CSC-213: PROF. CURTSINGER
FINAL PROJECT REPORT

ANNA BLINDERMAN, DAVID KRAEMER, ZACHARY SEGALL

Project Overview (<400 words)

- introduce Game of Life
- describe GUI
- describe GPU stencil update
- describe listener CPU threads
- describe evaluation strategies
- summarize evaluation results

We implemented a variation of Conway's Game of Life – a cellular automaton simulation created by the mathematician John Conway. Life is simply a series of iterations of a rendering of a grid of cells, all of which are either dead or alive. Each cell's state is determined by the states of its direct neighbors.

[david can you please write the paragraph on the ui don't you read through libraries in your spare time or something]

For each iteration, we use Conway's algorithm to determine the state (dead or alive) of every cell in our grid. Because this algorithm takes as input a cell along with the states of each of its eight immediate neighbors, this algorithm naturally lends itself to an "embarrasingly parallel" stencil computation on the GPU.

Design and Implementation (∼2 pages)

- summary of overall implementation structure
- Component 1: GUI
  - responsibilities/how fits into entire structure
  - rationale for choice
  - data structures/algorithms/libraries/other details
- Component 2: mouse/keyboard input
  - responsibilities/how fits into entire structure
  - rationale for choice
  - data structures/algorithms/libraries/other details

- Component 3: GPU stencil update
  - responsibilities/how fits into entire structure
  - rationale for choice
  - data structures/algorithms/libraries/other details
- Butler Lampson's Hints for Computer System Design (integrate?)
  - don't reinvent the wheel (+ how it did/didn't help)
  - be prepared to throw an entire thing out (+ how it did/didn't help)
- figures if appropriate

Our scheduler will need to handle the order and timing of the execution of various events integral to our simulation. For some examples, we must update the cells on the board in order to advance the simulation to the next iteration at regular intervals, add cells to the board in locations specified by user input, and pause, restart, or quit the simulation whenever the user inputs the appropriate keystroke. We will model our scheduler off of the one implemented by Zachary and Mattori for the Worm lab.

The algorithm used to advance a Life simulation from one iteration to the next naturally lends itself to an "embarrassingly parallel" implementation. Specifically, we will use a stencil pattern since a cell's next state depends on the states of its eight immediate neighbors and we can process whether or not a cell survives independent of this determination for any other cell on this board.

Regarding the more general mechanics of our project, we will use an `SDL_surface` to create a board as in the Galaxy lab. The cells on the board will be initially populated from either a random starting configuration, one of some set of preset starting configurations, or by the user clicking squares on the board. We will have a struct to represent the state of the board and will use a bitmap for the visualization of the board. Because our internal representation will be separate from our visualization, we will need to translate from the internal representation (the struct) to the visualization (the bitmap) at each interval of time. Further, this separation will allow us to more easily manipulate and update the representation and then simply update the visualization afterwards using Conway's original algorithm.

Evaluation

- describe experimental setup
  - hardware
  - software (libraries)
  - data-gathering methods
- explain what trying to measure (speedup)

  - vary block size
  - implement thing that only updates regions with living things?
  - implement serial version?
- include at least one graph with at least 8 data points
- discuss interpretation of results