Visualization of events from arbitrary spacetime perspectives

Michael D'Zmura*[a], Philippe Colantoni[b], Gregory D. Seyranian[a]

[a]Virtual Reality Lab, Univ. of California/Irvine, CA 92697

[b]Inst. d'Ingenierie de la Vision, Univ. Jean Monnet de St.-Etienne, France

ABSTRACT

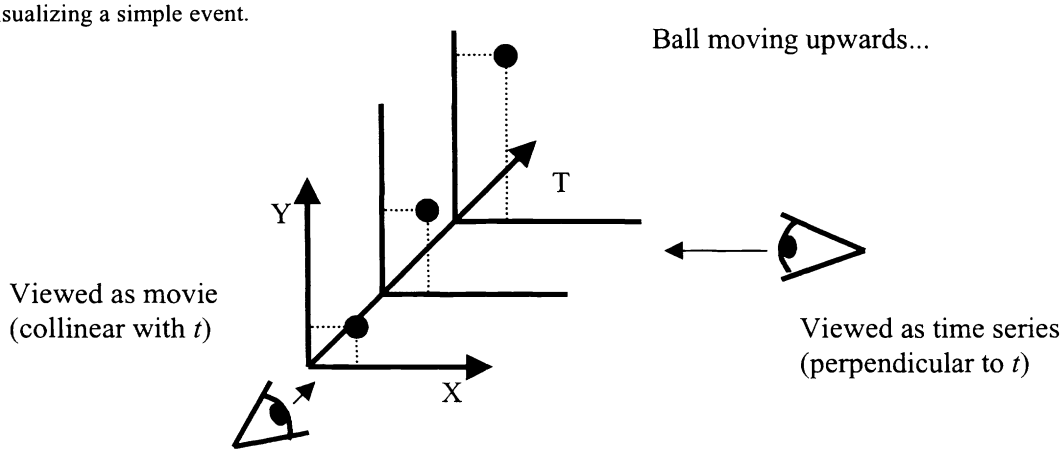
Methods for representing spacetime events as hyperobjects in four-dimensional (4D) space have been developed that let one examine events from arbitrary spacetime perspectives. Three-dimensional (3D) objects that move either rigidly or nonrigidly are extruded to create 4D hyperobjects. The boundaries of the 4D objects are represented using polyhedra, in much the same way that the boundaries of 3D objects may be represented using polygons. The user views one or more static, 4D hyperobjects using software which lets the user control in an interactive, realtime fashion the position and orientation of a 3D cross-section of the hyperobjects. The 3D cross-sections are rendered using standard techniques of 3D computer graphics. Extrusion of nonrigid 3D objects is useful for visualizing events that involve objects with time-varying shape. For instance, one can use nonrigid extrusion to visualize a character animation, in which multiple frames of a walking humanoid character are used to create a single, static hyperobject that represents the entire animation. With these methods, one need no longer be limited to watching a movie in which time is a hidden axis. Rather, one can view and alter events immersively from arbitrary vantage points.

Keywords: visualization, graphics, 4D, spacetime, event, immersive, polyhedron, animation, high-dimensional

1. INTRODUCTION

The display of time-varying events generally takes one of two forms, as illustrated in Figure 1. These forms are the movie and the time series. With the movie, objects are viewed through one or another form of visual projection on a flat, 2D surface. The way that the shapes, positions and identities of the objects change in time is not made available to the viewer all at once. Rather, the objects' configurations at individual moments of time are shown sequentially. If one considers three spatial axes X, Y, and Z and a time axis T to define a four-dimensional space, then it is the time axis that is "hidden" in the movie rendering process. At any one moment in time, the viewer of the movie is presented a 3D cross-section that lies orthogonal to the time axis. The second form of presentation is the time-series, in which the complete variation in time of the objects is presented simultaneously. The time-series display of spacetime events uses a cross-section of the four-dimensional space spanned by the X, Y, Z, and T axes that includes the last, temporal axis.

Figure 1. Visualizing a simple event.



* Correspondence: Email: mdzmura@uci.edu; WWW: http://www.cvr.uci.edu/dzmura; Telephone: 949 824 4055; Fax: 949 824 2663

In *Visual Data Exploration and Analysis VII*, Robert F. Erbacher, Philip C. Chen, Jonathan C. Roberts, Craig M. Wittenbrink, Editors, Proceedings of SPIE Vol. 3960 (2000) • 0277-786X/00/$15.00

Arbitrary perspectives on time-varying events may be provided by displaying appropriate 3D cross-sections of 4D spacetime. We have developed such display techniques in prior work on the interactive, immersive display of 4D virtual environments.[1,2] This paper sketches these techniques and their modification to allow for the display of arbitrary spacetime perspectives on events.

2. DISPLAY METHODS

The 4D computer graphics methods developed in the UC Irvine VR Lab display, at any one moment of time, a 3D cross-section of a 4D environment. The choice of 3D cross-section to be displayed depends on the position and orientation of the observer within the 4D environment. The position and orientation of the observer depend, in turn, on user interface activity. An interface which depends on mouse and keyboard activity in a manner reminiscent of 3D action games like Quake™ has been implemented for personal computers; we have also implemented an immersive interface for use with a head-mounted-display, head orientation tracker and pinch gloves.

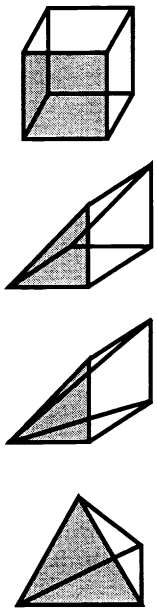


Figure 2. 4D graphics primitives.

The graphics methods use polyhedra rather than the polygons prevalent in 3D computer graphics. Primitive polyhedra include the hexahedron, triangular prism, pyramid, and tetrahedron, as pictured in Figure 2 at left. These are texture-mapped in one of two ways. The ideal way is to use 3D texture mapping. With 3D textures, each element within the volume of a polyhedron has a corresponding volumetric texture element. 3D texture-mapping works well in an interactive, real-time setting when supported by graphics hardware, and we have implemented it in software that runs on an SGI Onyx2™. However, most computer graphics hardware does not support 3D texture mapping. A simple way to texture-map the volume enclosed by a polyhedron with 2D textures is to stack the 2D texture to fill space.

2D texture-mapping is illustrated in Figure 3 below. A 2D image texture (left) is stacked, notionally, to occupy the volume within a polyhedral primitive (right). The 4D rendering process involves the determination of a 3D cross-section that produces polygonal cross-sections of the polygons. These polygonal cross-sections are projected onto the front face of the polyhedron to provide texture coordinates for 2D texture mapping, a process that is implemented well in commodity 3D graphics hardware.

The 4D graphics pipeline is shown in Figure 4 at the top of the next page. A 4D environmental database provides geometric information concerning primitive polyhedral elements and texture information, including texture images and information on how to position these properly on the corresponding polyhedra. Multitexturing hardware lets one designate more than one texture per polyhedral target; second textures are used most commonly for lightmaps, which simulate patterns of light and shading.

The position and orientation of the user within the 4D environment are used in several ways in the rendering process. The first is in culling polyhedra that are not visible, either because they lie behind the user or because they face away from the user. Remaining polyhedra are then subject to a cross-section computation that depends on user position and orientation. The cross-section involves the reduction of polyhedra into polygons and the calculation of 2D texture coordinates; these may be handled using standard methods of 3D computer graphics. Polygons with greater than three sides are triangulated, transformed from the global coordinate system to the local one erected about the user, and then sent to the 3D graphics pipeline, which we have implemented using OpenGL.[3]

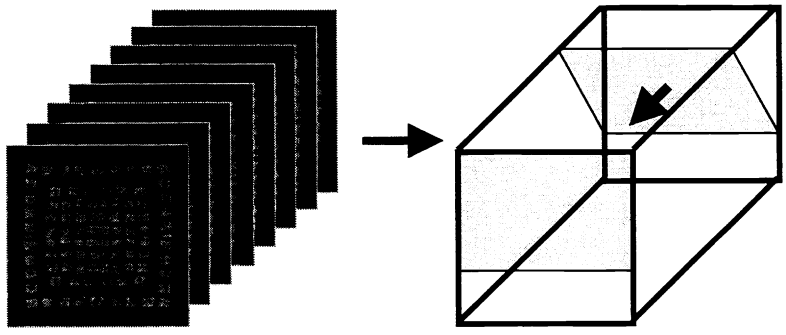
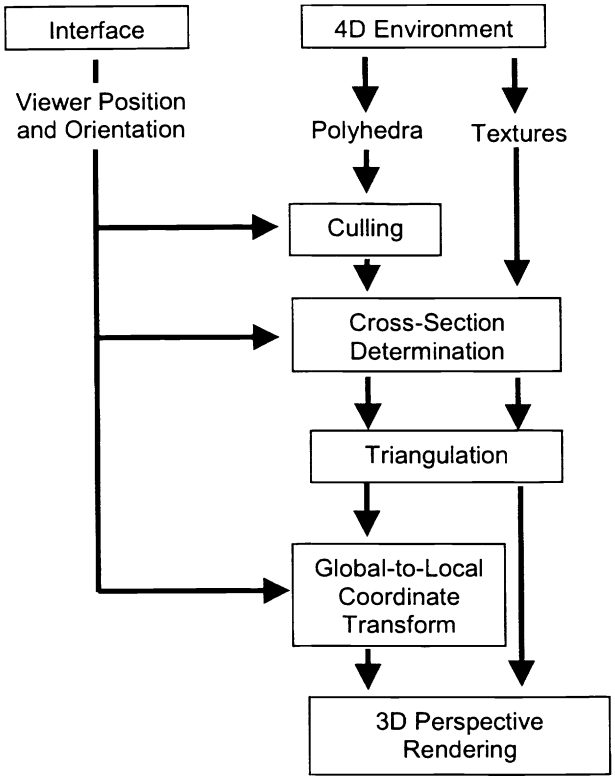


Figure 3. Stacking a 2D texture image to create a 3D texture. Polygonal cross-sections of polyhedra (right) are projected onto the front face to determine texture coordinates.

Figure 4. 4D computer graphics pipeline implemented with a 4D-to-3D cross-section.



The dominant feature of the user interface is its provision for motion within four spatial dimensions. As pictured in Figure 5 at right, X, Y, Z and W axes span the 4D space. One may move left and right along the X axis, up and down along the Y axis, back and forth along the Z axis, and *nim* and *bor* along the W axis. In the 4D virtual environments, gravity acts along the vertical Y axis to pull the user down onto a floor spanned by the three axes X, Z and W. Rotations of user orientation include change in pitch (rotation in the YZ plane), change in yaw (rotation in the XZ plane), and change in *zaw* (rotation in the ZW plane). Software provides collision detection to simulate solid floors, walls and other objects through simple ray-tracing.

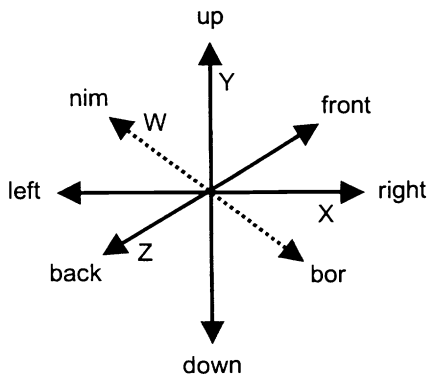
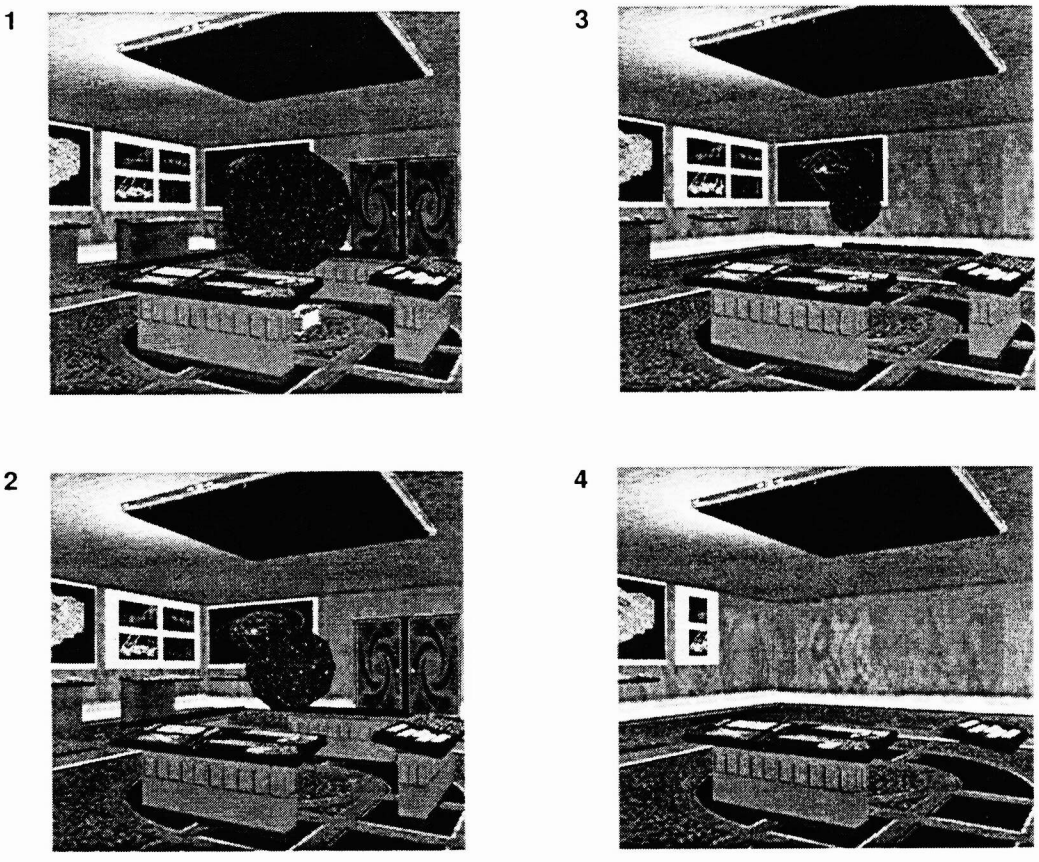


Figure 5. Directions along the four spatial dimensions.

3. MODELING METHODS

These 4D graphics methods have been used in an engine to display virtual environments that resemble 3D action games in their content and interactivity; users are able to walk, run, jump, fly and swim around the environments. A sample scene from such a 4D environment is shown in Figure 6 at the top of the next page. This environment and others were created using a 4D modeling and level-editing program, developed in the UC Irvine VR Lab, that we call *HyperEdit*. The program provides a number of primitive objects. These include the single polyhedra shown in Figure 2 as well as a number of primitive hyperobjects: the hypertetrahedron, hyperbox, hypercone, hypercylinder and hypersphere. Several of these are

Figure 6. Four screen shots illustrating rotation in the ZW plane (zaw), taken in sequence 1, 2, 3, 4. Visible are the disappearing spherical cross-sections of the central black hypersphere, and a door that passes out of view as the view orientation changes.



created (notionally) through an extrusion process. For example, the hypercylinder is built by taking the familiar sphere and extruding it along the axis orthogonal to the 3D space spanned by the sphere. The hypercone is made similarly; one shrinks the sphere along its extrusion path so that the sphere is a single point when the vertex is reached.

The basic operations that one can perform on these primitives are several in number. They include geometric transformations like scaling, translation and rotation. They also include the specification of material properties: color, specularity and transparency. Texture mapping is handled by a texture-mapping editor, which handles the positioning, scaling and rotation of textures and lightmaps on polyhedra. An object hierarchy editor helps to orchestrate the editing process through the provision of grouping and ungrouping. Finally, a number of functions help with level-editing, such as the specification of particular polyhedra as floors or walls or doors, *etc.*, and the attachment of subset-of-C language scripts for the dynamic objects in the environment.

Advanced editing operations include the tesselation of polyhedra into tetrahedra. Tetrahedralization aids with vertex-level editing and with the creation of terrain in 4D. A hole editor helps with the construction of holes of arbitrary shape that may be extruded so as to pass through one or more polyhedra. A 4D animation editor helps to coordinate keyframe animations. Finally, the modeling and level-editing program *HyperEdit* provides an extrusion editor that is able to take, in the simplest case, a 3D object represented in 3D Studio Max 3DS format and extrude it along an arbitrary path in 4D space. This extrusion editor that may also be used to create static 4D objects from 3D animations, and it is this feature that we use to visualize events from arbitrary spacetime perspectives.

4. EVENT VISUALIZATION

We use the 4D computer graphics methods embodied in the modeling and level-editing software to construct 4D static objects from 3D animations and view these from arbitrary spacetime perspectives. Figure 7 (below at right) and Figure 8 (next page) illustrate the display of an animation of a walking person, created in 3D Studio Max™ and exported as a 3DS file. As shown in Figure 7, the view orientation is initially collinear with the time axis T (0 deg), which is identified with the W axis of Figure 5. A single 3D cross-section--at some particular time value--resembles a frame of a movie. Such a frame is shown in the top left panel of Figure 8 (labeled zaw = 0 deg). One may rotate the viewpoint about the object, and illustrated in Figure 7 is a rotation in the ZT plane that changes the zaw angle. When the viewpoint is rotated is rotated a full 90 deg, the view orientation is collinear with the spatial Z axis and perpendicular to the T axis. The entire 3D animation, *sans* Z-axis, is visible from this "time-series" orientation. Intermediate rotation values result in displays like those shown Figure 8 for zaws of 40, 50, 60, 70 and 80 deg. While the Figures depict a change in viewpoint orientation that corresponds to changing zaw, there is no restriction on the viewpoint orientations that may be taken to view the 4D solid. The set of possible viewpoint orientations corresponds to a hypersphere.

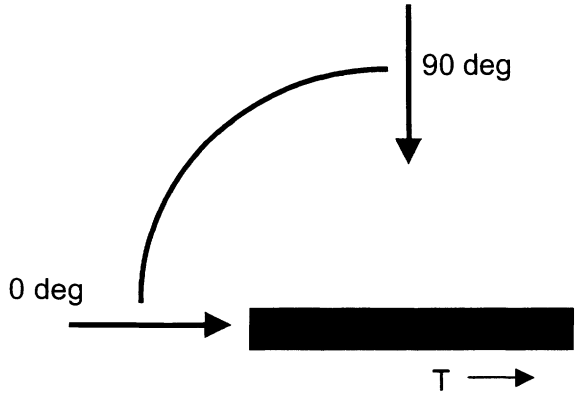


Figure 7. Animation (solid bar) viewed from movie-like viewpoint (zaw = 0 deg), in which the time axis T is the hidden axis, and from a time-series-like viewpoint (zaw = 90 deg) in which a spatial axis is suppressed.

A requirement by the existing software for the 4D extrusion of the animation created in 3D Studio Max is that vertices in adjacent frames be in one-to-one correspondence. The 4D software converts corresponding triangles in adjacent frames to triangular prisms; each prism inherits the 2D texture of its initial (earlier) face.

5. POSSIBLE AREAS OF APPLICATION

There are several possible application areas for visualization of this sort. One is the visualization of 3D flows like those about airfoils and hulls. An isocontour surface, found for the flow at some particular moment in time, can be represented by a 3D shape bounded by polygons. These polygons can be extruded to join time-adjacent frames of an isocontour movie to create a single 4D object that represents the spatial properties of the flow at all times. A second is the visualization of paths is 3D. For instance, one may visualize various problems in air traffic control as 4D spacetime structures. Shapes that change in time, such as the computer animation of Figure 8, are candidates for such visualization. In the scientific domain, one may wish to view time-varying molecular configurations or, more generally, chemical reactions, in such a fashion. Finally, one should not ignore the somewhat simpler problem of visualizing events in two spatial and one time dimension. Extruding line-segment boundaries in 2D images to create polygons which may then be viewed from arbitrary spacetime perspectives using standard 3D computer graphics methods may reveal structure not available from movie or time-series viewpoints.

ACKNOWLEDGMENTS

We thank Barb Krug, Geraldine Laurent, and Gordon Richins for their assistance with digital artwork. This work was supported by NSF DBI-9724595 and NIH EY10014 and by a fellowship from the Région Rhône-Alpes, France, to Colantoni.

REFERENCES

1. M. D'Zmura, P. Colantoni, & G. Seyranian, "Virtual environments with four or more spatial dimensions," *Presence*, submitted.
2. G. Seyranian & M. D'Zmura, "Search and navigation in environments with four spatial dimensions," *Perception & Psychophysics*, submitted.
3. Woo, M., Neider, J., Davis, T. & Shreiner, D. (1999) *OpenGL Programming Guide.* 3rd Ed. (Reading, MA: Addison-Wesley).

Figure 8. Animation of person walking that is viewed from a range of viewpoints in spacetime suggested in Figure 7.