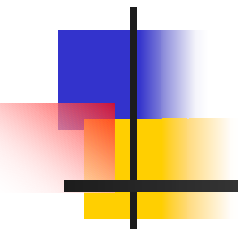
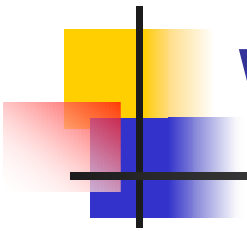


Cellular Automata as "Hello, World"

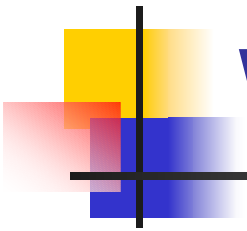
Dan Bentley

Ben Serebrin



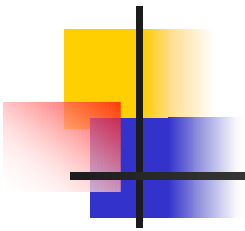
We're going to talk about:

- Why Conway's Game of Life is useful
- Language/Design Proposals
- Prose Composition (aka: Why it's not as easy as it looks to be an English major)
- Where we're going



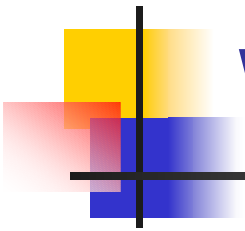
Why Life is Useful

- Introductory Program ("Hello, World") for new programmers
- Introductory Program for new systems
- Useful in this case for both roles
  - As we'll go into



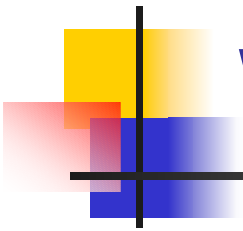
Bugs I found in Brook

- Copying
  - Copy-in, copy-out semantics
  - streamCopy
- Stencil-ing
  - 1-D
- Commenting



What I'd like to do:

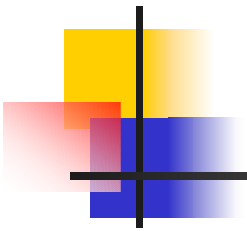
```
while(generations--) {
        streamShape(board, 2, size, size);
        streamStencil(boardp, board, STREAM_STENCIL_CLAMP,
                              2, -1, 1, -1, 1);
        Generation(board, boardp);
}
```



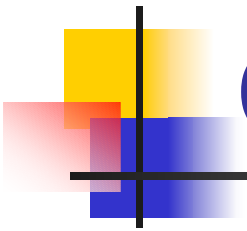
What I had to do:

```
while(generations--) {
        streamShape(board, 2, size, size);
        streamStencil(boardp, board, STREAM_STENCIL_CLAMP,
                              2, -1, 1, -1, 1);
        Generation(boardTemp, boardp);
        myCopy(board, boardTemp); //streamCopy
}
kernel void myCopy(out cell_s s1, cell_s s2) { s1 = s2; }
```

1-D Stenciling

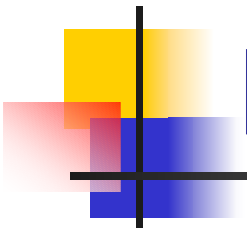


- Seems to not work
  - 80% sure that this isn't my error
- Get "random" values…
- 0, 16 and 134612816 popular
- Calling streamShape changes the values, but not to anything definite



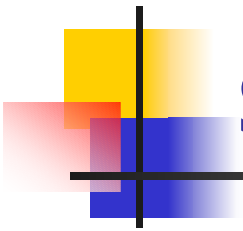
Commenting

- Meta-compiler doesn't like certain trigrams involving comments, nasty compile errors at C++ level
- }*/ and {*/ seem to be culprits
- Spaces sometimes work
- // also has its demons



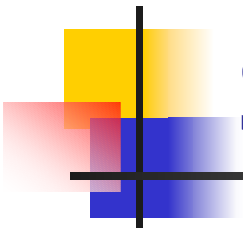
Language/Design Proposals

- stdout/in as FileStream
- Sequential kernels
- Stream Programs within kernels



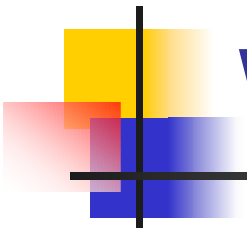
stdin/out as FileStreams

- Allow ability to create FileStreams from FILE*, not just filename.
- Trivial, but useful and important. Why?
- If we stray from C philosophy here, and stray from C philosophy there, then eventually we'll have enough un-orthoganality so people's conceptions don't hold



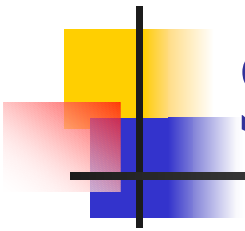
Sequential Kernels

- Example: Print out stream, newline at every n^{th} element
- Proposal: Use static keyword in kernels. Assures programmer of sequential semantics
- Why this isn't as harmful as it seems:



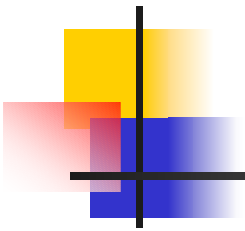
Why this isn't that bad:

- Used for book-keeping kernels (not much time)
- Allows programmers to not dump streams to memory, get back, etc.
- Only semantics guaranteed, may still be optimized



Stream Programs within

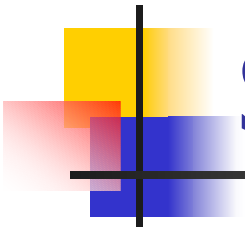
- Allow full stream programs within kernels, and all operations WITH a certain keyword, e.g. slow
- Allows kernels on different levels (TLP/DLP/ILP)
- Great Kernels have little kernels upon their backs to bite ‘em, and little kernels have lesser kernels, and so on ad infinitum



Example

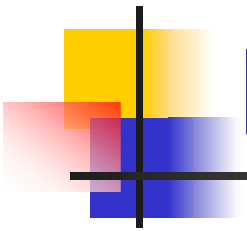
- kernel void Generation(out cell_s newBoard, cell_s cell) {

```
    int tally = 0;
    SumNeighbors(cell.neighbors, &tally);
    newBoard = computeNext(cell, tally);
}
```



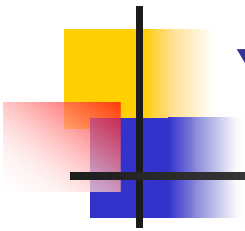
Stream Programs Within

- Also, circuit decomposition (from hardware design)
- Inter-procedural analysis
- Draw picture on board for this one



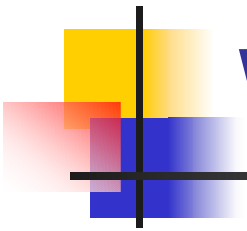
Prose Composition

- Problem: Consistent Nomenclature
- Goal: Lay out a clear, concise and consistent lexicon for the Stream community (at least in Stanford)
- Example: Stream Programming (the whole area) vs. Stream Functions (as opposed to kernel functions)



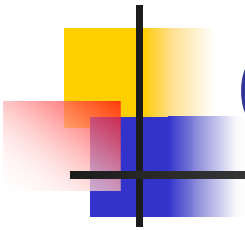
Your input

- What should mean what?
- Is there a better name for Stream Functions?
- My suggestion: Have a different name for the StreamC part of our duality
- Also: Should my intro focus on Stream Programming, or on Brook as an instance of Streaming Computation?



Where we're going

- Considering Irregular Grids more
- Writing more test-like programs for multiple dimensions?
- Other corner cases?



Conclusion

- Having a simple program makes it easy to diagnose more complex problems, by reducing the problem domain