1st International Conference on Mathematical and Computational Biomedical Engineering – CMBE2009
June 29 – July 1, 2009, Swansea, UK
P. Nithiarasu and R. Löhner (eds)

Computational Requirements of the Virtual Patient

Nigel W. John
School of Computer Science, Bangor University, n.w.john@bangor.ac.uk
Chris J Hughes
School of Computer Science, Bangor University
Serban R. Pop
School of Computer Science, Bangor University
Franck P. Vidal
APIS, INRIA Saclay Research Centre
Oliver Buckley
School of Technology, Oxford Brookes University

ABSRACT

Medical visualization in a hospital can be used to aid training, diagnosis, and pre- and intra-operative planning. In such an application, a virtual representation of a patient is needed that is interactive, can be viewed in three dimensions (3D), and simulates physiological processes that change over time. This paper highlights some of the computational challenges of implementing a real time simulation of a virtual patient, when accuracy can be traded-off against speed. Illustrations are provided using projects from our research based on Grid-based visualization, through to use of the Graphics Processing Unit (GPU).

Key Words: *Medical visualization, virtual environment, Grid, GPU*

1. INTRODUCTION

The medical domain provides excellent opportunities for communication and teaching of healthcare issues using computer graphics, visualization techniques, and virtual environments. Possible applications include anatomical educational tools; patient education; diagnostic aids; virtual autopsies; planning and guidance aids; skills training; and computer augmented reality. Both clinicians and patients can benefit from the appropriate use of tools that make use of these technologies.

The ability to render and manipulate medical data in 3D is one of the core requirements of a medical virtual environment, but traditionally volume and surface rendering algorithms have been expensive to compute. The ray casting algorithm, for example, is a $O(N^3)$ problem, where N is the dimension of the voxel data set. Despite the use of optimisation techniques, volume rendering on a single CPU is not an option for real time performance. A virtual patient must also simulate soft tissues and so the computer graphics rendering must also deform naturally during interaction with a tool, or as a consequence of physiological processes such as respiration. Finite Element Modelling is a popular choice for achieving soft tissue deformation, but is also too slow for real time. Mass spring models can be in real time, but produce lower fidelity results. Cutting of tissues is another challenge, which often needs a high resolution mesh-based model and the ability to

change the topology of that mesh. Fluid dynamics may also be employed for blood flow. In some applications, the operator and instruments being used will need to be tracked. This can be achieved with dedicated hardware (such as optical or magnetic tracking), robotic joystick, or by image processing techniques. This paper examines how the computational demands of all of these requirements can be met so as to deliver an interactive medical virtual environment. Often accuracy of the model used has to be sacrificed to achieve the required speed, but even so we can achieve a suitable fidelity of simulation. Examples are provided below from our research projects and a detailed survey of the principles and applications of computer graphics in medicine can be found in [1].

2. COMPUTING THE VIRTUAL PATIENT

The computational infrastructure that can be deployed within a hospital continues to evolve and benefit from recent technology advances. This section contains several examples from our research where we have taken advantage of and contributed to the state-of-the-art in the field.

2.1 Medical Visualization using the Computational Grid

Grid computing is designed to allow end users transparent access to a wide variety of high performance computing facilities and data acquisition devices. An excellent example of the Grid being deployed for visualization tasks is the RAVE system [2], which is now in production release. Our own first experiment with Grid-based visualization in a hospital setting, however, consisted of a remote visualization application, where a graphics server processes the patient data and the results are delivered in real time across the computer network to a client workstation. This can be achieved by streaming the contents of the frame buffer on the server (where the graphics primitives are rendered) in a similar fashion to how MPEG movie files are streamed across the Internet. OpenGL Vizserver [3] from SGI was the first product to support remote visualization, and we used this software for an intra-operative application to aid with hepato-pancreatic surgery [4]. A volume rendering of the patient's CT data was delivered to a laptop client in the operating theatre where it could be interrogated by the surgeon with an easy to use joystick driven interface. At the time, the CT data sets being used (700 slices at 512 x 512 pixel resolution, with a pixel size of 0.78 mm and an interslice distance of 1 mm.) were too large to be processed on a local PC. Through this approach, however, real time rendering was achieved across a 100baseT network link, with a physical distance of one mile from the server to the operating room. Grid middleware software provided support for scheduling computer time to coincide with the operation and for security of the patient data.

The lessons learnt from the above were invaluable in the e-Viz project [5] that explored the development of autonomic visualization and designed a generic adaptive infrastructure for Grid-based visualization. A particular application that was built on this infrastructure was an augmented reality (AR) interface for a procedure called transcranium magnetic stimulation (TMS) [6]. TMS requires an electro magnetic coil to be accurately positioned against a subject's head. Using AR, a 3D rendering of a subject's brain can be overlayed and registered onto a video stream of their head so helping the clinician to position the coil. A markerless interface was developed to achieve this, in which repeatable feature points are extracted from known views and then we match the best stored view to the user's viewpoint using the matched feature points to estimate the objects pose. Our research has shown that whilst an average desktop PC struggles to carry out the pose estimation, using remote resources can ensure real-time performance. Provided the visualization server is appropriate for the rendering task and network latency is low, then the e-Viz framework is able to return the rendered artefact to the user at a reliable 15 frames/second. On congested networks e-Viz uses stricter compression algorithms at a cost to the image quality to try and maintain these usable frame rates.

2.2 The GPU Age

A well known hardware acceleration technique for volume rendering is to use texture mapping hardware [7] and this method lends itself well for implementation on the Graphics Processing Unit (GPU) found on all modern PC graphics cards, and is often combined with use of per pixel, or fragment, shaders [8]. The latest PCI Express architecture allows efficient fetching of texture data from the main memory of the PC via the graphics bus. Special purpose hardware has also been designed for PCs, for example, the VolumePro 2000 (TereRecon, USA) implements shear warp rendering and supports memory capacities

ranging from 512MB to 16GB, which can handle up to 30000 CT slices. This performance can now often be matched by the GPU on a high-end graphics card, and even an inexpensive PC can achieve real time for a 256 cubed voxel data set.

In our simulator for ultrasound guided needle puncture [9], two haptic devices are used: one to manipulate a virtual ultrasound probe; one to represent the virtual needle. Ultrasound-like images are generated from the original patient CT data by transforming the appearance using GPU operations via the OpenGL Shading Language. A 2D multi-planar reconstruction (MPR) image is extracted from the CT voxel data based on the position and orientation of the virtual ultrasound probe being moved across the skin of the virtual patient. This can be efficiently achieved using the OpenGL frame buffer object (FBO) architecture. All voxels that have been penetrated by the virtual needle are assigned a high value corresponding to the metallic material of the needle shaft, which reflects ultrasound. Acoustic shadowing effects are simulated by post-processing the MPR image to compute a shadow mask, and high frequency noise is also added. Bright reflections may occur in ultrasound images at interfaces such as with bone, gas, and fat/tissue. This effect can be produced in the final image by detecting and enhancing horizontal edges in the MPR image using a Sobel filter. Finally, the MPR image, the shadow mask and the noise data are blended using multi-texturing. In an interventional procedure, the needle can then be used as a portal for the entry of other tools such as a guidewire and catheter. Fluoroscopy images are used in the operating room to keep track of these tools. To maintain real time performance we have also developed fluoroscopy simulation on the GPU from voxel data [10] or a polygon mesh [11]. In the latter case, this produces X-ray images using a three pass algorithm through the OpenGL pipeline. For each X-ray pixel, the first pass computes tissue penetration, the second computes an intermediary result required in the final pass to compute the cumulative attenuation using the Beer-Lambert law. This approach is extremely efficient. For example, using full floating point precision, our results show that the GPU is over 60 times faster than a CPU implementation when computing a 1024 x 768 pixel X-Ray image of a test object made up of 871,414 triangles, with no significant loss of accuracy (differences smaller than 0.3%).

2.3 Software Techniques

There has been a great deal of research into optimisation of computer graphics algorithms to gain speed increases – for an overview, refer to [1]. Taking soft tissue deformation as an example, the Chain Mail algorithm [12] has proved to be a popular alternative to FEM and mass-spring models. With this algorithm materials are modelled by adjusting deformation limits for individual elements. For a medical simulator, however, the requirement is not only for rendering speed – often you want to models forces too so that a haptics interface can be used. Haptics devices require a refresh rate of 1000 Hz to provide a smooth response and this must be run in parallel with the graphics rendering. In exploring this problem, we have developed a method based on particle systems, called the Charged Particle Model [13]. The soft tissue is considered as being composed of a large number of particles that are each assigned a virtual electrical charge. The haptic interface point is also assigned a virtual charge and forces are then simulated using the rules of electro-magnetic interaction. Using this approach we have demonstrated that real time performance for deformation with haptics can be achieved, typically using around 6000 charged particles surrounded by a Bezier surface of 250,000 points.

Current work at Bangor is looking at other ways of gaining speed through novel use of particle systems. There is a need to introduce blood flow into our simulators, for example, particularly for vascular intervention. Our model [14] is based on the idea that each layer of fluid behaves as a flock, interconnected by the parameters that govern the flow dynamics. At the macroscopic level blood can be considered as a Newtonian fluid and represented using an underlying particle system. Many similarities with existing particle dynamics systems for fluids are kept (e.g., the kernel function in smooth particle hydrodynamics (SPH) is replaced by the flock neighbourhood rule; however the search for nearby particles is still performed in the usual way). To conserve the mass of the system, we keep the number of particles inside the domain constant during the entire simulation. Each particle carries its own physical quantities such as mass, speed and position, which means that control is maintained over the main physical parameters of the fluid. The result is a real time visualization of blood flow with comparable results to commercial fluid dynamic software that takes 10-20 seconds to produce a single image.

3. CONCLUSIONS

The research projects described above have demonstrated that it is possible to compute a virtual patient for a real time graphics application. Visualization on the Grid is one possibility and provides the opportunity to combine 3D visualization with complex simulation algorithms in real time. However, the ever increasing computational power and very modest cost of desktop PCs make them the ideal platform for running many of the applications discussed in this paper. There remains a trade off between accuracy and speed with the latter taking precedence in a medical virtual environment. Despite this, validation studies that we are actively carrying out with our clinical collaborators show that a sufficient fidelity of simulation can be obtained for training and educational purposes.

REFERENCES

[1] F.P. Vidal, F. Bello, K.W. Brodlie, D.A. Gould, N.W. John, R. Phillips, and N.J. Avis, Principles and Applications of Computer Graphics in Medicine, *Computer Graphics Forum, Vol. 25 Issue 1*, pp113-137, 2006

[2] I.J. Grimstead, N.J. Avis, and D.W. Walker, RAVE: Resource-Aware Visualization Environment, *in Proc. of the UK e-Science All Hands Meeting*, Nottingham, UK, 2004

[3] C. Ohazama, OpenGL Vizserver White Paper, *White Paper*, Silicon Graphics, Inc, 1999

[4] N.W. John, R.F. McCloy, and S.Herrman, Interrogation of Patient Data delivered to the Operating Theatre during Hepato-Pancreatic Surgery using High Performance Computing, *Computer Aided Surgery 9(6)*, pp235-242, 2004

[5] K.W. Brodlie, J. Brooke, M. Chen, D. Chisnall, C. Hughes, N.W. John, M.W. Jones, M. Riding, N. Roard, M. Turner, and J.D.Wood, Adaptive Infrastructure for Visual Computing, *Proc. of Theory and Practice of Computer Graphics*, Eurographics, pp. 147-156, 2007

[6] C. J. Hughes and N. W. John, A Flexible Approach to High Performance Visualization Enabled Augmented Reality, *Proc. of Theory and Practice of Computer Graphics* 2007, Eurographics, pp181-186, 2007

[7] T.J. Cullip and U. Neumann, Accelerating Volume Reconstruction with 3D Texture Hardware. *Tech. rep.*, University of North Carolina, Chapel Hill, NC, USA, 1994.

[8] K. Engel, M. Hadwiger, J. M. Kniss, A. E. Lefohn, C. Rezk Salama and D. Weiskopf, Real-time volume graphics. *Tutorial 28 in SIGGRAPH* 2004.

[9] F.P. Vidal, N.W. John, A.E.Healey, and D.A. Gould, Simulation of Ultrasound Guided Needle Puncture using Patient Specific Data with 3D Textures and Volume Haptics, *Computer Animation and Virtual Worlds. Vol. 19, Issue 2*, pp111-127, 2008

[10] F.P. Vidal and N.W. John, Interactive Physically-Based X-Ray Simulation: CPU or GPU?, *Stud Health Technol Inform 125,* pp 479-481, 2007

[11] P.F. Villard, F. Bello, F.P. Vidal, N.W. John, C. Hunt, S. Johnson, D. Gould, Percutaneous transhepatic cholangiography training simulator with real-time breathing motion, *23rd International Congress of CARS - Computer Assisted Radiology and Surgery*, Berlin 2009

[12] S. Gibson, 3D Chainmail: A Fast Algorithm for Deforming Volumetric Objects, *Proc. Symposium on Interactive 3D Graphics*, pp 149--154, 1997

[13] O. Buckley and N.W. John, Efficient Soft Tissue Modelling Using Charged Particle Control Points, *Eurographics 2008 Short Paper*, Crete, ISSN 1017-4656, 191-194, 2008

[14] C.J. Hughes, S.R. Pop, and N.W. John, Macroscopic Blood Flow Visualization using Boids, *23rd International Congress of CARS - Computer Assisted Radiology and Surgery*, Berlin 2009