Dynamic Transparency for 3D Visualization: Design and Evaluation

Niklas Elmqvist[1], Ulf Assarsson[2] and Philippas Tsigas[2]

[1] *School of Electrical and Computer Engineering, Purdue University, West Lafayette, USA*
[2] *Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden*

***Abstract*—Recent developments in occlusion management for 3D environments often involve the use of dynamic transparency, or "virtual X-ray vision", to promote target discovery and access in complex 3D worlds. However, there are many different approaches to achieving this effect and their actual utility for the user has yet to be evaluated. Furthermore, the introduction of semi-transparent surfaces adds additional visual complexity that may actually have a negative impact on task performance. In this paper, we report on an empirical user study investigating these human aspects of dynamic transparency. Our implementation of the technique is an image-space algorithm built using modern programmable shaders to achieve real-time performance and visually pleasing results. Results from the user study indicate that dynamic transparency provides superior performance for perceptual tasks in terms of both efficiency and correctness. Subjective ratings are also firmly in favor of the method.**

***Index Terms*—Occlusion Management, Virtual X-Ray, 3D Visualization, Dynamic Transparency, GPU-Accelerated Visualization.**

I. INTRODUCTION

The ability to utilize the full 3D space as a canvas for information-rich [2] visualization applications is a mixed blessing—while 3D space on the one hand supports an order of magnitude of more layout opportunities for visual elements than 2D space, designers are on the other hand faced with a number of new challenges arising from the nature of 3D space that do not occur in 2D. More specifically, they must consider the *visibility* of objects when users wish to discover relevant objects, as well as their *legibility* when the user wants to access information encoded in a particular object [3]. For instance, whereas objects that do not intersect can never occlude each other in 2D space, this can occur in 3D space depending on the viewpoint and the relative position between the objects. Controlling the impact of this effect is known as *occlusion management* [4]. *Dynamic transparency*, also known as virtual X-Ray [4], has recently been proposed as a solution to this problem. The method achieves this by turning intervening surfaces semi-transparent as the user moves through the 3D world (see Fig. 1 for an example).

However, this approach may instead introduce additional visual complexity and reduce the user's depth perception. Furthermore, the actual utility of these techniques remains unknown. In this paper, we evaluate the usefulness of dynamic transparency for solving visual tasks in both abstract and realistic environments. Note that dynamic transparency cannot be realized using the standard model for transparency, and no real-time performance algorithm exists in the literature that fulfills all of our requirements. Therefore, we also present an algorithm for dynamic transparency that performs occlusion detection in the image space with real-time rendering performance. The effect is somewhat akin to the "X-ray vision" of a superhero.

We performed the evaluation by constructing two application examples depicting common scenarios within the visualization problem domain: an abstract 3D environment of simple geometric primitives similar to information visualization applications, and a 3D virtual walkthrough application for a complex building environment. We then conducted a controlled experiment in these two scenarios where we compared the time and correctness performance of subjects solving tasks using our technique to solving tasks while not using it.

The contributions of this paper are the following: (i) a model for dynamic transparency that captures a natural way of achieving high efficiency for perceptual tasks; (ii) an efficient image-space algorithm for dynamic transparency using programmable graphics hardware; and (iii) results from our formal user evaluation studying the impact of dynamic transparency on time performance and correctness for visual tasks involving discovery, access, and spatial relation of objects in 3D environments.

An earlier version of this work previously appeared at the INTERACT 2007 conference in Rio de Janeiro, Brazil [1]. The present article represents a consolidation of the whole research project, including an extended theoretical framework for dynamic transparency, a survey of existing dynamic transparency techniques, and all of the implementation details and study results that could not be included in the conference paper. In addition, this article incorporates all of the comments and feedback we have received on the work since its original conference presentation.

II. RELATED WORK

Bowman et al. define the concept of *information-rich virtual environments* (IRVEs) [2], a combination of information visualization within the framework of virtual environments, yet many

Manuscript Received on November 24, 2007.
E-mail: elm@purdue.edu

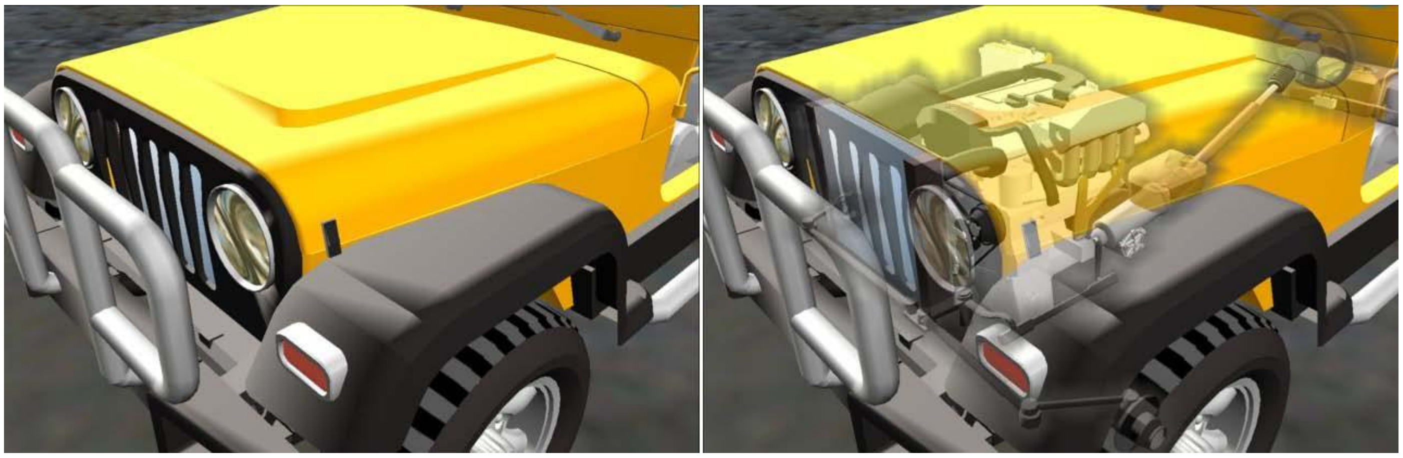

Fig. 1. Dynamic transparency uncovering an engine inside a jeep. (The left picture shows standard 3D rendering, the right picture shows the scene with dynamic transparency active.) (See Color Plate 11, 12)

of the challenges [3] presented in their work are directly applicable to any kind of 3D visualization application. The theoretical model for our technique is partly inspired by their work

2.1 Non-Photorealistic Rendering

Dynamic transparency techniques sacrifice some visual realism to increase the user's potential understanding of a 3D scene. In that sense, these techniques can be regarded as non-photorealistic rendering (NPR). Here, the emphasis lies on conveying important structural or semantic information about a 3D scene or visualization, not necessarily a high-quality visual appearance.

NPR has in recent years become a popular research area in computer graphics; examples include painterly rendering [5], hatching [6], and edge and silhouette extraction [7]. Although typically not employed directly for visualization, the approach has found use in computer-generated technical illustrations. Gooch et al. [8, 9] describe the use of NPR-based silhouette extraction and tone shading techniques for automatic and interactive technical illustrations. Nienhaus and Döllner [10] present the Blueprints method, which employs edge extraction and depth layering to outline and enhance both visible and occluded features of 3D models. Freudenberg et al. [11] introduce another tone-based NPR primitive that may be useful in this context.

2.2 Transparency

The general linear model for transparency in computer graphics was introduced by Kay and Greenberg [12]. However, transparent surfaces must be rendered in depth order to achieve correct results. Everitt [13] discusses the *depth peeling* image-space algorithm for achieving this on modern graphics hardware based on the virtual pixel map concepts introduced by Mammen [14] and the dual depth buffers by Diefenbach [15]. The Blueprints [10] technique mentioned earlier uses depth peeling to outline perceptually important geometrical features of complex models using transparency and edge detection. However, depth peeling is a computationally demanding method and interactive frame rates can only be achieved for a relatively low depth complexity. Even simple test scenes can have a depth complexity of over 15 (counting only front faces). This is currently much too high for real-time rendering using depth peeling, both regarding speed and memory cost, since each layer corresponds to a frame buffer.

Diepstraten et al. introduce view-dependent transparency [16] where NPR transparency techniques are employed for interactive technical illustrations. However, Diepstraten employs a fixed two-pass depth peeling step to uncover only the two foremost layers of transparent surfaces. Objects that are hidden by more than two layers of surfaces will remain hidden.

The use of alpha blending for exposing hidden content in windowing systems is well-known (e.g. [17]) but may result in loss of depth cues and legibility. Gutwin et al. [18] explore a dynamically adapting transparency mechanism based on the distance to the mouse cursor to avoid this. Multiblending [19] is a more advanced blending approach where many different image processing techniques are applied separately to different classes of graphical components. Ishak and Feiner [20] takes this a step further by introducing a content-aware transparency mechanism that dynamically adapts opacity depending on the importance of various parts of a window. Smooth gradients are employed to emphasize the continuity of the transparent objects and give some depth information. In addition, their system supports a magic lens-like [21] focus filter.

Semi-transparency is also commonly used in 3D games and virtual environments to allow users to see through occluding surfaces. Chittaro and Scagnetto [22] investigate this practice and conclude that see-through surfaces are more efficient than normal 3D navigation but not as efficient as bird's-eye views.

2.3 Cut-Away and Break-Away Views

One popular technique for traditional paper-based technical illustrations is called *cut-away* views, where parts of the depicted object are cut away to reveal interior objects that would otherwise be hidden. Diepstraten et al. present their work on computer-based cut-away illustrations [23], where a small set of rules are presented to generate an effective model for interactive technical visualization. Cut-away views are not view-dependent, however, and thus do not qualify as a *dynamic* transparency method.

In the same paper, the authors also present *break-away* views, where interior objects are made visible through the surface of containing objects through image-space holes. This technique is simplified by semantic knowledge of inside and exterior objects, and the fact that the break-away view is realized by a single hole. Their approach is to compute the convex hull of interior objects in a pre-processing step and use it as a clipping volume when rendering. However, this strategy does not handle the case when several targets line up and occlude each other.

Looser et al. [24] describe a 3D magic lens implementation for Augmented Reality that supports information filtering of a 3D model using the stencil buffer, allowing the user to utilize a looking glass to see through the exterior of a house and into its interior, for instance. This approach relies on the 3D model having semantically differentiated parts.

Coffin and Höllerer [25] present a similar technique with active interaction where the user is controlling a CSG volume that is dynamically subtracted from the surrounding world geometry, again using the stencil buffer. This work does not rely on any semantic target information at all and facilitates exploratory interaction. However, the depth of the volume cutout is limited and user-controlled, and no depth cues from the world geometry are retained other than the cutout border area.

2.4 Importance-Driven Rendering

A generalization of cut-away views, importance-driven rendering assigns importance values to individual objects in a 3D scene and renders a final image that is a composite of not only the geometrical properties of the objects, but also their relative importance. This can be used to achieve various effects for expressing spatial and semantic information about the scene; Viola et al. employ it for importance-driven volume rendering [26] (IDVR) to actively reduce inter-object occlusion. However, Viola's implementation does not support real-time performance, which is vital for interactive visualization applications.

In follow-up work, Viola et al. [27] present a model for attention-driven volume rendering, deriving characteristic viewpoints and finding a transparency balance between context and focused areas, but again lacking real-time performance.

2.5 General Occlusion Management

Our definition of dynamic transparency forms one specific strategy for occlusion management [4], the control of 3D views and worlds to reduce the impact of inter-object occlusion on visual perception tasks. Dynamic transparency mainly utilizes the image space by changing the transparency level of occluding distractors, but there exist a large number of additional strategies. Examples include approaches using view space [28], [29], object space [30, 31], and temporal space [32], guaranteed visibility [33], etc.

III. GENERAL DYNAMIC TRANSPARENCY

Dynamic transparency is based on the idea of guaranteeing the visibility of important targets regardless of occluding distractors. This is done by dynamically changing the transparency of intervening surfaces and objects. In this section, we present a model for the dynamic transparency approach. See Elmqvist [4] for a more in-depth treatment of general occlusion management.

3.1 Model

We represent the 3D world U by a Cartesian space $(x, y, z) \in R^3$. Objects in the set O are volumes within U (i.e. subsets of U) represented by boundary surfaces (typically triangles). The user's viewpoint $v = (M, P)$ is represented by the view and projection matrices M and P.

An object can be flagged either as a *target*, an information-carrying entity, or a *distractor*, an object with no intrinsic information value for the current task. Importance flags can be dynamically changed as the user task changes. Occluded distractors pose no threat to any analysis tasks performed in the environment, whereas partially or fully occluded targets do, resulting in potentially decreased performance and correctness. The surfaces defining an object volume have a transparency (alpha) function $\alpha(x) \in [0, 1]$. A line segment r passing through a surface at point p is *not* blocked if $\alpha(p) < 1$ and the cumulative transparency value α_r of the line segment is less than one. Passing through a surface increases the cumulative transparency of the line segment accordingly (multiplicatively or additively, depending on the transparency model).

3.2 Visual Tasks

The occlusion problem occurs in the following three *visual perception tasks*:

- *Target discovery:* finding targets $t \in O$ in the environment;
- *Target access:* retrieving graphically encoded information associated with each target; and
- *Spatial relation:* relating the spatial location and orientation of a target with its context.

More concretely, occlusion affects both the visibility and legibility of objects in a 3D environment. This has an impact on all of the above visual tasks.

3.3 Basic Mechanism

We define our model for dynamic transparency using four axioms that alter the standard rendering of a 3D environment:

- *Guaranteed target visibility:* managing visibility of targets;
- *Entity selection:* deciding which entities to turn semi-transparent;
- *Impenetrability:* exceptions allowing for impenetrable surfaces; and
- *Self-occlusion:* supporting object atomicity.

1) *Guaranteed target visibility*: All targets in the world U should be visible from any given viewpoint v. This is the most basic definition of dynamic transparency and it directly supports the target discovery task. It stipulates that no targets should be fully occluded from any viewpoint in the world. A target may still be hidden from the user if it falls outside the current view.

2) *Entity selection*: An occluded object is made visible by changing the transparency level of all occluding entities $e \in E$ from opaque ($\alpha(e)=1$) to transparent ($\alpha(e)= \alpha_T$). This axiom describes the mechanics of which objects should be turned transparent to uncover occluded targets. The selection of the set E is not specified. Depending on the application, this could be a

convex hull, circle, or ellipse that encloses the occluded object, or a projection of the target's outline on the viewing plane.

3) *Impenetrability*: Entities (objects, surfaces, or pixels) can be made impenetrable and will never be made transparent. This axiom provides a useful exception to the initial one—in some cases, we may want to limit the extent of the dynamic transparency mechanism using impenetrable surfaces (and objects). For example, it may not make sense to turn walls bounding a visualization transparent, or some parts of a visualization should be seen as atomic until the viewer comes sufficiently close (a little like semantic zooming [34] for dynamic transparency).

4) *Self-occlusion:* Targets are allowed to self-occlude. This is another refinement of the previous axioms: dynamic transparency operates on whole objects. Even if a part of a target is occluded by other parts of itself, none of its surfaces will be made transparent to show this.

3.4 Context and Granularity

The general idea behind dynamic transparency is simple: we can reduce the impact of occlusion by dynamically changing the transparency (alpha) value of individual entities occluding (either partially or fully) a target object. This results in fewer fully occluded objects in the environment and thus directly affects the object discovery visual task.

However, it is important to remember that distractor objects, while not vital for the current task, still provide context about the 3D environment that is useful for many tasks. The level of context and depth cues needed depends on the particular application.

In order to support this, our model of dynamic transparency can operate on several levels of granularity:

- *Region-level:* modify the transparency of a set of objects (typically grouped by regions in the 3D world);
- *Object-level:* modify the transparency of entire objects (e.g. vehicles, furniture, people);
- *Surface-level:* modify the transparency of individual 3D surfaces (usually triangles) that make up objects; and
- *Pixel-level:* modify the transparency of individual pixels that are rendered for each surface.

To give additional context, even occluding surface parts are not made fully transparent, but are set to a threshold alpha value α_T in order to shine through slightly in the final image. There is a tradeoff here: the use of semi-transparent occluders will make object access difficult since intervening surfaces will distort targets behind them. However, it is a necessity in order to maintain the user's context of the environment.

Object-level dynamic transparency is often easier to implement for a particular application and is typically less computationally expensive than surface-level or pixel-level dynamic transparency, but for some applications this may be too coarse a definition. In a 3D environment that includes a few large distractors, a very small target will cause whole distractors that happen to occlude it to be made transparent. The appearance and disappearance of these distractors may be confusing and disorienting for the user. On the other hand, in a 3D environment consisting of a large number of small objects (both targets and distractors), an object-level implementation may very well be sufficient.

Surface-level and pixel-level dynamic transparency typically retain increasingly more context since distractors are made transparent per-surface and per-pixel, respectively. In the example above, only a few triangles or a few pixels of the large, occluding distractor would be made transparent to show the small target. For applications like this, this functionality may be vital in order to retain important contextual information.

3.5 Operation Modes

Dynamic transparency can be used in either an active or a passive mode. *Passive* mode is when dynamic transparency is performed on the whole view visible to the user; all occluded objects are revealed automatically without the user having to do anything. This may cause quite a severe impact on the visual quality of the scene, however, and make it difficult for the user to gain an understanding of its layout and structure.

In *active* mode, on the other hand, the user controls a searchlight (essentially a 3D magic lens [21]) on the image plane of the scene specifying on which parts of the world dynamic transparency should be active. This is a less obtrusive mode of operation than passive mode and has less impact on the visual quality of the scene, but on the other hand requires direct manipulation and active discovery by the user.

3.6 Layer Control

The standard dynamic transparency mechanism, as described above, will peel away all intervening surface layers to reveal occluded targets in a scene. However, in some cases, we may want to control the maximum number of layers to be peeled away by the mechanism. By introducing this capability to the specification of dynamic transparency, we allow for special classes of visualizations, such as the one-layer depth technical illustrations discussed in Diepstraten et al. [16, 23].

IV. DYNAMIC TRANSPARENCY FOR VISUALIZATION

We are interested in the use of dynamic transparency for occlusion management in 3D visualization applications. This imposes a number of additional requirements on dynamic transparency implementations. Here follows a list of these, including their motivations:

R1. *View-dependent:* The technique should be dynamic, guaranteeing target visibility regardless of viewpoint since we cannot control the user's movement.

R2. *Unlimited depth:* A particular visualization application may have many targets or distractors lining up, requiring us to be able to handle uncovering targets that are hidden by a potentially large number of distractors.

R3. *Pixel-level granularity:* No assumptions can be made on the 3D environments of the visualizations, so we need pixel-level dynamic transparency to guarantee visibility in all situations.

R4. *Polygonal 3D representations:* We handle only polygonal 3D representations with no particular semantic information for objects beyond target and distractor information. The visualization applications we are concerned with have these properties—we disregard volume representations.

R5. *Real-time performance:* Visualization applications require interactive performance, so the dynamic transparency mechanism must allow for real-time rendering.

R6. *Passive mode:* We cannot require our users to manually control the technique since this will cause them to run the risk of missing important targets. Therefore, we stipulate that the technique must support passive mode.

R7. *Context and depth cues:* A dynamic transparency method must maximize the contextual information for each uncovered target and minimize the impact of decreased depth perception. Otherwise, users will have difficulties interpreting the visualization.

In this section, we discuss how existing dynamic transparency techniques fulfill these requirements. Furthermore, we look into two particular issues that are of special importance to the employment of dynamic transparency in 3D visualization: increased visual complexity and decreased depth cues.

4.1 Existing Techniques

As discussed in the related work (Section II), there exist a number of dynamic transparency techniques in the literature. Not all of these are suitable for use in visualization applications, however. In this section we will review the most important of these techniques in light of the requirements outlined above.

First of all, the four axioms presented in Section III cannot be fulfilled with standard transparency even if graphics hardware would support correct per-pixel sorting with transparency blending in back-to-front order, which it does not. We cannot simply make distractors covering targets transparent, since we want objects to self-occlude. That is, objects should still be rendered as solids with only the front-most surfaces visible. Back faces, insides, and self-occluded parts of closed surfaces of an object should not be rendered. This cannot be solved with simple back-face culling and depth culling.

Object-wise, the depth culling should only pass the frontmost surface elements per pixel. If these elements occlude a target or the close proximity of a target, they should be correctly blended in back-to-front order with a user-specified alpha in front of the target and fading to no transparency at a specific number of pixels from the target.

Table I reviews the existing dynamic techniques discussed earlier in this paper and summarizes whether they fulfill our requirements for use in visualization applications.

In general, standard transparency-based techniques (Section IIB) are insufficient for visualization purposes. Either they do not provide unlimited depth (R2), or doing so results in non-real-time performance (R5). Furthermore, none of them are targeted towards occlusion management in visualization, so they provide little or no context and depth information (R7). Some transparency techniques are designed only for 2D representations (R4).

Cut-away and break-away views (Section II-C) come closer to the mark by removing intervening surfaces on a per-pixel level. However, most techniques in this class are not well-suited for general visualization applications because they do not support unlimited depth layers; typically, they cannot handle the situation when targets line up in front of other targets (in this case, we must ensure that the furthermost target is always visible). Also, many provide poor context and depth information (R7) and require that users have prior knowledge of the features they are looking for—if not, the user is forced to conduct an exhaustive visual and possibly spatial search.

Finally, while importance-driven rendering in theory is perhaps the most powerful approach to dynamic transparency, the only existing implementation (IDVR [26]) is targeted at volume and not polygonal representations (R4) and does not provide real-time performance (R5).

4.2 Visual Complexity

The dynamic transparency mechanism reduces occlusion by making distractors semi-transparent on-demand in order to expose hidden targets. However, doing so will have an impact on the visual realism and complexity of the resulting image. The image will look less realistic than without dynamic transparency (after all, being able to see through walls and objects is different from our normal vision), and there may also be an increased amount of information in the image (what used to be an empty corridor may now become a mosaic of objects contained in the offices adjoining the corridor).

We are interested in empirically evaluating exactly how much impact this increased visual complexity will have user performance in visualization applications. While dynamic transparency in theory will be of great benefit to visualizations, we want to study whether there is a drawback in practice.

4.3 Depth Cues

Occlusion is an important depth cue when perceiving a 3D scene, so dynamic transparency may clearly have an impact on the way users understand the world. For visualization applications, depth ordering is clearly vital when trying to understand the spatial structure of the 3D environment. Implementations of dynamic transparency must ensure that occlusion is not eliminated entirely, or they end up with situations where distant objects occlude nearby objects, so-called *reverse occlusion*.

Fortunately, human perception relies on many more cues besides occlusion to disambiguate depth, such as stereopsis, motion parallax, relative size, atmospheric perspective, texture gradient, etc [35]. Even if we weaken the occlusion cue, other depth cues will help the viewer to correctly perceive the 3D scene. Nevertheless, we want to empirically examine this.

4.4 Classification

Dynamic transparency for visualization is an instance of the *virtual X-Ray* [4] design pattern for 3D occlusion management. Using the terms of this taxonomy, the approach has the following properties:

- *Primary purpose:* discovery
- *Disambiguation strength:* containment
- *Depth cues:* low
- *View paradigm:* single view
- *Interaction:* passive (or active depending on operation mode)
- *Target invariances:* location and geometry (appearance distorted)

More specifically, the purpose of dynamic transparency is to

TABLE 1: VISUALIZATION REQUIREMENTS FOR EXISTING DYNAMIC TRANSPARENCY TECHNIQUES.

Technique	*R1*	*R2*	*R3*	*R4*	*R5*	*R6*	*R7*
depth-peeling [13]	✓	–	✓	✓	✓	✓	–
Blueprints [10]	✓	–	✓	✓	✓	✓	–
view-dependent transparency [16]	✓	✓	✓	✓	✓	✓	✓
2D dynamic transparency [18]	✓	✓	✓	–	–	✓	–
multiblending [19]	–	✓	✓	–	–	✓	–
content-aware free-space transparency [20]	✓	✓	✓	–	✓	✓	✓
see-through surfaces [22]	✓	–	–	✓	✓	–	✓
interactive cut-away views [16]	–	✓	✓	✓	✓	–	–
interactive break-away views [16]	✓	–	✓	✓	✓	✓	–
3D Magic Lenses in AR [24]	✓	–	✓	✓	✓	–	✓
interactive perspective cut-away views [25]	✓	–	✓	✓	–	✓	–
importance-driven volume rendering [26]	✓	✓	✓	–	–	✓	✓

aid users in discovering objects. Its disambiguation strength is very high, up to objects contained inside other objects, but this comes at the cost of low depth cues. The technique is based on a single view paradigm. Both passive and active interaction is supported. Target location and geometry are retained, but appearance may be distorted due to alpha blending, even for uncovered targets.

V. IMAGE-SPACE DYNAMIC TRANSPARENCY

Since none of the previously presented methods fulfill our requirements, we here present a new algorithm for 3D dynamic transparency: image-space dynamic transparency.

An important observation that follows from our model of occlusion from the previous section is that occlusion can be detected in the image space by simply shooting a ray through the scene for every pixel that is rendered and checking the order it intersects objects in the scene. In modern graphics hardware, this essentially amounts to detecting whenever we are overwriting pixels in the color buffer or discarding pixels due to depth testing. Thus, programmable fragment shaders are perfectly suited for realizing dynamic transparency.

However, correct blending of transparency is order-dependent, and thus our algorithm, as well as most algorithms for transparent objects, requires the objects to be rendered in back-to-front order. This is a classical problem, since current graphics hardware cannot do the sorting for us (although suggestions for solutions exist [36]). Usually, depth sorting is performed on triangle-level. In our algorithm, for non-intersecting objects, it is sufficient to sort on object-level for normal objects that are opaque by default. For intersecting objects, sorting must be performed on a per-triangle-level. Intersecting objects are however rare and usually non-physical. Objects fully contained within other objects, like objects in a suitcase or nested Russian dolls, can be correctly treated by specifying a fixed sort order (explained below).

5.1 Algorithm Overview

Our algorithm has the following basic rendering loop structure:

1) Sort the scene in back-to-front order (Painter's algorithm).
2) All objects are blended into the frame buffer using the alpha-channel, which defaults to 1 (opaque).
3) Blend targets into the framebuffer using a special *alpha map* that is rendered to an off-screen buffer.

Given our axioms and the requirements specified previously in this paper, the algorithm needs to fulfill these criteria:

- All parts of objects (target or distractor) in front of a target object should be transparent.
- Object should be rendered as solids, i.e. only the front-most surfaces should be visible. Thus, the objects cannot be rendered using transparency in an ordinary sense. Back-facing triangles, or more distant front-facing triangles, should not be visible through transparent frontmost triangles.
- There should be a gradual transition from no transparency to a predefined transparency in an n-pixel outline region around each target object.
- Some surfaces may be flagged to be impenetrable.

Algorithm 1 shows this algorithm in pseudocode.

```
Input:   set of groups $G$
Output:  correctly rendered dynamic transparency scene
BubbleSort(G), taking advantage of frame coherence
for all groups g ∈ G do
  for all objects o ∈ g do
    if o is a target then
      renderTargetObject()
    else
      renderDistractorObject()
```

Algorithm 1. Main rendering algorithm.

5.2 Rendering Order

We divide the scene into groups. By default, a group contains one object. All groups are sorted with respect to their center point, which is precomputed once. The sorting metric is the signed distance to the group from the eye along the view vector. This is better than sorting by only the distance from the eye, because the former corresponds better to how the z-buffer works. We use bubble sort, since frame coherency brings the resorting down to an average cost corresponding to *O(n)*.

If some objects are known never to have target objects behind them, like floors, ceilings, and outer walls, those objects can safely be rendered to the frame buffer first. This mechanism is also used for impenetrable surfaces.

In certain cases, like for Russian dolls, the sort order between the dolls should be from the innermost to the outermost. A fixed rendering order between the dolls is then user-defined by putting them into the same group with a predefined rendering order, for instance by the order of appearance in the group. In other words, the innermost doll should be rendered first and the outermost doll last. This results in correct transparency, since only the frontmost triangles of the dolls are visible (unlike for classic transparency). This mechanism gives the user a tool to specify which objects should be regarded as solids.

5.3 *Object Rendering*

Initial requirements for rendering both targets and distractors are that (i) the alpha buffer is initiated to 1 for each pixel at the start of each frame, (ii) rendering is done back-to-front on object level, and (iii) the alpha buffer contains the desired blending factor (transparency) at each pixel. Given these preconditions, we render distractor objects in the following way:

1) Render object to the z-buffer to mask out front faces.
2) Blend object to the color buffer.

The first step selects the frontmost surfaces of the object. The second blends these surfaces to the frame buffer, with blending using the alpha values stored in the frame buffer. These alpha values are 1 by default and less in front of, and in an *n*-pixel region around, target objects.

In contrast, target objects are rendered in the following way:

1) Render step 1 and 2 as for distractor objects.
2) Render alpha mask, i.e. multiplicatively blend an alpha mask (Fig. 2) to the alpha channel of the frame buffer.

The final step ensures that the rendered target is visible by creating a mask that essentially protects the target from being fully overdrawn by subsequently rendered objects.

5.4 *Alpha Mask*

As discussed in the general model of dynamic transparency, targets are made visible by changing the transparency level of a selection of the distracting entities occluding the target. In our image-space implementation, we perform entity selection on a per-pixel level. This is done using an *alpha mask* that modulates target pixels with the intervening distractor pixels.

Multiplying a constant alpha value to the pixels covered by the target object is easily done by simply rendering the object to the alpha-channel only and using a color with the alpha value set appropriately. Creating the alpha mask is a little trickier.

The alpha mask can be any type of shape exposing the underlying target, such as an ellipse or circle. We have experimented with all of these shapes (see Fig. 2). The drawback with circles is that a bounding circle for an oblong object will cause a high degree of wasted space being exposed. The same is true for an axis-aligned ellipse, and even an object-aligned ellipse will be problematic for a large cross shape.

Due to these reasons, we instead choose the expanded outline of the object with a transparency gradient as the alpha mask shape. To achieve this, we render to two external off-screen buffers alternately to create a border around the target object with a smooth transition to full opacity. The resolution can be allowed to be quite low; we use a size of 128 × 128. See Algorithm 2 for pseudocode for the alpha mask algorithm and refer to Algorithm 3 for the fragment shader code.

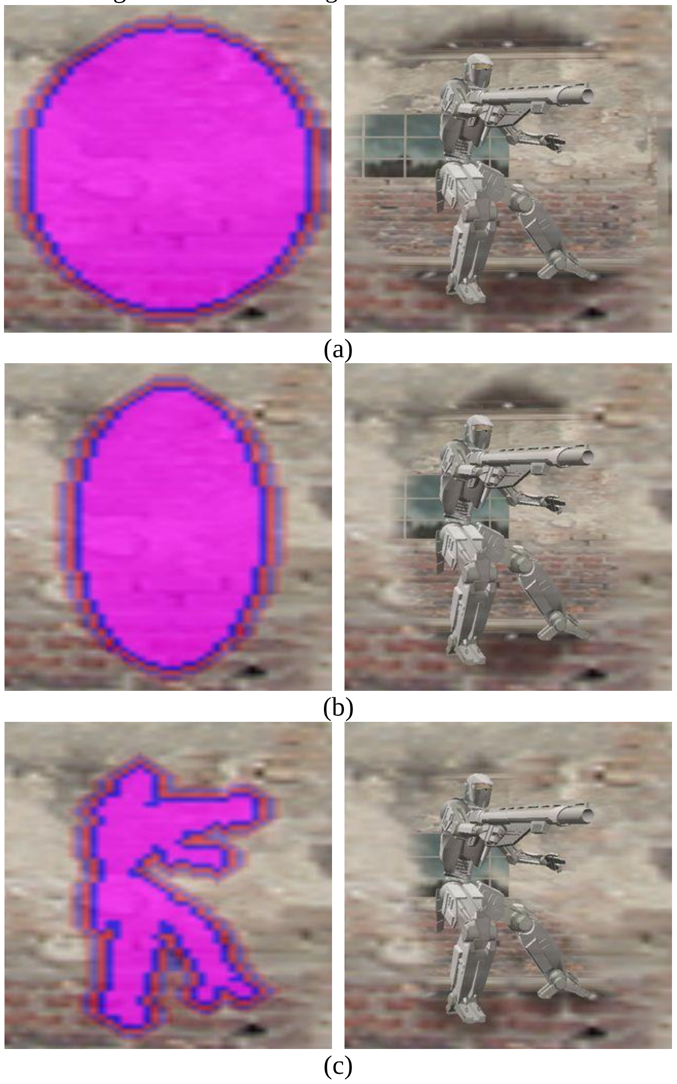

Fig 2. Alpha mask creation for an occluded target being made visible using dynamic transparency. (a) Circle. (b) Ellipse (axis-aligned). (c) Outline.

We found that it often looks better to have the transition from full opacity to a low start alpha value α_0 for the gradient outline, while keeping a higher threshold opacity α_T for pixels occluding the target. This maximizes both context and discovery.

Fig. 3 shows a more complex example with an ellipse-shaped alpha mask uncovering a tank hidden by a tree.

5.5 *Performance*

Table II shows the performance of three example applications with and without dynamic transparency active (an abstract environment, an architectural walkthrough, and the Jeep visualization example in Fig. 1). The test was performed on an Intel Pentium 4 desktop computer with 1 GB of memory running Microsoft Windows XP and equipped with an NVidia Geforce 7800 GTX graphics adapter. As can be seen from the measurements, only the Jeep application is fillrate-limited (the bottleneck seems to be buffer switching). For the Walkthrough application, we are performing dynamic transparency on 50 complex objects, so 11 FPS is acceptable, if not quite real-time.

```
Input:   target object $o$, mask width $n$, two buffers $B_1$ and $B_2$.
Output:  $128 \times 128$ alpha mask blended to the frame buffer.
1  Enable buffer $B_1$.
2  Render the target object $o$ to the alpha channel only, setting
   the alpha values to $\alpha_T$, the threshold transparency for objects
   in front of target objects.
3  Set buffer $B_1$ as texture.
4  Enable rendering to buffer $B_2$.
5  for each layer {1 ...n} of mask do
6      Render buffer-sized quad with the fragment shader
       specified in Algorithm 3.
7      Set the rendered buffer as texture and enable rendering
       to the other buffer. Each iteration adds one pixel-wide
       layer of the transition.
8      Increase the border alpha value $\alpha_B$ in the shader
       incrementally starting from $\alpha_0$ to 1.0.
9  Disable buffer and activate standard color buffer.
10 Multiplicatively blend the screen-size buffer texture to the
   color buffer (alpha values). Note that resolutions may differ,
   but linear filtering quite efficiently hides zooming artifacts.
11 Render the target region again to avoid jaggedness at the
   border of the target object due to differences in resolution
   between the color and mask buffers. (Line 2).
```

Algorithm 2. Rendering the alpha mask.

```
Input:   border alpha $\alpha_B$, frame buffer $F$, screen position $P$
Output:  alpha value $\alpha_P$ for pixel at position $P$
1  bool IsBorderPixel ← false
2  for each neighbor N of position P do
3      IsBorderPixel ← F(N).Alpha != 1.0 or
       IsBorderPixel
4  IsBorderPixel ← (F(P).Alpha == 1.0) and
   IsBorderPixel
5  output IsBorderPixel ? $\alpha_B$ : 1.0
```

Algorithm 3. Fragment shader.

TABLE 2: PERFORMANCE FOR THREE EXAMPLE APPLICATIONS.

Application	*#polys*	*Screen resolution*	*Standard (FPS)*	*DynTrans (FPS)*
Abstract	13k	800 x 600	87	33
		1280x1024	87	33
Walkthrough	464k	800x600	40	11
		1280x1024	40	11
Jeep	115k	800x600	300	140
		1280x1024	188	90

VI. USER STUDY

From a purely theoretical viewpoint, it seems clear that dynamic transparency will make it significantly easier to discover and access targets in a 3D environment. However, as mentioned in Section IV, the method has two important side-effects: increased visual complexity, and reduced depth perception. While we have designed our image-space implementation to minimize these, we cannot be sure of how well it will perform in practice. Therefore, we conducted a controlled experiment comparing the image-space dynamic transparency technique to unaided 3D navigation.

In other words, our primary motivation for this user study is not to prove the superiority of our technique over other dynamic transparency techniques, but rather to measure the usefulness of dynamic transparency and verify that it has no significant weaknesses. While it might have been interesting to compare our technique to other approaches, the fact is that no existing dynamic transparency technique is designed for visualization use and thus does not support all of our requirements outlined in Section IV.

6.1 Predictions

Despite the possibility of dynamic transparency having a negative impact on user performance, we formulate the following optimistic predictions (in relation to unaided 3D navigation with no access to dynamic transparency):

P1. *Dynamic transparency will allow for faster performance.* Certainly, we believe that our dynamic transparency implementing will help participants to use less time for solving visual perception tasks.

P2. *Dynamic transparency will not cause decreased accuracy.* We claim that the increased visual complexity and loss of depth information introduced by dynamic transparency will not have a significant impact on the accuracy of participants solving visual perception tasks. In fact, for some tasks (such as object discovery), dynamic transparency will allow for better accuracy.

P3. *Dynamic transparency will not cause decreased static depth perception.* The depth cues retained in our dynamic transparency implementation will not result in significantly reduced depth perception for a static 3D scene.

6.2 Participants

We recruited 16 paid subjects for the study (3 female, 13 male). The subjects were drawn primarily from the undergraduate student pool at our university and were screened to have at least basic computer knowledge. Subject ages ranged from 20 to 35 years of age. All subjects had normal or corrected-to-normal vision, and no participants reported color-blindness.

6.3 Apparatus

The experiment was conducted on an Intel Centrino Duo laptop computer equipped with 2048 MB of memory running the Microsoft Windows XP operating system. The display was a 17inch widescreen LCD display running at 1920 × 1200 resolution and powered by an NVidia Geforce 7800 GO graphics card. Input devices were a standard Microsoft mouse and the laptop keyboard.

6.4 Scenarios

We designed the study to include two widely different scenarios: an abstract 3D world and a virtual walkthrough in a 3D building, and four different tasks. In this way, we aimed to be able to measure not only basic target discovery, but also the more complex visual tasks of access and spatial relation.

Abstract 3D World: The first scenario (ABSTRACT) was intended to portray an abstract 3D visualization application and consisted of a cubic 3D volume of size $100 \times 100 \times 100$ filled with n = 200 objects of randomized position and orientation (see Fig. 4 for a screenshot). The objects were simple 3D primitives: spheres, cones, boxes, and torii. Objects were allowed to intersect

=

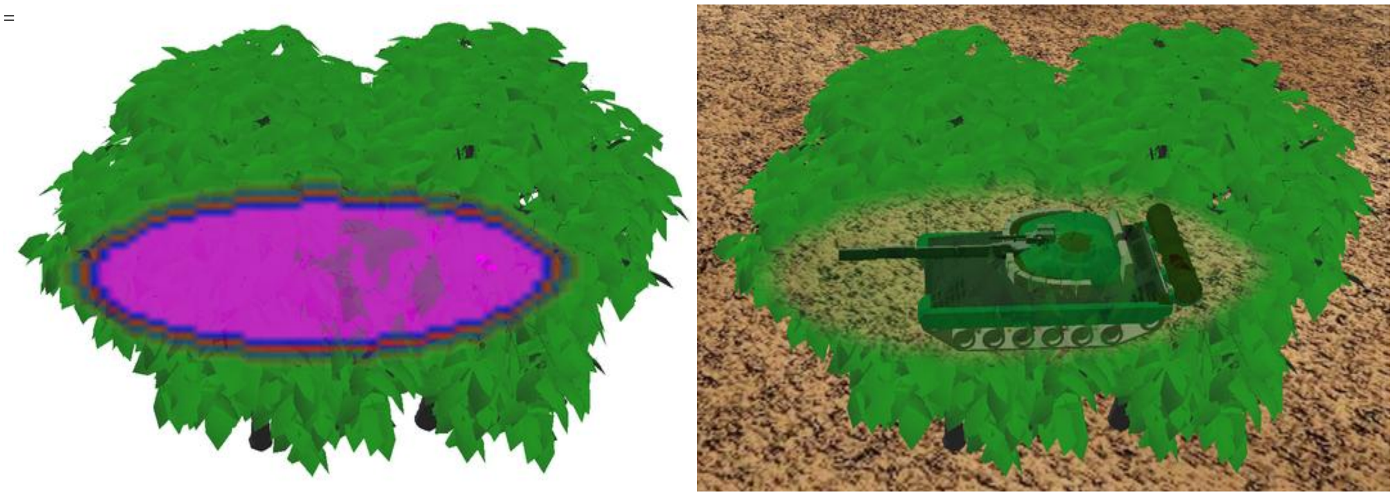

Fig. 3. Ellipse alpha mask for a tank 3D model occluded by a tree. (See Color Plate 13, 14)

but not full enclose each other. A random ratio of 10% to 20% of the objects were flagged as targets and the remainder as distractors. Distractor objects were randomly assigned green and blue color component values, while targets were set to a pure red color and made visible using dynamic transparency.

The user view was fixed at a specific distance from the center of the environment cube so that no object could fall outside of the view frustum, and could be freely orbited around the focus point to afford view from all directions. Orbiting was performed by left-dragging the mouse.

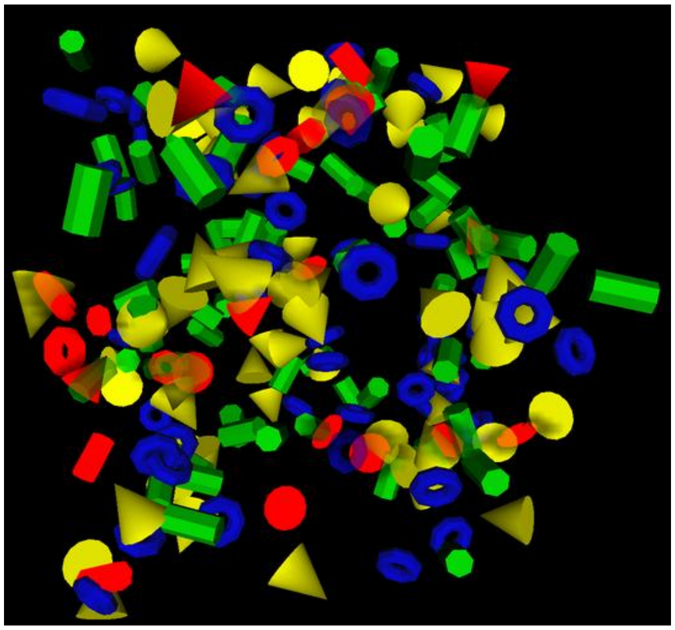

Fig 4. Abstract application with dynamic transparency active.

Virtual Walkthrough: The second scenario (Walkthrough) was a little more complex in nature and designed to mimic a real 3D walkthrough visualization application more closely. Here, a one-level floor plan was randomly generated from a simple 16 × 16 grid, creating walls, floors and ceiling as well as ensuring that all rooms were connected with all of its adjacent neighbors through doorways (see Fig. 6 for an example). A number of n = 50 objects were generated and placed in the environment, and all objects are made visible through the walls using dynamic transparency. The 3D objects chosen for this scenario were more complex 3D models, including pets, vehicles, and furniture, yet were easily distinguishable from each other.

The user started each instance in the center of the environment and navigated using 3D game-like controls involving the mouse and keyboard (mouse to pan the camera around the vertical axis, arrow keys to move, no strafing allowed). The view was constrained to floor level with only yaw (no pitch or roll control) and there was no collision detection with walls or objects.

6.5 Tasks

Tasks were designed to exercise all three visual perception tasks (Section III-B), and differed for the two scenario types. For the abstract 3D world, participants performed the following tasks:

T1 Count the number of targets (red objects) [*discovery*]

T2 Identify pattern formed by the targets (red cones) [*relation*]

For task T1, all red objects were targets and were to be counted. For T2, on the other hand, only red cones were targets. There existed red objects in other shapes (exposed using dynamic transparency), but these were distractors and were not part of the global pattern to be identified.

The pattern was one of the five capital letters C, K, R, X, and Y, rasterized in a 5 × 7 horizontal grid of the same scale as the environment and rotated in an arbitrary fashion around the vertical axis (Fig. 5). The subject was informed of the possible letters prior to performing the task, but not the exact renderings.

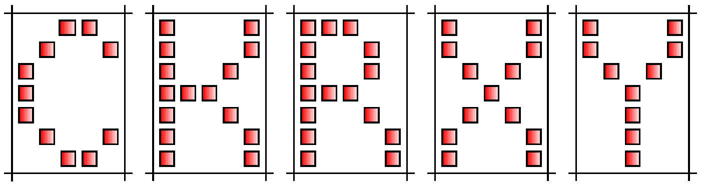

Fig 5. The five patterns used in task T2.

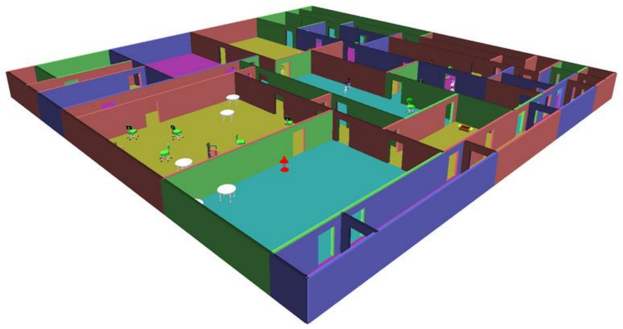

Fig 6. Example floor plan for the WALKTHROUGH application.

For the walkthrough, subjects performed the following tasks:

T3 Find the unique target [*discovery*]
T4 Count the number of targets [*discovery, relation*]

For T3, one of the objects in the environment was unique and the user was asked to find this target. The current target was shown in the upper left corner of the screen. After finding the target, the user proceeded to mark its location on a 2D floor plan of the environment (seen from above). Fig. 7 gives a screenshot.

For the counting task (T4), a random number of the objects in the environment were of the same type and the user was asked to count the occurrences. The current object type was again shown in the upper left corner of the screen. After having estimated that all occurrences were found, the subject entered the amount into the application.

6.6 Experimental Design

The study used a 2 × 2 within-subjects design. The order of the conditions was counterbalanced using a Latin square: each level of each factor ocurred an equal number of times in every position in the sequence. In summary, the factors were the following:

- Dynamic transparency: *active* or *inactive*
- Scenario: *abstract* or *walkthrough*

The tasks for each condition depended on the scenario. With 16 participants and 3 trials per condition, there were 384 tasks recorded in total. The experimental system automatically collected completion time and a correctness measure for each task. This measure depended on the actual task:

- *relative error:* target count error divided by the total number of targets (T1, T4);
- *correctness:* correctness measure (true/false) (T2); and
- *error distance:* distance between participant answer and actual target position (T3).

Abstract scenarios were dynamically generated using a random generator for each trial. Walkthrough scenarios were static, but since all conditions were counterbalanced, any differences in complexity between individual scenarios cancelled each other.

6.7 Procedure

Every task set was preceded by a training session lasting up to five minutes where the subject was instructed in the current task and was allowed to explore the scenario as well as ask questions. Each task set consisted of three trials per condition. During the execution of the actual task set, only general questions were allowed. A session lasted up to 60 minutes.

After finishing the test, participants were asked to fill out a post-test questionnaire about their subjective ratings. As part of this questionnaire, they were also asked to perform a static depth perception test from the WALKTHROUGH scenario (Fig. 8). This test asked subjects to arbitrate between the visible objects to state which object was closest to the current viewpoint.

VII. RESULTS

Analysis of the collected measurements indicates that both our hypotheses are correct: subjects are more efficient (i.e. use less time) and more correct when performing visual search tasks using dynamic transparency than without.

7.1 Completion Time

Overall, the average completion time with inactive dynamic transparency was 65.17 (s.d. 27.75) seconds, compared to 28.69 (s.d. 11.02) with active dynamic transparency. Analysis using a one-way analysis of variance (ANOVA) shows that this was also a significant difference (*F(1,15) = 49.54,p < .001*). Each of the individual tasks also showed significantly shorter average completion times for active dynamic transparency compared to inactive dynamic transparency down to *p < .05*. See Table III and Fig. 9 for a summary.

TABLE 3. AVERAGE COMPLETION TIME FOR ALL FOUR TASKS (STANDARD DEVIATIONS).

Task	*Standard*	*DynTrans*	*F*	*p*
T1	56.26 (38.72)	40.44 (20.99)	7.54	*
T2	22.30 (16.20)	15.80 (10.21)	5.28	*
T3	62.78 (35.63)	23.21 (12.01)	22.98	**
T4	140.0 (61.75)	40.80 (24.16)	48.61	**

** = p < 0.05, ** = p < 0.01*

7.2 Correctness

For the counting tasks (task T1 and T4), we defined correctness in terms of average relative error, i.e. the ratio between the absolute error and the total number of targets for all trials. The absolute error was the absolute difference between the sum of the targets and the sum of the subject answers for the trials. Overall, for task T1 and T4 combined, the average relative error was .100 (s.d. .141) when dynamic transparency was inactive compared to .027 (s.d. .045) when it was active. One-way ANOVA shows that this is also a significant difference (*F(1, 15) = 6.28, p = .024*).

Task 1 in particular showed average relative error of .042 (s.d. .046) for inactive dynamic transparency and .017 (s.d. .018) for active. This too was significant (*F(1,15) = 4.74, p = .046*).

Task 4 showed .123 (s.d. .184) and .034 (s.d. .074) average relative error, respectively, not a significant difference (*F(1,15) = 4.12, p = .061*).

For task 2, we define correctness as whether or not the subject identified the pattern as the correct one. This figure was .963 (s.d. .109) for no dynamic transparency and .963 (s.d. .150) for active. This is obviously not a significant difference (Friedman test, *p =1.0*).

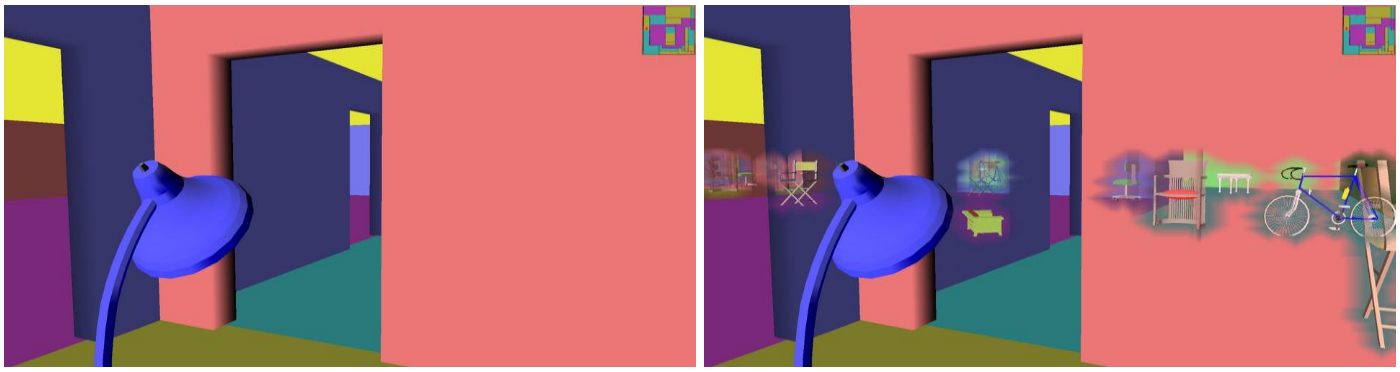

Fig. 7. First-person view of the WALKTHROUGH application with dynamic transparency inactive (left) and active (right).

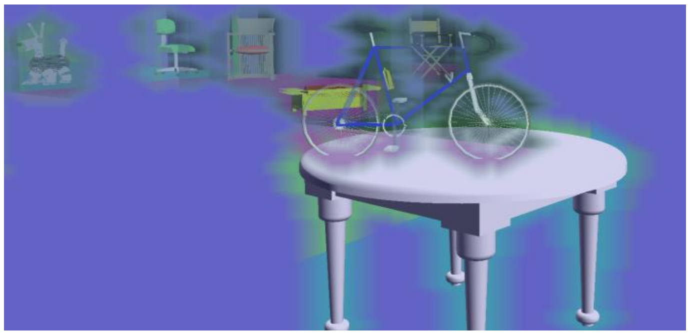

Fig 8. Static depth perception image for dynamic transparency. Participants were asked to arbitrate between the following objects: white table and bike; sofa and director's chair; white table and dog.

Finally, for task 3, we define correctness as the average Euclidean distance (in world units) from the real position of the target and the point marked on the map by the subject for each trial. With dynamic transparency inactive, this average distance was 16.99 (s.d. 14.44), as opposed to 16.21 (s.d. 8.88). This difference is not significant (*F(1,15) = .068, p = .797*), and indicates that the spatial understanding of the subjects was not negatively affected by the use of dynamic transparency.

7.3 Subjective Ratings

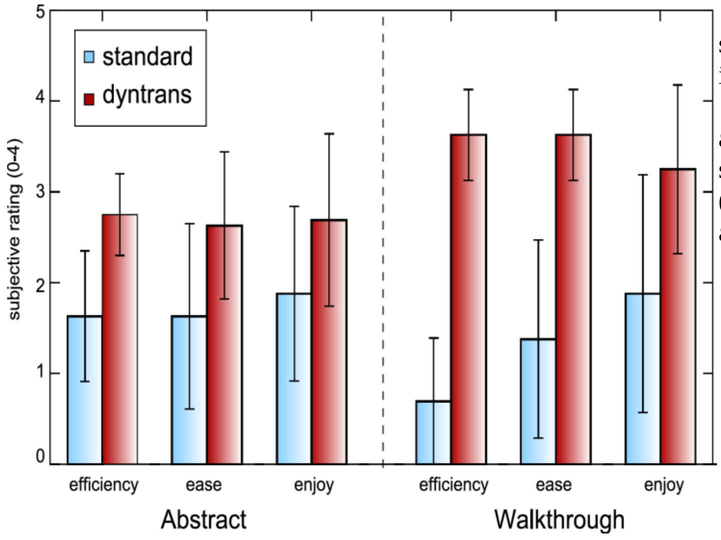

Fig 9. Average completion times for all tasks (error bars show standard deviation).

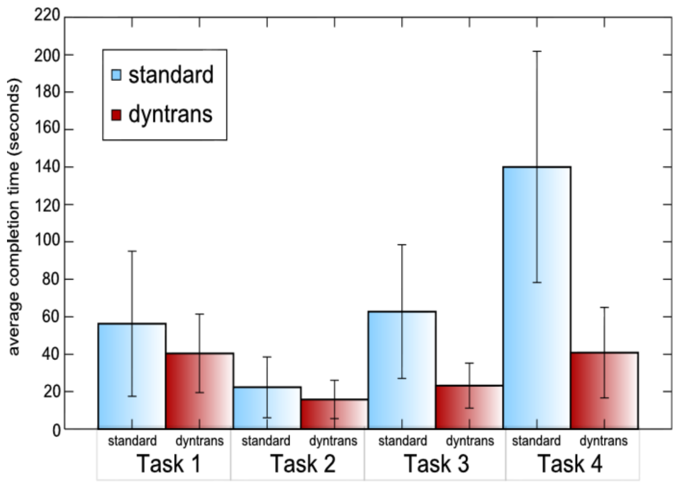

Fig 10. Average subjective rating for both applications (error bars show standard deviation).

Fig. 10 summarizes subjective ratings of dynamic transparency compared to unaided 3D navigation for each scenario. These were all significant differences using Friedman Tests (*p < .05*). Overall, unaided navigation had a mean rating of 1.52 (s.d. .59) as opposed to 3.09 (s.d. .38) for dynamic transparency. This difference is significant (Friedman Test, *p < .01*).

Overall preference for dynamic transparency as opposed to standard vision was .94 (s.d. .25), with one participant indicating a neutral preference.

Static depth perception across all participants gave an average of 2.94 (s.d. .25) out of 3 correct answers. The average self-reported depth perception on a scale from 0 to 4 was 2.75 (s.d. .45). Thus, participants generally felt they still had acceptable depth perception even with transparency active.

VIII. DISCUSSION

The results from our user study can be summarized as follows:

- The completion time for all tasks was significantly shorter for participants using dynamic transparency than unaided 3D navigation.
- Participants were significantly more accurate for some tasks using dynamic transparency. In no task was unaided 3D navigation more accurate.
- Static depth perception was high for dynamic transparency, and self-reported depth perception was acceptable.

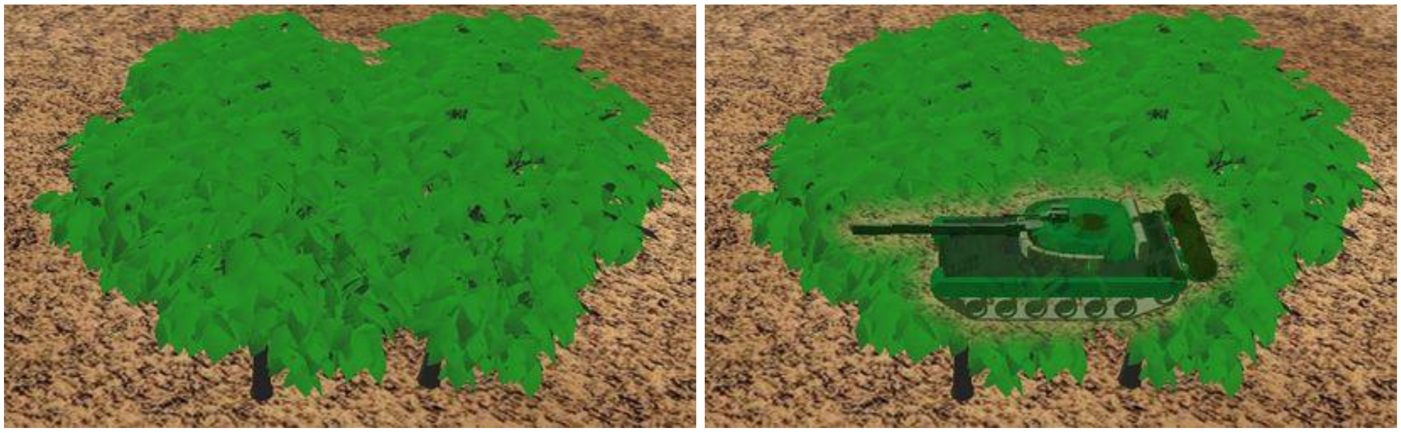
Fig. 11. Applying the image-based dynamic transparency algorithm to units in a 3D real-time strategy game.

These findings in turn all confirm our predictions P1, P2, and P3. In the following sections, we will try to explain and generalize these results. We will also briefly discuss limitations and uses for dynamic transparency in practice as well as our future work.

8.1 Explaining the Results

The results from the user study confirm all of our predictions. In other words, our image-space dynamic transparency implementation fulfilled our goals for the technique without falling prey to the potential weaknesses of decreased depth perception and increased visual complexity. Furthermore, the subjective results indicate a strong preference for dynamic transparency.

As has been emphasized several times in this paper, occlusion is an important depth cue that humans use to determine the spatial relation of objects in our environment. The introduction of dynamic transparency can then cause "reverse occlusion", i.e. the phenomenon that distant objects all of a sudden occlude nearby objects instead. In our implementation, we took care to avoid this extreme, but it is clear that depth perception is still weakened. How did participants manage to achieve such good accuracy results, on a par with unaided 3D navigation?

The explanation for this lies in the redundancy in depth information. As discussed earlier, human perception relies on many more factors besides occlusion to disambiguate depth; examples include stereopsis, motion parallax, atmospheric perspective, texture gradient, etc. Even if we weaken the occlusion cue, other depth cues will help the viewer to perceive the 3D scene correctly (for instance, in Fig. 8, relative positioning and size plays a large role in aiding depth perception).

8.2 Generalizing the Results

In light of the general model for dynamic transparency presented in this paper, it is interesting to investigate whether the results we collected for our implementation also generalize to the whole class of dynamic transparency techniques. The image-space algorithm described in this paper does not make use of any special functionality to achieve these results, so any other dynamic transparency implementation that conforms to the requirements in Section IV should see the same results.

Our choice of scenarios for the user study may also affect how we can apply these results to other situations. The intention behind the user study design was to capture an ecologically valid selection of application domains. However, it can be argued that a wider selection of perhaps three or four scenarios taken from real visualization applications would have constituted a better design. We leave this extended analysis for future work.

Furthermore, there is a limit where the findings from the study do not hold. Add enough objects, or make the scene complex enough, and the visual complexity and lack of depth information will make dynamic transparency impractical. We can only claim generalizability for the same order of object complexity and quantity that we involved in the user study, but—limitations of the scenario 3D environments non-with-standing—we believe these to be good approximates of real visualization applications.

The shape of the alpha mask may also have an impact on the results. In our study, we only used the outline alpha mask, but it is conceivable that different shapes may yield different results. Given that other shapes are useful for particular applications, it would be interesting for future studies to investigate this effect.

8.3 Limitations to Dynamic Transparency

One fundamental limitation with dynamic transparency lies in the selection of targets and distractors for the passive mode of operation. The model requires semantic knowledge of which objects are targets and which are distractors. In some applications, this dichotomy may be known in advance, but other more general applications may want to let the user make this selection. It is not clear exactly how to allow for this interaction. In this regard, the active mode of operation is more powerful because it turns this decision into a direct manipulation task. On the other hand, active dynamic transparency gives rise to a new range of issues, such as choosing the depth or the number of layers uncovered by the dynamic transparency magic lens. The relative efficiency of these two modes should be evaluated in future studies.

The model for dynamic transparency described in this paper is clearly very useful for many visualization tasks. However, it is important to remember that dynamic transparency, for all of its virtues, has a direct impact on the visual realism of a 3D scene.

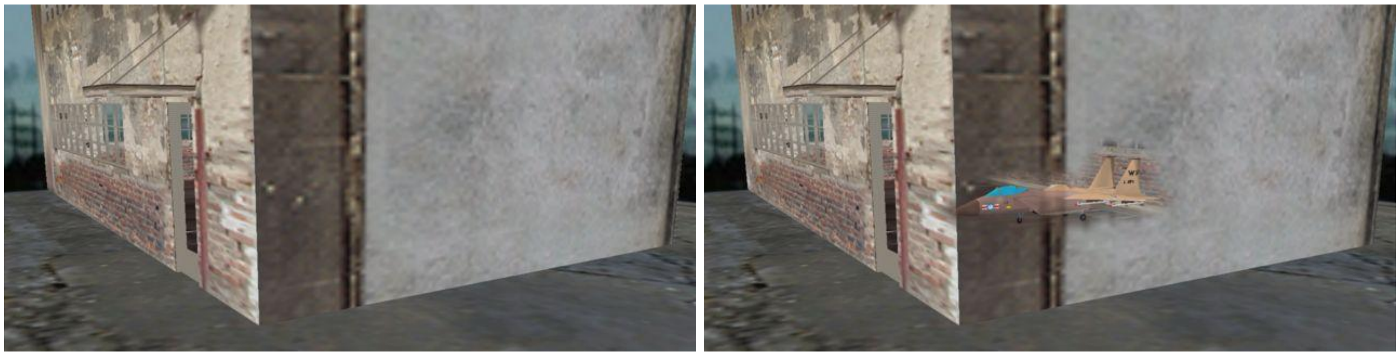

Fig. 12. Simple 3D scene showing dynamic transparency alpha maps uncovering an F-15 fighter hidden inside a building.

Walls, vehicles, and other objects with semi-transparent holes in them simply do not look realistic, so in a sense we sacrifice some realism to achieve these benefits. As discussed in this paper, this sacrifice may be even more tangible: scenes become more "visually busy" and understanding the structure of the scene may become exceedingly difficult.

As an example of this, some subjects in our study had the interesting behavior of "respecting" the world less with dynamic transparency active. When it was inactive, they would use the doors in the virtual walkthrough rather than going through walls. However, with dynamic transparency active, they would not hesitate to pass through walls. While this is an informal observation, this behavior might indicate that the impact that dynamic transparency has on visual realism causes the world to become less believable to the users, thus making them ignore the implicit rules of the environment.

8.4 Applications for Dynamic Transparency

Beyond the visualization aspects that this paper focuses on, the concept of dynamic transparency can be applied to a broad range of other domains. Examples include simulation, training, and command and control applications, among others.

In particular, the method could be useful in computer games, where we may want to temporarily suspend graphical realism by removing intervening objects in order to improve game play. Fig. 11 shows an example from a mockup 3D strategy game, rendered in real-time using our algorithm (Fig. 3 shows the same scene with an ellipse-shaped alpha mask). The player-controlled tank hiding under the cover of the forest is made visible through the foliage of the trees in order to help the user see the friendly units. Also see Fig. 12 for another example of a game-like scenario with an F-15 fighter aircraft hidden inside a hangar being exposed to the player using dynamic transparency.

8.5 Future Work

We envision improving our model for dynamic transparency with a more general interest-based importance scale, allowing users and applications to dynamically specify the relative importance of individual parts of 3D objects to a very high degree (possibly along the lines of the IDVR [26] importance model). We will continue working on techniques for reducing the impact of occlusion in 3D environments, including the automatic generation of view-dependent animated exploding diagrams as well as the generation of occlusion-free grand tours of a 3D environment. We are also interested in pursuing similar avenues for providing superhuman vision capabilities in visualization applications. These occlusion management techniques could be useful for 3D user interface development [37].

IX. CONCLUSIONS

We have presented an evaluation of the use of dynamic transparency for managing occlusion of important target objects in 3D visualization applications. In the absence of existing real-time algorithms for dynamic transparency that are suitable for interactive visualization, we have further devised an image-space algorithm and implementation realizing the model. The algorithm uses the standard framebuffer as a cumulative alpha buffer, rendering the scene back-to-front and blending in alpha masks of target objects to allow for see-through surfaces. Our evaluation consisted of a comparative user study measuring efficiency and correctness gains from using the technique as opposed to standard 3D navigation controls. Our results clearly show that having access to dynamic transparency yields significantly more efficient (faster) performance. Users are typically also more correct with the technique than without.

ACKNOWLEDGMENT

The authors would like to thank Per A. Jonasson for valuable insights into superhero X-ray vision, as well as the members of the IIR group at Georgia Tech for their comments on the intermediate stages of this work. We also thank the people who have offered feedback on this work since its initial presentation at the INTERACT conference in Rio de Janeiro, Brazil.

REFERENCES

[1] N. Elmqvist, U. Assarsson and P. Tsigas. Employing dynamic transparency for 3D occlusion management: Design issues and evaluation, in *Proceedings of INTERACT*, ser. LNCS, C. Baranauskas, P. Palanque, J. Abascal, and S. D. J. Barbosa, Eds., Springer, vol. 4662, pp. 532–545, 2007.

[2] D. A. Bowman, C. North, J. Chen, N. F. Polys, P. S. Pyla and U. Yilmaz. Information-rich virtual environments: theory, tools, and research agenda, in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pp. 81–90, 2003.

[3] N. F. Polys and D. A. Bowman. Design and display of enhancing information in desktop information-rich virtual environments: challenges and techniques, *Virtual Reality*, vol. 8, no. 1, pp. 41–54, 2004.

[4] N. Elmqvist and P. Tsigas. A taxonomy of 3D occlusion management techniques, in *Proceedings of the IEEE Conference on Virtual Reality*, pp. 51–58,2007.

[5] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes, in *Proceedings of the ACM Conference on Computer Graphics (SIGGRAPH '98)*, pp. 453–460, 1998.
[6] E. Praun, H. Hoppe, M. Webb and A. Finkelstein. Real-time hatching, in *Proceedings of the ACM Conference on Computer Graphics (SIGGRAPH 2001)*, pp. 581–581, 2001.
[7] T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte. A developer's guide to silhouette algorithms for polygonal models, *IEEE Computer Graphics and Applications*, vol. 23, no. 4, pp. 28–37, 2003.
[8] A. Gooch, B. Gooch, P. Shirley and E. Cohen. A non-photorealistic lighting model for automatic technical illustration, in *Proceedings of the ACM Conference on Computer Graphics (SIGGRAPH '98)*, pp. 447–452, 1998.
[9] B. Gooch, P.-P. J. Sloan, A. Gooch, P. Shirley and R. F. Riesenfeld. Interactive technical illustration, in *Proceedings of the ACM Symposium on Interactive 3D*, pp. 31–38, 1999.
[10] M. Nienhaus and J. Döllner. Blueprints: Illustrating architecture and technical parts using hardware-accelerated non-photorealistic rendering, in *Proceedings of Graphics Interface*, pp. 49–56, 2004.
[11] B. Freudenberg, M. Masuch and T. Strothotte.Real-time halftoning: A primitive for non-photorealistic shading, in *Proceedings of the 13th Eurographics Workshop on Rendering*, Eurographics Association, pp. 227–232.
[12] D. S. Kay and D. P. Greenberg. Transparency for computer synthesized images, in *Computer Graphics (SIGGRAPH '79 Proceedings)*, pp. 158–164, 1979.
[13] C. Everitt. Interactive order-independent transparency, NVIDIA Corporation, 2001, see *http://developer.nvidia.com*.
[14] A. Mammen. Transparency and antialiasing algorithms implemented with the virtual pixel maps technique, *IEEE Computer Graphics and Applications*, vol. 9, no. 4, pp. 43–55, July 1989.
[15] P. J. Diefenbach. Pipeline rendering: Interaction and realism through hardware-based multi-pass rendering, Ph.D. thesis, Computer Graphics, University of Pennsylvania, 1996.
[16] J. Diepstraten, D. Weiskopf and T. Ertl. Transparency in interactive technical illustrations," *Computer Graphics Forum*, vol. 21, no. 3, pp. 317–325, 2002.
[17] B. L. Harrison, G. Kurtenbach and K. J. Vicente. An experimental evaluation of transparent user interface tools and information content, in *Proceedings of the ACM Symposium on User Interface Software and Technology 1995*, pp. 81–90, 1995.
[18] C. Gutwin, J. Dyck, and C. Fedak . The effects of dynamic transparency on targeting performance," in *Proceedings of Graphics Interface*, pp. 105–112, 2003.
[19] P. Baudisch and C. Gutwin. Multiblending: displaying overlapping windows simultaneously without the drawbacks of alpha blending, in *Proceedings of the ACM CHI 2004 Conference on Human Factors in Computing Systems*, pp. 367–374, 2004.
[20] E. W. Ishak and S. K. Feiner. Interacting with hidden content using content-aware free-space transparency," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp. 189– 192, 2004.
[21] E. A. Bier, M. C. Stone, K. Pier, W. Buxton and T. DeRose. Toolglass and Magic Lenses: The see-through interface, in *Computer Graphics (SIGGRAPH '93 Proceedings)*, pp. 73–80, 1993.
[22] L. Chittaro and I. Scagnetto. Is semitransparency useful for navigating virtual environments?, in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pp. 159–166, 2001.
[23] J. Diepstraten, D. Weiskopf, and T. Ertl. Interactive cutaway rendering, in *Proceedings of Eurographics*, pp. 523–532, 2003.
[24] J. Looser, M. Billinghurst and A. Cockburn. Through the looking glass: the use of lenses as an interface tool for augmented reality interfaces, in *Proceedings of GRAPHITE*, pp. 204–211, 2004.
[25] C. Coffin and T. Höllerer. Interactive perspective cut-away views for general 3D scenes, in *Proceedings of the IEEE Symposium on 3D User Interfaces*, pp. 25–28, 2006.
[26] I. Viola, A. Kanitsar and E. Gröller. Importance-driven volume rendering, in *Proceedings of the IEEE Conference on Visualization*, pp. 139–145, 2004.
[27] I. Viola, M. Feixas, M. Sbert and E. Gröller. Importance-driven focus of attention, *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 933–940, Sept./Oct. 2006.
[28] N. Elmqvist and P. Tsigas. View-projection animation for 3D occlusion management," *Computers and Graphics*, vol. 31, no. 6, pp. 864–876, 2007.
[29] R. Stoakley, M. J. Conway and R. Pausch. Virtual Reality on a WIM: Interactive worlds in miniature, in *Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems*, pp. 265– 272, 1995.
[30] N. Elmqvist and M. E. Tudoreanu. Occlusion Management in Immersive and Desktop 3D Virtual Environments, *International Journal of Virtual Reality*, vol. 6, no. 2, pp. 21–32, 2007.
[31] H. Sonnet, M. S. T. Carpendale and T. Strothotte. Integrating expanding annotations with a 3D explosion probe, in *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pp. 63–70, 2004.
[32] C. Andújar, P.-P. Vázquez and M. Fairén. Way-finder: guided tours through complex walkthrough models, in *Proceedings of Eurographics*, pp. 499–508, 2004.
[33] M. S. T. Carpendale, D. J. Cowperthwaite and F. D. Fracchia. Distortion viewing techniques for 3D data, in *Proceedings of the IEEE Symposium on Information Visualization*, pp. 46–53, 1996.
[34] K. Perlin and D. Fox. Pad: An alternative approach to the computer interface, in *Proceedings of the ACM Conference on Computer Graphics (SIGGRAPH '93)*, pp. 57–64, 1993.
[35] E. B. Goldstein, *Sensation and Perception*, 6th ed. Wadsworth-Thomson, 2002.
[36] L. Carpenter. The A-buffer, an antialiased hidden surface method," *Computer Graphics*, vol. 18, no. 3, pp. 103–108, July 1984.
[37] D. A. Bowman, J. Chen, C. A. Wingrave, J. Lucas, A. Ray, N. F. Polys, Q. Li, Y. Haciahmetoglu, J.-S. Kim, S. Kim, R. Boehringer and T. Ni. New Directions in 3D User Interfaces," *International Journal of Virtual Reality*, vol. 5, no. 2, pp. 3–14, 2006.

Niklas Elmqvist is an Assistant Professor in the School of Electrical and Computer Engineering at Purdue University in West Lafayette, IN, USA. Having joined Purdue in fall 2008, he was previously a Microsoft Research postdoctoral researcher in the AVIZ team of INRIA Saclay - Île-de-France located at Université Paris-Sud in Paris, France. He received his Ph.D. in December 2006 from the Department of Computer Science and Engineering at Chalmers University of Technology in Gothenburg, Sweden. His research specialization is information visualization, human-computer interaction, and visual analytics. He is a member of the IEEE and the IEEE Computer Society

Ulf Assarsson is an Assistant Professor in the Department of Computer Science and Engineering at Chalmers University of Technology in Göteborg, Sweden. His area of research is computer graphics, focusing primarily on real-time and non-real-time soft shadows as well as raytracing and global illumination. He is a member of the IEEE and the IEEE Computer Society.

Philippas Tsigas is an Associate Professor in the Department of Computer Science and Engineering at Chalmers University of Technology in Göteborg, Sweden. He is the scientific leader of the Distributed Computing and System research group and his research interests include distributed and parallel computing and systems, as well as general information visualization. He is a member of the IEEE and the IEEE Computer Society.