Institut für Computergraphik und Algorithmen
Technische Universität Wien
Karlsplatz 13/186/2
A-1040 Wien
AUSTRIA
Tel: +43 (1) 58801-18601
Fax: +43 (1) 58801-18698

Institute of Computer Graphics and Algorithms
Vienna University of Technology
email:
technical-report@cg.tuwien.ac.at
other services:
http://www.cg.tuwien.ac.at/
ftp://ftp.cg.tuwien.ac.at/

TECHNICAL REPORT

An Error Metric for Layered Environment Map Impostors.

Stefan Jeschke Michael Wimmer

TR-186-2-02-04
February 2002

Keywords: impostors, real-time rendering, virtual environments

An Error Metric for Layered Environment Map Impostors.

Stefan Jeschke* Michael Wimmer †

Vienna University of Technology

Abstract

Impostors are image-based primitives commonly used to replace complex geometry in order to accelerate the rendering of large virtual environments. This paper describes a "layered impostor technique" used for representing distant scene-parts when seen from a bounded viewing region. A special layer placement is derived which bounds the geometric error introduced by parallaxes to a defined value. In combination with a special technique for image generation, a high-quality impostor representation without image artifacts can be obtained.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing Algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality

Keywords: impostors, real-time rendering, virtual environments

1 Introduction

Impostors are image-based primitives that are primarily used in real-time rendering systems for the faster display of complex scene parts. Impostors are especially useful where other rendering-acceleration methods do not provide satisfying results. For instance, visibility calculations do not work well in scenes with low occlusion (like flyovers etc.), and geometric simplification approaches often provide insufficient results when simplifying many disconnected objects like street objects, trees etc.. The advantage of image-based representations is their low geometric complexity while at the same time providing almost the same appearance as conventional geometry rendering. Impostors are valid (i.e., indistinguishable from the geometric objects) only from a bounded viewing region (a so-called *view cell*). To generate an impostor, an image is rendered from a so-called *reference viewpoint* and combined with geometric information. The amount of geometric information can be different for individual impostor techniques, including simple quadrilaterals (so-called *planar impostors* [Regan and Pose 1994]), polygonal meshes (so-called *Textured Depth Meshes* [Darsa et al. 1997; Sillion et al. 1997]), or per-pixel geometry (called a *Depth Image* [Max and Ohsaki 1995; Aliaga and Lastra 1999]). Problems of impostor techniques are high memory requirements, incompatibility with conventional rendering hardware, and artifacts due to missing information about scene parts hidden from the reference viewpoint [Decoret et al. 1999]. For including hidden geometry in the representation, impostor techniques with multiple colors and depth values per texel position [Max 1996; Shade et al. 1998] were introduced.

Several impostor techniques have been proposed in previous work. However, the problem of generating an impostor that is guaranteed to show no image artifacts for a large view cell has not been solved in a satisfying way. In this paper, a layered impostor technique is described which addresses these issues simultaneously. It is based on partitioning the scene part into multiple image layers with varying depth [Lacroute and Levoy 1994; Meyer and Neyret 1998; Regan and Pose 1994]. In 1998, Schaufler [Schaufler 1998] introduced layered planar impostors. Objects are sliced into layers with increasing distance from the reference viewpoint, and for every layer a single planar impostor is generated by rendering its content into a transparent texture. This texture is assigned to a quadrilateral that is placed in the middle of the respective layer. Figure 1 shows an example of a layered impostor. With this technique, parallax errors are split to different layers so that the impostors are valid for a larger view cell compared to using only a single quadrilateral.

However, the original method suffers from image gaps between the layers, who become visible as the observer moves within the view cell. In this paper, we overcome this drawback with an optimal layer placement which bounds parallax errors in the impostor to a defined value, and by use a new sampling algorithm for recording the individual impostor layers. This is done without relying on any knowledge about the geometric structure of the original objects. As a result, all potentially visible scene parts are sampled at the proper reso-

*jeschke@cg.tuwien.ac.at

†wimmer@cg.tuwien.ac.at

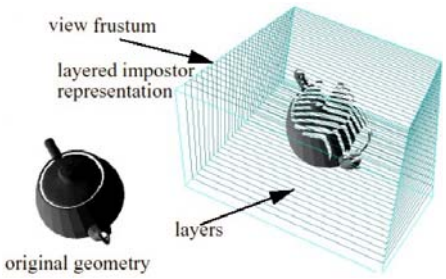

Figure 1: Layered impostor technique [Schaufler 1998].

lution, which makes it possible to produce high-quality representations without visible artifacts.

2 Layered Impostors Without Image Artifacts

The layered impostor generation process requires as input

- the scene part to be represented,
- the view cell (shape, size and position),
- the output image resolution and the field of view used during impostor display.

Two different shapes of view cells are considered here. *Box-shaped* view cells have widely been used in computer graphics. *Shaft-shaped* view cells have been used implicitly [Jakulin 2000; Aubel et al. 1999; Schaufler and Stürzlinger 1996; Shade et al. 1996]. Figure 2 shows an example for a shaft in 2D. The shaft apex lies in the center of the represented scene part. It is defined by a direction, an apex angle and a minimum distance to the object (see Figure 2). Shafts address the fact that

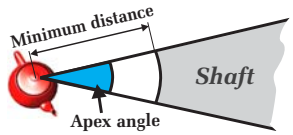

Figure 2: A shaft-shaped view cell.

errors introduced by the impostor are much less apparent when increasing the view distance. Consequently, the impostor can be displayed for a very large viewing region, especially compared to box-shaped view cells.

Before the impostor generation starts, the camera for recording the impostor is set to the reference viewpoint within the view cell. The camera frustum is defined so that it tightly encloses the scene part to be represented. The technique presented here for preventing image artifacts is based on two ideas:

- **Appropriate layer spacing:** The distance between two adjacent impostor layers is chosen so that they do not move more than one texel against each other when seen from within the view cell. We call this property the *one-texel layer spacing* (see Section 2.1).
- **Complete scene recording:** The rasterization process used for impostor layer recording is modified so that no scene parts are missed (see Section 2.2). In addition, for scene parts that are present in two adjacent layers, the transition between the layers is drawn using identical texel positions.

In summary, because every texel exposes no more than the texel behind it and all layer transitions are represented with identical texels, it is not possible to look "through" a transition from within the view cell. For this reason we can guarantee that no impostor shows any image gaps, especially if it represents a continuous surface in multiple layers.

2.1 Layer Placement Calculation

This section describes the calculation of the position and the depth range of every impostor layer. Two issues have to be taken into account in this connection. First, the one-texel layer spacing must be fulfilled in order to guarantee an artifact-free representation. Therefore, the following subsection gives a scheme for calculating the positions of the impostor layers so that this requirement is always fulfilled. The second issue are parallax errors introduced by the impostor which must be quantified and limited. This issue is important for calculating the depth range of every layer, as will be presented further below.

2.1.1 One-Texel Layer Spacing

The goal is to place the layers close enough so that for every view within the view cell, each texel uncovers at most the texel behind it in the following layer. Without loss of generality, for all further illustrations the object is assumed to be centered in front of the view cell. Furthermore, the reference viewpoint is positioned at a

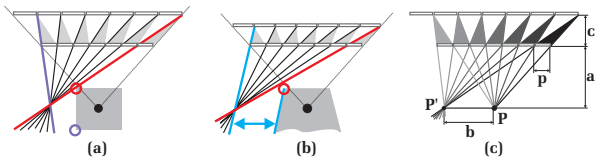

Figure 3: Movements of one texel when seen from within a rectangular (a) and shaft-shaped (b) view cell. The lines indicate for every texel when the texel behind it is completely uncovered when moving to the left. (c) shows the setup for the derivation of texel movements between two layers.

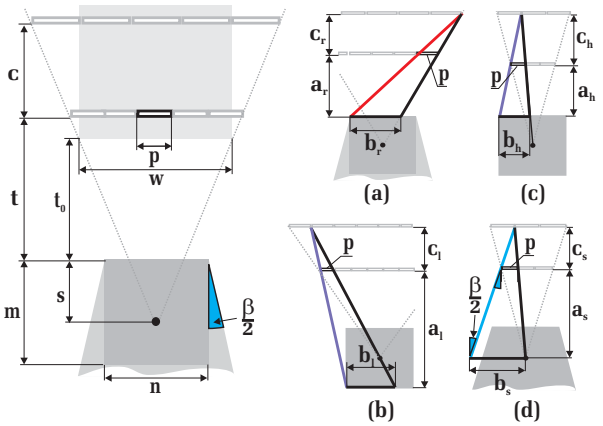

Figure 4: Left: setup for the layer spacing calculation based on the maximum allowable texel movement. Right: (a) for the right boundary line (identical for rectangular and shaft shaped view cell), (b) and (c) for the left boundary line for a rectangular view cell, (d) for the left boundary line for a shaft shaped view cell.

vertical line through the view-cell center as is shown in Figure 3 (left and middle). This is the best horizontal position because all errors introduced by the impostor are equally distributed to the left and right side. The optimal vertical position will be discussed further below. For this symmetric setup, all following considerations can be reduced to one side of the view cell (we choose the left side for explanation).

Figure 3 ((a) and (b)) shows for every texel in a layer a line indicating where the texel in the following layer is completely uncovered when moving to the left. This means an image gap will occur behind a texel when viewed from a position to the left of this line. The one-texel layer spacing has to be calculated so that no such line crosses the view cell.

First we will show that all these lines meet in a point at the same height as the reference viewpoint, because this reduces the consideration to only the outermost lines. Assume a given reference viewpoint P and two impostor layers with distances a and $a+c$ as shown in Figure 3 (c). The size of a texel in the layer with distance a is defined as p. For every texel, a triangle is formed by the line that indicates when the texel behind it is completely uncovered, and a line from P to the right border of that texel. This triangle is shown in different colors for every texel in Figure 3 (c). From similar triangles we get:

$$\frac{c}{a+c} = \frac{p}{b}.$$

Because p, c and a are constant, b is also constant, i.e., independent from the actual texel position in the layer. Consequently, all lines meet the single point P' as can be observed in Figure 3 (c).

This in turn means that the line of the leftmost and the rightmost texel define the right boundary of *all* lines (see Figure 3). In order to avoid image gaps, it is sufficient to ensure that these two lines do not cross the view cell.

For the following considerations we will call them *left boundary line* and *right boundary line*, respectively.

The layer calculation assumes the following input parameters (see Figure 4 (left)):

- Either a rectangular view cell with width n and depth m, or a shaft-shaped view cell with an apex angle β. Furthermore, the distance s from the reference viewpoint to the view-cell border.
- The distance t_0 between the object and the view cell, and the width w of the object. The frustum used for the impostor generation is assumed to tightly fit the bounding box, as is depicted in Figure 4 (left).
- The impostor texture resolution *res*, which is chosen so that the impostor texture is not magnified when seen from any viewpoint within the view cell. The resolution can be calculated using the method of Schaufler [Schaufler 1995]. The size of a texel p at distance t (see Figure 4 (left)), is then defined as:

$$p = \frac{w(t+s)}{res(t_0+s)}.$$

Given a layer distance t, the distance c from that layer to the following one is calculated so that the boundary lines just touch the view cell. c is calculated independently for the left and right boundary line, and the smaller value

is used. This is done for all cases by using similar triangles:

$$\frac{c}{a+c} = \frac{p}{b}, \quad (1)$$

Figure 4 ((a)-(d)) shows the respective configurations for a and b for the left and right boundary line, which are now discussed.

For the right boundary (see Figure 4, (a)) the definition is

$$\begin{aligned} a_r &= t, \\ b_r &= \frac{n}{2} + \frac{sw}{2(t_0+s)}. \end{aligned}$$

Substituting a and b in Equation 1 results in the distance c_r:

$$c_r = \frac{t}{\frac{res}{2(t+s)}\left(\frac{n(t_0+s)}{w}+s\right)-1}. \quad (2)$$

This equation holds for the right boundary of rectangular as well as of shaft-shaped view cells.

For the left boundary line, the layer computation is first considered for a rectangular view cell. A distinction must be made depending on whether the left boundary of the impostor layer ends to the left or to the right of the left boundary of the view cell. This indicates whether the left boundary line touches the lower (see Figure 4 (b)) or upper (see Figure 4, (c)) view-cell corner.

The definition of a and b for the lower view-cell corner (Figure 4, (b)) is

$$\begin{aligned} a_l &= t+m, \\ b_l &= \frac{n}{2} + \frac{(m-s)\left(\frac{w}{2}-\frac{w}{res}\right)}{t_0+s}. \end{aligned}$$

Equation 1 results in the distance c_l:

$$c_l = \frac{t+m}{\frac{res}{2(t+s)}\left(\frac{n(t_0+s)}{w}+(m-s)\left(1-\frac{2}{res}\right)\right)-1}. \quad (3)$$

The definition for the upper view-cell corner is analogous (Figure 4, (c))

$$\begin{aligned} a_h &= t, \\ b_h &= \frac{n}{2} - \frac{s\left(\frac{w}{2}-\frac{w}{res}\right)}{t_0+s}, \end{aligned}$$

with Equation 1 resulting in the following distance c_h:

$$c_h = \frac{t}{\frac{res}{2(t+s)}\left(\frac{n(t_0+s)}{w}+s\left(\frac{2}{res}-1\right)\right)-1}. \quad (4)$$

For shaft-shaped view cells, in order to ensure that the view cell is not intersected by the left boundary line, the line must be *parallel* to the left border of the shaft (see Figure 4, (d)). a and b are defined as follows for this case:

$$\begin{aligned} a_s &= t+s, \\ b_s &= \frac{(t+s)w}{2(t_0+s)} + (t+s)\tan\left(\frac{\beta}{2}\right). \end{aligned}$$

Equation 1 results in c_s:

$$c_s = \frac{t+s}{res\left(\frac{1}{2}+\frac{t_0+s}{w}\tan\left(\frac{\beta}{2}\right)\right)-1}. \quad (5)$$

Up to now, the layer placement calculation was presented for one dimension. The extension to the general 2D case is quite simple because all texel movements are independently for the x and y direction. This means that c must be calculated for *both* directions and the smaller value is used in order to ensure that no gaps occur in any direction.

An interesting question is the choice of an optimal reference viewpoint, i.e., the choice of s. The best choice maximizes the inter-layer spacing for the left *and* right boundary line, because the smaller value has to be used. Figure 5 (a-c) shows an example how the position of the reference viewpoint affects the inter-layer spacing. The

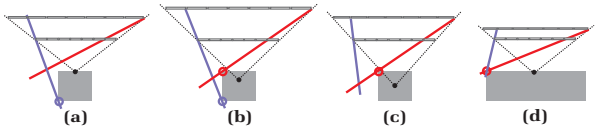

Figure 5: Layer spacing for different reference viewpoints: (a) too close to the object, (b) optimal placement, (c) too far from the object. (d) If the impostor width is smaller than the view-cell width, the optimal viewpoint is always nearest to the object.

spacing c_r for the right boundary line always decreases with increasing s. This can be shown using the derivation $c_r'(s)$ of Equation 2 which is

$$c_r'(s) = \frac{2reswt(n(t_0-t)-wt)}{(res(n(t_0+s)+ws)-2w(t+s))^2}.$$

The denominator is always positive and the numerator is always negative, because $t > t_0$ holds. Consequently, c_r decreases monotonically with increasing s. Therefore, the best value for s for the right boundary line is 0. Note that placing the reference viewpoint in front of the view cell (i.e., $s < 0$) is not possible, because

distant impostor layers would appear magnified in the output image. Fortunately, if $s = 0$, Equation 4 always results in exactly the same value for c_h as Equation 2 does for c_r. This means, as long as the left boundary of every layer does not exceed the left boundary of a rectangular view cell, the optimal viewpoint is $s = 0$ as is shown in Figure 5 (d). This is also the best viewpoint considering the sampling of the impostor texture: if the viewer moves to the view-cell border nearest to the impostor (where $s = 0$), the texels in every impostor layer are equally-sized on the screen, thus making best use of texture memory. In contrast, if the reference viewpoint is placed further apart from the object (i.e., $s > 0$), distant layers are never viewed at full resolution which is a (minor) waste of memory. For all other cases (for the lower rectangular view-cell corner and a shaft-shaped view cell), no general optimum can be found. This is because the optimal reference viewpoint is different for each layer (as it depends on the layer distance t). However, changing the position of the reference viewpoint for different layers is not compatible with the artifact-free layer recording technique (see Section 2.2). However, in practice, a value for rectangular view cells of $s = \frac{m}{2}$ and for shaft-shaped view cells of $s = \frac{n}{2}$ was found to provide satisfying results.

2.1.2 Parallax Errors

The layering technique provides a reproduction of parallax movements within the represented scene parts. However, because every texel represents a certain depth range, parallax errors still occur within every layer. This error is quantified with the *parallax angle* α, which is the angle between the *true* 3D position of the point and its projection to the impostor [Schaufler 1998]. The goal is to limit α for the whole impostor, typically to the angle of a pixel in the output image. In order to achieve this for layered impostors, the depth range for every layer has to be set up appropriately.

For the following considerations, we distinguish between two cases: first, we derive the depth ranges *between* two adjacent layers, i.e., where to place the border between two impostor layers. Afterwards, we determine the depth ranges of the outermost layers, i.e., in front of the first (nearest) layer and behind the last (most distant) layer.

Concerning the depth ranges between two layers, the one-texel layer spacing (see Section 2.1.1) already guarantees that the parallax errors that occur between adjacent impostor layers are always smaller than α. This is because the relative movement of a texel against the texel at the same position in the previous and following layer is limited to less than a texel. Given the fact that every texel is always visible with an angle smaller than α, no parallax error larger than a texel can occur. A result of this consideration is that the choice where to place the depth border (i.e. the clipping plane) between two adjacent impostor layers is arbitrary.

However, to keep the introduced parallax errors as small as possible, parallax movements should be distributed *equally* between adjacent layers. This means that for the viewpoint were the maximum parallax angle occurs, it should be equally distributed to the closer and more distant layer, for every "view direction" from that point. Figure 6 (a) shows a configuration where the border is placed so that this is *not* achieved: the parallax angle that occurs for the closer layer is much larger than the one for the distant layer. In contrast, Figure 6 (b) shows a configuration where the border is placed so that this is achieved, independently from the view direction. Note that it is *not* desired to equally distribute the parallax angle for a particular texel position.

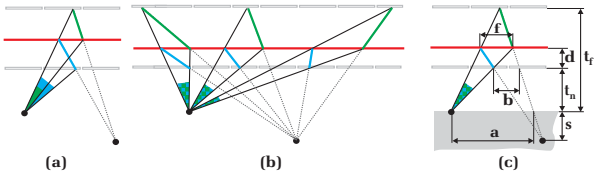

Figure 6: (a): different parallax angle in two adjacent layers. (b): equal parallax angle in two layers, independently from the view direction. (c): calculation scheme for optimally placing the border.

We assume that the largest parallax angle occurs if the viewer is located somewhere on the view cell border closest to the impostor (as Figure 6 (c) shows), because the impostor appears in maximum magnification for this configuration. Given two impostor layers with distances t_n and t_f to the view cell, the distance d from the near layer to the border between the layers is calculated using three pairs of similar triangles (see Figure 6 (c)):

$$\begin{aligned} \frac{t_f}{a} &= \frac{t_f - t_n - d}{f}, \\ \frac{t_n + d + s}{f} &= \frac{t_n + s}{b}, \\ \frac{d}{b} &= \frac{t_n + d}{a}. \end{aligned}$$

Solving the equation system for d (by eliminating a and b, which automatically eliminates f) results in

$$d = \frac{-t_n^2 - t_n s + \sqrt{t_f t_n (t_f + s)(t_n + s)}}{t_n + t_f + s}. \quad (6)$$

Note that d only depends on t_n, t_f and s. This means that the parallax angle is equally distributed between the layers for *all* view directions of any viewpoint on the view cell border, as is shown in Figure 6 (b).

The final task is the derivation of the depth range in front of the nearest and behind the farthest impostor layer. We assume a plane at a given distance, which is either the near border of the scene part to be represented for calculating the first impostor layer, or the last impostor layer for calculating to which distance that layer can represent the scene. The goal is to choose the depth range so that the parallax angle never exceeds α when seen from the view cell. Therefore, the setup for which the *largest parallax angle* occurs must be found. Figure 7

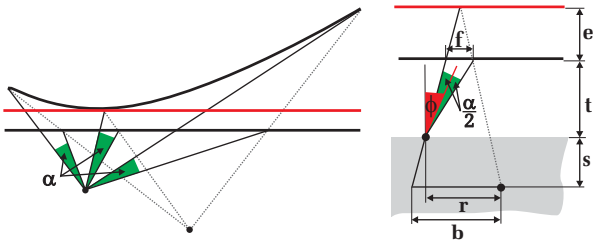

Figure 7: Left: points that are represented with an angle of α with respect to a layer form a curve. Right: setup for calculating the maximum allowable depth range.

(left) shows for every point on some plane a corresponding point which is represented with an error angle of α when seen from a new viewpoint. All such "α-points" form a curve, as the figure shows. If the depth range is set to the minimum distance of all α-points to the plane (the red line in the figure), the parallax error will never exceed α for the whole depth range.

We again assume that the maximum parallax angle occurs for a viewpoint on the view cell border close to the impostor as Figure 7 (right) shows. The depth range e is described in dependence of an angle ϕ, which can be interpreted as the view direction from the new viewpoint. Given the plane at distance t to the view cell and a viewpoint of r left to the reference viewpoint (see Figure 7 (right)), for an arbitrary ϕ we get the depth range e by similar triangles:

$$\begin{aligned} \frac{e}{f} &= \frac{e+t+s}{b}, \text{ where} \\ f &= t\tan\left(\phi + \frac{\alpha}{2}\right) - t\tan\left(\phi - \frac{\alpha}{2}\right), \\ b &= r + s\tan\left(\phi - \frac{\alpha}{2}\right). \end{aligned}$$

This results in

$$e = \frac{2t\sin(\alpha)(t+s)}{r(\cos(2\phi)+\cos(\alpha)) + s(\sin(2\phi)-\sin(\alpha)) - 2t\sin(\alpha)} \quad (7)$$

The minimum value for e is obtained by differentiation with respect to ϕ:

$$e'(\phi) = \frac{4t\sin(\alpha)(t+s)(r\sin(2\phi) - s\cos(2\phi)}{(r(\cos(2\phi)+\cos(\alpha)) + s(\sin(2\phi)-\sin(\alpha)) - 2t\sin(\alpha))^2},$$

and solving $e'(\phi) = 0$. This results in

$$\phi_{min} = \frac{1}{2}\arctan\left(\frac{s}{r}\right).$$

With the second derivative, it can be shown that this is a minimum for e, as Figure 7 (left) shows. Note that ϕ_{min} only depends on the position of the new viewpoint relative to the reference viewpoint, i.e., it is independent from the distance between the impostor and the view cell, and from the pixel angle α. In order to obtain the minimum depth range e_{min}, ϕ_{min} is inserted in Equation 7, resulting in

$$e_{min} = \frac{2t\sin(\alpha)(t+s)}{r\cos(\alpha) - (2t+s)\sin(\alpha) + \sqrt{r^2+s^2}}.$$

Note that e_{min} is decreasing with increasing r. This means that r has to be set to the maximum possible value in order to guarantee a parallax angle smaller than α for every viewpoint in view cell. This maximum value is obtained in the corner of the view cell, which is intuitive because this is the largest distance to the reference viewpoint.

An interesting characteristic is that starting from a certain distance, the parallax movements in the whole scene part farther away than this distance are smaller than α, so that the whole scene part can be represented by a single layer, regardless of its extent. This is the case if $f \geq b$ (see Figure 7 (right)), because the two lines that enclose f (starting in the reference viewpoint and in the new viewpoint) do not meet each other.

All considerations up to this point work in a 2D plane. In contrast to texel movements, parallax movements cannot be treated independently for the x and y direction

for the general 3D case. However, the viewpoint where the largest parallax effect occurs is still the view cell corner near the impostor, but now in 3D space. The calculations presented above can still be done in 2D, but in a plane which is defined by the reference viewpoint and the two diagonal view-cell corners near the impostor. Consequently, the value for r must be changed to mean the distance to a view cell corner in 3D space (see Figure 8). Using this setup, the calculations remain as described above.

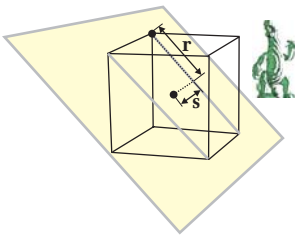

Figure 8: Parametrization for the 3D case for calculating the depth ranges concerning parallax angles.

2.2 A Rasterization Method for Guaranteed Layer Connectivity

Because every layer is recorded separately, scene parts hidden from the reference viewpoint are included in the representation (in contrast to Schaufler [Schaufler 1998] who used only a single image). However, image cracks still appear. The problem is basically caused by the specification of polygon rasterization in current graphics hardware: a pixel is only drawn during polygon rasterization if its *center* is covered by the polygon. One effect of this definition is that surfaces viewed from an acute angle may not be rendered at all if they fall between two adjacent pixel centers. In this case, information is missing which can lead to large image gaps. Furthermore, small gaps appear between adjacent layers representing a single primitive (i.e., polygon) because of the following fact: the intersection of the primitive with the clipping plane separating the two layers forms a *clipping edge*. The definition of rasterization entails that each texel of the clipping edge is rasterized *either* in one *or* in the other layer, as is shown in Figure 9, left. This instantly leads to gaps between adjacent layers if the viewpoint is moved.

The elimination of such image cracks is obtained in the following way: texels are drawn even if they are covered only partly by a primitive, especially if the texel center is not covered. This ensures that *all* scene parts

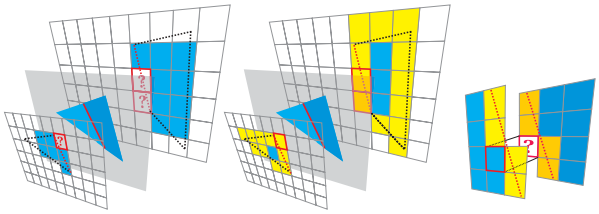

Figure 9: Left: the clipping edge of the polygon (drawn in red) is not represented in both adjacent layers. Middle: after separately drawing the outline of the clipped polygon (yellow texels), a common representation of the clipping edge is generated in both layers. Right: holes at each clipping edge step are filled manually to guarantee an artifact free representation.

are acquired (regardless whether they cover a texel center or not) and that *all* clipping edges are present in *both* involved layers.

The way to ensure this is to *manually* clip each polygon to the corresponding near and far layer border (Sutherland-Hodgeman clipping can be used, for example), and draw all polygon outlines explicitly, for example using the OpenGL edge primitive (see Figure 9, middle). The outlines have to be drawn in a predefined direction (e.g., from left to right) to ensure that identical texels are rasterized for the clipping edges in every layer. In order to ensure that all line endpoints are drawn as well, the polygon vertices are rasterized separately as points.

Although this removes most of the image cracks, sporadic ones might still appear if the viewer moves in *diagonal* direction within the view cell. This is caused by the rasterization of the clipping lines (an “eight-connected” line) as is shown in Figure 9 (right). Such cases can be manually identified by considering each 2x2 texel block in both layers: while the diagonal texels in one direction are present in *both* layers (Figure 9 (right) shows the upper-left to lower-right case), the texels in the orthogonal direction are only present *either* in one *or* in the other layer. The gap can be closed by copying the texel color from the closer layer to the more distant layer (thus forming a “four-connected” line).

The result is that for every object, all borders between adjacent layers are drawn using identical texel positions. In combination with the one-texel layer spacing (see Section 2.1.1) the representation shows no cracks or image gaps as can be observed in Figure 10 (right). In order to achieve this property, Schaufler [Schaufler 1998] proposed to let the depth ranges of adjacent layers overlap. However, this gives no guarantee that image gaps do not occur, as Figure 10 (middle) shows. For

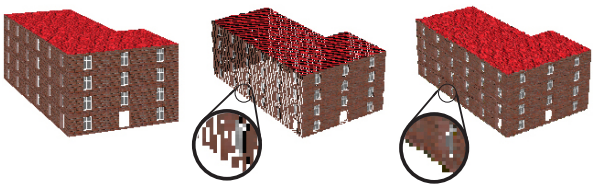

Figure 10: Left: original object seen from the reference viewpoint. Middle: impostor recorded using layers widely overlapping in depth. Note that image cracks are not avoided. Right: the same impostor generated using the new method with all image cracks eliminated.

this example, even a very large depth overlap of half a layer does not remedy the problem. Another approach by Meyer and Neyret [Meyer and Neyret 1998] tries to solve the problem in image space by estimating object contours in every layer and filling the inside of each such contour. However, because the algorithm does not exploit information about the original scene parts, it is unclear how well the result resembles the original scene.

2.3 Discussion on the Number of Layers

The number of impostor layers is important for the efficiency of the layered impostor technique since it strongly correlates to the required impostor memory as well as the geometric complexity of an impostor. Therefore, this section describes the basic factors that influence this value.

The number of required impostor layers depends on two main factors:

- The parallax movements within the scene part, which depend on the size of the view cell and the distance between the view cell and the represented scene part. Because parallax movements are "mimicked" by the impostor layers, the number of required layers increases proportionally with the amount of parallax movements.
- The size of an impostor texel, defined by the output image resolution and field of view. Because each layer is not allowed to move more than one texel (see Section 2.1.1), the number of layers grows with decreasing texel size.

In order to show the influence of parallax movements, the number of layers is calculated for the model of a dragon (about 20 m high and consisting of 108,500 polygons) for varying distances between the model and a cubic view cell with a sidelength of 10m. Figure 11 shows some views from a point outside the view cell in order to show the number of layers. Diagram 12

Figure 11: Example for layered impostors for different object distances to the view cell. From left to right: 120, 60, 30, 15, 7, 1 layers generated for 42, 59, 96, 142, 209 and 563 m distance.

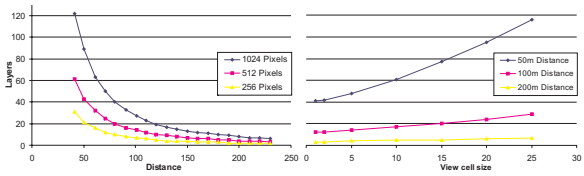

Figure 12: Number of layers required for the dragon model in dependence of the distance between the view cell and the model and for different output image resolutions (left), as well as for some fixed distances but varying view-cell sizes (right).

(left) shows the number of layers, obtained in each test for varying output resolutions (discussed further below). It can be seen that the number of layers falls hyperbolically with increasing distance between the model and the view cell. This was expected since parallax movements increase analogously. In a second test, the view-cell size was varied for three fixed distances. It can be observed that the number of layers grows roughly linearly with the view-cell size for all distances. A result from these tests is that the distance between the scene part and the view cell in combination with the view-cell size has a major impact on the efficiency of the resulting impostor: while for distant objects and/or small view-cell sizes only few layers are needed to cover large scene portions, near objects and/or large view-cell sizes result in very high number of layers. The distance has a greater impact than the view-cell size.

In the first test described above (Figure 12, left), the output resolution was varied between 256 and 1024 pixels. It can be seen that the number of layers increases roughly linearly with the number of pixels. This can also be derived from the Equations 2 to 4.

Note that all trends discussed here also hold for shaft-shaped view cells because of their similar parallax and texel movement characteristics.

3 Application: Layered Environment Map Impostors

The layered impostor technique can be used for rendering acceleration in several ways. Aside from the use for rectangular or shaft-shaped view cells, layered impostors can also be used in architectural models as portal impostors. For *Layered Environment Map Impostors*, the impostor layers are arranged in a concentric way around a cubic view cell, forming "cubic" environment maps. This means that the impostors are used to represent distant scene parts. Figure 13 shows an example for the Vienna model, where impostors were placed in four orthogonal directions parallel to the ground plane.

Figure 13: Layered environment map impostor.

4 Summary

In this paper, a new method for generating layered impostors without image artifacts was presented. This is achieved by a special layer placement in combination with a special layer recording method that ensures that every scene part is present in each layer it covers. This ensures that no holes become visible between the layers, so that image artifacts never occur.

References

ALIAGA, D. G., AND LASTRA, A. 1999. Automatic image placement to provide a guaranteed frame rate. In *SIGGRAPH 99 Conference Proceedings*, Addison Wesley, A. Rockwood, Ed., Annual Conference Series, ACM SIGGRAPH, 307–316.

AUBEL, A., BOULIC, R., AND THALMANN, D. 1999. Lowering the cost of virtual human rendering with structured animated impostors. In *WSCG'99 Conference Proceedings*, Univ. of West Bohemia Press, V. Skala, Ed.

DARSA, L., SILVA, B. C., AND VARSHNEY, A. 1997. Navigating static environments using image-space simplification and morphing. In *1997 Symposium on Interactive 3D Graphics*, ACM Press, M. Cohen and D. Zeltzer, Eds., ACM SIGGRAPH, 25–34. ISBN 0-89791-884-3.

DECORET, X., SILLION, F., SCHAUFLER, G., AND DORSEY, J. 1999. Multi-layered impostors for accelerated rendering. *Computer Graphics Forum (Proc. Eurographics '99) 18*, 3 (Sept.), 61–73. ISSN 1067-7055.

JAKULIN, A. 2000. Interactive vegetation rendering with slicing and blending. In *Proceedings of Eurographics 2000 (Short Presentations)*, A. de Sousa and J. Torres, Eds., Eurographics.

LACROUTE, P., AND LEVOY, M. 1994. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *SIGGRAPH 94 Conference Proceedings*, 451–458.

MAX, N., AND OHSAKI, K. 1995. Rendering trees from precomputed Z-buffer views. In *Rendering Techniques '95*, Springer, 45–54.

MAX, N. 1996. Hierarchical rendering of trees from precomputed multi-layer Z-buffers. In *Rendering Techniques '96 (Proceedings of the Eurographics Workshop on Rendering 96)*, Springer-Verlag Wien New York, X. Pueyo and P. Schröder, Eds., Eurographics, 165–174. ISBN 3-211-82883-4.

MEYER, A., AND NEYRET, F. 1998. Interactive volumetric textures. In *Rendering Techniques '98 (Proceedings of the Eurographics Workshop on Rendering 98)*, Springer-Verlag Wien New York, G. Drettakis and N. Max, Eds., Eurographics, 157–168.

REGAN, M., AND POSE, R. 1994. Priority rendering with a virtual reality address recalculation pipeline. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, ACM Press, A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, 155–162. ISBN 0-89791-667-0.

SCHAUFLER, G., AND STÜRZLINGER, W. 1996. A three-dimensional image cache for virtual reality.

Computer Graphics Forum (Proc. Eurographics '96) 15, 3 (Sept.), 227–235. ISSN 0167-7055.

SCHAUFLER, G. 1995. Dynamically generated impostors. In *GI Workshop on Modeling, Virtual Worlds,*, D. W. Fellner, Ed., 129–135.

SCHAUFLER, G. 1998. Per-object image warping with layered impostors. In *Rendering Techniques '98 (Proceedings of the Eurographics Workshop on Rendering 98)*, Springer-Verlag Wien New York, G. Drettakis and N. Max, Eds., 145–156.

SHADE, J., LISCHINSKI, D., SALESIN, D., DEROSE, T., AND SNYDER, J. 1996. Hierarchical image caching for accelerated walkthroughs of complex environments. In *SIGGRAPH 96 Conference Proceedings*, Addison Wesley, H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, 75–82. held in New Orleans, Louisiana, 04-09 August 1996.

SHADE, J. W., GORTLER, S. J., HE, L., AND SZELISKI, R. 1998. Layered depth images. In *SIGGRAPH 98 Conference Proceedings*, Addison Wesley, M. Cohen, Ed., Annual Conference Series, ACM SIGGRAPH, 231–242. ISBN 0-89791-999-8.

SILLION, F., DRETTAKIS, G., AND BODELET, B. 1997. Efficient impostor manipulation for real-time visualization of urban scenery. *Computer Graphics Forum (Proc. Eurographics '97) 16*, 3 (Aug.), 207–218. ISSN 1067-7055.