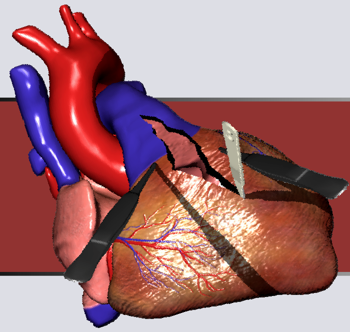


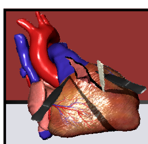
Implementation of a GPU Based Surgical Simulator for Open Heart Surgery

Thomas Sangild Sørensen

Associate Professor

Dept. Computer Science and Institute of Clinical Medicine

University of Aarhus, Denmark



On behalf of

Technical staff

- Thomas Sangild Sørensen
- Jesper Mosegaard
- Allan Rasmusson
- Dagur Ballisager
- Bo Carstensen

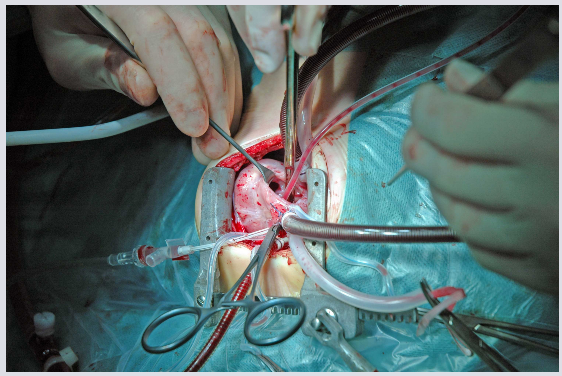
Clinical staff

- Ole Kromann Hansen
- Vibeke Hjortdal
- Gerald Greil
- Ludger Sieverding
- Conal Austin
- and others…

University of Aarhus, Denmark
University of Tübingen, Germany
King's College London School of Medicine, United Kingdom

Congenital heart disease

- Intracardiac surgery on infants and small children
- Complex individual morpholgy
- Surgical outcome will influence an entire life-time

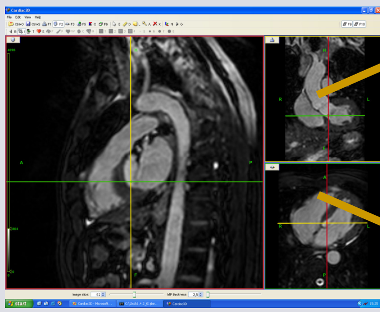


Preoperative planning essential

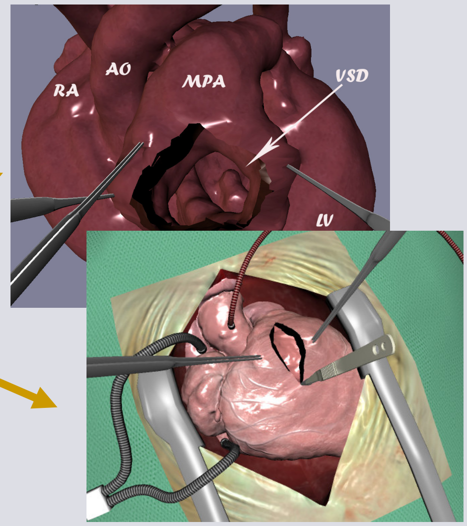
- Gold standard – 2D imaging
  - Echocardiography
  - Catheterization
  - MRI / CT
- But surgery is 3D
  - Can we use “virtual cardiotomy” to evaluate surgical strategies?

Virtual cardiotomy

Virtual cardiotomy



3D MRI



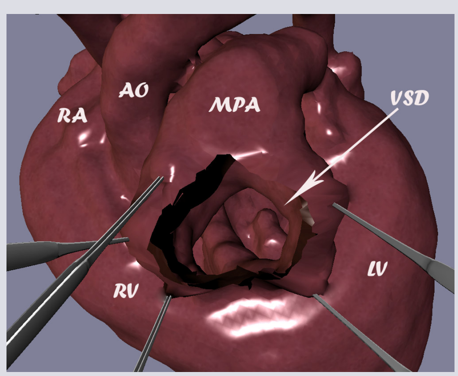
Training and education

Virtual cardiotomy

2 months old boy

Double outlet right ventricle
VSD / septum deviates to the right

Biventricular repair possible? Switch or intracardiac repair?



The movie

Sørensen et al. Circulation. 2007.

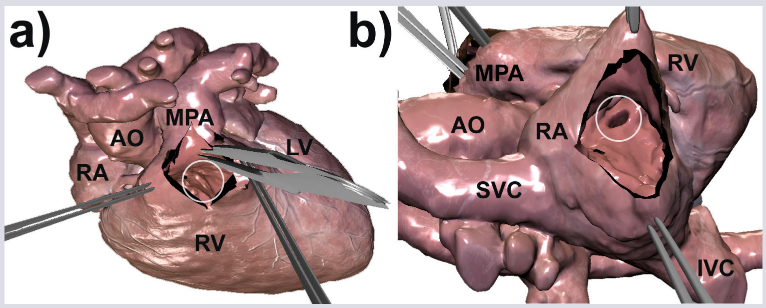
VSD closure

- Evaluation study in 40+ patients
    - One hour of segmentation needed

Work in review.

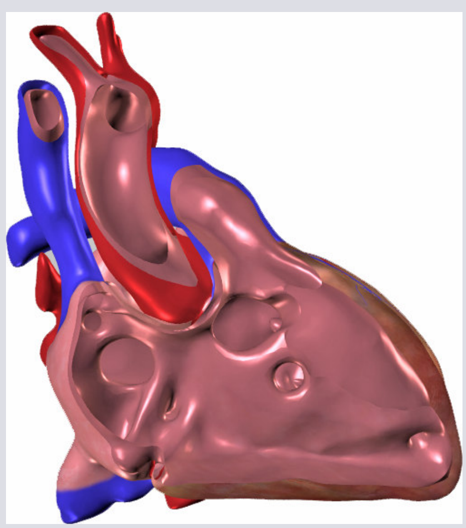
Teaching surgical scenarios

- 3D MRI of a volunteer
  - Septal defect (VSD, circle) added manually
- How can we best access the VSD?
  - Trans-ventricular or trans-atrial incision?



Sørensen et al. Interact Cardiovasc Thorac Surg. 2006.

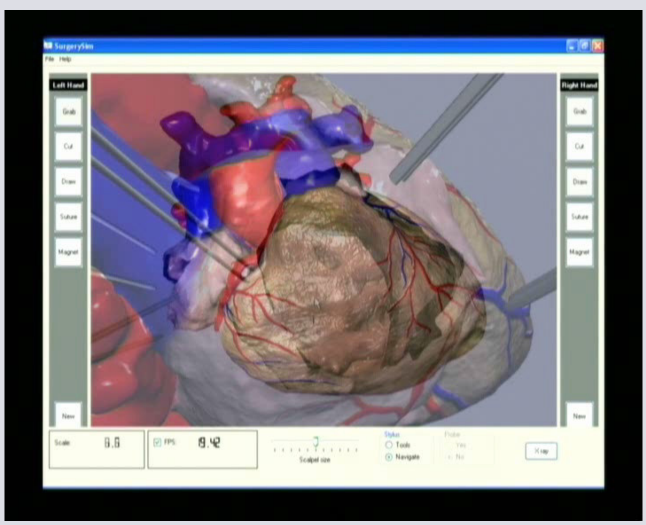
“Configurable” septal defects



Movie example

Sørensen & Mosegaard. ACM SIGGRAPH 2006.

An older model – playing around



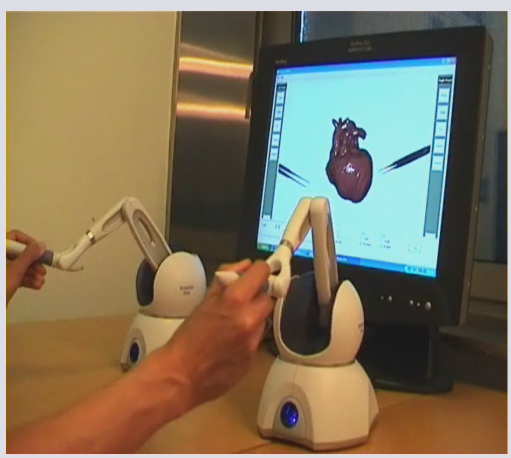
Challenge

- How to model and compute tissue elasticity sufficiently fast in a very complex organ such as the heart?
  - Real-time interaction is <u>essential</u>
  - Can we simulate say 50000 nodes?
- (Our) Answer
  - Implement the simulation engine on the GPU
    - Deformable model
    - Visualization
    - Haptic feedback

Virtual Surgery – Surgical Simulation

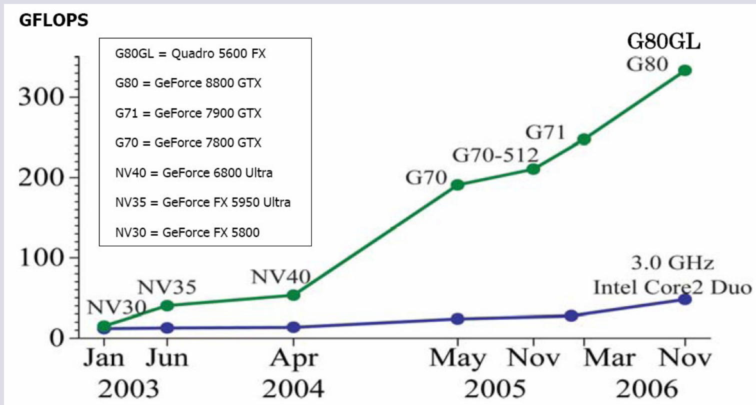
<u>Setup</u>

- State of the art pc with high-end graphics card
- One or two Phantom Omnis for force feedback



Motivation

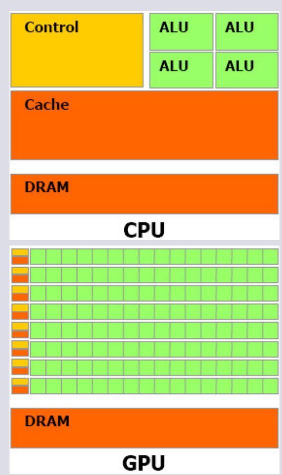
Floating point operations per second (FLOPS)



Nvidia CUDA Programming Guide.

GPU vs CPU

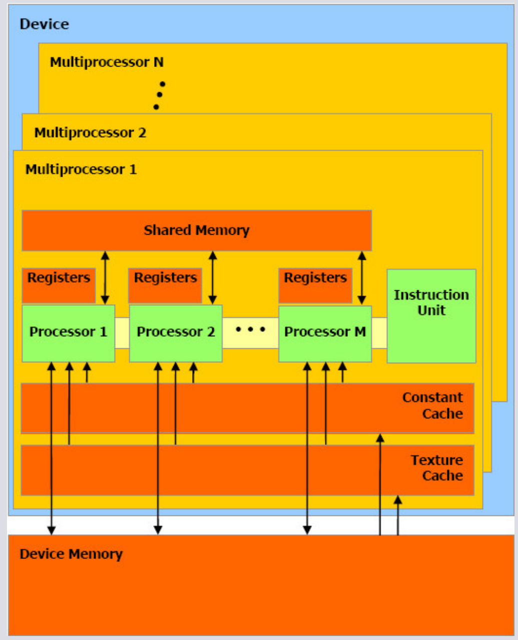
- The GPU utilizes parallel computation
  - Execution on many elements concurrently
- Single Instruction – Multiple Data (SIMD) architecture, i.e. limited flow control requirements
- Memory latency hidden by computation, i.e. limited cache requirements



Nvidia CUDA Programming Guide.

Hardware implementation

- A Set of SIMD multiprocessors with on-chip shared memory



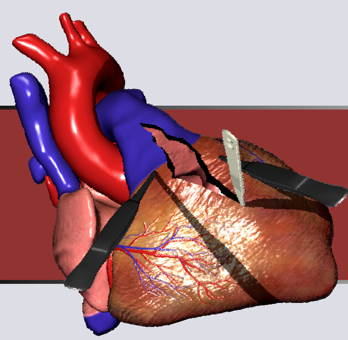
Nvidia CUDA Programming Guide.

CUDA API

- Standard C/C++
  - Templates
  - Classes
  - Overloading
- Only a few language extensions to define the interface to the GPU
- High-level libraries
  - BLAS
  - FFT
- Emulation mode for debugging

Language extensions

- Roughly speaking only four additions to standard C
  - Function type qualifiers to specify whether a function executes on the host or on the device
  - Variable type qualifiers to specify the memory location on the device
  - A new directive to specify how a kernel is executed on the device
  - Four built-in variables that specify the grid and block dimensions and the block and thread indices



Deformable model

Deformable model

- Spring-mass model with damping
  - Second order ordinary differential equation
  - The acceleration of a particle is determined by spring forces and damping forces

$$\mathbf{M}\ddot{\mathbf{u}} = \mathbf{f}_{spring} + \mathbf{f}_{damping}$$

  - $\mathbf{f}_{damping}$ is proportianal to the velocities

$$\mathbf{f}^i_{spring} = \sum_{neighbors} k_{ij}(||\mathbf{p}_i\mathbf{p}_j|| - l_{ij})\frac{\mathbf{p}_i\mathbf{p}_j}{||\mathbf{p}_i\mathbf{p}_j||}$$

- With explicit time integration
  - Verlet integration updates particle positions from
    - Positions of the two previous iterations
    - Force/acceleration vector

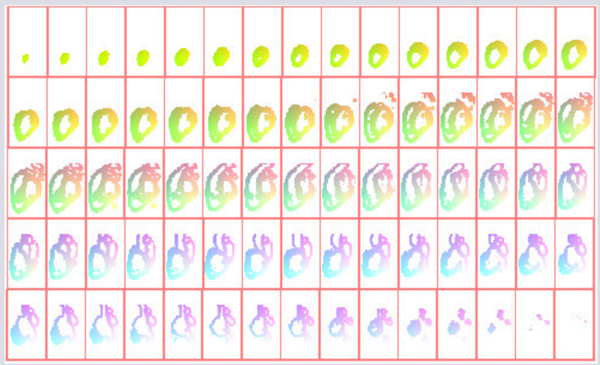
Mapping to the GPU

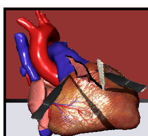
- OpenGL
  - Rendering computer graphics off-screen
  - The “graphics” is then the desired computation
- CUDA
  - New programming platform for Nvidia GPUs
  - “**C**ompute **U**nified **D**evice **A**rchitecture”
  - Simple extension of C/C++

OpenGL

Parallel computation

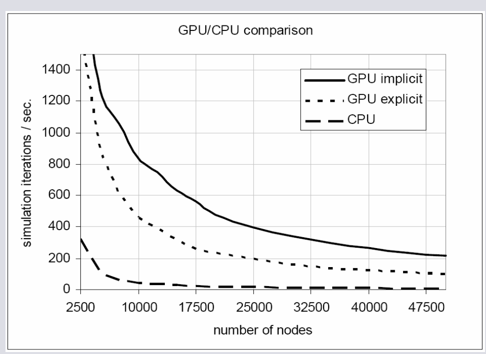
- One processor pr pixel (conceptually)
- Image (rgb) corresponds to positions (xyz)
- Kernel invoked on all pixels (at the same time)





Performance

- **Geforce 6800 Ultra**
- The GPU is up to 30 times faster as the CPU
- We are now several GPU generations further on…
- **Geforce 8800 GTX:**
  - 30.000 particles / 18 neighbours each
  - **>4000 iterations/sec**



Mosegaard et al. IEEE Virtual Reality 2005.

CUDA

- Different programming model on the *same* hardware
  - How fast can the spring-mass system be simulated?
  - Does it compare to the OpenGL performance?
  - General memory model for read/write to global memory plus fast on-chip shared memory
    - Which is the better implementation?

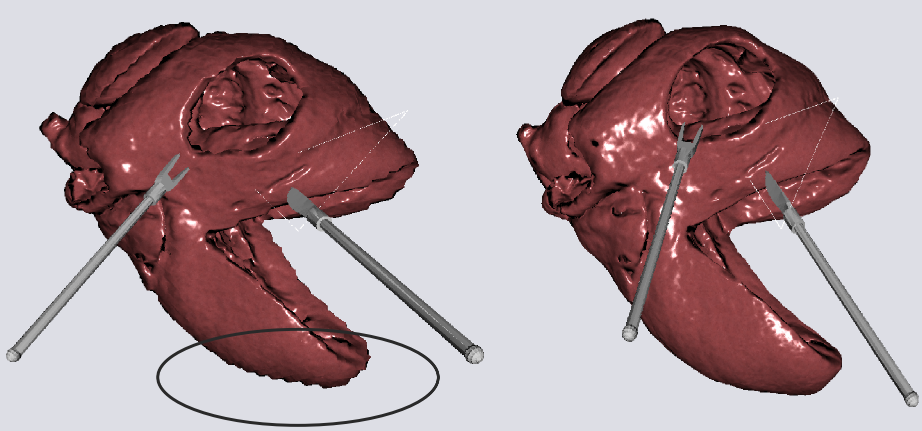
CUDA

- Does performance compare to the OpenGL performance?
  - Yes!

Visualization

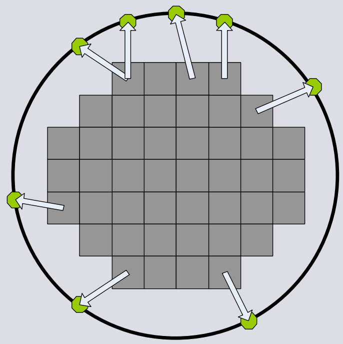
Visualization

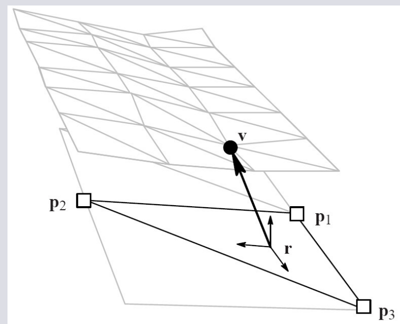
- Decoupled from the physical simulation
  - Visualisation is most often desirable at a higher resolution as what can be simulated in real time.



Decoupling of simulation and visualisation

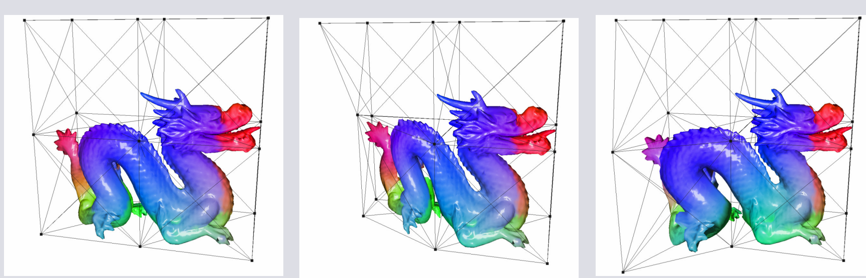
- Smooth geometry
  - Circle (left) / high resolution mesh (right)
- Represented by
  - A “tangent space” offset from the simulation nodes





Example

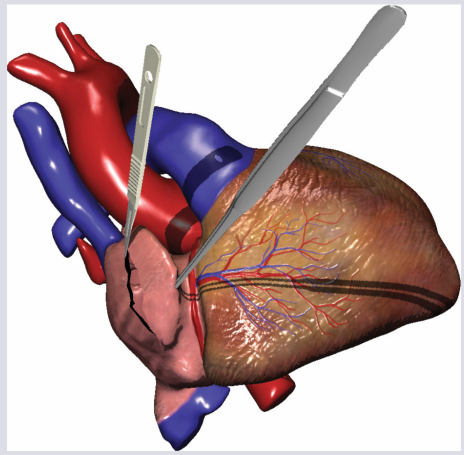
- Deformation of a detailed surface model based on a very low resolution volumetric mesh.



Mosegaard et al. Eurographics Virtual Environments 2005.

Texturing

- Details do not have to be modelled geometrically
  - Normal maps for
    - Coronaries
    - Muscle structure



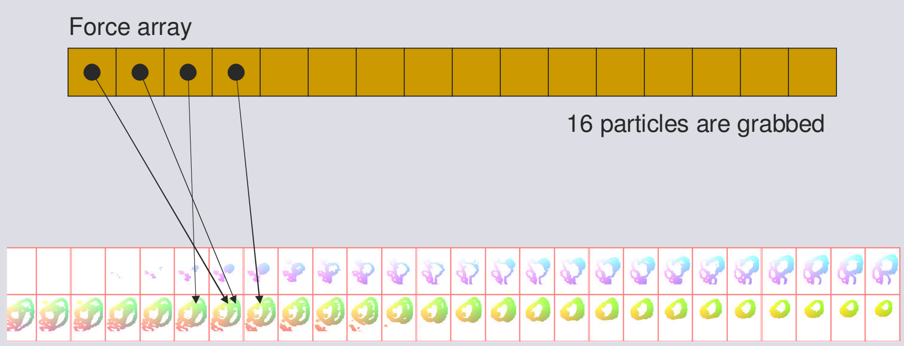
Force Feedback

Force feedback

- Compute forces on the GPU
  - Read back result to the CPU (device)
  - Minimize the required bus bandwidth
- Two gestures
  - Grab (easy)
  - Probe (tricky)

Grabbing

- Only limited sized array is read back



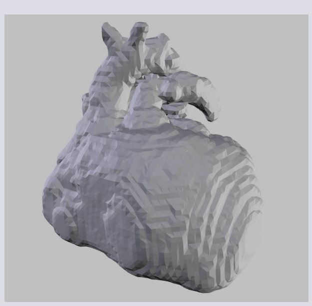
Array of particle positions (recompute spring forces)
or summed spring forces per particle (from last simulation step)

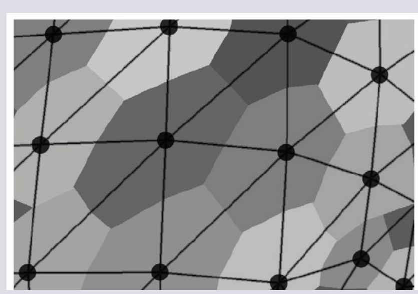
Probing

- The list of particles underlying the force feedback changes in each frame
  - Assuming this list was known we could proceed as in the case of grabbing
  - The challenge is then to determine which particles are touched by the instrument

Picking for probing

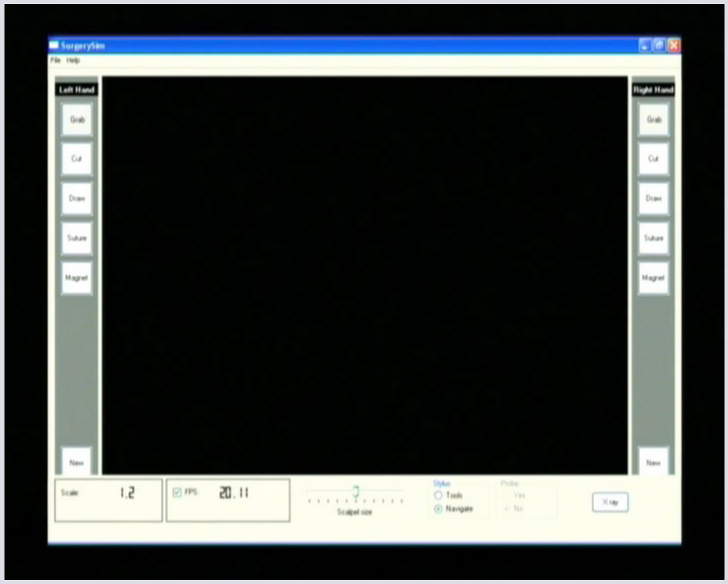
- Render the “simulation surface” from the tip of the instrument
  - Shading indicates the particle “coordinates” of the nearest particle in GPU memory
  - Update rate << simulation rate





Sørensen et al. MMVR 2006.

Picking buffer movie



Putting it all together

- Using the settings from the movie…

Geforce 7900 GTX

- Visualisation
  - 80.000 faces
  - Updated at 25 Hz
- Simulation
  - 30.000 nodes / 18 springs
  - 20 simulation steps for each visualization step
  - Updated at 500 Hz
- Force feedback
  - After each simulation step
  - Picking buffer updated after each visualization

Geforce 8800 GTX

- Visualisation
  - 80.000 faces
  - Updated at 75 Hz
- Simulation
  - 30.000 nodes / 18 springs
  - 20 simulation steps for each visualization step
  - Updated at 1500 Hz
- Force feedback
  - Turned off

Summary

- We have shown that GPU based surgical simulators are indeed feasible
  - Deformable model
  - Visualization
  - Force feedback
- The CUDA framework makes GPU programming much easier
  - C/C++ programming with minor language extension to define the CPU/GPU interface
  - Templates / classes / overloading