

Putting Visualization First in Computational Steering

Helen Wright

Simulation and Visualization Research Group
Department of Computer Science, University of Hull, HULL HU6 7RX, UK

Abstract

This paper challenges the tradition that the part played by visualization within computational steering is limited purely to looking at output results, highlighting the potential of visualization to handle the input of simulation parameters as well. By considering the principles of human-computer interaction that are involved in steering, it proposes a model which puts visualization first, rather than last, in the computational steering pipeline and demonstrates how such an arrangement may benefit both the user and the designer of computational steering systems. The potential of two projects, MPII and SuperVise, to contribute to the realisation of this model on the grid is described.

1. Introduction

Computational steering is a valuable mechanism for scientific investigation in which the parameters of a running program can be altered and the results visualized immediately. In contrast with the usual approach which is to post-process results produced by a batch job, the immediacy of computational steering allows rapid assimilation of the simulation's properties by the scientist.

The grid offers a number of benefits to the simulation community, the most obvious of which is the potential to bring a wide range of resources to bear on increasingly complex problems. The commonest criticism levelled at the computational steering paradigm is the small number of problems that can be manipulated at interactive rates – the potential for the grid to increase the scope of computational steering is therefore obvious. This paper challenges the tradition that in computational steering visualization is purely for looking at results, perpetuated at least in part by the batch philosophy that has characterised grid computing to date. It first summarises current approaches and then puts the case for a model which places visualization first, rather than last, in the computational steering pipeline, reviewing two recent projects at the University of Hull that can contribute. The paper concludes with an assessment of the feasibility of the ideas for real problems and a summary of the principles of re-use and usability that this approach embodies.

2. Computational Steering Pipeline

The visualization model proposed by Upson et al. (1989) and in a different form by Haber and McNabb (1990) recognises the pipeline nature of transformations that raw data undergoes in order to emerge as an image. Any visualization system, even if it is monolithic in design, will incorporate these transformations within it; in Modular Visualization Environments (MVEs) the user interface implements the visualization model explicitly resulting in a dataflow paradigm. The conventional model of computational steering, rather than collecting data and post-processing it, closely couples the simulation which is producing the data into the pipeline (Figure 1).

Steering typically begins with a start-up phase, when a fairly large number of parameters are investigated quickly, followed by a steering phase, when fewer parameters are varied exhaustively. Within the outer loop of the search process (each set of parameter values tried) there is an inner loop of analysis (visualization) undertaken by the scientist. Their role is therefore not only to guide the investigation but also, by default, to synchronise these inner and outer loops.

Several different approaches to computational steering can be identified. One, demonstrated by Pickles et al. (2004), is to instrument the code to deliver data for remote systems to visualize on-line; another (see for example Kong et al. (2003)) is to take simulation and visualization elements implemented as components and compose applications dynamically. Computational steering within MVEs can be accomplished by

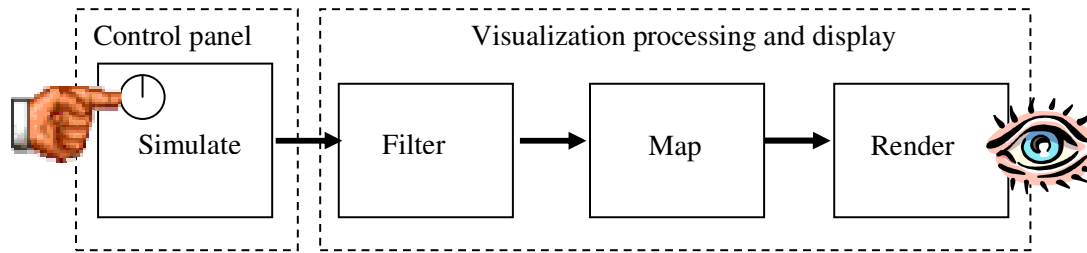


Figure 1. Conventional computational steering models envisage interaction with parameters at one end of the pipeline followed by viewing of results at the other end

including the code as a module with parameters exported to a control panel whose appearance is consistent with the remainder of the system. Wood et al. (2003) have demonstrated such an arrangement where the computational element utilised grid resources but with the interface and display handled locally within the visualization environment. In some approaches the model of Figure 1 can therefore be seen to reflect the steering architecture employed in practice, whilst in others it simply provides a convenient description of the activity.

3. A Visualization-First Approach

The conventional model of computational steering, in addition to the many technical hurdles that must be overcome, presents two principal usability difficulties for the investigating scientist. Firstly, interaction with the parameters of the simulation is usually via dials and sliders, whose different values cause changes in the graphical output, viewed in a separate window. Although windows may be nearby on the user interface, the scientist usually still has to switch context from one to another to accomplish their task, and logically the interaction and visualization stages are perceived as occurring at two opposite ends of the pipeline. Secondly, the model works on the ‘push’ principle, whereby new parameter values are specified, flow down the pipeline and produce a different visualization. If the computation is slow or is carried out on a remote machine, then problems can occur because the repeated and rapid specification of parameters by the scientist can overload the pipeline.

Hart and Kraemer (1999) have characterised this problem of computational steering as *lag*, comprising the sum of *steering* and *presentation* lag, which respectively represent the time taken for the user’s interaction to affect the computation and the time taken for new results to reach the viewer. They point out that where steering is performed in order to understand the

parameterisation of the computation, then minimising lag is crucial. They also recognise *consistency* as an issue in computational steering, though their emphasis is on maintaining inter-process consistency of distributed applications. Our experience finds that synchronising the interaction (displaying the current value of a steered parameter) with the system state (shown in the visual results) is also a pressing consistency requirement, and is just as important to a usable system as minimising lag.

3.1 A mental model of steering

Looking at the problem from the point of view of the scientist merits a new approach, attempting first of all to elicit their mental model of steering. We limit the scope to a consideration of graphical user interfaces supporting a direct manipulation style, so following Schneiderman (1983) will employ object-action modelling, utilising a notation based on that suggested by Newman and Lamming (1995).

A typical scenario begins with users aware there may be lag but not knowing its extent. They make an experimental change to a parameter and await the results. If this takes longer than some arbitrary judgement of what is reasonable, they may try another parameter value in the belief that the original change was for some reason not entered into the system or not sufficient to see a difference. In reality, of course, the original change might not yet have taken effect. Such observations reveal an initial mental model of a parameter object with a change action that affects its results attribute, but with an incomplete understanding of the time taken to effect the change (Figure 2). Whether or not the user is eventually successful in completing their mental model depends to a great extent on how long the actual lag is, the degree of feedback afforded by the user interface elements, and how obvious the results of their parameter changes are. Steering may

continue quite successfully once the extent of lag has been assessed, but changes that affect lag can interrupt progress, such as when the system load varies or a different parameter is steered.

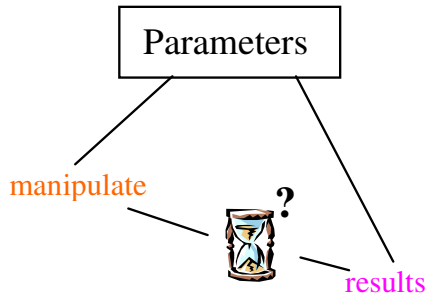


Figure 2. An incomplete mental model of computational steering. The user is aware that the action (red) of manipulating a parameter affects its results attribute (magenta), but has yet to understand the extent of lag

The resolution of this problem is to promote a mental model which reflects the reality of steering, designing the user interface so as to present a consistent picture, even in the presence of lag. Schneiderman points out the expectation to see an immediately visible reaction when operating a direct manipulation system and it is this that is missing in the presence of lag. Some surrogate for the arrival of new results is needed that signifies a parameter change has been accepted, and it turns out that the parameter object itself holds the key. A parameter object that changes its appearance and becomes inactive after it gets a new value (user manipulation) can cue that re-computed results are pending; when these do eventually arrive they reverse the process. If desired, the equivalent visual cue can be applied to the display of results that are out-of-date, compared with those that are up-to-date. This leads to a more complex, but also more accurate, mental model that requires the results of the simulation to be considered an object in their own right. The parameter object and the results object respectively acquire Boolean attributes called ‘active’ and ‘out-of-date’ that, along with their appearance, switch state under the ‘manipulate’ and ‘re-compute’ actions. Figure 3 shows the object-action diagram for this new model.

3.2 Enabling technologies

The mental model of Figure 3 could be adequately supported by a user interface toolkit comprising the usual dials and sliders, but whose elements were able to change their appearance and active state as required.

However, returning to the first of the usability problems described in section 3 implies another requirement – to manipulate parameters directly via the rendered image. Multi-Purpose Image Interaction (MPII; Chatzinikos and Wright, 2003) provides interactors in the form of geometry which can be rendered along with the display of results. The work targets MVEs and consists of a toolkit of commonly needed interactors – vector, scalar and positional manipulators. A process at the head of the pipeline on the left-hand side of Figure 1, as well as sending down its output data, transmits its input parameter requirements too. This information is used by *InsertInteractor*, a Map module which is also part of the toolkit, in order to place the required elements in the rendered scene. When activated by the user the interactor contacts its originator with new parameter values, generating new results which flow to the display.

The toolkit has found applicability both in computational steering and in visualization to post-process data. Figure 4 shows a data slicing tool that has been converted from having its conventional control on a separate portion of the panel to using a geometrical vector normal included in the image. The conversion was carried out to minimise context switching during full-screen interaction with the wall-sized display in the Hull Immersive Visualization Environment (HIVE, 2003). Effort required for the conversion was modest, consisting of changes to a transform matrix generator to accept three vector components from the interactor instead of 2D mouse coordinates from the inset panel.

Having software tailored to particular hardware arrangements potentially increases complexity for the user when selecting from the different versions available, and is contrary to the original aim of improving usability. SuperVise (Osborne and Wright, 2004) addresses this concern, employing grid-based tools to advise on appropriate resources. SuperVise has access to information about various nodes’ capabilities held in a repository based on Metacomputing Directory Services, a subsystem of the Globus 2 toolkit (Globus, 2002). Rather than containing the usual information about operating system and installed grid software, the repository developed includes each node’s display capabilities. Thus, for example, when a user selects a facility such as stereo output that is not available on their current hardware, an information window advises where such facilities can be found within the HIVE. Conversely if the facility is

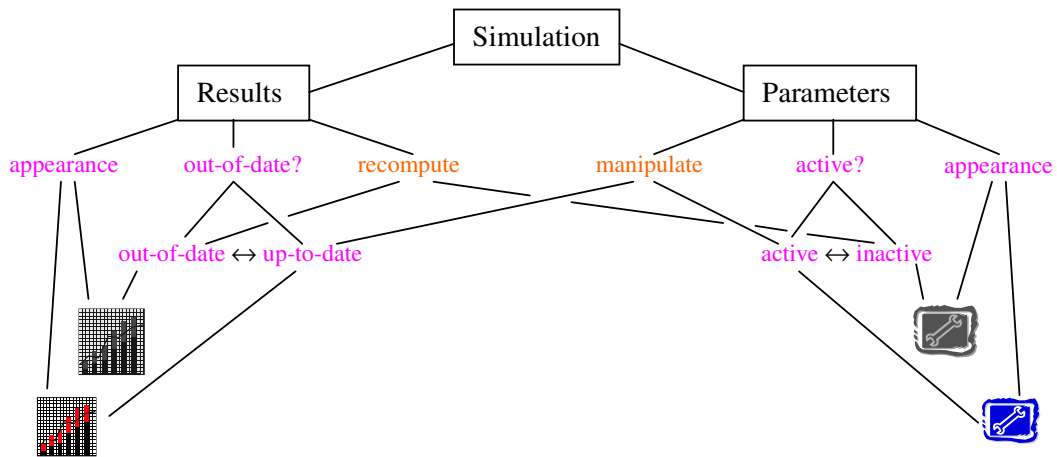


Figure 3. A more complete mental model of steering than Figure 2, showing attributes (magenta) and actions (red) pertaining to ‘results’ and ‘parameters’, sub-objects of the simulation object. Manipulating a parameter simultaneously makes it inactive and the results out-of-date, prompting a re-computation which reverses the process

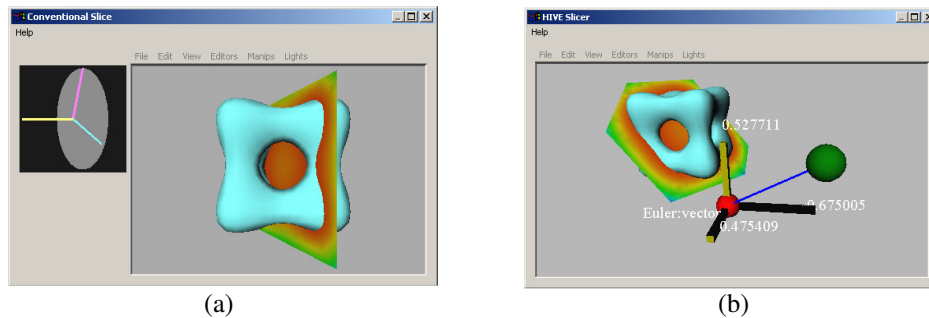


Figure 4. A data slicing tool controlled (a) by means of a separate window and (b) by incorporating a vector interactor in the image. In (a) the visual link between the slice and its yellow plane normal is broken as soon as the image is rotated. In (b) this link is maintained because the two elements of geometry, namely the image and the interactor, move together

available, appropriate software is fired up to drive the device. The same repository also holds information on visualization processing capabilities and SuperVise has been used to distribute different parts of the visualization pipeline to different hosts using Grid Resource Access Management and Grid-FTP.

3.3 Modified steering pipeline

Combining MPII and SuperVise makes feasible a model for steering on the grid that, in contrast with the conventional model, recognises visualization as both the logical output *and* the input route (Figure 5). Collocation of input parameters and graphical output provides a convenient way to preserve mutual knowledge of their respective states and may therefore alleviate some of the technical difficulties associated with synchronising processes. Built

primarily to minimise context switching, MPII steering applications to date have not exploited this knowledge of state but potentially it also solves the usability problems associated with interface inconsistency described in 3.1. MPII’s focus was not remote processing but, using SuperVise, elements of the pipeline could be distributed across available resources. SuperVise, with extensions to its node capabilities repository, could also help distinguish the different software versions required to drive different input devices. For example, SuperVise currently has an entry `Output: stereo | mono` in its node capabilities repository; a complementary entry `Input: 2DOF | 6DOF` would allow to switch support between standard mouse input and a 3D device more suited to MPII’s geometrical interactors.

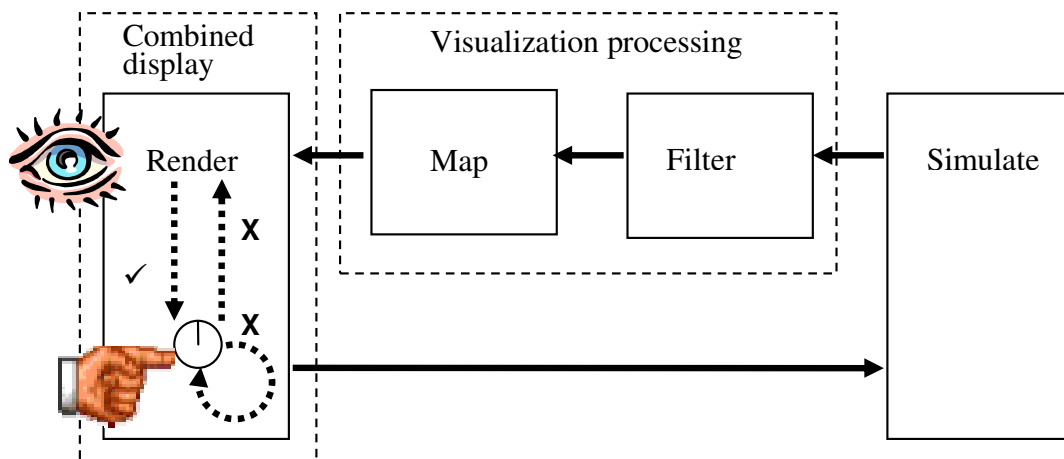


Figure 5. A model for computational steering that collocates parameter interaction and rendering of results in a single display process. Changing a parameter value toggles the interactor state to ‘inactive’ and the state of the rendered results to ‘out-of-date’. The simulation is notified and computes new results, whose arrival sets the interactor state back to ‘active’. This ‘pull’ model, in contrast with the ‘push’ model of the conventional pipeline, ensures the parameters seen by the scientist are always consistent with the corresponding results, regardless of where the computation runs or how long it takes

3.4 More complex steering scenarios

This paper has so far addressed the problems caused by the potential to make repeated requests that overload the simulation – in the configuration shown in Figure 5 this would represent an excess of data flowing left-to-right. Similar problems can arise during ‘tracking’ (Marshall et al., 1990; termed ‘monitoring’ by Hart and Kraemer (1999)), where simulation-initiated results flow too quickly right-to-left and overload the renderer. Wood et al. (2003) note some strategies for managing a right-to-left overload, including dropping some visualizations, caching and slowing down the simulation whilst the display catches up. Real applications thus typically comprise both steering (in the strict sense of altering parameter values) and tracking modes, with repeated parameter interactions effecting changes in the results that are tracked in the meantime. Modelling this situation is quite complex – if the simulation has not been configured for immediate interrupts, or if caching is in place, the user may encounter a lag in the actuation of their parameter change which is nonetheless accompanied by apparently new results arriving at the display. Figure 6 shows a possible sequence of events at the renderer, using the GRASPARC History Tree (Brodie et al., 1993) to demonstrate how several ‘snapshots’, or time-steps, of the simulation might be visualized whilst the interactor is inactive. Its re-enablement is by means of the arrival of results originating specifically from the tree’s

branch point, which signifies the point where the parameter change is actually applied to the simulation. Treatment of this situation is therefore feasible using the model of Figure 5, provided the *particular* set of results arising from the changed parameter can be identified amongst all of those flowing to the renderer. It is likely however that a visual indication of out-of-date results (see section 3.1) would not be helpful here, conflicting with the changes seen anyway by virtue of tracking the simulation.

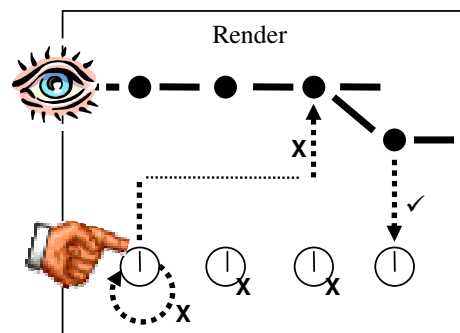


Figure 6. A model for computational steering based on that in Figure 5 but incorporating delay in the application of parameter changes. In this example the inactive interactor is re-rendered twice in succession along with cached ‘snapshots’ of the simulation results, until the branch point of the history tree is reached. Arrival of results from the head of the new branch sets the interactor state back to ‘active’

4. Conclusions

This paper has described a paradigm for computational steering that highlights the potential for the visualization process to unify the input of parameters with the output of results. More than simply a juxtaposition of dials and images, the display envisaged has knowledge of the respective states of parameters and results, and plays a key role in synchronising steering activity in the presence of lag, relieving the user of this responsibility. The paper has also described the toolkit of MPII interactors that can be added to existing applications to allow them to be steered via the image. Using vector, scalar and positional manipulators it is possible to convert a range of simulations without having to perform a complete re-implementation of the visualization software for each one. Combination of these tools with the products of SuperVise will fulfil the long-term aim of this research – to produce usable steering applications for the grid.

5. Acknowledgements

MPII and SuperVise are the work of Fotis Chatzinikos and James Osborne respectively. The author acknowledges useful discussions with them and the assistance of other colleagues working at the Department of Computer Science, University of Hull, in particular James Ward, Research Fellow in the HIVE.

References

Brodie K, Poon A, Wright H, Brankin L, Banecki G & Gay A. (1993). In: Nielson G M & Bergeron D (eds), Proceedings of IEEE Visualization 1993 Conference, pp 102 - 109, IEEE Computer Society Press. *GRASPARC: A Problem Solving Environment Integrating Computation and Visualization*

Chatzinikos F & Wright H. (2003). In: Erbacher R F, Chen P C, Roberts J C, Grölin M & Börner K (eds), SPIE Vol 5009: Visualization and Data Analysis 2003, pp 455 – 462, SPIE Press. *Enabling Multi-Purpose Image Interaction in Modular Visualization Environments*

Globus. (2002). <http://www.globus.org/training/grids-and-globus-toolkit/IntroToGridsAndGlobusToolkit.ppt>. *Introduction to Grid Computing and the Globus Toolkit*

Haber R B & McNabb D A. (1990). In: Nielson G M, Shriver B & Rosenblum L J (eds), Visualization in Scientific Computing, pp 74 - 93, IEEE Press. *Visualization Idioms: A Conceptual Model for Scientific Visualization Systems*

Hart D & Kraemer E. (1999). International Journal of Parallel and Distributed Networks and Systems, Volume 2 No. 3, pp 171 – 179, ACTA Press. *Consistency Considerations in the Interactive Steering of Computations*

HIVE. (2003). <http://www.hive.hull.ac.uk/>. *HIVE (Hull Immersive Visualization Environment)*

Kong G, Stanton J, Newhouse S & Darlington J. (2003). In: Cox S J (ed), Proceedings of the UK All Hands e-Science Meeting 2003, pp 393 – 396, EPSRC. *Collaborative Visualisation over the Access Grid using the ICENI Grid Middleware*

Marshall R, Kempf J, Dyer S & Yen C. (1990). Computer Graphics, Volume 24 No. 2, pp 89 – 97, ACM Press. *Visualization Methods and Simulation Steering for a 3D Turbulence Model for Lake Erie*

Newman W M & Lamming M G. (1995). Addison-Wesley. *Interactive System Design*

Osborne J A & Wright H. (2004). In: Wyrzykowski R, Dongarra J, Paprzycki M, Wasniewski J (eds), Lecture Notes in Computer Science Vol 3019: 5th International Conference on Parallel Processing and Applied Mathematics (PPAM 2003), pp 856-863, Springer-Verlag. *SuperVise: Using Grid Tools to Simplify Visualization*

Pickles S M, Blake R J, Boghosian B M, Brooke J M, Chin J, Clarke P E L, Coveney P V, González-Segredo N, Haines R, Harting J, Harvey M, Jones M A S, McKeown M, Pinning R L, Porter A R, Roy K & Riding M. (2004). In: Hinke T, Cox S J, Hood R T & Towns J (eds), Workshop on Case Studies on Grid Applications, GGF10. *The TeraGyroid Experiment*

Schneiderman B. (1983). IEEE Computer, Volume 16 No. 8, pp 57 – 69, IEEE Computer Society. *Direct Manipulation: A Step Beyond Programming Languages*

Upton C, Faulhaber T, Kamins D, Laidlaw D, Schlegel D, Vroom J, Gurwitz R & van Dam A. (1989). IEEE Computer Graphics and Applications, Volume 9 No. 4, pp 30 – 42, IEEE Computer Society. *The Application Visualization System: A Computational Environment for Scientific Visualization*

Wood J, Brodie K & Walton J. (2003). In: Cox S J (ed), Proceedings of the UK All Hands e-Science Meeting 2003, pp 164 – 171, EPSRC. *GViz – Visualization and Steering for the Grid*