# Seamless tiling in OpenGL based Quicktime VR applications

Gerald Schröcker
schroeck@icg.tu-graz.ac.at

Computer Graphics and Vision
Graz University of Technology
Graz / Austria

## Abstract

This article deals with the development of a method to warp cylindrical panoramas with the use of the OpenGL graphic system. Under consideration of the recording and reproduction geometry it is demonstrated that this technique is equivalent to conventional image warping algorithms. Problems with high resolution textures and OpenGL are discussed and a solution is shown.

**KEYWORDS:** image-based rendering, cylindrical panoramas, texturing

## 1 Introduction

In computer graphics *rendering* means creating a digital image with the use of a computer - to be more exact the generation of a view of a virtual scene under an arbitrary viewing direction. Conventional methods use a scene description based on geometrical primitives to create a view. McMillan uses the term *geometry-based rendering* for these methods [MB95]. With those methods regular shaped 'artificial' objects can easily be described. The modeling of curved surfaces, however, is more complex. These surfaces have to be approximated with a polygonal mesh. Natural objects as trees or items of daily life as clothing must be laboriously constructed with thousands of single faces.

Recently there have been great advances in three-dimensional scanning methods, which promise to simplify the process of model building. However, this automated model creation also verifies our suspicion: the geometry of the real world is extremely complex [MB95]. Therefore the modeling process is a laborious and time consuming task. But not only the modeling, also displaying the scene is algorithmicaly complex and needs high computational power. To rotate or to move an object at interactive frame rates, it is necessary to generate at least 5 images/sec. For complex scenes this can only be done with dedicated hardware.

In contrast, *image-based rendering* uses pictures for describing the scene. These images can be photographs, video takes or synthetic images. The modeling process is replaced by image recording. New views are produced through modifica-

tion (interpolation, reprojection etc.) of existing input data. A great advantage of *image-based rendering* is that the effort to display an object is independent of the complexity of its internal representation. A tree is represented by just as much pixels as a cube (assuming both are the same virtual size). Thus, in an image-based rendering system the computational costs of displaying new images are constant and independent of scene complexity. In Table 1 the differences of the two methods are summarized [Kan97].

| geometry-based rendering | image-based rendering |
|---|---|
| use of 3-D models | uses a collection of images |
| effort depends on scene complexity | effort independend of scene complexity |
| sophisticated software for realism | realism depends on input images |
| dedicated hardware | only main processor necessary |
| conventional rendering pipeline | pixel projection and pixel interpolation |
| | only static scenes |
| | not metric |

Table 1: Comparison between geometry-based rendering and image-based rendering

# 2 Survey on image-based rendering

Based on [Kan97] image-based rendering techniques can be classified into four areas: offline synthesis (*mosaicking*), online synthesis (pixel *reprojection*) and image-interpolation (*morphing* resp. *interpolation*).

## 2.1 Morphing

This category is not physically based and 3-D geometry is not considered at all. It simply interpolates between a pair of possibly unrelated images. This technique is used most widely in the advertising and entertainment industry. During the morphing process the images are warped so that the source shape slowly assumes the target shape, while maintaining a visual appealing mix. An example can be found in [BN92].

## 2.2 Mosaicking

At least two different images are combined to get a larger image. The resulting image (*mosaic*) has a wider field of view than its constituent images. It is a more compact representation that allows new views of the scene quickly to be generated. The simplest subsection of this technique are rectilinear panoramas. But they are problematic for wide view angles greater than $180°$. For a complete surround view spherical or cylindrical panoramas are more suitable. However these representations have to be warped prior to viewing to show a geometrical exact view of the scene.

## 2.3 Interpolation

The idea behind this class of methods is, to build up some kind of a *lookup table*. This table includes many image samples of a scene from a lot of different viewpoints. A new view from an arbitrary viewpoint is synthesized by *interpolating* the data stored in the lookup table.

The advantage of this class of methods is that unlike all other methods, pixel correspondence is not necessary. In addition, the lookup table is an approximation of the *plenoptic function* $P(X, Y, Z, \theta, \phi)$ [AB91]. The plenoptic function is a 5-D description of the flow of light at every 3-D position and 2-D viewing direction. Because the process of image synthesis is restricted to a search in the lookup table with following interpolation, fast visualization can be achieved.

Disadvantages are the high number of image samples resulting in high memory requirements and the necessity of knowing the exact camera position and orientation for every sample during data acquisition. Two recently described approaches are *light field rendering* [LH96] and *the lumigraph* [GGSC96].

## 2.4 Reprojection

These techniques use a relative small number of images, but additionally geometric constraints from the scene geometry are applied. For synthesizing a view from a virtual camera position, the image pixels are *reprojected* appropriately. The geometric constraints can be of the form of *known depth values* at each pixel [CW93], or epipolar constraints between pairs of images are used (*fundamental matrix* [LDFP93], [LF94] or constraints between pairs of cylindrical panoramas [MB95]). It is also possible to use three images with *trilinear tensors* [AS98].

# 3 Implementation

There are many possible surfaces upon which perspective projections can be mapped [GG99]. The most natural one is a sphere centered about the viewpoint. The problem of a spherical projection, is the representation of the surface of the sphere in a form which is suitable for storage and fast access on a computer. This is particularly difficult because a uniform (i.e. equal area for all elements) discrete sampling is desirable. This difficulty is reflected in various distortions which arise in cartography for planar projections of world maps. The mappings which are uniform, allow no systematic access and those which map to a plane distort significantly.

Another possibility is a set of six planar projections in the form of a cube with the projection center in the middle. While this representation can be easily stored and accessed by a computer, it is difficult to achieve the precise camera position and orientation. Also the planar cubic mapping does not represent a uniform sampling, it is considerably oversampled at the edges and corners. It is difficult to avoid artifacts from discontinuities at the image borders.

Therefore a projection on the surface of a cylinder has been suggested. One advantage of the cylinder is, that it can be easily unrolled into a simple planar map, making computer access easy. Another advantage is that a cylindrical panorama can be obtained more easily than a spherical one. Most recording systems support this class of panoramic images. A drawback is the limited vertical field of view.

## 3.1 Virtual camera

With a cylindrical panorama two of the three rotational degrees of freedom can be emulated completely and the third one partly. A full rotation about the vertical axis is possible. With a rotation about the horizontal axis a limited view upwards and downwards can be realized. But the vertical field of view can not be greater than 180°. Therefore a view straight up or down is not possible. A roll motion can also be simulated, but this rotation of the image is not reasonable, because this motion is not common in conventional photography. By modifying the field of view of the artificial camera, a zooming effect can be achieved. But through this image magnifying no new details can be seen. With bilinear filtering jagged edges resulting from the limited image resolution can be reduced.

## 3.2 Geometry

Due to the curved projection surface used when making cylindrical panoramic images strong distortions are inescapable. To generate new views, these distortions have to be corrected. For that purpose the mantle of a cylinder is covered with the panoramic image and viewed through a central projection from the center of the cylinder.

In other systems (e.g. Quicktime VR [Che95]) a custom image warping algorithm has been used for this task. However the goal of this work is to use the OpenGL graphic system and therefore a standard rendering pipeline. One can produce arbitrary image distortions by texturing a uniform polygonal mesh and transforming the vertices appropriately [HS93]. To warp the panoramic image, a cylindrical surface is approximated with a triangular mesh and the synthetic camera is placed in the center of the cylinder (Figure 1).

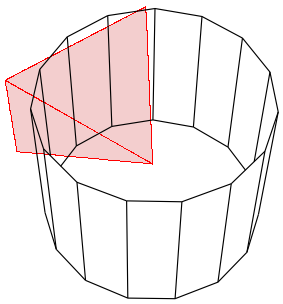


Figure 1: camera position in the approximated cylinder

## 3.3 Recording

By taking a point in 3-D world coordinates $(X, Y, Z)$ and mapping this point with a central projection with the center $O$ onto the surface of a cylinder (see Figure 2) we get formula (1). This gives the transformation in cylindrical 2-D coordinates $(\theta, \upsilon)$ [SS97].

$$\begin{aligned} \theta &= \arctan(X/Z) \\ \upsilon &= f \cdot \frac{Y}{\sqrt{X^2 + Z^2}} \end{aligned} \tag{1}$$

$\theta$ corresponds to the rotation angle and $\upsilon$ to the scanline. Whereas in the resulting panoramic image $\theta$ corresponds to the $x$-coordinate and $\upsilon$ corresponds to the $y$-coordinate. The cylinder radius $r$ is equal to the focal length $f$.

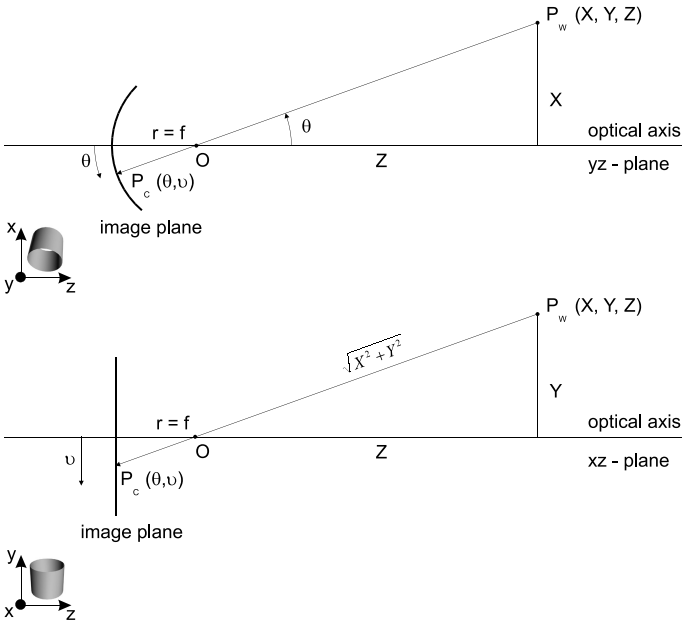



Figure 2: recording system

## 3.4 Reproduction

In order to warp the panorama for viewing, the image is projected from the cylindric surface to a plane, which is normal to the optical axis and tangents the mantle at the point $H$ (Figure 3). A point $P_C(\theta, \upsilon)$ on the cylinder mantle is transferred after formula (2) into a point $P_P(x_E, y_E)$ on the plane [Hof99].

$$\begin{aligned} x_P &= f \cdot \tan\theta \\ y_P &= \frac{\upsilon}{\cos\theta} \end{aligned} \tag{2}$$

The process of warping can now be accomplished by an algorithmic operation, called warp operation [Che95], [Hof99]. As one sees in Figure 3, the projection on a plane and following display with a central projection is equivalent to a direct projection of the mantle points through $O$ (the points lie on the same ray). Formula

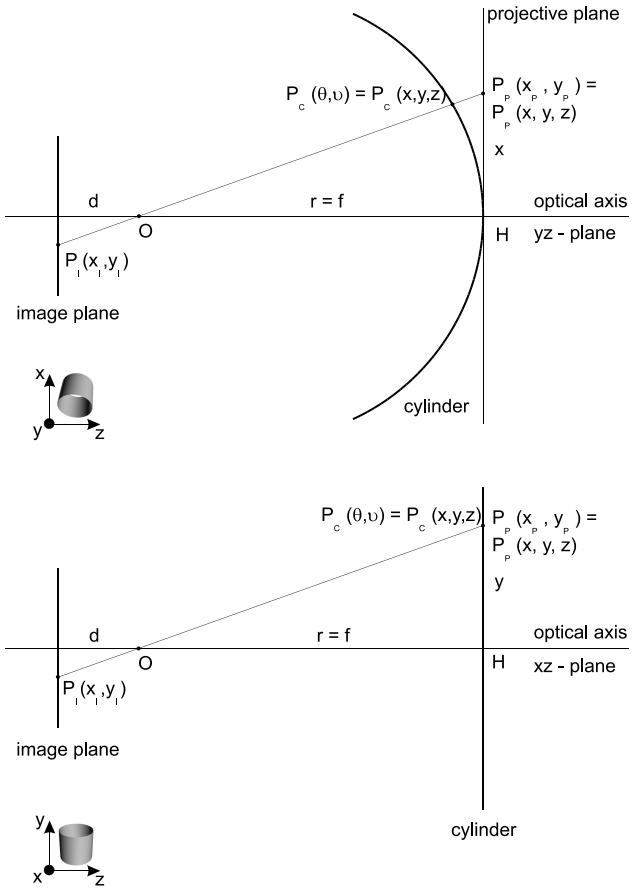



Figure 3: reproduction system

(3) gives the mapping from cylindrical coordinates $(\theta, \upsilon)$ to cartesian coordinates $(x, y, z)$ with the origin in the center of the cylinder. If one looks at these points under a central projection (projection center $O$, focal length $d$) this gives (4).

$$\begin{aligned} x' &= f \cdot \sin\theta \\ y' &= f \cdot \upsilon \\ z' &= f \cdot \cos\theta \end{aligned} \tag{3}$$

$$\begin{aligned} x'_I &= d \cdot \frac{x'}{z'} = d \cdot \frac{\sin\theta}{\cos\theta} = d \cdot \tan\theta \\ y'_I &= d \cdot \frac{y'}{z'} = d \cdot \frac{\upsilon}{\cos\theta} \end{aligned} \tag{4}$$

In order to proof that both methods provide the same results, the image plane coordinates $(x_I, y_I)$ from (2) are transformed with formula (5) to projective coordinates $(x'', y'', z'')$. If these points are seen under a central projection $(O, d)$ this

gives (6).

$$\begin{aligned} x'' &= x_E = f \cdot \tan\theta \\ y'' &= f \cdot y_E = f \cdot \frac{v}{\cos\theta} \\ z'' &= f \end{aligned} \tag{5}$$

$$\begin{aligned} x''_I &= d \cdot \frac{x''}{z''} = d \cdot \tan\theta \\ y''_I &= d \cdot \frac{y''}{z''} = d \cdot \frac{v}{\cos\theta} \end{aligned} \tag{6}$$

As expected, both methods provide the same result: $P_I(x'_I,y'_I)=P_I(x''_I,y''_I)$. Now it remains to be seen, that every warped view generated from the distorted panoramic image data is actually a central projection.

$$\begin{aligned} x_I &= f \cdot \tan\theta = f \cdot \frac{X}{Z} \\ y_I &= \frac{v}{\cos\theta} = \frac{f\frac{Y}{\sqrt{(X^2+Z^2)}}}{\cos(\arctan\frac{X}{Z})} = f \cdot \frac{Y\sqrt{\left(1+\frac{X^2}{Z^2}\right)}}{\sqrt{(X^2+Z^2)}} = f \cdot \frac{Y}{Z} \end{aligned} \tag{7}$$

Inserting of formula (1) in (2) gives the central projection in formula (7). The rendering of the undistorted view gives in fact the same picture that a normal camera would take.

## 3.5 Texturing

The polygon mesh, whose shape was derived in the previous section, has to be textured with the panoramic image. This leads to the following problem: the panoramic image can be very big (e.g. $13000 \times 3000$ Pixel). The rendering hardware however can only work with small sized textures. Thus the input data has to be divided into suitable parts. These parts are sent as single textures to the hardware. If bilinear filtering is used, new problems occur at the texture border. In bilinear filtering [SA99, S. 125ff] the weighted median of the color of the four nearest pixels is calculated. At the texture border these neighborhood values are not accessible. This results in *filtering artifacts*. Figure 4 shows a magnified cutting from a synthetic test image. The artifacts caused by incorrect filtering are clearly visible. There are two possibilities to support neighborhood data:

- use the built-in OpenGL *Texture-Border* [SA99, p. 114]. This feature is not implemented in hardware on common consumer graphic cards, so the OpenGL system switches to software emulation. This results in great speed losses.
- scale the texture coordinates in a way, that only the inner part of the texture is used. An invisible border of one pixel width remains outside. This border is taken into account by bilinear filtering.

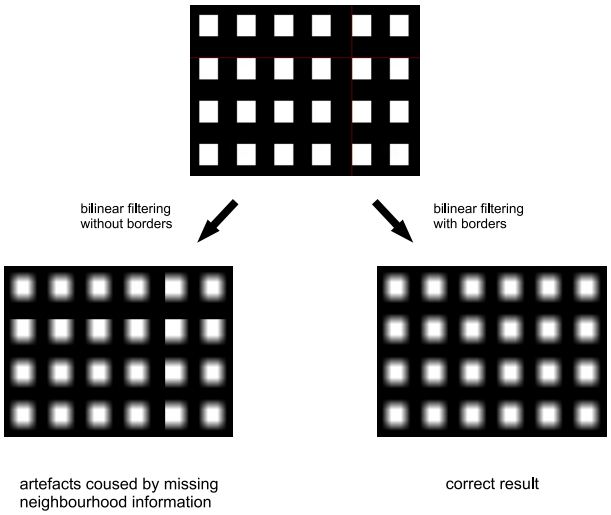



Figure 4: filtering artefacts

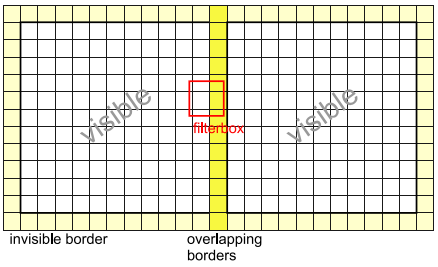



Figure 5: overlapping texture borders

As the test implementation should not only run on high end systems, the second method was used. In figure 5 the invisible overlapping texture border can be seen. The given texture coordinates are modified with the texture matrix [SA99, p. 34], this matrix transforms the $u, v$ - coordinates and determines which part of the texture is visible. The texture matrix $T$ in (8) down-scales the visible part by moving the texture coordinates by the equivalent of a discrete texel inward, thus an one texel thick invisible border remains. This border is only used for bilinear filtering.

$$T = \begin{bmatrix} \frac{w-2}{w} & 0 & 0 & -\frac{1}{w} \\ 0 & \frac{h-2}{h} & 0 & -\frac{1}{h} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

By using the texture matrix for modifying texture coordinates, switching between hardware texture border and software texture border is easy. When using the former method the texture matrix must be the identity matrix.

## 3.6 Memory management

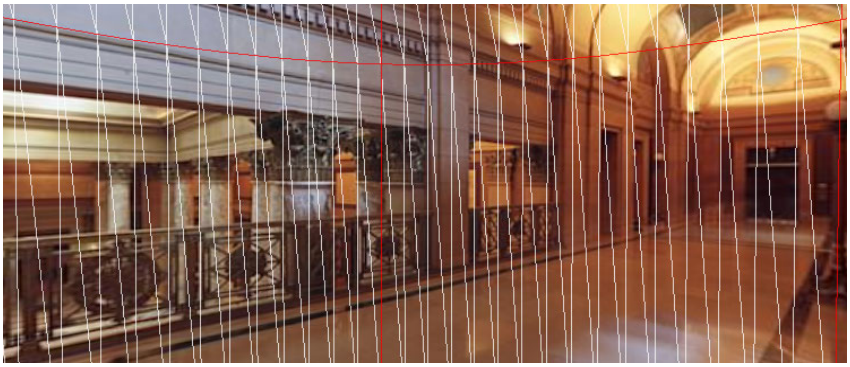
Memory management is another important aspect, because texture data can reach a size of 50 MByte or more. To achieve good efficiency the implementation uses OpenGL texture objects [SA99, S. 132ff]. The texture objects are handled standalone from OpenGL with a priority system. If all texture objects have the same priority most of the OpenGL implementations apply a *last recently used* strategy. This means currently used texture objects remain resident, unused texture objects are swapped out to less efficient memory areas. That is sufficient to achieve good response times. Moreover by using specific prioritizing improvements are eventually possible. The OpenGL system tries to hold the texture data (visible image parts) in texture memory. If the user moves the panorama, new texture data has to be loaded and old texture data has to be deleted. This is a time consuming task which leads to noticeable framerate reduction. For systems with *unified memory architecture* this is not true, they show a constant framerate under all conditions. In these systems the main memory is also used as texture memory, there is no bottleneck transporting data over the system bus.

# 4 Results

Figure 6 shows results from the test implementation. The panorama is warped in a correct way: the bent corridor (upper left image) is displayed - as it is in the real scene - as a straight corridor (upper right image). The lower image shows the underlying geometry of the textured cylinder overlapped to the warped image.

distorted view

undistorted view



geometry of the approximated cylinder
(white: polygon boundary, gray: texture boundary)

Figure 6: results

# 5 Summary and future work

After giving a short survey on image-based rendering techniques, a method to warp cylindrical panoramas with the use of the OpenGL graphic system was presented. With this approach fast warping of high resolution images can be done on personal computers with consumer OpenGL hardware.

A possible extension of the presented work would be the display of stereo panoramas to give the user a realistic sense of depth. In [HH98] a system is described which is capable to record stereo panoramas. The method uses two cameras and corrects discrepancies of the epipolar geometry or image differences automatically. As a result one gets two cylindrical panoramic images. They could be used directly for the technique described in this article. For that purpose the left eye and the right eye should see different textures of the cylinder. For example this could be achieved with LCD shutter glasses.

# References


[AB91] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. *M. Landy and J. A. Movshon, (eds) Computational Models of Visual Processing*, 1991.

[AS98] Shai Avidan and Amnon Shashua. Novel view synthesis by cascading trilinear tensors. *IEEE Transactions on Visualization and Computer Graphics*, 4(4), October 1998.

[BN92] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. *Computer Graphics*, 26(2):35–42, July 1992.

[Che95] Shenchang Eric Chen. Quicktime VR - An Image-Based Approach to Virtual Environment Navigation. *Proceedings of SIGGRAPH 95*, pages 29–38, August 1995.

[CW93] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 279–288, August 1993.

[GG99] Georg Glaeser and Eduard Gröller. Fast generation of curved perspectives for ultra-wide-angle lenses in VR applications. *The Visual Computer*, 15(7/8):365–376, November 1999.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, Addison Wesley, August 1996.

[HH98] Ho-Chao Huang and Yi-Ping Hung. Panoramic stereo imaging system with automatic disparity warping and seaming. *Graphical Models and Image Processing*, 60(3):196–208, May 1998.

[Hof99] Gerald Hofer. 3D Editor mit 3D Eingabegerät. Master's thesis, Institut für Maschinelles Sehen und Darstellen, Technische Universität Graz, June 1999.

[HS93] Paul Haeberli and Mark Segal. Texture mapping as a fundamental drawing primitive. In Michael F. Cohen, Claude Puech, and Francois Sillion, editors, *Fourth Eurographics Workshop on Rendering*, pages 259–266. Eurographics, June 1993.

[Kan97] Sing Bing Kang. A survey of image-based rendering techniques. Technical Report CRL 97/4, DEC Cambridge Research Laboratory, 1997.

[LDFP93] Q. T. Luong, Rachid Deriche, Olivier Faugeras, and Theodore Papadopoulo. On determining the fundamental matrix : analysis of different methods and experimental results. Technical Report RR-1894, Inria, May 1993.

[LF94] Stephane Laveau and Olivier Faugeras. 3-D scene representation as a collection of images and fundamental matrices. Technical Report RR-2205, Inria, February 1994.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 31–42. ACM SIGGRAPH, Addison Wesley, August 1996.

[MB95] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 39–46. ACM SIGGRAPH, Addison Wesley, August 1995.

[SA99] Mark Segal and Kurt Akeley. The OpenGL graphics system: A specification (version 1.2.1). Technical report, Silicon Graphics, Inc., Mountain View, CA,USA, April 1999.

[SS97] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic mosaics and environment maps. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 251–258. ACM SIGGRAPH, Addison Wesley, August 1997.