# Fire Simulator and Fractals: using a visualization library to introduce CUDA

**Tia Newhall, Andrew Danner**
**Computer Science Department**
**Swarthmore College**

We present two assignments that introduce CUDA [5] programming to upper-level undergraduate students. The assignments use an OpenGL 4.x [6] library we developed for visualizing computation on the GPU. The library allows students to see their CUDA programs execute on the GPU, which aids in their debugging of their first CUDA programs. Students also enjoy these visual assignments and they often add interesting extensions to the required parts.

Our OpenGL library uses ideas from the "CUDA By Example" [2] text and transparently allocates a display grid on the GPU that allows students to color individual pixels via a CUDA kernel call. CUDA programs that use this flexible library may allocate additional GPU buffers to store non-color data. Students then write one or more CUDA kernels to update their data and the display grid. The library then repeatedly calls these kernels in an animation loop. The library is designed to be simple enough for students to use knowing only the basics of CUDA, and no knowledge of OpenGL is required.

**Fire Simulator Program:** The Fire Simulator program is our parallel version of a discrete event simulation [8] of a two-dimensional grid world consisting of forests and lakes. Each grid cell is classified as water, forest, fire, or burnt. Forest cells transition to fire cells based on some probability if one or more of their neighbors are on fire. Fire cells transition to burnt cells after some number of time steps. Each step of the computation computes in parallel the values for all cells for the next time step and updates the display grid. The program uses random numbers in CUDA to determine if a cell catches fire. Command line arguments specify parameters of the world and fire simulation including the burn rate and the probability a cell catches fire.

The primary TCPP [7] concepts covered in this application are: Stream-GPU architectures; gpgpu computing; synchronization; heterogeneous systems; SPMD; and memory management. The assignment is used in our Parallel and Distributed Computing course [3], an upper-level undergraduate course. The main purpose of this lab is to introduce CUDA programming to students as one of several programming models that they could use in their independent course projects. Other labs in the course introduce client-server socket programming, pthreads, MPI, and OpenMP.

As an introduction to CUDA, we find that this assignment works very well. In particular, the visualization helps students to easily see bugs in their CUDA grid-block-thread mappings of the CUDA kernels that they write. We have also had students implement impressive extensions that simulate more realistic burning functions, create interesting starting worlds, and visualize the fire simulation in more realistic ways. This assignment could be easily modified for any type of discrete event simulation application. This assignment often results in many students wanting to use CUDA in their main course project. The main weakness of this project is that it requires using cuRAND library. However, we give them almost all of the cuRAND code that they need to use with the assignment starting point code. The other main weakness is that it requires an OpenGL visualization library [4], though we hide all the OpenGL details in our library implementation.

**Fractal Generation:** Though our library does not assume familiarity with OpenGL and abstracts away all the OpenGL details, students familiar with OpenGL can still use our library to explore CUDA and connect TCPP topics back to core computer graphics concepts. Internally, the display grid in our library is both a CUDA GPU buffer and an OpenGL texture map. In our Computer Graphics course [1], we have students develop multiple kernels to compute and render 2D Julia sets [4], similar to examples of Sanders and Kandrot [2]. Our library also includes an abstract timing class so students can easily compare performance of CPU only approaches and multiple CUDA kernels for generating images. Instead of seeing a static 2D image, graphics students enjoyed seeing that their CUDA image was indeed a texture map and could be applied to arbitrary shapes including animated spheres and cubes.

This assignment covers the same TCPP topics as the fire simulation assignment, but additionally makes connections between CUDA and more traditional OpenGL programming.

**Assignments and Code:** More information about these assignments, including example assignment write-ups and full starting point code for both assignments is available online [4].

## References

[1] Andrew Danner. CS40: Computer Graphics. https://www.cs.swarthmore.edu/~adanner/cs40/, 2016.

[2] Edward Kandrot Jason Sanders. CUDA by Example. Addison Wesley, 2010.

[3] Tia Newhall. CS87: Parallel and Distributed Computing. https://www.cs.swarthmore.edu/~newhall/cs87/, 2018.

[4] Tia Newhall and Andrew Danner. CUDA Fire Simulation and Fractals: Assignments and Resources. https://www.cs.swarthmore.edu/edupar18, 2018.

[5] NVIDIA. NVIDIA CUDA Compute Unified Device Architecture. https://developer.nvidia.com/about-cuda, 2018.

[6] opengl. OpenGL Overview. http://www.opengl.org/about, 2018.

[7] Prasad S. et al. NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing - core topics for undergraduates. http://www.cs.gsu.edu/~tcpp/curriculum/, 2012.

[8] Angela B. Shiflet. Spreading of Fire. http://nifty.stanford.edu/2007/shiflet-fire/, 2007.