# Cell Powered Games

Jim Tilander
Lucasarts

## Takeaways

- Some insights into CELL games development.
- Are new languages the answer?

## Exciting times

- Increased processing power
- Multicore is now standard
- Memory systems slowly catching up

## CELL

- Main memory bandwidth (20GB/s)
- Programmable LS
- 6 cycle LS
- Unified vector registers

## Movies

- Simulation is key
  - Complex motion
- Games have been the kid brother
- Scaled simulations can be transitioned to games
- Catching up (unification)

## Simulations right now in games

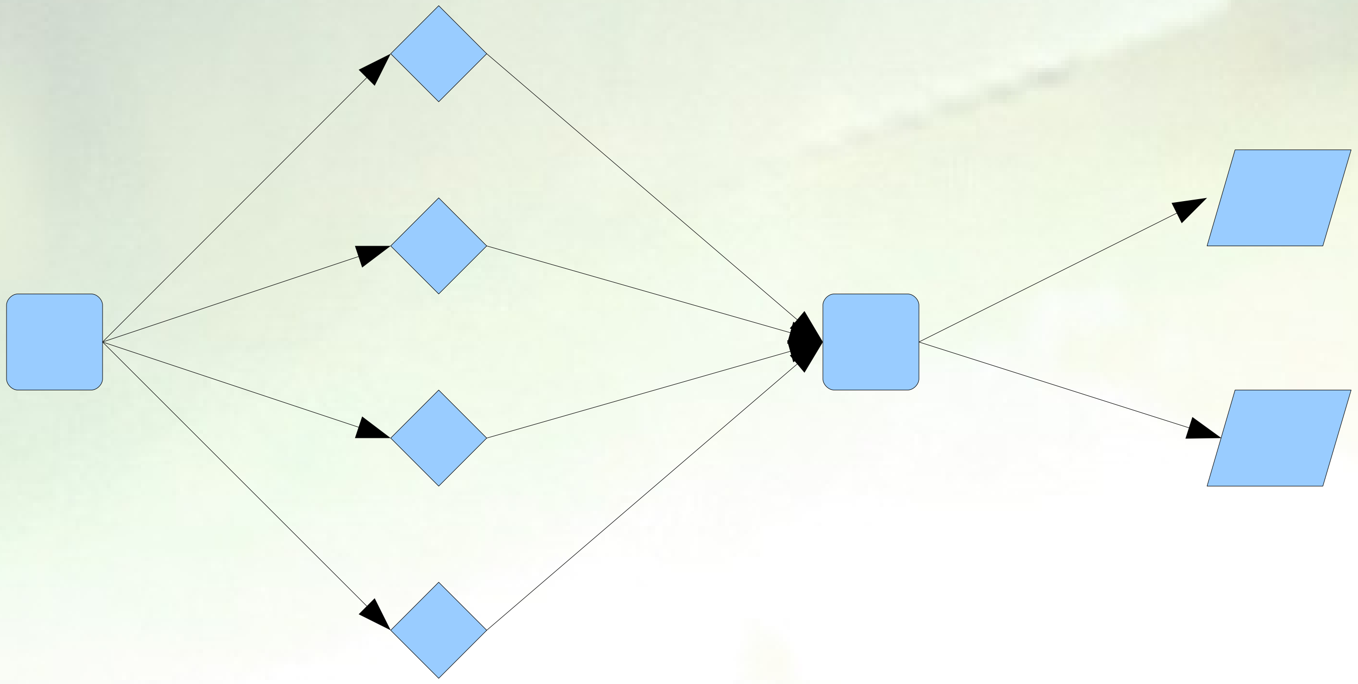
- Cloth
- Rigid body physics
- Soft body physics (FEM)
- Fluids and gas
- Hair
- Motion driven animation
- Artificial Intelligence

## Current strategies

- Large legacy non CELL codebases
- Move system by system to take advantage of the CELL.
- Time consuming
- Error prone
- Large changes often necessary to ... straighten out the data structures.

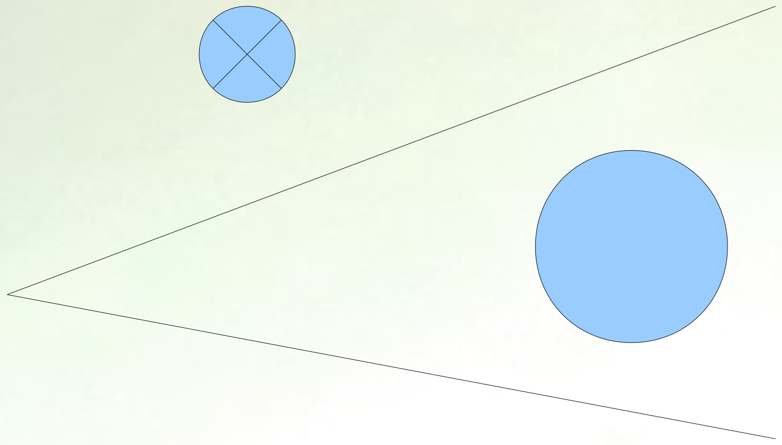
# Current strategies cont.

- Go wide
- Schedule individual components
- Poor intra component parallelism.



# Rethinking the algorithm

- Example: visibility culling
- Spatial partitioning, make sure it's a win
- Consider the worst case scenario
- Perhaps faster to just compute all items regardless



## Taking a look at the code...

- C's motto “a foot and a gun for everyone”
- C++ motto “an object and a shotgun for everyone”
- A lot of abstract code written in a low level language.

## Code size

- When I say large, what do I mean?
- 15y ago: ~20 KLOC (1 floppy, 880kb)
- 3y ago: ~1 MLOC
- Today: 3+ MLOC

## C++ code weaknesses

- No deep analysis of code (even alias doesn't work)
- Rooted in C, foot + gun
- Easy to write race conditions and deadlocks
- I don't want to be friends with “eieio”
- Schrödinger's cat bugs are nasty

## Is there help?

- Most problems stem from C++
- Why are we using it? (turns out C)
- C is just portable asm.
- Can we change the abstraction level?
- Can we change the language?

## Extending the C++ language?

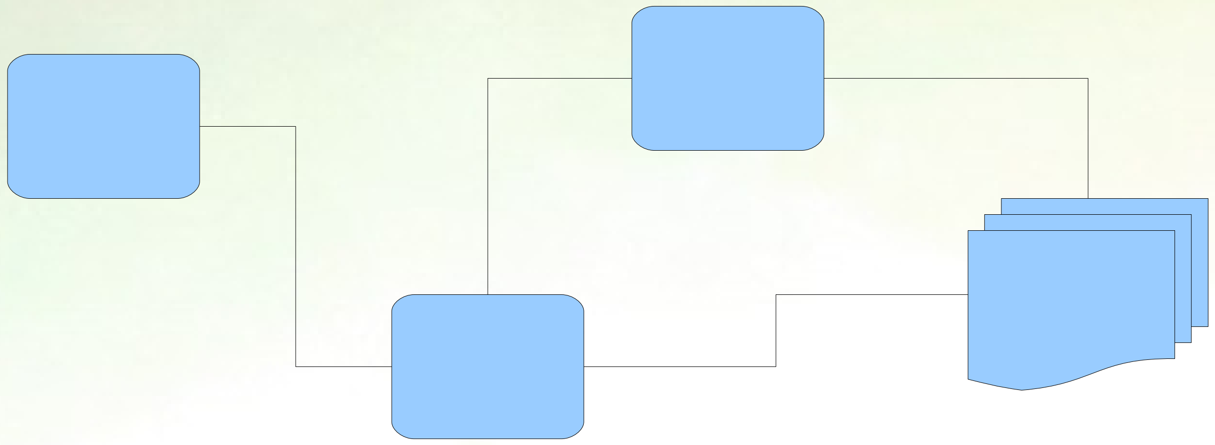
- Herb Sutter had the concur project (new keywords to C++)
- OpenMP
- Codeplay's compiler + runtime

## Moving up the alphabet

- L as in Lisp. Functional languages.
- Functional constructs like map/filter/reduce
- Control side effects
- ML only used for ML compilers

# Data flow languages

- Easy to understand
- Wires are data flows, nodes are procedures.
- Maps well onto the SPUs
- An old concept (1966)
- (In)famous product is NI's LabVIEW (G, 1986).



## Visual Basic for CELL?

- Visual interface to a dataflow language.
- Targeting non CELL programmers.
- SPU constraints can be build into the language.
- Visual debugging and single stepping.
- Race condition analysis, deadlock analysis

## Why not switch?

- Writing a top notch compiler is hard!
- Writing top notch debuggers/profilers hard!
- Finding people with the right skills is hard!
- Writing C/asm is cool.

## In closing...

- C++, C and asm will not go away tomorrow.
- New languages might open up a way for faster, more reliable code on the SPUs which ultimately will result in cooler, more fun games!

## Thank you for listening!

My emails:

jim@tilander.org

jtilander@lucasarts.com