ALICE on the Eightfold Way: Exploring Curved Spaces in an Enclosed Virtual Reality Theater

George K. Francis[1], Camille M.A. Goudeseune[2], Henry J. Kaczmarski[2], Benjamin J. Schaeffer[2], and John M. Sullivan[1]

[1] Department of Mathematics, University of Illinois, Urbana, IL, USA, gfrancis@uiuc.edu, jms@uiuc.edu

[2] Integrated Systems Laboratory, University of Illinois, Urbana, IL, USA, cog@uiuc.edu, kacmarsk@uiuc.edu, schaeffr@uiuc.edu

Summary. We describe a collaboration between mathematicians interested in visualizing curved three-dimensional spaces and researchers building next-generation virtual-reality environments such as ALICE, a six-sided, rigid-walled virtual-reality chamber. This environment integrates active-stereo imaging, wireless motion-tracking and wireless-headphone sound. To reduce cost, the display is driven by a cluster of commodity computers instead of a traditional graphics supercomputer. The mathematical application tested in this environment is an implementation of Thurston's eight-fold way; these eight three-dimensional geometries are conjectured to suffice for describing all possible three-dimensional manifolds or universes.

1 Introduction

Successful visualization projects involve two pieces, scientists with interesting content and facilities that provide the services needed to realize that content. In this paper we explore a collaboration between mathematicians interested in visualizing curved three-dimensional spaces and researchers building next-generation virtual reality environments. In many ways, the ultimate test of new visualization technology is putting it into the hands of scientists and making it useful for their projects.

The Integrated Systems Laboratory (ISL) at the University of Illinois (UIUC) provides hardware and software tools to researchers in a variety of disciplines. This particular collaboration involves a six-sided, rigid-walled virtual reality (VR) chamber, the Adaptive Laboratory for Immersive Collaborative Experiments (ALICE), which has been built by the ISL over the last year. This is a traditional VR environment, integrating active stereo, wireless tracking of a user's position and orientation, and wireless-headphone sound. To reduce cost, the display is driven by a cluster of commodity computers instead of a traditional graphics supercomputer. To meet the software challenges of the cluster environment, the lab has a toolkit for cluster-based VR, called Syzygy, under active development. A preliminary version of Syzygy was provided to the mathematicians to aid in porting their software to the cluster.

We describe several aspects of the project, starting with the challenges of constructing the physical facility. Our collaboration was a case study in porting code from a shared-memory architecture (SGI Onyx) to a cluster architecture (Syzygy). Finally, we describe the tangible benefits of a fully enclosed environment for visualizing curved three-dimensional spaces; this project could not work as well in a less immersive environment.

We gratefully acknowledge the programming assistance of Ben Bernard and Matt Woodruff, without whose work this project would not have been possible.

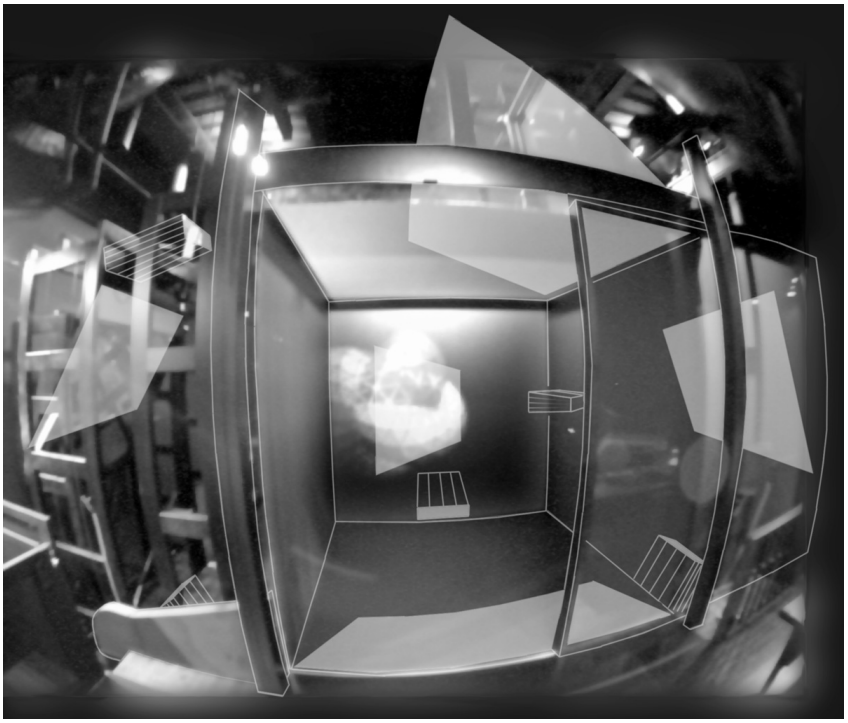

Fig. 1. A fisheye view of the ALICE VR theater, highlighting five of the six mirrors and projectors. The view is from near the sixth projector, which aims at the sliding door (halfway open in this photograph).

2 Other fully enclosed virtual reality theaters

The fully enclosed systems built so far are cubes about 3 meters on a side. Binocular images are rendered via "active stereo", which rapidly alternates

left-eye and right-eye projected images which are then directed to each eye by similarly flickering LCD shutter glasses.

ALICE is driven by a cluster of commodity desktop computers instead of a large Silicon Graphics Onyx computer. ALICE has rigid walls instead of fabric, which is prone to flutter and sag when made this large.

Similar theaters we know of include the following:

- The C6 at Iowa State University's Virtual Reality Applications Center [1] uses wireless motion tracking.
- The VR-CUBE at KTH Royal Institute of Technology's Center for Parallel Computers (Stockholm) [2] uses tethered motion tracking.
- The HyPI-6 system is located in the Virtual Reality Laboratory of the Fraunhofer Institute for Industrial Engineering [3]. In one configuration it uses a cluster of PC's, two per wall, to draw images with passive stereo.
- The COSMOS system at the VR Techno Center in Gifu (Japan) [4] has a motorized sliding rear wall. Motion tracking is tethered (cables go through a small hole at one of the corners of the floor). Dual CRT projectors on each face of the cube double the brightness of the images viewed.

3 Cluster architecture

A PC cluster architecture provides a cost effective means of driving multiple graphics pipes but has disadvantages over a shared-memory SMP architecture. With SMP, shared memory is an efficient means of communication between the rendering processes, the single instance of the operating system can be used to easily manage the software, and time-tested software libraries, such as the CAVELib [5, 6], exist to aid the applications programmer. Many of these substantial advantages disappear when considering a PC cluster architecture. Communication between rendering processes can now become a bottleneck because it occurs over a network instead of a memory bus. Managing the different application components spread across the network can be a challenge, especially if they are running under different operating systems. Finally, there is no standard library for writing VR software for the cluster architecture.

Fortunately, there are several interesting software projects that may address this in the near future. Of particular note for the VR community is the NetJuggler [7] extension to VR Juggler [8]. This allows multiple copies of a VR Juggler program (satisfying certain conditions) to run synchronized across a cluster. Another set of tools to run multiple synchronized copies of an application comes from the Princeton Omnimedia group [9]. While their focus is not on VR, their software could be easily adapted. Similarly, the WireGL effort [10], which replaces the OpenGL shared library with a stub that sends data over a network for rendering, is a distributed graphics library with potential for VR applications.

The projects just described take one of two paths: distributing the application, as in the case of NetJuggler, or distributing the data, as in the case of WireGL. Distributing the application, namely running synchronized copies of an application on multiple nodes, can use far less network bandwidth than a data distribution strategy. This advantage is balanced, however, by the restrictions that typical synchronization schemes place on application design. For instance, multithreading might not be allowed, or the programmer might be constrained to change application state only in certain ways, for instance through interfaces to input devices like trackers or mouse/keyboard.

The Integrated Systems Lab is conducting ongoing research into creating a software toolkit for virtual reality on PC clusters. A very early version of the library was used to drive a head-mounted display in an human-factors study at ISL [11]. The toolkit's current incarnation is called Syzygy, is licensed under the GNU LGPL, and can be freely downloaded from the web [12]. While the software is still changing and many architectural decisions are as yet unmade, the lab has made a prototype available to help researchers port their visualizations to the PC cluster environment. This is the software backbone supporting the visualization of curved three-dimensional spaces described in this paper.

In order to produce graphics for ALICE, the six-sided VR theater at ISL, Syzygy controls six PCs running Windows NT with Wildcat 4210 graphics cards. Each PC produces an active stereo image which is viewed using standard shutter glasses. It is important that all the video images be produced in complete synchrony, with their vertical refresh locked together by means of an external *genlock* signal. The Wildcat card's ability to do this is critical for us, since active stereo across multiple displays will not work without such a feature.

Instead of trying to wrap existing libraries and hide the parallelism inherent in the PC cluster, Syzygy attempts to expose the parallelism and provide the programmer with tools to manage it. Since distributing the application tends to require fewer network resources than distributing data, the lab focused on building tools to help programmers write applications that can run synchronized across a network. As such, Syzygy does not address questions of load-balancing. The same application runs on each rendering node, with the only differences being the different point-of-view for each screen and the division of the nodes into a master and slaves. The master node is in charge of I/O and managing the synchronization of the group.

The application distribution framework for Syzygy assumes that a program can be conceptualized as a infinte loop with four phases: precomputation, data exchange between the master application instance and the slaves, post-computation, and synchronization. A simple API lets the programmer place the data-exchange and synchronization points wherever desired. The API also provides ways to determine which node is the master, to interface with a tracker, to configure the different nodes for rendering the different

walls of a cube, and to compute different viewing frustums based on head position.

To use the data-exchange API, the programmer first defines the format of the packet that will be broadcast from the master to the slaves. For some applications, this can be extremely simple, possibly even a single navigation matrix, as is the case with the visualization of three-dimensional geometries described in this paper. At the data-exchange point, the master application packs this data packet and sends it to each slave. At this point, the slave applications, conversely, receive the data packet and unpack it into local storage. The data-exchange API is built on a lower layer of software that handles the details of the network connection, the data format conversion between different machine architectures, and other chores.

The underlying synchronization infrastructure needs to accomplish two goals. The computers doing the rendering not only need to produce a sequence of frames in lockstep but also need to make sure that they are drawing a consistent world at each point in time. The Syzygy synchronization primitive is used to guarantee that the data sent by the master at a given data exchange step is used by all rendering nodes to draw the next frame; this ensures consistency. The same synchronization call is used to control graphics buffer swaps. Without external genlock, the control would be imperfect. Out-of-phase vertical refreshes on the displays would mean that buffer swaps occur in a slightly staggered fashion. However, external genlock of the display video cards eliminates this problem.

Synchronization in Syzygy is accomplished by a single API call, sync(). When a slave calls sync(), it sends the master a packet via UDP and then blocks while waiting for a response. When the master calls sync(), it waits until all slaves have requested release from the barrier and then broadcasts a UDP packet to them. The master now continues, as do the slaves when they receive the broadcast packet. In GLUT-based programs, like the mathematical visualizations described here, it makes sense to put the sync() call just before the call to glutSwapBuffers(). In practice, this synchronization method yields very good coherence even on graphics hardware without genlock capabilities. Experiments with highly animated scenes on a 2×2 video wall run by non-genlocked PCs show extremely small amounts of jitter. Indeed, the jitter is only perceptible to viewers within $3\,m$ of a $2\,m \times 1.5\,m$ image. When graphics hardware with genlock capabilities is used, as in ALICE, the synchronization is perfect. No jitter whatsoever can be perceived. One should note that this level of quality is achieved using 100Mbps Ethernet and not some more exotic networking technology.

Another challenge met by the Syzygy toolkit is in management of the distributed system. Our PC cluster is heterogeneous, with the rendering nodes running Windows NT, some nodes controlling interface devices running Linux and some running Windows, and additional support nodes running Linux. Syzygy provides a remote-execution daemon that works on either Windows

or Unix. It also provides an interface to send messages to running processes, which is used to provide a kill signal and also to make running applications reload their parameters. This simple interface provides a way to start, stop, and generally manage distributed applications on the PC cluster. To aid in running visualizations while enclosed in a VR theater, the lab has developed a Java interface to these functions that can run on a wireless handheld device. In this way, one can cycle through visualizations without leaving the CAVE and moving to a console; this is very convenient.

Recently, the ISL conducted an open house of its VR facility ALICE. Our visualization of three-dimensional hyperbolic space was one of the demos shown. The lab hosted almost 400 visitors over a two-day period. Eleven PC's were part of the distributed system, and that system stayed running the entire time. At any particular time, about 20 software components were running, dispersed across the system but cooperating in producing a visualization. Over the two-day period, over 10 000 seperate software components were started, connected to the system, and terminated once their work was over.

We now briefly describe the specific implementation in Syzygy of the mathematical visualizations described below. The only communication needed between the nodes is a description of the viewer's position and orientation. For the simplest geometries this is encoded as a 4×4 matrix expressing a projection from a representation of the curved space into ordinary Euclidean space (technically, into the observer's tangent space) [13]. (For the more complicated geometries slightly more information, perhaps a pair of matrices, will be needed [14].) During the pre-computation phase, the master node retrieves information from a wireless joystick and uses this to change the 4×4 matrix. The data-exchange phase distributes this matrix to all the slave nodes. In the post-computation phase, each rendering node applies the given projection and then applies the camera transformation appropriate for the point-of-view of the wall it controls. Finally, synchronization occurs and the loop repeats.

4 Mathematical Visualization of Three-dimensional Geometries

Topology is the study of deformable shapes. Topologically, a surface is anything that locally looks like a patch of the Euclidean plane. The surfaces of a round ball, a cylindrical can or a rectangular box are all topologically equivalent, since one could be deformed into another. This topological space is called the two-sphere and its most symmetric geometric realization is as a round sphere $\mathbb{S}^2$, with constant (positive) curvature.

The most interesting topological surfaces are those which, like the sphere, are compact (not infinite) but complete (with no edges). Topologically, the surfaces of a donut, an inner-tube, or a mug with a handle are also equivalent; this compact, complete space is called the two-torus. One might think that

the round inner-tube gives the nicest geometry for a torus, but, in fact, by going outside the realm of geometries obtained from surfaces embedded in our ordinary three-dimensional world, we can find a much more symmetric geometry for the torus.

From video-games, we are all familiar with the notion of a wrap-around screen. When a game token flies off the top or left edge of the screen, it reappears at the bottom or right edge and proceeds at the appropriate angle. The rectangular screen, with these identifications of opposite sides, becomes topologically a torus. And it clearly has (from the original rectangle) a flat geometry, that of the Euclidean plane $\mathbb{E}^2$.

Let us ignore non-orientable surfaces, like the Möbius band, where a left-handed object can become right-handed merely by traveling around some loop on the surface. It is then a classical result [15] that any orientable, compact, complete surface other than the sphere and torus must be a multiple torus. A multiple torus is a surface with more than one handle, obtained by connecting tori together.

We have seen that the sphere and the torus each admit nice geometries (round and flat, respectively). To put an equally symmetric geometry on a multiple torus requires the use of hyperbolic geometry, $\mathbb{H}^2$, the non-Euclidean geometry discovered in the 1800s by Lobachevskii, Bolyai and Gauss. It is not hard to explicitly write down a hyperbolic geometry (with constant negative curvature) for each multiple torus. (A deep result of Poincaré and Klein, called the uniformization theorem, says that any geometric shape for a surface is in fact conformally equivalent to one of the symmetric ones we have described.)

What about three-dimensional worlds, called three-manifolds? Locally, a 3-manifold looks like a block of Euclidean three-space. Our own world is part of some 3-manifold, though we don't know which one [16]. We know from Einstein's theory of relativity that our world is curved, albeit very gently. If we could explore our universe geometrically, unconstrained by physical limitations such as the speed of light, we might find that it closes on itself like a sphere, or a torus, or some more complicated possibility. Although the classification of surfaces is relatively easy, the classification of possible 3-manifolds remains an active area of mathematical research. Many interesting open problems remain, including the Poincaré conjecture. This says there are no "fake" three-spheres, and carries a million-dollar prize for its solution [17].

Again, the three-dimensional sphere admits a nice round geometry, called $\mathbb{S}^3$, obtained when it bounds a round ball in four-dimensional space. And a three-dimensional torus admits a flat geometry modeled on $\mathbb{E}^3$, obtained by identifying opposite faces of a cubical room. Again, "most" 3-manifolds have a hyperbolic geometry, $\mathbb{H}^3$.

But some three-manifolds admit no nice geometry. In general, one must first perform a decomposition (along two-spheres and two-tori) into so-called irreducible and atoroidal pieces. The famous Geometrization Conjecture of

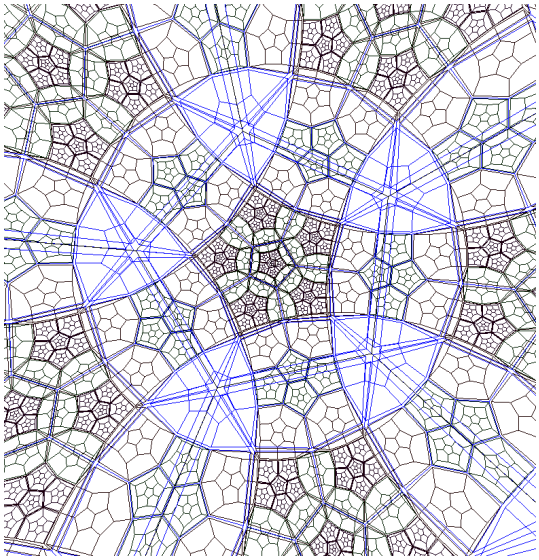

Fig. 2. The view of a sample hyperbolic manifold, with geometry $\mathbb{H}^3$, as seen on one wall of ALICE. This hyperbolic space is tiled by right-angled dodecahedra, meeting 8-to-a-vertex, just as Euclidean space can be tiled with cubes.

Bill Thurston says that each such piece should carry a nice geometry; it has been proved for many classes of manifolds, and seems likely to be true.

Thurston's list of possible geometries [18, 19], however, is an eight-fold one. Some manifolds admit one of the isotropic geometries $\mathbb{S}^3$, $\mathbb{E}^3$ or $\mathbb{H}^3$ mentioned above. Although hyperbolic geometry is in some sense again the generic case, for 3-manifolds (unlike for surfaces) even spherical geometry can be used for infinitely many different manifolds [20], and there are several Euclidean possibilities.

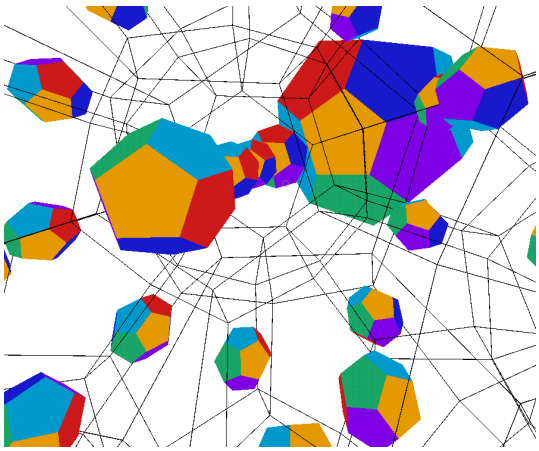

Fig. 3. The view of a sample spherical manifold, with geometry $\mathbb{S}^3$, as seen on one wall of ALICE. This space, formed by identifying opposite faces of a dodecahedron, is almost a counter-example to the Poincaré conjecture. An inhabitant of this space would see 120 repeating images of each object in the space, as in this regular tiling of the sphere by 120 dodecahedra.

For certain other 3-manifolds, however, the nicest geometry they can carry is nonisotropic, though still homogenous. There are two product geometries, $\mathbb{S}^2 \times \mathbb{E}^1$ and $\mathbb{H}^2 \times \mathbb{E}^1$, which are relatively easy to understand. The remaining three geometries on Thurston's list are the twisted geometries of the three-dimensional Lie groups Nil, Sol and $\widetilde{\mathsf{SL}}_2\mathbb{R}$. These Lie groups arise in many other areas of mathematics and physics, especially Nil, also known as the Heisenberg group.

It turns out that the isotropic geometries can all be modeled by the 4×4 projection matrices built into graphics systems like OpenGL, even though these systems were built only for $\mathbb{E}^3$. This fact was investigated by Charlie Gunn and Mark Phillips at the Geometry Center [21, 13, 22], and we demonstrated it in the four-sided CAVEs at SIGGRAPH'94 [23]. Jeff Weeks has written an exposition for computer graphics programmers, describing the surprisingly simple mathematics underlying these models of the isotropic geometries [24] and his software for exploring spherical, flat, and hyperbolic spaces at home on an ordinary PC is freely available [25].

The anisotropic product geometries can be implemented without too much more difficulty, using 5×5 matrices. The three twisted geometries do have matrix representations [14], but these are not necessarily efficient. Real-time exploration of these fascinating worlds presents a challenge of considerable mathematical importance.

Our ten-year experience from frequent public demonstrations of hyperbolic and spherical geometry in a conventional (4-walled) CAVE suggests that the illusion presented in that CAVE is not adequate to truly experience non-Euclidean geometry. Too much has to be explained to visitors, and too few questions are asked. Out of the corners of their eyes, people see the empty space above, and the dimly lit room to the rear of the convential CAVE.

The exotic geometries, being anisotropic, are more subtle. Physical or geometric laws there do depend on orientation, or, to misquote George Orwell, some directions are more equal than others. We look forward to implementing these in ALICE, since the sensation of navigating these anisotropic worlds will be totally different from any previous experience.

Our experience in the conventional CAVE has, however, proved that the gravity-based sense of an absolute "down" can be fleetingly overcome by the visual suggestion to the contrary. In an illusion without "above" and "rear" to confirm the gravitational "down" we expect to overcome the Euclidean reference frame, and truly immerse visitors in the exotic three-dimensional geometries.

References

1. Virtual Reality Applications Center. C6. Iowa State University, Ames. http://www.vrac.iastate.edu

2. Center for Parallel Computers. The PDC cube. Royal Institute of Technology, Stockholm. http://www.pdc.kth.se/projects/vr-cube
3. Fraunhofer IAO. HyPI-6. Fraunhofer Institute for Industrial Engineering, Stuttgart. http://vr.iao.fhg.de/6-Side-Cave/index.en.html
4. K. Fujii, Y. Asano, N. Kubota, and H. Tanahashi. User interface device for the immersive 6-screens display COSMOS. Sixth International Conference on Virtual Systems and Multimedia, 2000. http://wv-jp.net/got/media/COSMOS/
5. C. Cruz-Neira, D.J. Sandin, T.A. DeFanti, R.V. Kenyon, and J.C. Hart. The CAVE: Audio-Visual Experience Automatic Virtual Environment. *Communications ACM*, **35**(6):65–72, 1992.
6. C. Cruz-Neira, D.J. Sandin, and T.A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. *Computer Graphics (Proc. SIGGRAPH '93)*, 1993.
7. J. Allard, L. Lecointre, V. Gouranton, E. Melin, and B. Raffin. Net Juggler. http://www.univ-orleans.fr/SCIENCES/LIFO/Members/raffin/SHPVR/NetJuggler.php
8. A. Bierbaum, C. Just, P. Hartling, and C. Cruz-Neira. Flexible application design using VR Juggler. *SIGGRAPH*, 2000. Conference Abstracts and Applications.
9. Y. Chen, H. Chen, D. W. Clark, Z. Liu, G. Wallace, and K. Li. Software environments for cluster-based display systems. 2001. http://www.cs.princeton.edu/omnimedia/papers.html
10. G. Humphreys and P. Hanrahan. A distributed graphics system for large tiled displays. *IEEE Visualization*, 1999.
11. B. Schaeffer. A Software System for Inexpensive VR via Graphics Clusters. 2000. http://www.isl.uiuc.edu/ClusteredVR/paper/dgdpaper.pdf
12. Integrated Systems Laboratory. Syzygy. University of Illinois, Urbana. http://www.isl.uiuc.edu/ClusteredVR/ClusteredVR.htm
13. M. Phillips and C. Gunn. Visualizing hyperbolic space: Unusual uses of 4×4 matrices. In *Symposium on Interactive 3D Graphics (SIGGRAPH)*, **25**:209–214, New York, 1992.
14. E. Molnar. The projective interpretation of the eight 3-dimensional homogogeneous geometries. *Beiträge zur Algebra und Geometrie*, **38**(2):261–288, 1997.
15. J.R. Weeks and G.K. Francis, Conway's ZIP proof. *Amer. Math. Monthly*, **106**(5):393–399, 1999.
16. J.R. Weeks. *The Shape of Space*. Dekker, 1985.
17. Clay Mathematics Institute. The Poincaré conjecture. http://www.claymath.org/prizeproblems/poincare.htm
18. P. Scott. The geometries of 3-manifolds. *Bull. London Math. Soc.*, **15**:401–487, 1983.
19. W. P. Thurston. *Three-Dimensional Geometry and Topology*. Princeton University Press, Princeton, New Jersey, 1997.
20. E. Gausmann, R. Lehouq, J.-P. Luminete, J.-P. Uzan, and J. Weeks. Topological lensing in spherical spaces. 2001. http://www.arXiv.org/abs/gr-qc/0106033
21. C. Gunn. Visualizing hyperbolic geometry. In *Computer Graphics and Mathematics*, pp 299–313. Eurographics, Springer Verlag, 1992.
22. C. Gunn. Discrete groups and visualization of three-dimensional manifolds. *Computer Graphics (Proc. SIGGRAPH '93)*, 255–262, 1993.

23. G. Francis, C. Hartman, J. Mason, U. Axen, and P. McCreary. Post-Euclidean walkabout. In *VROOM – the Virtual Reality Room*. SIGGRAPH, Orlando, 1994.
24. J. Weeks. Real-time rendering in curved spaces. *IEEE Computer Graphics and Applications* **22**(6):90–99, 2002.
25. J. Weeks. Curved spaces software. http://www.northnet.org/weeks/CurvedSpaces/