Computers & Graphics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Hybrid visibility compositing and masking for illustrative rendering ☆

Stefan Bruckner [a,*], Peter Rautek [a], Ivan Viola [b], Mike Roberts [c], Mario Costa Sousa [c], M. Eduard Gröller [a]

[a] Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria
[b] Department of Informatics, University of Bergen, Norway
[c] Department of Computer Science, University of Calgary, Canada

ARTICLE INFO

Keywords:
Compositing
Masking
Illustration

ABSTRACT

In this paper, we introduce a novel framework for the compositing of interactively rendered 3D layers tailored to the needs of scientific illustration. Currently, traditional scientific illustrations are produced in a series of composition stages, combining different pictorial elements using 2D digital layering. Our approach extends the layer metaphor into 3D without giving up the advantages of 2D methods. The new compositing approach allows for effects such as selective transparency, occlusion overrides, and soft depth buffering. Furthermore, we show how common manipulation techniques such as masking can be integrated into this concept. These tools behave just like in 2D, but their influence extends beyond a single viewpoint. Since the presented approach makes no assumptions about the underlying rendering algorithms, layers can be generated based on polygonal geometry, volumetric data, point-based representations, or others. Our implementation exploits current graphics hardware and permits real-time interaction and rendering.

1. Introduction

Digital compositing was arguably one of computer graphics' first mainstream commercial applications. Areas such as motion picture post-production greatly benefited from automated processing. The ability to flexibly combine multiple sources ultimately lead to the widespread adoption of digital special effects which are now ubiquitous in the film industry. Today, virtually every image editing software package has the ability to arrange elements in layers, modify alpha channels, control blending, and apply effects to individual layers.

In this paper, we focus on the compositing of dynamic 3D content. Instead of combining static elements such as images, movie sequences, or pre-rendered 3D animations, an interactive environment allows the modification of properties such as the viewpoint for individual layers which are rendered on-the-fly. Common software tools such as Adobe Photoshop have recently introduced the ability to embed dynamically generated layers based on 3D models. However, for the purpose of composing, the layer content is still treated as a 2D image even though additional information would be available. One reason for this choice is the fact that the conventional layered compositing approach, which assumes 2D layers, is deeply incorporated into these software packages and the workflow of their users. In this paper, we present a concept for the integration of 3D layers which preserves this intuitive notion, but allows artists to take advantage of 3D information by extending the operator set of traditional compositing approaches. We deliberately make minimal assumptions about the algorithms and data structures used to generate layer content to facilitate non-invasive integration into conventional image manipulation software.

One of our target applications is the generation of technical, medical, and scientific illustrations which frequently make use of selective occlusion overrides and blending in order to produce an expressive image. Using our approach, an illustrator can modify 3D properties of the content layers without having to go through the entire compositing process again. However, the presented approach is general and not restricted to this particular scenario. While recent work in illustrative visualization has put special emphasis on the role of methods employed by graphics artists and illustrators, many approaches are limited to specific types of scientific data (e.g., scalar volume data) and/or application domains. Although our approach can handle dynamically changing 3D layers, it does not require knowledge about the underlying rendering algorithms. This enables the flexible integration of different data representations such as polygonal meshes, volumetric-, and point-based data as well as different rendering algorithms such as painterly, photorealistic, or illustrative rendering.

The remainder of this paper is structured as follows: Section 2 reviews related work. In Section 3, we discuss the concepts behind our hybrid visibility compositing approach. Section 4 presents a technique for masking of dynamic 3D layers. Section 5 details our implementation and Section 6 presents further results. We discuss our approach in Section 7 and conclude the paper in Section 8.

☆ Funded by: FWF.
* Corresponding author. Tel.: +43 1 58801 18643; fax: +43 1 58801 18698.
E-mail address: bruckner@cg.tuwien.ac.at (S. Bruckner).

0097-8493/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2010.04.003

2. Related work

The work presented in this paper is related to several fields. Our approach is based on the large body of research in the area of digital compositing. Masking of 3D layers is related to image-based rendering techniques which attempt to generate novel views of 3D objects based on partial information. Furthermore, we were also inspired by illustrative and non-photorealistic rendering techniques which aim to reproduce the aesthetic quality of manually generated artwork using computer graphics.

Digital compositing: Digital compositing dates back to the early days of computer graphics as a scientific discipline. Alvy Ray Smith and Ed Catmull combined two images using a third image of coverage values which subsequently lead to the notion of the alpha channel [1]. Wallace [2] extended the approach to recursive blending allowing layers to be composited in any order that obeys associativity. Porter and Duff [3] introduced the concept of pre-multiplied alpha and formulated the compositing algebra which is in widespread use today. For the purpose of anti-aliased combination of 3D rendering results, Duff [4] proposed the rgbαz representation which, in addition to color and alpha channels, also includes a depth value for every pixel. Recent work by McCann and Pollard [5] extends the flexibility of traditional compositing by enabling stacking decisions on a per-overlap basis. However, their approach is designed to provide more control over the compositing of 2D layers, while the concept presented in this paper aims at facilitating the integration of 3D content in a consistent manner.

Image-based rendering: The idea of avoiding expensive rendering passes by using compositing to combine parts of a scene gave rise to the area of image-based rendering. Image-based rendering approaches attempt to synthesize novel views which closely approximate correct visibility from information captured during the generation of a single image. Lengyel and Snyder [6] proposed a factorization of 3D scenes into independent 2D sprites which could have different update rates. While their approach attempts to identify independent 2D layers, our method makes use of the available depth information and therefore allows full control over intersecting objects. Nailboards additionally store a depth value for every pixel of a sprite to enable the rendering of interpenetrating 3D objects with correct visibility [7]. Layered depth images contain multiple pixels along each line of sight to enable the generation of novel views with higher fidelity [8]. These approaches use layers with additional spatial information to combine parts of a scene with correct visibility. However, the focus of our work is different: we want to provide the user with the ability to selectively override occlusion relationships as it is common in technical, medical, and scientific illustrations.

Illustrative rendering: Several systems for the generation of illustrations using computer graphics have been developed. Dooley and Cohen [9,10] presented approaches for the automatic generation of semi-transparent line and surface illustrations from 3D models. Pioneering work by Seligman and Feiner [11–13] first treated the topic of visibility constraints. Their work employed cutaways and ghosting to resolve visibility conflicts. Preim et al. [14] presented Zoom Illustrator, a semi-interactive tool for illustrating anatomic models. Their approach focuses on the integration of three-dimensional graphics and textual representations. Diepstraten et al. [15,16] proposed rendering algorithms for ghosting and cutaway effects. Owada et al. [17,18] developed a system for modeling and illustrating volumetric objects. They semi-automatically generate artificial cutting textures based on surface models. Viola et al. [19] introduced the notion of importance-based rendering for improved visualization of features in volume data. Extending this approach, Bruckner and Gröller [20] presented a flexible interactive direct volume illustration system. Rautek et al. [21,22] proposed the use of semantic layers defined using a fuzzy logic rule base. Cole et al. [23] proposed a technique for generating architectural illustrations featuring a stylized focus area through local variations in shading effects and line qualities. Kalkofen et al. [24] used stylized overlays for focus+context visualization in augmented reality applications. Li et al. [25,26] presented geometric methods for generating high-quality cutaway and exploded view diagrams. Raman et al. [27] discussed a system which uses layer-based effects to enhance the visualization of volume data. Similar to our approach, the ClearView system presented by Krüger et al. [28] uses layered rendering to generate a number of different transparency effects inspired by traditional illustrations. However, their approach relies on a globally defined layer order, for instance nested isosurfaces of a volume dataset. Furthermore, their method only allows the use of a single spherical focus region.

In this paper we contribute with a new approach to combining interactively rendered 3D output based on the communication goals and stylization requirements of technical, medical and scientific illustrations. We introduce the notion of hybrid visibility compositing which allows integration of layered 2D compositing with 3D visibility operations in a flexible and intuitive manner. Additionally, we propose a new method for performing common masking operations based on this concept. The resulting framework enables the interactive generation of 3D illustrations featuring effects and techniques typically only available in 2D compositing software.

3. Compositing

Duff [4] was the first to propose the rgbαz representation for compositing 3D rendered images. In such a representation each pixel stores, in addition to its color and alpha value, a depth value. In a way, such an rgbαz image is a generalization of a 2D sprite [29]—points with color, transparency, and depth, but without any thickness information. Even though rgbαz layers are not a complete description of a general 3D object, they are a useful extension of conventional 2D layers. One of the main reasons why we choose this representation is that it requires minimal information about the actual data structures and algorithms used to provide layer content. A layer may be generated through ray tracing, rasterization of polygonal models, point-based rendering, or virtually any other technique capable of producing color and depth information.

We therefore choose this representation as one of the basic building blocks of our compositing framework. Each 3D layer is bound to a renderer instance and captures its output as an rgbαz image at any time. The content of such a layer may change dynamically, e.g., due to user interaction or animation. The compositing engine then decides how these layers are combined to form the final image. Since current graphics hardware allows us to easily access its color and depth buffers, one advantage of employing an rgbαz representation is that no modifications to the rendering stage are required. This means that the compositing engine can be used to combine layers produced by a variety of different rendering algorithms.

3.1. Implicit visibility

In contrast to 2D compositing where the stacking order of layers is solely specified by the user, rgbαz layers have an *implicit visibility* defined by the relative depth values of their pixels. The general technique for compositing multiple rgbαz layers with correct visibility is through a per-pixel application of the painter's

Please cite this article as: Bruckner S, et al. Hybrid visibility compositing and masking for illustrative rendering. Computers and Graphics (2010), doi:10.1016/j.cag.2010.04.003

algorithm: for each pixel, the corresponding depth values of all layers are sorted and then blended together using the *over* operator, i.e., each layer overdraws the layers located behind to a degree specified by its alpha channel. While such a compositing algorithm permits the combination of many different rendering techniques, it does not provide the same level of flexibility as 2D compositing in which the user has full control over layer order and blending operators.

3.2. Explicit visibility

Another approach to compositing rgbαz layers is to employ *explicit visibility* by ignoring the per-pixel depth values and defining a stacking order in which blending operators are applied. This means that the layers are treated as flat images. Their operators are applied in the same order for all pixels. Employing explicit visibility for 3D content can be useful for creating illustrations when a particular layer should be emphasized by overlaying over layers depicting occluding structures. However, it also completely discards the additional information provided by the depth values.

3.3. Hybrid visibility

One of the main complications of implicit visibility compositing is that there is no consistent layer order. Using explicit visibility, layers can be moved in the stacking order to control which structures appear in front of each other and this relationship remains true for all pixels of an image. For implicit visibility, however, there is an inherent layer order which may be different for each pixel. Ignoring the depth information sacrifices all the advantages of 3D layers while relying on the implicit visibility severely limits the range of possible operations. In order to provide the user with a more intuitive interface based on familiar 2D compositing metaphors while preserving the ability to render with correct occlusion, we use a *hybrid visibility* approach which represents a flexible combination between implicit and explicit visibility.

As illustrated in Fig. 1, our framework allows the user to specify a stacking order for the input layers and group them hierarchically. Just like in conventional 2D approaches, each layer and group can be assigned a blending operator. Additionally, an optional layer mask, discussed in detail in Section 4, can be specified. Compositing is performed by traversing the layer hierarchy starting with the bottommost layer and blending the layers using their associated operator. The depth value of an intermediate image pixel always corresponds to the last layer which makes a visible contribution to it. For the purpose of integrating hybrid visibility into this familiar setup we provide a special set of blending operators which take into account spatial relationships.

Fig. 2 illustrates the advantages of hybrid visibility for the generation of illustrations. In the first row, a manually generated illustration of a sports car is depicted in the left column. The center column shows the implicit visibility of a similar 3D model. In the right column, four individual layers of the car (chassis, tires, interior, and details) are shown. The second row shows an example of explicit visibility using the following stacking order from bottom to top: chassis, tires, interior, details. Even though a result similar to the manual illustration can be generated by employing explicit visibility, it does not translate to other viewpoints. The third row depicts results generated using our hybrid visibility approach which allows us to closely mimic the essential features of the manually generated image. Interior and tires form a *visibility chain* which uses implicit visibility. The result is combined with the chassis and the details using *occlusion-based blending*. These concepts are discussed in detail in the following sections.

3.3.1. Visibility chains

A visibility chain is simply a group of layers where, for each pixel, compositing is performed with implicit visibility. It is specified using the *visibility* operator. The chain starts with a layer that has its operator set to *visibility* and ends with the first subsequent layer in stacking order which uses a different operator—this layer terminates the chain. The compositing result of the visibility chain is combined with the intermediate image using the operator specified for the terminating layer. Compositing then proceeds normally with the next layer. The advantage of visibility chains is that they allow groups of layers to exhibit correct occlusion relationships among themselves while still being embedded in the specified layer hierarchy.

For each pixel within a visibility chain, our algorithm first performs a depth sort of its input layers. Compositing is then performed by blending the individual layers in visibility order using the *over* operator. For additional control, we use a smooth distance-based weight similar to the blurred *z*-buffering approach proposed by Luft et al. [30]. The color rgbα used for compositing a layer L_i is a distance-weighted sum of the color of all layers in the visibility chain:

$$\mathrm{rgb}\alpha = L_i \cdot \alpha \frac{\sum_j (1-\Delta z(L_i \cdot z, L_j \cdot z, L_j \cdot \omega)) L_j \cdot rgb\alpha}{\sum_j (1-\Delta z(L_i \cdot z, L_j \cdot z, L_j \cdot \omega)) L_j \cdot \alpha} \quad (1)$$

Note that in Eq. (1) each layer's color is pre-multiplied by its alpha value and that the result will also be an opacity-weighted color. The function $\Delta z(z_0, z_1, \omega) \in [0,1]$ is a user-selectable function which controls the nature of the depth transition. We require the function to be monotonically increasing with the absolute difference between its first two arguments. The third argument $\omega \in [0,1]$ allows additional control over the particular shape of this function—increasing ω should lead to a sharper transition. Different types of such transition functions are possible, similar

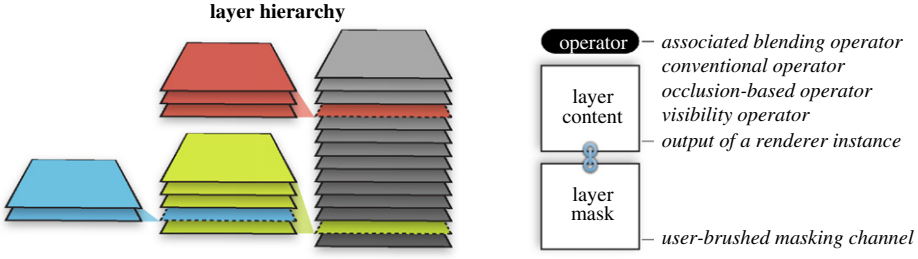

Fig. 1. Conceptual overview of our compositing setup. A layer definition is comprised of the layer content in form of an rgbαz image, an additional layer mask, and an associated blending operator. Layers are arranged in a hierarchical blending tree as exemplified in the figure: the upper highlighted layer in the stack consists of three sublayers and the lower highlighted layer consists of five sublayers one of which is itself composed of two sublayers.

Please cite this article as: Bruckner S, et al. Hybrid visibility compositing and masking for illustrative rendering. Computers and Graphics (2010), doi:10.1016/j.cag.2010.04.003

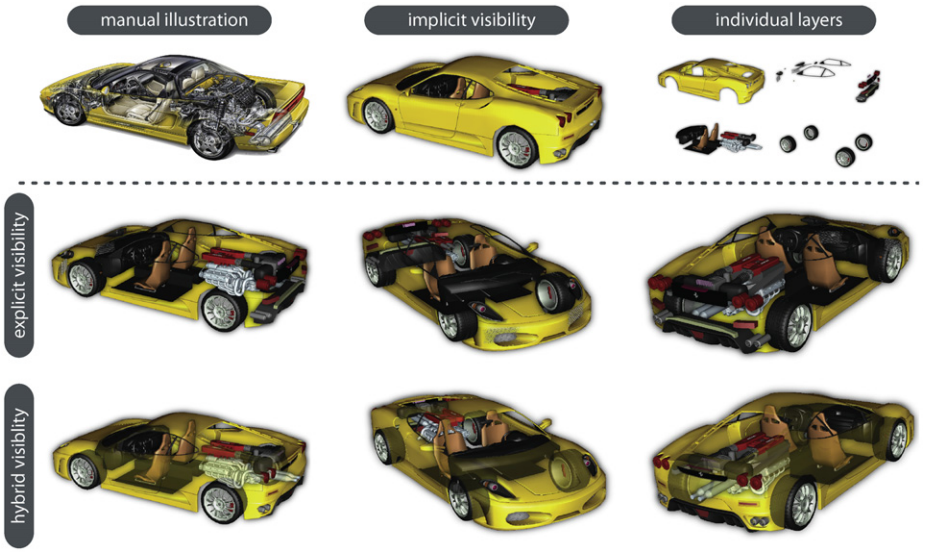

Fig. 2. Comparison of implicit, explicit, and hybrid visibility approaches to compositing. *Top row*: Left—manually generated illustration of a sports car. Center—implicit visibility of a similar 3D model. Right—four individual layers of the model. *Middle row*: Explicit visibility of the layers from three different viewpoints. *Bottom row*: Hybrid visibility of the layers from three different viewpoints. While implicit visibility alone does not capture the subtle effects used in the manual illustration, explicit visibility leads to distracting results when changing the viewpoint. Hybrid visibility avoids the drawbacks of both approaches. *Manual illustration courtesy of ©Kevin Hulsey Illustration, Inc.*

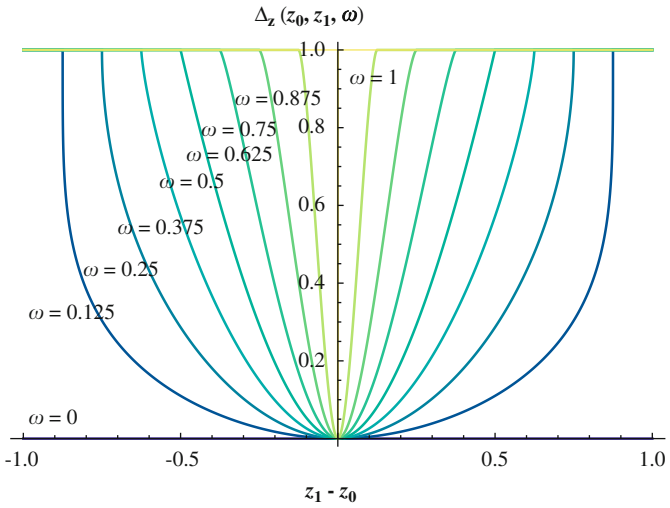

Fig. 3. Graphs of the Δz- function for different values of ω.

to easing curves in animation. In our current implementation, we use the following definition:

$$\Delta z(z_0,z_1,\omega) = 1-(\text{smoothstep}(\omega,1,1-|z_1-z_0|))^{\omega} \quad (2)$$

where smoothstep (a,b,x) is OpenGL's smoothstep function typically implemented as $u^2(3-2u)$ with $u = \text{clamp}(0,1,(x-a)/(b-a))$.

If $\omega = 0$, the value of Δz is always zero. For $\omega = 1$, the function value is zero only if $z_0 = z_1$ and one otherwise. Fig. 3 depicts graphs of the Δz-function for different values of ω.

If a pixel has the same depth in two layers, the resulting color will be the opacity-weighted average of the two layers' colors. Conversely, if the depth of the pixel in both layers is sufficiently different, their influence on each other will be zero. The user can control the distance weighting for each layer in a visibility chain by modifying its ω parameter. The approach can be used to effectively combat z-fighting, but it also offers an additional degree of artistic freedom. For instance, the weight may be altered on a per-layer basis to give a better indication of spatial relationships or to suggest the softness of a particular object. Fig. 4 shows an example. The teapot's body, handle, lid, and spout are rendered into separate layers and ω is globally set to 1, 0.5, 0.25, and 0.

3.3.2. Occlusion-based blending

In addition to the *visibility* operator, we provide a simple but powerful extension of conventional blending operators which allows them to make use of the additional spatial information. This includes the operators of the Porter–Duff algebra, such as *over*, *atop*, *in*, and *out*, as well as further operators typically present in image manipulation software (e.g., *multiply*, *screen*, or *overlay*). Our framework allows the use of all these operators in combination with a blending weight based on the distance between the layer's depth z and the current depth of the intermediate composite z_f. The layer's opacity is multiplied by the blending weight w_o which is computed by

$$w_o = \begin{cases} 1 & \text{if } \beta z < \beta z_f \\ 1-\Delta z(z,z_f,|\beta|) & \text{otherwise} \end{cases} \quad (3)$$

where $\beta \in [-1,1]$ is a user-controlled parameter of the operator. If β is zero (the default value), the operator will behave exactly like its two-dimensional counterpart. If $\beta > 0$, the parts in front of the intermediate image will be shown and parts behind it will decrease in opacity with increasing distance. Conversely, if $\beta < 0$ parts behind the current depth of the intermediate composite will be shown with full opacity and parts in front of it will decrease in opacity with increasing distance. This enables smooth fading of layers based on occlusion relationships. For instance, two layers containing different representations of the same object can be used to make it shine through an occluding layer with a different appearance. This effect is demonstrated in Fig. 10 where an occlusion-based *plus* operator is used to show an X-ray style representation of the hand where it is occluded by the lens of the magnifying glass.

Please cite this article as: Bruckner S, et al. Hybrid visibility compositing and masking for illustrative rendering. Computers and Graphics (2010), doi:10.1016/j.cag.2010.04.003

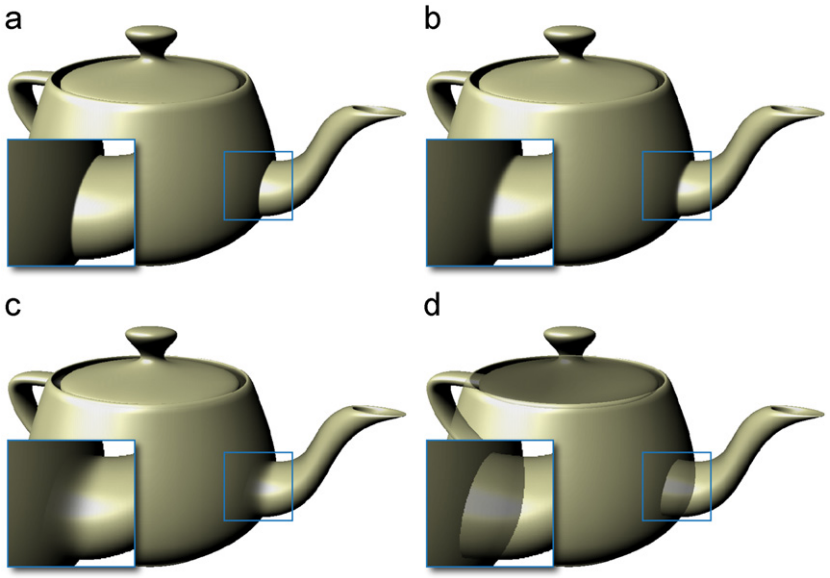

Fig. 4. Soft depth compositing using different values for ω: (a) $\omega=1$; (b) $\omega=0.5$; (c) $\omega=0.25$; and (d) $\omega=0$.

4. Masking

A common technique frequently employed for the compositing of 2D images is masking. A layer mask enables the artist to modify visibility independent of layer content. It can be utilized to make structures semi-transparent using smooth transitions, give selective emphasis to certain objects, or to remove unwanted parts. Common software packages feature flexible brushing tools to perform these manipulations (e.g., Adobe Photoshop's eraser).

When dealing with 3D layers which are generated on-the-fly and allow interactive modification of the viewpoint, however, the extension of such tools is not straight-forward. A purely 2D approach would be invariant to any camera changes and therefore frequently lead to undesired results when the viewpoint is modified. When we attempt to operate in object-space, on the other hand, we face the problem that a rgbαz layer is not a complete representation of a 3D object. The only 3D information available is the first visible surface of the object for the current viewpoint. While the renderer has complete information about the structure of the object, leaving the task of masking to each renderer would be prone to much duplication and potential inconsistencies as well as requiring modification of each rendering algorithm. For instance, a layer generated using volume rendering would need to handle masking operations in an entirely different manner than a layer generated by rendering polygonal geometry.

Our approach for masking represents a hybrid image-space/object-space approach which does not require additional information other than that provided by rgbαz layers. It can therefore be used for any kind of layer, irrespective of layer content. The user simply selects the target layer for the masking operation and can then paint on it to establish the mask. As soon as a stroke is placed by brushing over an area, the depth value of the layer is read. Together with the image-space position this gives us the 3D location of the stroke under the current viewing transformation. Applying the inverse viewing transformation transforms the stroke location into object space. This position, together with the current brush settings, is stored in a list associated with the selected layer. Whenever a layer has been updated by the corresponding renderer (e.g., due to a change of the viewing transformation) the list of strokes for the layer is traversed and rendered using splatting [31]. Each stroke is rasterized as a view-aligned impostor under the current viewing transformation. For each fragment of the stroke, we now have its intensity i and depth z. The intensity i is determined by the brush parameters and can be, for example, a 2D Gaussian with its peak centered at the stroke's image-space position. The depth z is simply the depth of the impostor fragment.

Then, for each fragment of the stroke, the depth z_l of the layer at the fragment's position is read. As this value is the first intersection of the viewing ray with the three-dimensional object represented by the layer, we can use it to estimate how much influence this fragment of the stroke should have for the current viewpoint. For instance, if the surface point we originally placed our stroke on is now occluded by another part of the surface, the difference between z and z_l will be high. Conversely, if the same point on the surface we placed the stroke on is still visible in the novel viewpoint, the difference will be zero at that location. Fig. 5 illustrates this behavior. It depicts two stroke centers rendered from two different viewpoints. From view 1, both stroke centers lie on the surface, i.e., $z=z_l$. For view 2, stroke 2 still lies on the visible surface. The position of stroke 1, however, is occluded by another part of the object, i.e., the difference between z and z_l is large.

As we want our strokes to vary smoothly in intensity when the view is changed, we choose to modulate the stroke intensity i using a weight w_b based on the difference between z and z_l. This weight is computed using the previously discussed Δz-function:

$$w_b = 1-\Delta z(z,z_l,\gamma) \tag{4}$$

where $\gamma \in [0,1]$ is a user-controlled parameter of the brush. An intuitive feature of this approach is that the brush is sensitive to the properties of the visible surface. If a large brush is chosen the rendered imposter will be a large flat disc centered at the stroke position. As the distance between the disc's depth and the surface depth modifies the brush intensity, depth discontinuities will tend to be preserved. Since we store brush strokes using a point-based representation there are other advantages of our approach: as the brush strokes are rasterized for every novel view, aliasing is avoided. Furthermore, parameters such as brush intensity, size, or shape can be modified after the strokes have been placed.

Typically, the masking channel is used to modulate layer opacity, i.e., it is multiplied with the α value of the corresponding layer pixel. To enable further control over the effect of masking,

Please cite this article as: Bruckner S, et al. Hybrid visibility compositing and masking for illustrative rendering. Computers and Graphics (2010), doi:10.1016/j.cag.2010.04.003

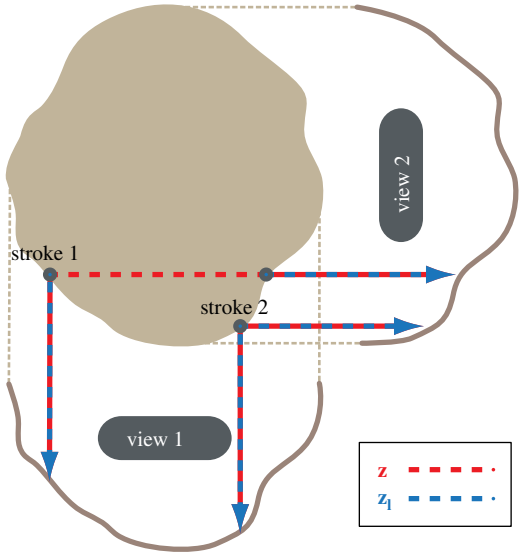

Fig. 5. Example of distance-based weighting for brush strokes. Two brush strokes (stroke 1, stroke 2) generated from the viewpoint view 1 are shown. In view 1, $z=z_l$ for both strokes, i.e., both strokes receive the maximum weight. When a novel viewpoint (view 2) is chosen, stroke 1 has $z \neq z_l$ due to occlusion, i.e., it receives a lower weight, while stroke 2 remains visible.

we provide an additional parameter s in the range $[-1,1]$ which the user can modify independently for each layer. If $s \geq 0$, the α value for each pixel is additionally multiplied by $1-si$, where i is the brushed intensity for the layer pixel. If s is negative, α is multiplied by $1-si+s$. If s is positive, the layer becomes more transparent with higher brush intensity. Negative values of s invert the effect: the layer is transparent where the brush intensity is zero and becomes more opaque with increasing values. Setting s to zero disables any effect of masking.

5. Implementation

The techniques described in this paper were implemented as an extension to an existing rendering framework written in C++ and OpenGL/GLSL. In order to make use of all available renderers of the framework, the basic display routine was modified to supply each renderer instance (which represents a layer) with an offscreen buffer for color and depth instead of the visible framebuffer. This is possible using the EXT_framebuffer_object OpenGL extension. For the renderers, this change was completely transparent—the rendering code did not require any changes. In fact, the framework now allows the compositing engine to be switched at runtime. The offscreen buffers are allocated as an array texture (an array of 2D textures which can be accessed similar to a 3D texture) provided by the EXT_texture_array extension. When a renderer instance needs to update itself, it simply renders a new image into its offscreen buffer–all other images are left unmodified. This also means that when a user interaction occurs, only those renderers which are affected by the change need to re-execute.

The compositing algorithm then uses the current values stored in the array texture. It is executed for every frame. The significant improvements concerning conditionals and loops introduced by the latest generation of graphics hardware allowed us to implement the whole compositing algorithm in a single-pass fragment program. This fragment program first reads colors, depth values, and masking information for every layer. Completely transparent pixels can be culled during this early stage which serves as a great source for performance improvement as they do not have to be considered in the subsequent steps. Compositing is then performed corresponding to the specified stacking order and grouping hierarchy. For visibility chains, depth sorting is performed in the fragment program. An additional offscreen buffer is kept for each layer where masking has been applied. When the appropriate input event occurs, the current masking parameters together with the determined object-space stroke locations are stored in a list associated with the selected layer. Every time a renderer has updated itself, this information is used to execute the algorithm described in Section 4. For rendering the strokes as imposters, we employ OpenGL's ARB_point_sprite extension which allows for textured as well as analytically defined brush tips. One advantage of having the result of several renderers available as separate layers is that different effects can be applied selectively. Again, we draw inspiration from 2D image manipulation software which offers a wide variety of layer effects. As our layers also store depth information even more options are available. Our framework provides a flexible interface for integrating these effects. For example, we employ the depth-based image enhancement approach presented by Luft et al. [32] which has proven to be a natural extension of common two-dimensional glow or drop shadow filters.

Although the current implementation of our compositing algorithm was not optimized for performance, it performs at frame rates above 20 frames/s for up to eight layers with a window size of 800×600 pixels on a GeForce 8800 GTX GPU. The overall performance is heavily dependent on the algorithms and models used to generate the layer content. For all the results shown in this paper the frame rate was over 5 frames/s for re-rendering all layers and effects. As the modification of a layer mask only requires a re-execution of the compositing pass, it is independent of layer content.

6. Results

In order to evaluate the utility of the presented techniques, we consulted a professional medical illustrator with over 25 years of experience in the field. We attempted to recreate effects and techniques commonly found in scientific and technical illustrations using 3D models.

The illustration shown in Fig. 6(a) depicts the female reproductive system. The purpose of the illustration was to clearly show internal reproductive organs while indicating their placement within the body. The illustrator used 2D renditions of the individual elements which were combined in Adobe Photoshop. First, the body contours were placed on the bottommost layer and a drop shadow was added to lift the image off the background. The pelvis was then added as a second layer, its opacity was lowered, and a drop shadow filter was applied. Additionally, a mask layer was added to preserve the contour of the body around the genital area. Reproductive organs were added as a third layer and a layer mask was employed to lower the opacity of the uterus as it passes behind the pelvis. Instead of completely masking out the structures behind the pubic symphysis, the artist chose to keep this area slightly visible while still indicating to the viewer that these regions are located behind the pelvis.

Using our approach the process of creating a similar illustration, as shown in Fig. 6(b), is analogous. Given a suitable 3D model, the user assigns the respective objects to individual layers. The same three layers are used: body contours, pelvis, and internal reproductive organs. The opacity of the pelvis layer is adjusted and our masking tool is applied, just like in the 2D workflow. However, instead of manually specifying a mask to achieve the desired see-through effect for the internal structures, pelvis and reproductive organs simply form a visibility chain

Please cite this article as: Bruckner S, et al. Hybrid visibility compositing and masking for illustrative rendering. Computers and Graphics (2010), doi:10.1016/j.cag.2010.04.003

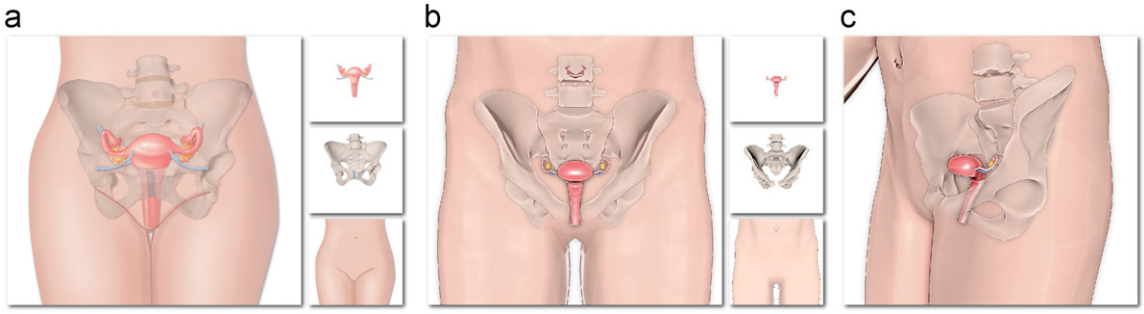

Fig. 6. Female reproductive system: (a) 2D illustration generated using Adobe Photoshop; (b) 3D illustration generated using our compositing approach; and (c) different viewpoint of the 3D illustration. *Illustrations courtesy of ©Kari C. Toverud, MS, CMI.*

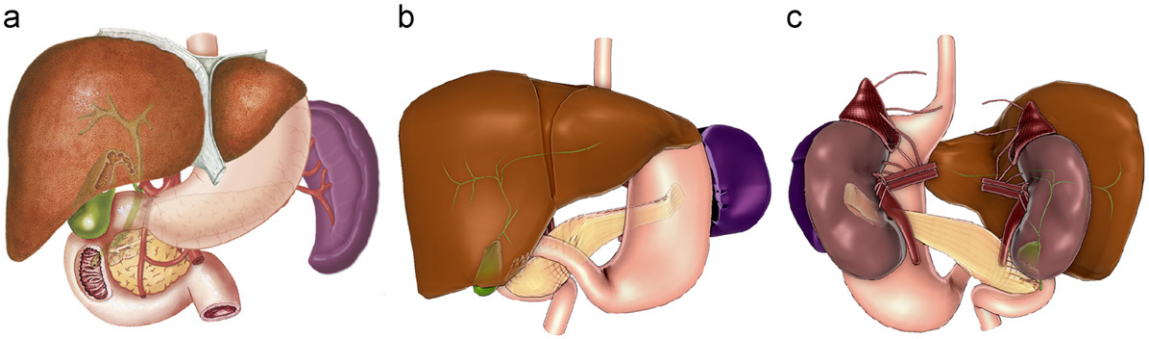

Fig. 7. Upper gastrointestinal tract: (a) 2D illustration generated using Adobe Photoshop; (b) 3D illustration generated using our compositing approach; and (c) different viewpoint of the 3D illustration. *Illustrations courtesy of ©Kari C. Toverud, MS, CMI.*

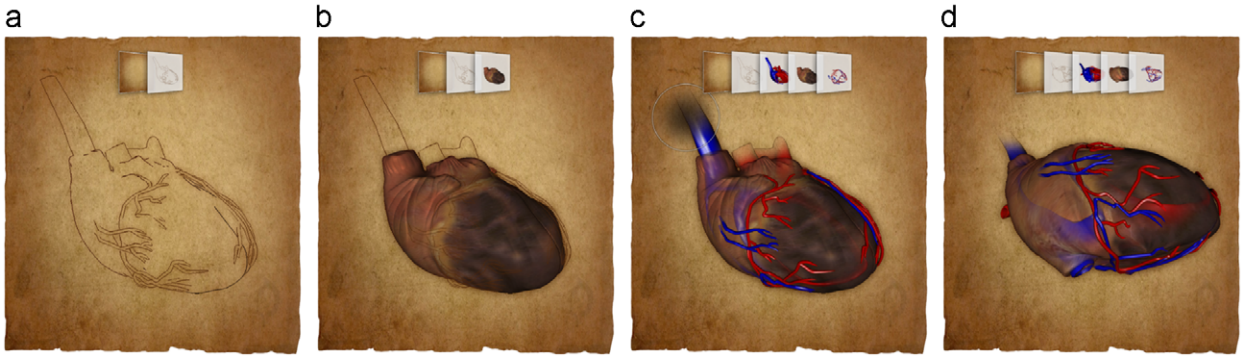

Fig. 8. Generation of a 3D illustration of the human heart: (a) line drawing layer; (b) combination with muscle layer; (c) applying masking; and (d) after rotation.

which is combined with the body layer using the *over* operator. This enables the generation of novel views without requiring any changes, as shown in Fig. 6(c). Fig. 7 depicts a further example of how our approach allows the user to employ the same effects as in 2D illustrations (see Fig. 7(a) and (b)), but enables the easy generation of novel views with the same hybrid visibility order. Additional illustrations showing the same topic are therefore easily created, as demonstrated in Fig. 7(c).

In Fig. 8(a), we show an example of a human heart model rendered as a line drawing. Then, in Fig. 8(b), a layer depicting the pericardium (heart muscle) is added. In Fig. 8(c), additional layers depicting arterial and venous system are enabled. As no visibility overrides are required, all heart layers form a visibility chain which terminates with the *over* operator. The ω parameter of the pericardium is adjusted to make the inner structures of the heart close to the surface shine through. Masking is then applied to the line drawing and vascular layers making them fade into the background. Fig. 8(d) demonstrates that the brushed mask smoothly translates to other viewpoints.

Fig. 9 depicts an illustration of the human vascular system. There are five layers: background, skin, skeleton, arterial system, and venous system. The background layer resides at the bottom, the skin layer uses the *over* operator. Skeleton, arteries, and veins form a visibility chain. The vein layer terminates the chain using the *over* operator. Masking has been applied to make these layers smoothly fade into the skin. Additionally, the ω parameter of the skeleton layer has been adjusted to show blood vessels passing closely behind bones.

The illustration depicted in Fig. 10 demonstrates that our approach can also be used to easily generate interactive effects such as magic lenses. This setup contains two layers generated by volume rendering of a human hand CT dataset. The first one uses non-photorealistic isosurface rendering of the skin while the second one uses maximum intensity projection to achieve an X-ray effect. The magnifying glass model is split into two layers: lens and body. The X-ray layer resides in a separate layer group with the lens and uses the *plus* operator and a negative β parameter so only the parts of the layer located behind the lens are added. The result forms a

Please cite this article as: Bruckner S, et al. Hybrid visibility compositing and masking for illustrative rendering. Computers and Graphics (2010), doi:10.1016/j.cag.2010.04.003

visibility chain with skin and body which is combined with the background layer using the *over* operator.

7. Discussion

One goal of the work discussed in this paper was to provide a practical way of incorporating illustrative rendering techniques into the workflow of illustrators and artists. Many high-quality illustrative techniques have been presented in recent years. However, these methods frequently rely on particular data structures and algorithms which makes their integration into professional software tools difficult. While it is possible to generate many of the effects presented in this paper using specialized algorithms, our contribution is a general concept which allows seamless integration of 3D layers into 2D software. Traditional compositing tools see the increasing need to provide 3D integration. The demand for this kind of functionality is evidenced by its recent incorporation into widely popular applications such as Adobe Photoshop. However, in current implementations 3D layers behave as 2D images with respect to other layers—there is no way to make use of the visibility information between two 3D layers. Using our approach, this functionality could be greatly extended in a non-invasive manner while still covering the full range of 2D operations.

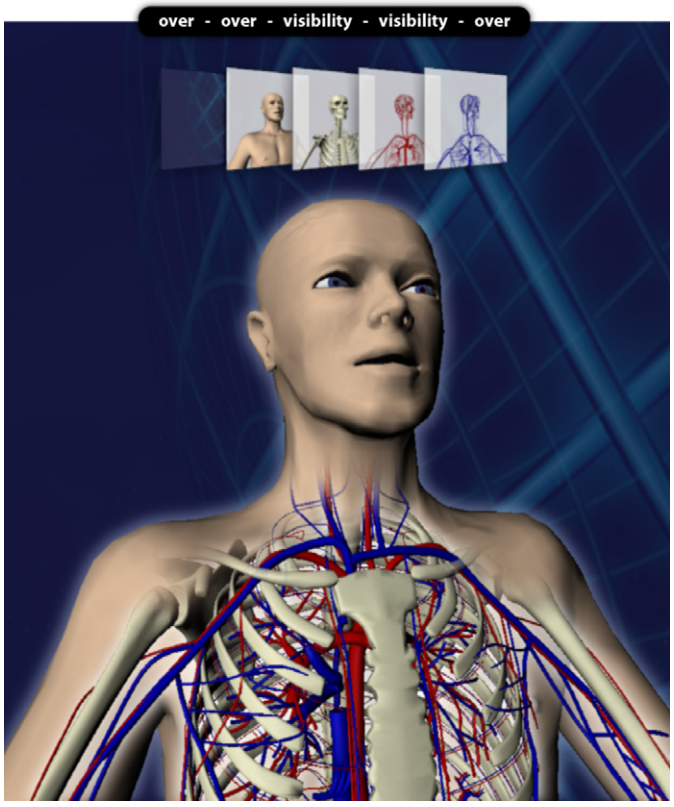

Fig. 9. Interactively generated 3D illustration of the human vascular system.

In a 2D workflow, artists frequently employ manually drawn layer masks to emulate visibility information for the generation of see-through and ghosting effects. Visibility chains and occlusion-based blending have shown to be effective tools to reduce the number and complexity of manually specified layer masks by taking advantage of the existing spatial information. Based on the artist's intent, however, additional masking is still useful in many cases. Our hybrid image-space/object-space method for brushing layer masks closes this gap by also exploiting spatial information for this operation. During experiments we found that our masking approach is very similar in behavior to analogous tools in 2D applications. One has the impression to be manipulating an image, but masking information smoothly transfers to nearby viewpoints in a consistent manner. However, it is impossible to predict the intent of the user in all cases. For example, if a user paints on one side of a radially symmetric object, it might be desirable to automatically have the object appear transparent from all viewing directions along the axis of symmetry. As our system is completely interactive, these cases can be easily resolved by rotating the object and placing new strokes. Our general approach also allows easy integration of additional specialized tools for this purpose.

We received positive feedback on the utility of our prototype implementation and the general concept of hybrid visibility for generating illustrations. As shown in the examples in Section 6, our approach is capable of closely mimicking the traditional 2D compositing workflow. The ability of being able to alter an

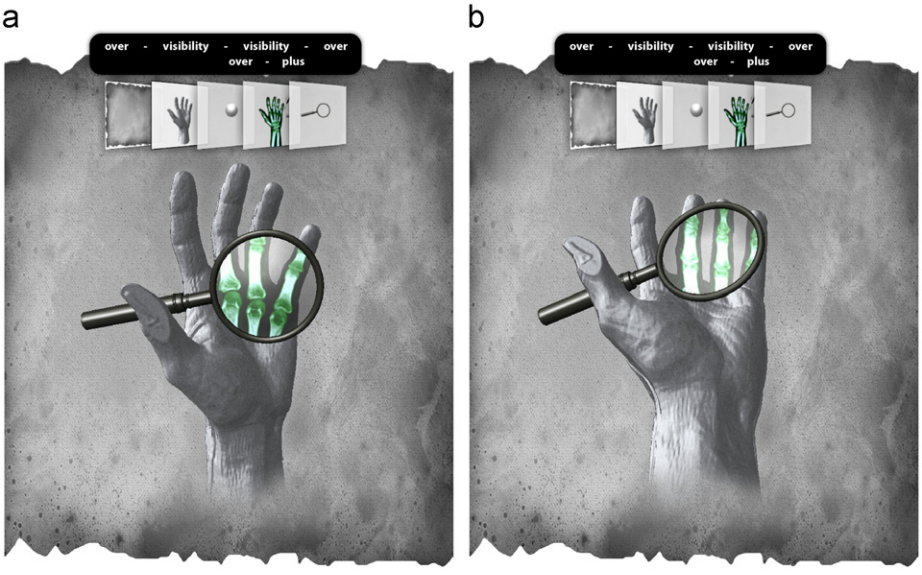

Fig. 10. Two viewpoints for a magic lens effect generated using our compositing framework.

Please cite this article as: Bruckner S, et al. Hybrid visibility compositing and masking for illustrative rendering. Computers and Graphics (2010), doi:10.1016/j.cag.2010.04.003

existing illustration by modifying the viewpoint was greatly appreciated and, given the availability of suitable 3D models, considered to have a high potential of speeding up the production process. However, as our research prototype does not encompass the full range of functionality featured in standard software packages, only the integration of the proposed concepts into a commercial product is likely to facilitate widespread adoption.

8. Conclusion

In this paper we presented a simple concept for illustrative compositing of dynamic 3D layers in an interactive environment. Our approach enables a wide variety of different effects such as selective occlusion overrides commonly employed in the generation of scientific and technical illustrations. With our method these operations can be performed in 3D using an extension of the familiar layer metaphor. We also proposed a novel technique for masking of 3D layers. It enables the generation of opacity transitions which smoothly extend beyond a single viewpoint. The presented framework makes minimal assumptions about the underlying algorithms used for rendering the individual layers. By exploiting the performance of current graphics hardware, high-quality illustrations of 3D objects can be generated interactively.

Acknowledgments

The work presented in this publication was supported by the Austrian Science Fund (FWF) Grant no. P21695—ViMaL: The Visualization Mapping Language, the Norwegian Research Council (Project no. 193170/S10) and by the MedViz Initiative in Bergen, as well as the iCORE/Foundation CMG Industrial Research Chair in Scalable Reservoir Visualization and the Discovery Grants Program from the Natural Sciences and Engineering Research Council of Canada.

We want to thank CF Lietzau 3D Special (http://www.anatomium.com) for permission to use the P1 human anatomy model depicted in Figs. 6–9. Furthermore, we want to express our gratitude to Kari C. Toverud for providing her time and expertise.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version of 10.1016/j.cag.2010.04.003.

References

[1] Smith AR. Alpha and the history of digital compositing. Technical Memo 7, Microsoft Corporation, Redmond, Washington.
[2] Wallace BA. Merging and transformation of raster images for cartoon animation. ACM Computer Graphics 1981;15(3):253–62.
[3] Porter T, Duff T. Compositing digital images. ACM Computer Graphics 1984;18(3):253–9.
[4] Duff T. Compositing 3-D rendered images. ACM Computer Graphics 1985; 19(3):41–4.
[5] McCann J, Pollard NS. Local layering. ACM Transactions on Graphics 2009;28(3):1–7.
[6] Lengyel J, Snyder J. Rendering with coherent layers. In: Proceedings of ACM SIGGRAPH '97, 1997. p. 233–42.
[7] Schaufler G. Nailboards: a rendering primitive for image caching in dynamic scenes. In: Proceedings of the Eurographics Workshop on Rendering '97, 1997. p. 151–62.
[8] Shade J, Gortler S, He L-W, Szeliski R. Layered depth images. In: Proceedings of ACM SIGGRAPH '98, 1998. p. 231–42.
[9] Dooley D, Cohen MF. Automatic illustration of 3D geometric models: lines. In: Proceedings of the symposium on interactive 3D graphics '90, 1990. p. 77–82.
[10] Dooley D, Cohen MF. Automatic illustration of 3D geometric models: surfaces. In: Proceedings of IEEE Visualization '90, 1990. p. 307–14.
[11] Seligmann DD, Feiner SK. Automated generation of intent-based 3D illustrations. In: Proceedings of ACM SIGGRAPH '91, 1991. p. 123–32.
[12] Feiner SK, Seligmann DD. Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations. The Visual Computer 1992;8(5 & 6): 292–302.
[13] Seligmann DD, Feiner SK. Supporting interactivity in automated 3D illustrations. In: Proceedings of the international conference on intelligent user interfaces '93, 1993. p. 37–44.
[14] Preim B, Ritter A, Strothotte T. Illustrating anatomic models—a semi-interactive approach. In: Proceedings of the international conference on visualization in biomedical computing '96, 1996. p. 23–32.
[15] Diepstraten J, Weiskopf D, Ertl T. Transparency in interactive technical illustrations. Computer Graphics Forum 2002;21(3):317–25.
[16] Diepstraten J, Weiskopf D, Ertl T. Interactive cutaway illustrations. Computer Graphics Forum 2002;22(3):523–32.
[17] Owada S, Nielsen F, Nakazawa K, Igarashi T. A sketching interface for modeling the internal structures of 3D shapes. In: Proceedings of the international symposium on smart graphics '03, 2003. p. 49–57.
[18] Owada S, Nielsen F, Okabe M, Igarashi T. Volumetric illustration: designing 3D models with internal textures. ACM Transactions on Graphics 2004;23(3): 322–8.
[19] Viola I, Kanitsar A, Gröller ME. Importance-driven feature enhancement in volume visualization. IEEE Transactions on Visualization and Computer Graphics 2005;11(4):408–18.
[20] Bruckner S, Gröller ME. VolumeShop: an interactive system for direct volume illustration. In: Proceedings of IEEE visualization '05, 2005. p. 671–8.
[21] Rautek P, Bruckner S, Gröller ME. Semantic layers for illustrative volume rendering. IEEE Transactions on Visualization and Computer Graphics 2007; 13(6):1336–43.
[22] Rautek P, Bruckner S, Gröller ME. Interaction-dependent semantics for illustrative volume rendering. Computer Graphics Forum 2008;27(3): 847–54.
[23] Cole F, DeCarlo D, Finkelstein A, Kin K, Morley K, Santella A. Directing gaze in 3D models with stylized focus. In: Proceedings of the eurographics symposium on rendering '06, 2006. p. 377–87.
[24] Kalkofen D, Mendez E, Schmalstieg D. Interactive focus and context visualization for augmented reality. In: Proceedings of the IEEE international symposium on mixed and augmented reality '07, 2007. p. 1–10.
[25] Li W, Ritter L, Agrawala M, Curless B, Salesin D. Interactive cutaway illustrations of complex 3D models. ACM Transactions on Graphics 2007; 26(3):3:11–31:11.
[26] Li W, Agrawala M, Curless B, Salesin D. Automated generation of interactive 3D exploded view diagrams. ACM Transactions on Graphics 2008;27(3): 1–7.
[27] Raman S, Mishchenko O, Crawfis R. Layers for effective volume rendering. In: Proceedings of the international symposium on volume and point-based graphics '08, 2008. p. 81–8.
[28] Krüger J, Schneider J, Westermann R. Clearview: an interactive context preserving hotspot visualization technique. IEEE Transactions on Visualization and Computer Graphics 2006;12(5):941–8.
[29] Smith AR. A sprite theory of image computing. Technical Memo 5, Microsoft Corporation, Redmond, Washington.
[30] Luft T, Deussen O. Real-time watercolor illustrations of plants using a blurred depth test. In: Proceedings of the international symposium on non-photorealistic animation and rendering '06, 2006. p. 11–20.
[31] Westover L. Footprint evaluation for volume rendering. In: Proceedings of ACM SIGGRAPH '90, 1990. p. 367–76.
[32] Luft T, Colditz C, Deussen O. Image enhancement by unsharp masking the depth buffer. ACM Transactions on Graphics 2006;25(3):1206–13.

Please cite this article as: Bruckner S, et al. Hybrid visibility compositing and masking for illustrative rendering. Computers and Graphics (2010), doi:10.1016/j.cag.2010.04.003