

Multi-Perspective Images for Visualisation

Scott Vallance

Paul Calder

School of Informatics and Engineering
Flinders University of South Australia
PO Box 2100, Adelaide 5001, South Australia

{vallance,calder}@infoeng.flinders.edu.au

Abstract

This paper describes the concept, and previous realisations, of multi-perspective images in nature, art and visualisation. By showing how distortions have been used for visualisation, it motivates the use of multi-perspective images, which are similar in effect to object based distortions. A new API being developed to facilitate multi-perspective rendering is presented, with particular reference to its suitability for interactive applications. This API is demonstrated in a simple example of a multi-perspective image, where five faces of a cube are shown at once. Further work necessary to make multi-perspective images for visualisation a reality is discussed.

1 Introduction

We define a multi-perspective image as multiple views of a single scene from different perspectives. These views are joined seamlessly to form an image that is a coherent whole, without discrete subsections. The concept of continuously joined views is not new. The idea has been realised in many different forms; for example, reflections on curved objects and lens effects can constitute natural multi-perspective views. Apart from the inherent aesthetics of the concept, we seek to explore its use as a visualisation tool.

The motivation for exploring multi-perspective rendering comes from the limitations of human sight in a 3D world. The view we can see from our two eyes can be in many situations very limited. We cannot see over tall objects, nor through opaque objects, and we cannot see forwards and backwards at the same time. In a virtual world, the first two limitations can easily be dismissed – we can fly over any object, and turn what we wish transparent. The third limitation is what this paper specifically addresses. We seek not just to be able to see forwards and

backwards at the same time, but to see from different places – without having multiple separate views.

Real world multi-perspective images are used as visualisation aids already. An obvious example is curved mirrors on roads, which provide views of both directions at T-junctions. We wish to allow for even more flexibility in virtual worlds, so that visualisers can effectively look out from arbitrary curved mirrors.

2 Previous Work

Previous work falls into three sections: work relating to viewing from multiple viewpoints continuously, techniques for distorting objects, and methods for rendering reflections on curved objects. While the previous work on multi-perspective images gives a clearer understanding on the nature of the images, it is the work on distortions that prompts our visualisation focus. Previous work on reflections on curved objects relates to the practical implementation of multi-perspective rendering.

2.1 Multi-Perspective Images

In art, Chinese landscape painting, Cubism, and M. C. Escher have explored the idea of multi-perspective images. Chinese landscape paintings contain different focuses, or sub-images, which are seamlessly joined. These paintings are similar to the panoramas used for cartoon drawing and image resynthesis, as is discussed in Chu and Tai (2001). For example, in Figure 1 the perspective shifts from left to right, following the path of the stream.

M. C. Escher depicted a view with multiple vanishing points, or perspectives, in his work “High and Low”

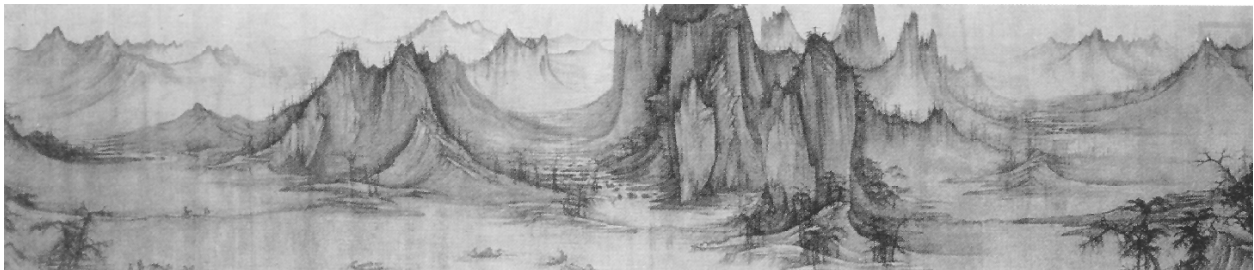


Figure 1: “Fisherman’s Evening Song” by Xu Daoning, Circa 11th Century.

Escher (1992). This work, see Figure 2, has five different vanishing points: top left and right, centre, and bottom left and right.

While the automatic generation of an image like this from 3D geometry may not be practical, it illustrates the concept and the aesthetic potential.

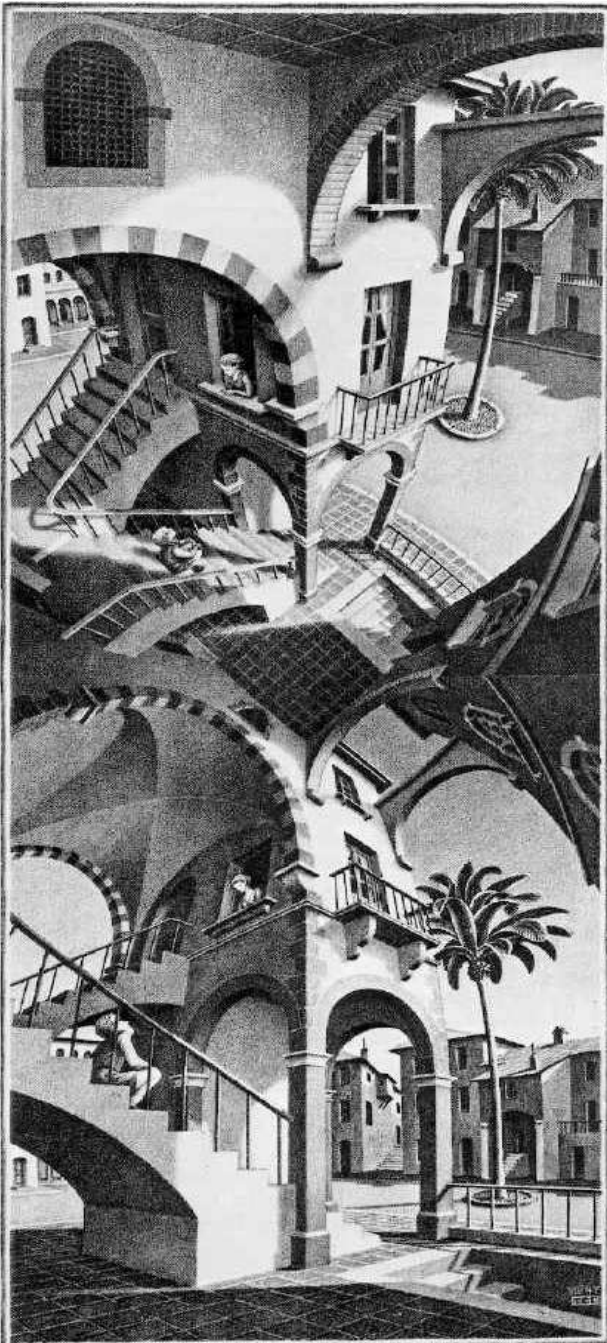


Figure 2: "High and Low" by M. C. Escher

Hand-drawn and computer-generated panoramas with multiple points of view have been used as the basis of image resynthesis. Cartoon animation from panoramas is an early example of resynthesis, which is explained and adapted to computer-generated images in Wood, Finkelstein, Hughes, Thayer and Salesin (1997). When a small subsection of the panorama is viewed it approximates a standard single viewpoint. If these subsections are taken over a path on the panorama, they give the appearance of motion when sequenced into an animation. This is because the viewpoint shifts continuously in a multi-perspective panorama.

Rademacher and Bishop (1998) present more generalised multi-perspective images as the basis for resynthesis, the advantage being a variable level of sampling without multiple separate images. The paper calls these Multiple-

Centre-Of-Projection images. One section of the panorama can be from close to a portion of the image, giving a high sampling for that area, while others are further away and capture more of the object. Moving a virtual camera through the scene generates the multi-perspective panorama. At regular intervals the camera captures a single line of pixels for the final panorama. These lines, either rows or columns, are placed next to each other, so that the viewpoint smoothly changes from one to the next. This is effectively a virtual strip camera, which is explained below. Figure 3 shows a Multiple-Centre-Of-Projection image of an elephant. The virtual camera path goes from one side of the elephant to the other, and we can simultaneously see both sides and the front.



Figure 3: Multi-Centre-Of-Projection image of an elephant taken from Rademacher and Bishop

Strip cameras are used in surveillance and mapping. These cameras have a continuous roll of film that slides past a slit as a picture is being taken. The camera may be moved whilst the shooting, providing a change in point of view from one section of the film to another. If used from a moving plane these cameras can capture a long section of curved earth as if it were flat. The cameras have also been used for artistic purposes, capturing strange and unusual images, such as in Robert Davidhazy's work (Figure 4).



Figure 4: A strip camera image showing a head from all sides taken from Davidhazy (2001)

In Loffelmann and Groller (1996) the idea of rendering from multiple viewpoints with ray tracing is examined. By developing an extended camera for ray tracing, the authors present multi-perspective images with visualisation as an application. Essentially an extended camera is a set of 3D rays. Each ray has a starting position and a direction, and these rays are used by a conventional ray-tracer to draw the scene. Unlike a

conventional ray-tracer, these rays do not necessarily originate from the same point.

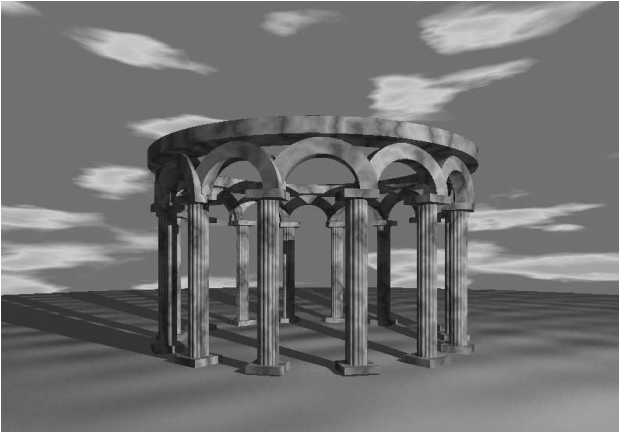


Figure 5: A conventionally rendered set of columns

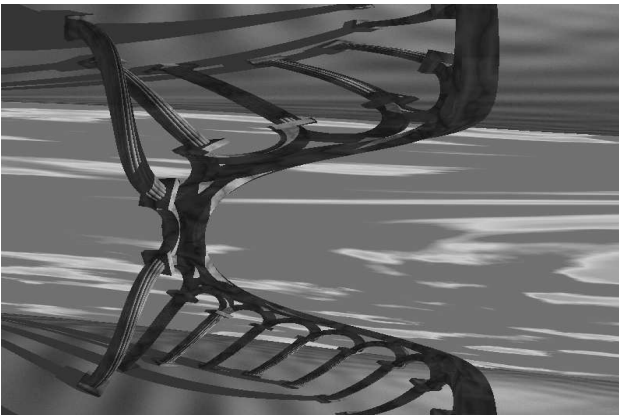


Figure 6: Columns rendered from a torus surface

Figures 5 and 6, both taken from Loffelmann (2001) show the extended camera ray tracing in practice. The first image is rendered normally, however the second is rendered out of a torus camera. The rays used all start on the surface of the torus and point in the direction of the surface normal.

The extended camera is comprised of three sections: an object space transformer, a picture space transformer and a parameter space transformer. The object space transformer is a function that returns the 3D start position of a ray given its corresponding index on the 2D screen. This function defines a surface in 3D, and the scene is rendered “out of” this surface. Defining a multi-perspective image from a rendering surface is a convenient representation, so we have adopted it for our work. A multi-perspective image could otherwise be defined as a camera path, such as in Rademacher and Bishop (1998) described previously. However, the camera path can be described as a surface, and vice versa.

2.2 Distortions

Distortions of the data, while being fundamentally different in implementation than rendering multi-perspective images, highlight the potential and applications for multi-perspective images. Both seek to present the data in a changed way so that previously unseen properties become apparent.

Distorting an object to view it better is most commonly illustrated with the Mercator projection. This takes a sphere, generally the earth, and transforms it to a 2D map. In this map directions are conserved, though sizes are not, to allow for easy sea navigation. Although the distortion of size is an artefact, it allows for a better understanding of some aspects of the globe. This is ultimately the point of distorting the data, either directly or by rendering through a curved surface – to better illustrate certain properties of the data.

In Hurdal, Bowers, Stephenson, Sumners, Rehm, Schaper and Rottenberg (1999) a scan of a brain was distorted into a nearest approximation flat projection. This allows a better appreciation of the layout of the brain from medical imaging, with the intention of improving surgical planning. This type of distortion is highly dependant on the nature of the data, and does not translate well to the visualisation of arbitrary scenes.

Distortion Orientated Displays are a general visualisation tool based around the distortion of data. These displays seek to show detail and context simultaneously. The general problem is that when detail is shown, much of the screen is filled with that detail. If the surrounding data is shown at the same level of detail, it would not fit on the screen. To accommodate this, in a distortion view there is a region of focus at a certain detail level, which smoothly transitions to a region of context at a lower detail level. This can be seen in Smith (1997). Using a distortion called a frustum display, the author was able to achieve levels of detail sufficient for a city level road map, whilst showing the context of the whole of Australia.

In 3D, both Keahey (1998) and Winch, Calder and Smith (2000) expanded the idea of distortion orientated displays to allow for regions of zoom – regions where the scene data was expanded to a larger size. These prove useful for highlighting sections of particular detail in a scene without zooming and therefore cutting out periphery data.

The idea of detail and context is to have at least two different perspectives on the data. In this case the perspectives are not literal changes in viewpoint, but in operating conditions. In Vallance and Calder (2001) the idea of distorting a mainly planar world onto the inside of cylinder was examined. This was proposed, in the application of virtual maze navigation, so that two different perspectives on the maze could be simultaneously realised: a local view of the undistorted maze walls, and a navigational view of the distant maze perpendicular to the users viewpoint. Figure 7 shows an example of the maze distortion.

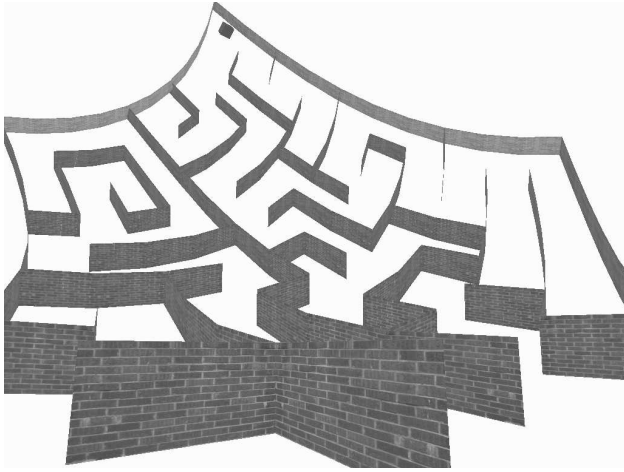


Figure 7: A maze distorted in a cylindrical fashion to show context

The cylindrical distortion works only because the data lies mainly on a single plane. For arbitrary 3D data it is unclear how the mapping onto a cylinder would be helpful.

2.3 Reflections on Curved Objects

Reflections on curved surfaces are a natural form of multi-perspective image. Computer graphics has long been interested in realistically rendering virtual scenes including reflections. Research into generating reflections on curved objects, especially for real-time graphics, shows how multi-perspective rendering can be implemented. However reflections on curved objects are a subset of multi-perspective images. Moreover they need not be entirely accurate (they need only look appropriate) and form only a small portion of the screen. Multi-perspective images for visualisation have different accuracy requirements and take up more of the screen.

Ray tracing is the most direct way to render reflections from curved surfaces. When a ray intersects a reflective surface it bounces off in a direction determined by the angle of intersection with surface. Reflections drawn in this manner are accurate, however ray tracing tends to be very slow. Various methods exist for accelerating the ray tracing. These methods range from reducing the number of ray-surface intersection tests with hierarchical space subdivision, to parallelising the calculations. Without massively parallel hardware it seems unlikely real-time ray tracing will be achieved soon, the problem is discussed in Jansen (1993).

Environment mapping was initially suggested by Blinn Blinn and Newell (1976), and is often used to approximate curved reflections for real-time graphics. When a scene is rendered from a particular viewpoint, the light coming into that point is sampled. With enough samples for a particular viewpoint, the technique can approximate the colour of any ray shot from that point. A popular implementation of environment mapping involves rendering to the six faces of a cube centred on a point. When used for generating reflections, this centre point is the centre of the reflective object. For each ray that bounces off the reflective object, a ray is shot from

the centre point in the direction of the reflected ray into the cube. This approximation moves each reflected ray to the centre of the cube. For scenes where the reflected objects are far away from the reflective object the technique works well; otherwise the approximation is obvious. For the purposes of multi-perspective images environment maps are not sufficiently accurate. The light is sampled at only one point, making it unsuited for multi-perspective, or multi-viewpoint, rendering.

An extension to the concept of environment mapping is proposed by Cho (2000) to provide more accurate images. Instead of simply sampling the light coming into a point, a depth-mapped image is calculated for each of the six cube faces of the environment map. Reflection rays can then be traced into the 3D depth map, without needed to approximate the start point of the ray. This technique amounts to ray tracing the scene, though through a modified representation of the geometry (the depth map) that provides a significant performance enhancement in some cases. Static scenes are required for this technique, as otherwise the depth maps need to be recomputed at each frame, which is very expensive. In a multi-perspective image for visualisation, the surface will be moving and not static in relation to the scene, so extended environment maps would be unsuited.

Another type of technique based on environment mapping is presented in Hakura, Snyder and Lengyel (2001). In this technique, layers of environment maps are used. Different environment maps may be used according to the viewer's location and direction of view, to avoid the failings of standard environment maps. Once again, this technique requires a static scene relative to the reflective object to be effective. Rendering these reflections from a series of stored images is actually image based rendering. Other image based rendering techniques, such as the Lumigraph (Gortler, Grzeszczuk, Szeliski and Cohen 1996) can be useful for rendering reflections. These techniques represent the light in the space of a scene, and are sampled with 2D slices to generate a particular view. The drawback of such a system is that it is currently not used for many visualisation purposes, has a large memory overhead, and needs an unchanging scene.

The most intriguing approximation of rendering reflections on curved surfaces is described in Ofek and Rappoport (1998). In this method, objects are transformed by the reflective surface so that they may be rendered from a single viewpoint. In essence the data is distorted to an approximation of how it will look after being viewed from a reflective surface, and then rendered. It requires an appropriate tessellation of both reflective surface and scene object so that lines that should now appear curved do. The performance of this technique is sufficient for real-time rendering of moderate scenes. The technique works on standard polygon scenes making it suited for visualisation tasks, and easy integration into current applications.

3 An API for Multi-Perspective Rendering

In developing an API to facilitate rendering from multiple perspectives the key concerns were:

- To separate, as much as possible, technique from calling interface.
- To allow easy integration into existing visualisation applications.
- To allow for expansion of functionality and techniques.

To achieve interactive rendering of multi-perspective images a variety of different techniques will need to be evaluated. The API is designed to be flexible enough so that different techniques can be used without major changes to the visualisation code that uses the API.

One of the most popular API's for 3D graphics is Silicon Graphics Inc.'s OpenGL. It provides a clear and powerful set of instructions for building graphical programs. The specification is described in Segal and Akeley (1998). This API forms the inspiration for our design, and many of the commands resemble OpenGL syntax. By basing the API on OpenGL it should allow for easy integration into existing visualisation programs that use OpenGL.

The multi-perspective rendering API covers two main tasks: describing a surface to render from and a scene to be rendered. Two abstract classes define the base level interfaces provided by the API for these two tasks.

Geometry: Describes the scene to be rendered.

Functions	Description
Begin()	Start of a geometry description block. Vertex, normal and colour may only occur between a Begin() and an End()
End()	End of a geometry block
Vertex3f(float x, y, z)	Place a vertex of a triangle in the scene with the current normal and colour
Normal3f(float x, y, z)	Specify the current normal
Color3f(float a, r, g, b)	Specify the current color
<i>Other</i>	Various other commands to specify textures and other graphical properties

Surface: Describes the surface to render from.

Functions	Description
BeginSurface()	Start specifying a surface, vertex and normal commands may only appear between a Beginsurface and Endsurface

EndSurface()	End a surface specification block
Vertex(float x, y, z)	A vertex in the surface, the exact meaning is dependant on the type of rendering technique
Normal(float x, y, z)	Current normal for each vertex
Viewport(float l, r, t, p)	The viewing dimensions of the screen. Determines how the surface is mapped to the screen

These classes are accessed through the Renderer class, which is a conglomeration of all the interface functions so that they may be accessed without reference to particular Geometry or Surface objects. The rendering technique implemented in the Renderer class, which uses specific Geometry and Surface objects to generate an image. This is done so that the interface looks and feels more like OpenGL.

These abstract classes are inherited by specific implementations. For instance, the rendering of multi-perspective images can be done with a ray tracer, the surface can be specified as a set of rays and the geometry as a set of triangles. This defines three classes that inherit from the base classes: RayTraceRenderer, RaySet and TriStore. Figure 8 shows the relationships of the classes. The solid arrows indicate inheritance and the dashed arrows show usage.

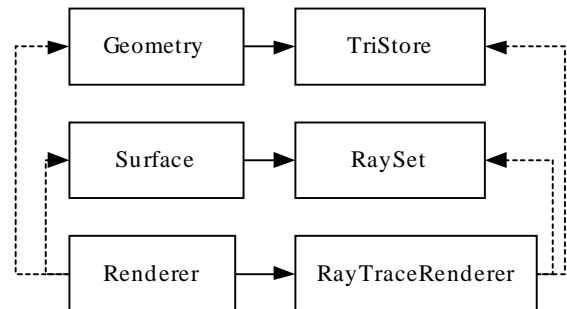


Figure 8: Diagram of base classes and a raytracing renderer

In the surface class, like the geometry class, the main mechanism for passing information to the lower level rendering technique is the vertex and normal commands. This symmetry is deliberate, so that, with the right technique, a surface can be used to render from or to draw with little modification. Also this OpenGL-style immediate-mode specification of surfaces suits an interactive application where the surface may be changing from frame to frame.

The types of rendering surfaces that can be represented easily with only vertex and normal commands is similar

to the types of conventional surfaces easily specified with those primitives. Simple point-based geometry is analogous to a simple rayset surface, where vertex and normal commands denote a ray's origin and direction.

Triangular patches are probably the most common form of 3D geometry and rendering surfaces can also be described in this manner. While polygon patches are not curved, it is common practice to make them appear so by interpolating properties between the vertices of the constituent triangles. By taking the normal values at a particular point as denoting the direction of a ray origin at that point, and interpolating these normals between the vertices, a graduated or curved surface is approximated.

Parametric surfaces are also easily specified with vertex and normal commands, with vertices interpreted as control points of a patch. It is hoped that these different ways of specifying surfaces will be sufficiently rich for most purposes.

To illustrate how the API works here is an example based on using the RayTraceRenderer described in Figure 8:

```
RayTraceRenderer rtr;

void SpecifySurface() {
    rtr.Viewport(0,0,1,1);
    rtr.BeginSurface(); {
        rtr.Normal3f(0.0,0.0,-1.0);
        rtr.Vertex3f(0.0,0.0,0.0);
        rtr.Normal3f(0.0,0.0,1.0);
        rtr.Vertex3f(0.0,0.0,-1.0);
    } rtr.EndSurface();
}

void SpecifyScene() {
    rtr.Begin(); {
        rtr.Normal3f(0.0,0.0,1.0);
        rtr.Vertex3f(1.0,0.0,-1.0);
        rtr.Vertex3f(-1.0,0.0,-1.0);
        rtr.Vertex3f(0.0,1.0,-1.0);
    } rtr.End();
}
```

These functions, appropriately called, trace two rays into a scene comprising of a single triangle. The RaySet and TriStore objects are hidden by the RayTraceRenderer and are accessed through that class.

4 Raytracing Implementation

As a first case implementation a ray tracing algorithm was used, with rendering surfaces defined as a set of rays. This implementation was used to render a simple scene with a cube a tiled floor, shown here in Figure 9. While basic, the scene has some properties that make it an appropriate demonstration of multi-perspective rendering. First the cube has six numbered sides, in a normal perspective view at most three sides can be seen at once, due to self occlusion. Second, the tiled floor provides a reference to the effect of the rendering surface that is easily perceived.

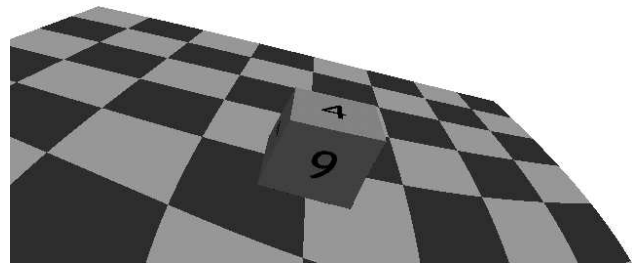


Figure 9: The cube scene

To simultaneously show five sides of the cube, a curved surface is placed over the scene. This surface is constructed initially from 16 control point Bezier patch, which is decomposed in a set of rays. The surface can be seen here in Figure 10, and is roughly hemispherical in nature. Figure 11 shows the cube scene rendered through the surface, with five of the cube's faces visible.

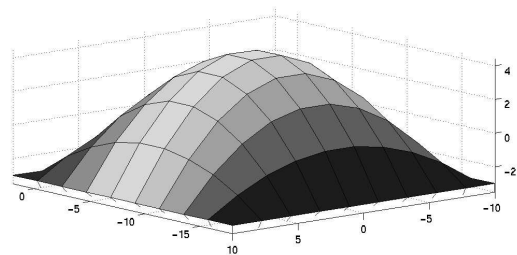


Figure 10: A surface for visualising all sides of a cube

The surface spans the cube scene, with the rays pointing inwards in a direction normal to the surface. The generation of the rays from the Bezier equations and control points is a significant performance cost in itself.

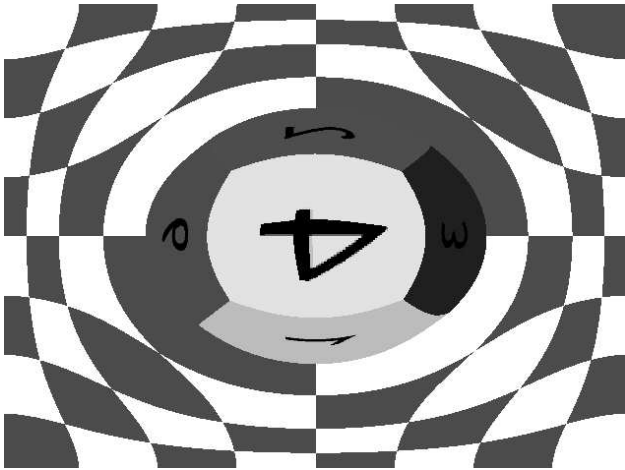


Figure 11: The cube scene rendered from the surface in Figure 10

The surface was constructed by manually entering appropriate values for the Bezier control points. This is undesirable from an interaction point of view, one of the key aims of our research. As a first experiment in the interactive specification of rendering surfaces, a 1-dimensional control is proposed. A slider dictates the amount of perspective on a normal rendering surface. A zero value corresponds to an orthographic projection, and the positive values correspond to an increasingly severe perspective projection. Negative values correspond to a 'reverse' perspective projection where distant objects seem larger, and closer objects smaller. Figures 12 through 14 show a perspective view, an orthographic view and finally a 'reverse' perspective, respectively.



Figure 12: A perspective projection

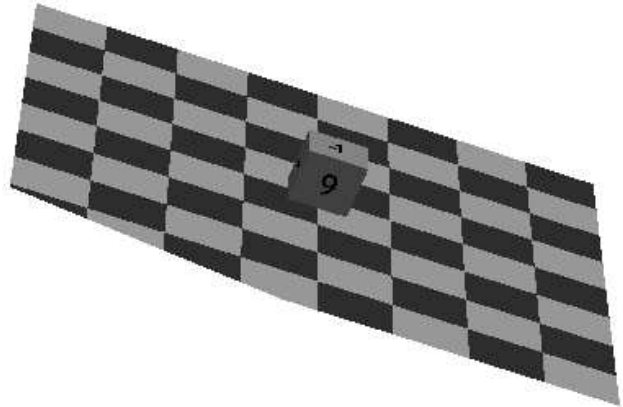


Figure 13: An orthographic projection

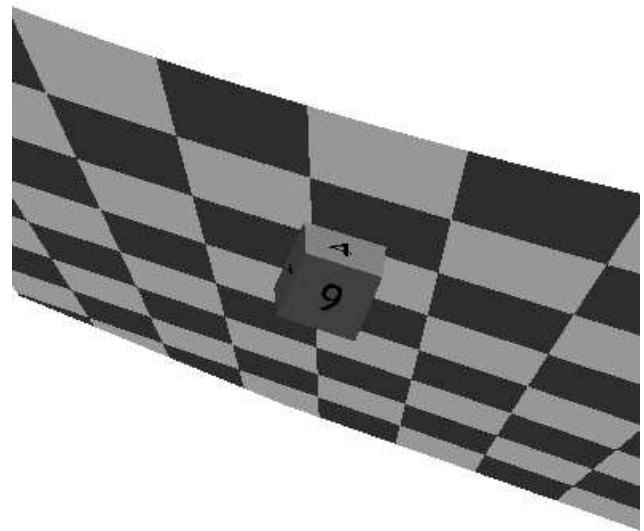


Figure 14: A 'reverse' perspective projection

The floor of the cube scene appears to be upside down in Figure 14, as the distant top edge is now larger than the closer bottom edge. A simple 1-dimensional slider barely captures the possibilities of interactively controlling the rendering surface, but does allow for an evaluation of such things as API design and system performance. The OpenGL like multi-perspective rendering API is well suited to interactively specified rendering surfaces because the surface is specified in a manner similar to that of dynamic geometry.

The rendering performance of the ray tracing implementation of the API is too slow for interactive performance. Frames take seconds to render each, even over such a simple scene. The ray tracing implementation is admittedly naïve, and does not include such common speedups as BSP culling of scene geometry. The API is designed to easily allow for the development and integration of new techniques, and these are already being worked on.

5 Conclusions and Further Work

While the performance of the API in terms of frames per second is far from interactive, the design and conceptual groundwork has been laid for more detailed investigation into multi-perspective images from visualisation. By extending previous work on interactive reflections on curved objects to multi-perspective images for visualisation performance issues will be addressed. The raytracing implementation described in this paper forms an important base line for measuring the accuracy of faster implementations.

The interactive manipulation of viewing surfaces is an unexplored field, the simple 1-dimensional interaction described here is a start, though things will be much more complicated with more dimensions of freedom. Another aspect this paper does not touch on is in the useful application of multi-perspective images. The cube world in Figure 11 demonstrates clearly one of our reasons for pursuing multi-perspective images, the ability to see more than is otherwise possible. With appropriate tools and interfaces we believe multi-perspective images will be a valuable tool for visualising complex data.

6 References

- BLINN, J.F. and M.E. NEWELL (1976): Texture and Reflection in Computer Generated Images. *Communications of the ACM*, v19, n10, p542-547.
- CHO, F. (2000): *Towards Interactive Ray Tracing in Two- and Three-Dimensions*. PhD Thesis. University of California at Berkeley.
- CHU, S. H. and C. L. TAI (2001): Animating Chinese Landscape Paintings and Panorama using Multi-Perspective Modeling, *Proceedings of Computer Graphics International 2001*, Hong Kong, IEEE Press.
- DAVIDHAZY, A. (2001):, *Peripheral Portraits and Other Strip Camera Photographs*, Retrieved October, 2001: <http://www.rit.edu/~andpph/exhibit-6.html> .
- ESCHER, M. C. (1992): *The Graphic Work*. Evergreen, Germany.
- GORTLER, S., R. GRZESZCZUK, R. SZELISKI and M. COHEN (1996): The Lumigraph. In *Proceedings of SIGGRAPH 96*, pp. 43-54.
- HAKURA, Z., SNYDER, J., and LENGYEL, J. (2001): Parameterized Environment Maps. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics, I3D01*.
- HURDAL, M. K., P. L. BOWERS, K. STEPHENSON, D. W. L. SUMNERS, K. REHM, K., SCHAPER, D. and A. ROTTENBERG (1999): Quasi-conformally flat mapping the human cerebellum, in C. Taylor and A. Colchester (eds), *Medical Image Computing and Computer-Assisted Intervention, Vol. 1679 of Lecture Notes in Computer Science*, Springer, Berlin, pp. 279-286.
- JANSEN, F. W. (1993): Realism in real-time? In *Proceedings Fourth Eurographics Workshop on Rendering*, Cohen, Puech and Sillion (eds).
- KEAHEY, T. A. (1998): The Generalized Detail-In-Context Problem. *Proceedings of the IEEE Symposium on Information Visualization*. IEEE.
- LOFFELMANN, H. and E. GROLLER (1996): Ray tracing with extended cameras. *The Journal of Visualization and Computer Animation*, 7(4): pp. 211-227.
- LOFFELMANN, H. (2001): *Diploma Thesis, "Extended Cameras for Ray Tracing"*. Retrieved October 1, 2001, <http://www.cg.tuwien.ac.at/~helwig/projects/dipl/>
- OFEK, E. and A. RAPPOPORT (1998): Interactive reflections on curved objects. In *Proceedings SIGGRAPH 98*, pp 333-342.
- RADEMACHER, P., and G. BISHOP (1998): Multiple-center-of-projection images. In *Proceedings of SIGGRAPH 98*, pp 199-206. ACM.
- SEGAL, M., and K. AKELEY (1998): The OpenGL Graphics System: A Specification (Version 1.2). <ftp://sgigate.sgi.com/pub/opengl/doc/opengl1.2/opengl1.2.pdf>
- SMITH, R. (1997): *Distortion Oriented Displays for Demanding Applications*. PhD Thesis, Gippsland School of Computing and Information Technology, Monash University.
- VALLANCE, S. and P. CALDER (2001): Context in 3D Planar Navigation. In *Proceedings of the Australasian User Interface Conference AUIC 2001*, Queensland, pp. 93-99.
- WINCH, D., P. CALDER, and R. SMITH (2000): Focus + Context³: Distortion-orientated displays in three dimensions. In *Proceedings of the 1st Australasian User Interface Conference AUIC 2000*, Canberra, Australia, pp. 126-133.
- WOOD, D., A. FINKELSTEIN, J. HUGHES, C. THAYER, and D. SALESIN (1997): Multiperspective Panoramas for Cel Animation. In *Proceedings of SIGGRAPH 97 Conference*, pp 243-250, ACM