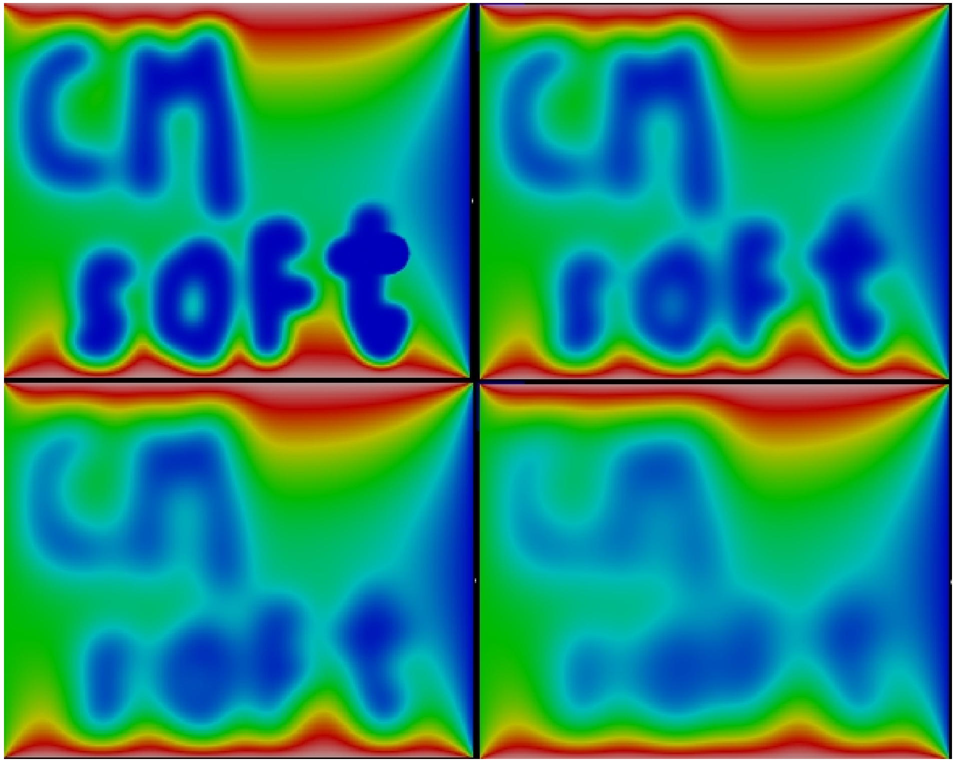
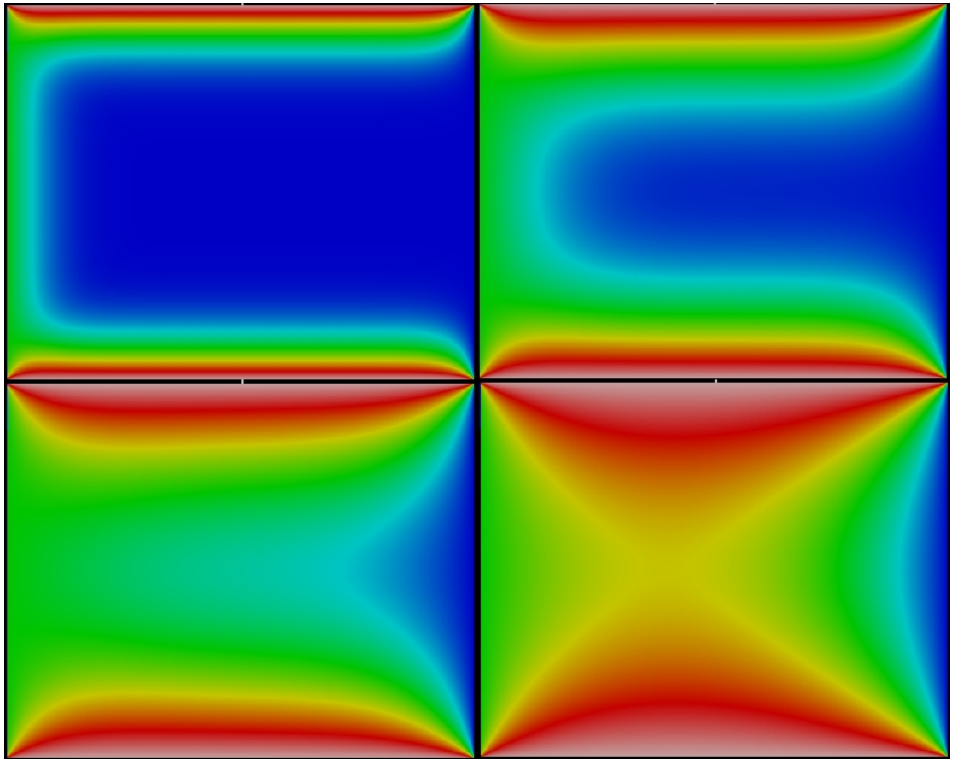
CMSoft Case Study: Fast Simulation of Heat Transfer using C# and OpenCL

Simulation of heat transfer in a solid plate with Dirichlet conditions via finite differences using GPU acceleration





Time evolution of temperature distribution in a plane plate with fixed temperatures along the edges and varying initial conditions

Douglas Coimbra de Andrade (douglas@cmsoft.com.br) **july/2012**

Demonstration

- Heat transfer simulation;
- Changing initial conditions using the mouse;
- Changing the heat conductivity on the fly;
- (Demonstration).

Index

- Prerequisites;
- Overview of heat conduction;
- Finite differences discretization;
- Sharing OpenCL/OpenGL objects;
- Computing color representation of intensities;
- Initial values and boundary conditions;
- Simulating the system;
- Final remarks.

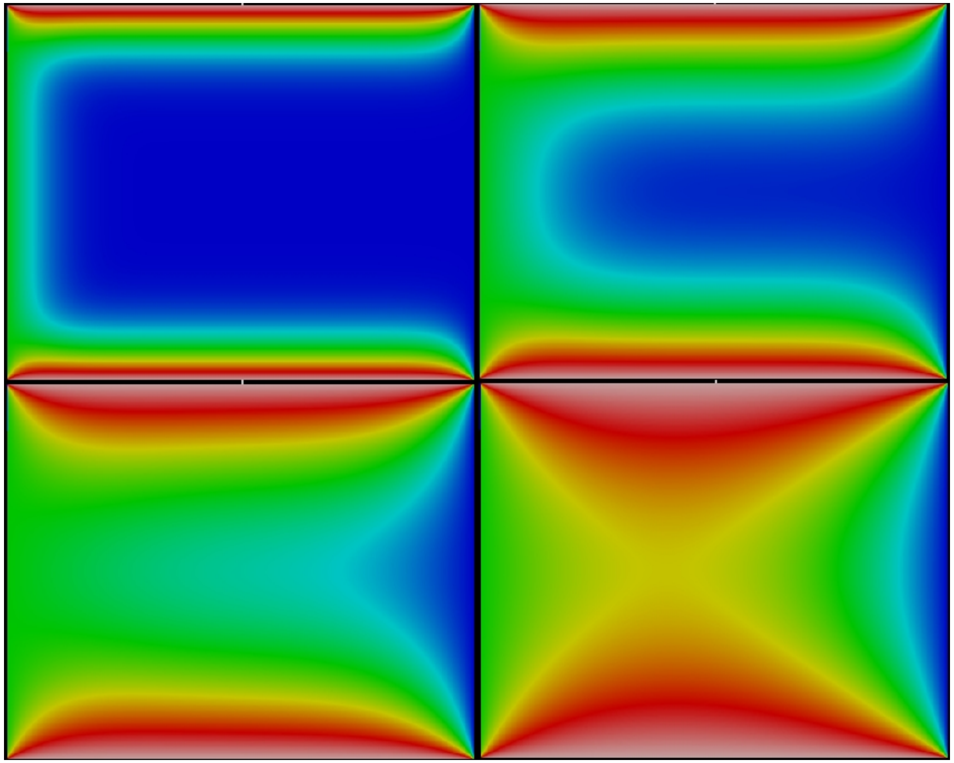
Prerequisites

- Working knowledge of C/C#/C++;
- Familiarity with OpenCL data-parallel algorithms and the notion of Host/Device code;
- Basic OpenGL vertex buffer objects;
- Basic calculus and partial differential equations;
- Sharing OpenCL and OpenGL objects.

- Basics at CMSoft OpenCL tutorial using C#, Cloo and OpenCLTemplate: http://www.cmsoft.com.br/index.php?option=com_content&view=category&layout=blog&id=41&Itemid=75

Overview of heat conduction

- Heat flow intuition: temperature difference, material properties;



- The heat equation:

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$

Finite differences discretization

- Reworking the heat equation:

$$\frac{\partial}{\partial t} u(t,x,y) = \alpha \cdot \nabla^2 u(t,x,y) = \alpha \cdot \left(\frac{\partial^2}{\partial x^2} u(t,x,y) + \frac{\partial^2}{\partial y^2} u(t,x,y) \right)$$

- Discrete approximation of time and space derivatives:

$$\frac{\partial}{\partial t} u(t,x,y) \cong \frac{u(t+\Delta t,x,y) - u(t,x,y)}{\Delta t}$$

$$\frac{\partial^2}{\partial^2 x} u(t,x,y) \cong \frac{u(t,x+\Delta x,y) - 2\,u(t,x,y) + u(t,x-\Delta x,y)}{\Delta x^2}$$

$$\frac{\partial^2}{\partial^2 y} u(t,x,y) \cong \frac{u(t,x,y+\Delta y) - 2\,u(t,x,y) + u(t,x,\mathrm{y}-\Delta \mathrm{y})}{\Delta y^2}$$

- Final equation (which we can simulate):

$$u(t+\Delta t,x,y) = u(t,x,y) + \Delta t \cdot \alpha \cdot \left(\frac{\partial^2}{\partial x^2} u(t,x,y) + \frac{\partial^2}{\partial y^2} u(t,x,y) \right)$$

Finite differences discretization

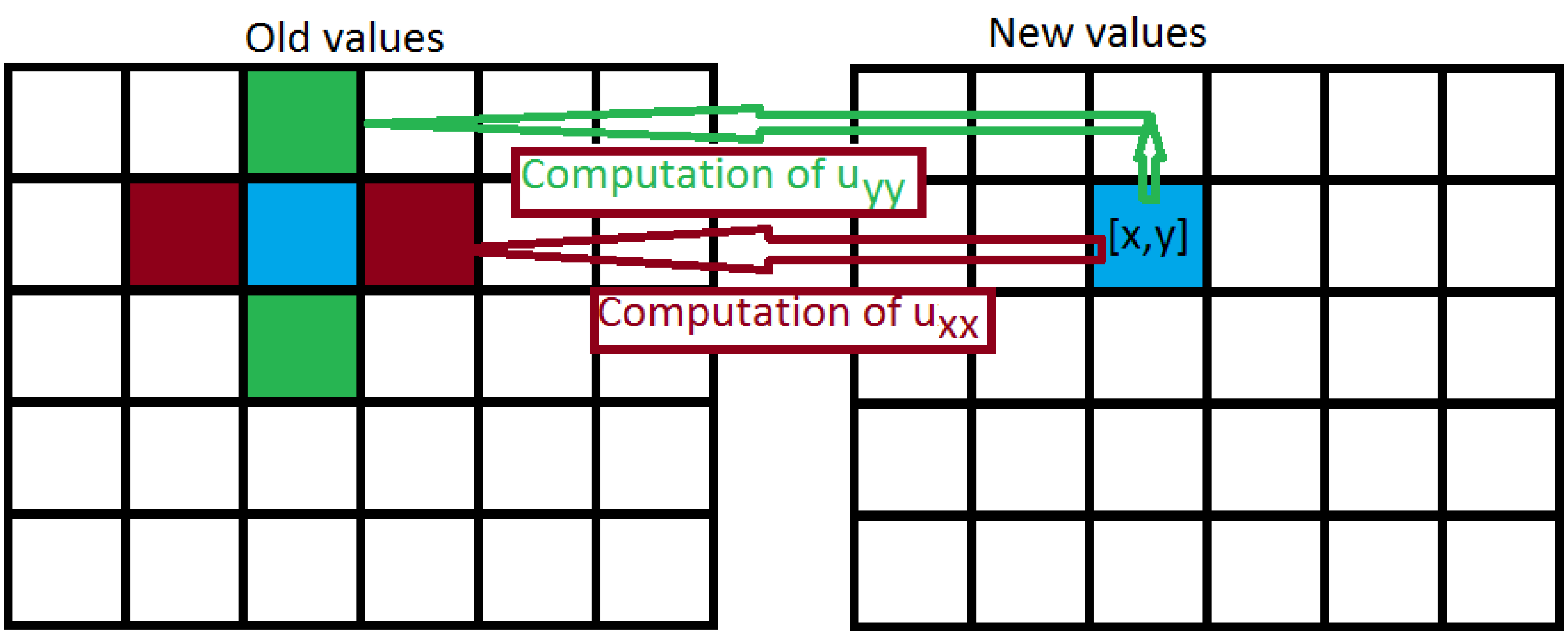
- Stepping forward in time:

$$u(t+\Delta t,x,y) = u(t,x,y) + \Delta t \cdot \alpha \cdot \left(\frac{\partial^2}{\partial x^2} u(t,x,y) + \frac{\partial^2}{\partial y^2} u(t,x,y) \right)$$

$$\frac{\partial^2}{\partial^2 x} u(t,x,y) \cong \frac{u(t,x+\Delta x,y) - 2\,u(t,x,y) + u(t,x-\Delta x,y)}{\Delta x^2}$$

$$\frac{\partial^2}{\partial^2 y} u(t,x,y) \cong \frac{u(t,x,y+\Delta y) - 2\,u(t,x,y) + u(t,x,y-\Delta y)}{\Delta y^2}$$

(demonstration)



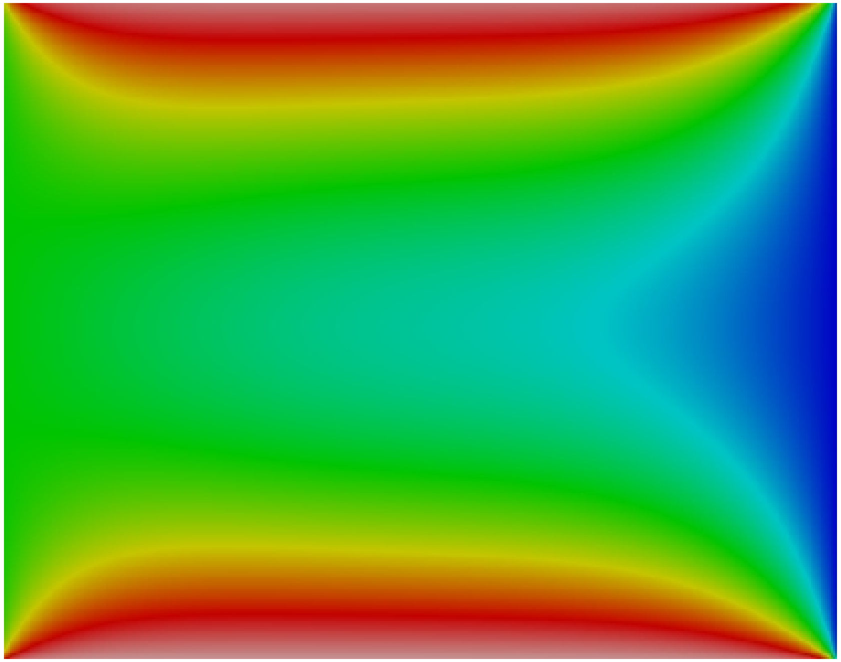
Sharing OpenCL/OpenGL objects

- Easy using OpenCLTemplate: bind texture to 3D model and retrieve OpenCL image2d;
- <C# code>

Color representation of intensities

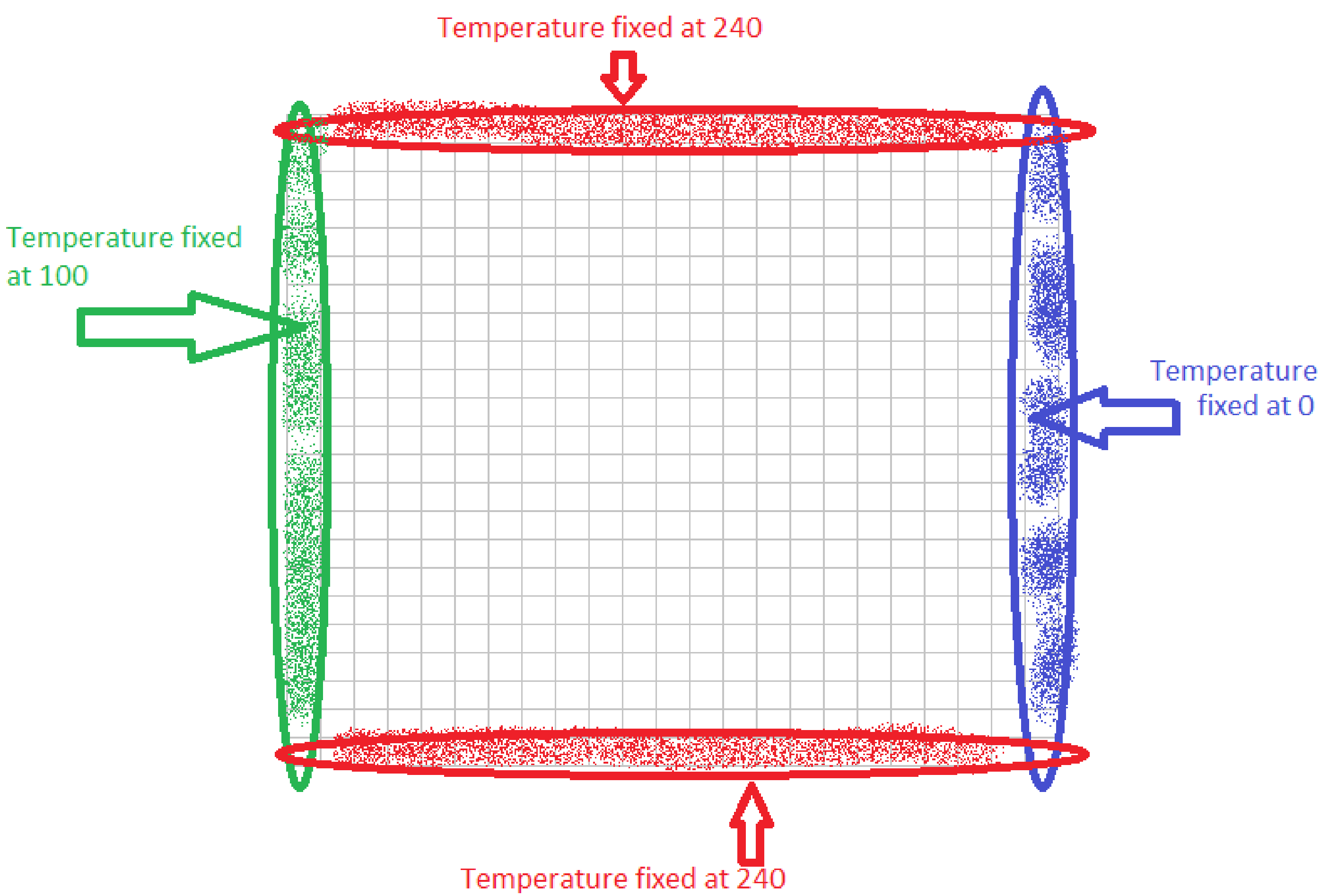
- Need an intuitive graphical representation of temperatures (hot/cold);
- OpenCL code:

```
float4 getColor(float hMax, float hMin, float t)
{
    t = clamp(t, hMin, hMax);

    float invMaxH = 1.0f / (hMax - hMin);
    float zRel = (t - hMin) * invMaxH;

    float cR = 0, cG = 0, cB = 0;


    if (0 <= zRel && zRel < 0.2f)
    {
        cB = 1.0f;
        cG = zRel * 5.0f;
    }
    else if (0.2f <= zRel && zRel < 0.4f)
    {
        cG = 1.0f;
        cB = 1.0f - (zRel - 0.2f) * 5.0f;
    }
    else if (0.4f <= zRel && zRel < 0.6f)
    {
        cG = 1.0f;
        cR = (zRel - 0.4f) * 5.0f;
    }
    else if (0.6f <= zRel && zRel < 0.8f)
    {
        cR = 1.0f;
        cG = 1.0f - (zRel - 0.6f) * 5.0f;
    }
    else
    {
        cR = 1.0f;
        cG = (zRel - 0.8f) * 5.0f;
        cB = cG;
    }

    return (float4)(cR, cG, cB, 1.0f);
}
```



Initial values and boundary conditions

- Need initial values;
- Can't compute off the bounds of the matrix.



Simulating the system

- CLHeatTransfer.cs:
- Initialization;
- Step phase;
- <C# and OpenCL code analysis>

Final remarks

- Other implementation details: see source code;
- Easy to add, for example, a circle of fixed temperature or heat sources;
- Easy to extrapolate to wave equation;
- High performance: 7000 steps per second on a reasonable 500x400 grid;
- Demonstration of how OpenCL GPU computing coupled with OpenCL object sharing can be a powerful visualization tool.

References

- [1] Heat equation. http://en.wikipedia.org/wiki/Heat_equation retrieved 09 jul 2012
- [2] Finite difference. http://en.wikipedia.org/wiki/Finite_difference retrieved 09 jul 2012
- [3] OpenCL OpenGL interoperation. http://www.cmsoft.com.br/index.php?option=com_content&view=category&layout=blog&id=99&Itemid=150 retrieved 09 jul 2012
- [4] OpenCLTemplate OpenCL/GL Interop Framework http://www.cmsoft.com.br/index.php?option=com_content&view=category&layout=blog&id=137&Itemid=197 retrieved 09 jul 2012
- [5] Partial Differential Equations. http://www.math.odu.edu/~jhh/chs9.pdf retrieved 09 jul 2012