
gluOrtho2D(0, 1, 0, 1);  
glViewport(0, 0, WIDTH, HEIGHT );  
glMatrixMode( GL_MODELVIEW );  
glLoadIdentity();
```

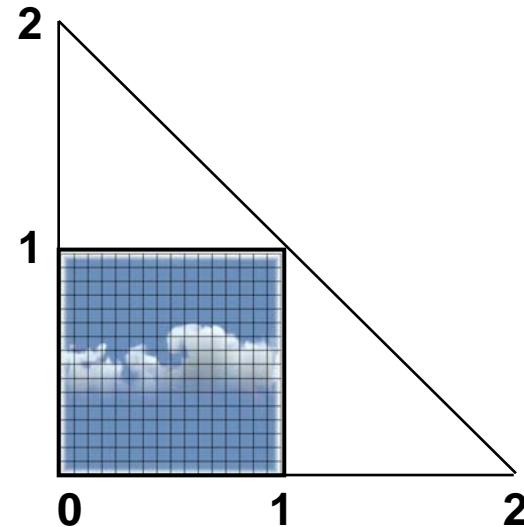
- 5) Bind `image` and `blurKernel` (texture and fragment program)
- 6) Bind `blurImage` as render target

“Hello World“ GPGPU Example

- GPU Version

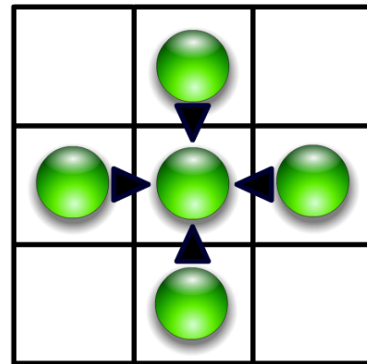
- 7) Execute kernel on each stream element
Draw quad of size [WIDTH x HEIGHT]

```
glBegin( GL_TRIANGLES );  
    glVertex2f(0, 0);  
    glVertex2f(2, 0);  
    glVertex2f(0, 2);  
glEnd();
```



“Hello World“ GPGPU Example

- **What happened?**
 - `blurKernel` executed on each element of `image`
 - Rendering replaced outer two loops of CPU version
 - `blurKernel` performed *gather* operation at each element



Gather

- Result (`blurImage`) was written to framebuffer / texture

“Hello World“ GPGPU Example

- **Get the source code for GPGPU examples**
 - <http://www.gpgpu.org/developer/>
 - http://download.nvidia.com/developer/SDK/Individual_Samples/samples.html
 - gpgpu_fluid
 - gpgpu_disease
 - gpu_particles
 - http://www.ati.com/developer/sdk/RadeonSDK/Html/Samples/OpenGL/HW_Image_Processing.html

Questions?

- GPU as stream processor?
- Stream programming model?
- “Hello World” example?

References

- ATI Developer web site, <http://www.ati.com/developer/>
- GPGPU Developer web site, <http://www.gpgpu.org/developer>
- N. Goodnight, C. Woolley, G. Lewin, D. Luebke, G. Humphreys, “A Multigrid Solver for Boundary Value Problems Using Programmable Graphics Hardware,” Graphics Hardware 2003, pp. 102–111
- M. Harris, W. Baxter, T. Scheuermann, A. Lastra, “Simulation of Cloud Dynamics on Graphics Hardware,” Graphics Hardware 2003, pp. 92–101
- W.D. Hillis and G. Steele Jr., “Data Parallel Algorithms,” Comm. ACM, 29(12), December 1986, pp. 1170–1183
- U. Kapasi, W. Dally, S. Rixner, P. Mattson, J. Owens, B. Khailany, “Efficient Conditional Operations for Data-parallel Architectures”. In Proc. of the 33rd Ann. Int'l Symp. on Microarchitecture (2000), pp. 159–170
- A. Lefohn, J. Kniss, C. Hansen, R. Whitaker, “A Streaming Narrow-Band Algorithm: Interactive Deformation and Visualization of Level Sets,” IEEE Transactions on Visualization and Computer Graphics, Jul./Aug. 2004 , pp. 422–433
- NVIDIA Developer web site, <http://developer.nvidia.com/page/home>