Interactive Context-Driven Visualization Tools for Augmented Reality

Erick Mendez*
Graz University of Technology
Institute for Computer Graphics and Vision, Inffeldgasse 16a
Graz, 8010, Austria

Denis Kalkofen†
Graz University of Technology
Institute for Computer Graphics and Vision, Inffeldgasse 16a
Graz, 8010, Austria

Dieter Schmalstieg‡
Graz University of Technology
Institute for Computer Graphics and Vision, Inffeldgasse 16a
Graz, 8010, Austria

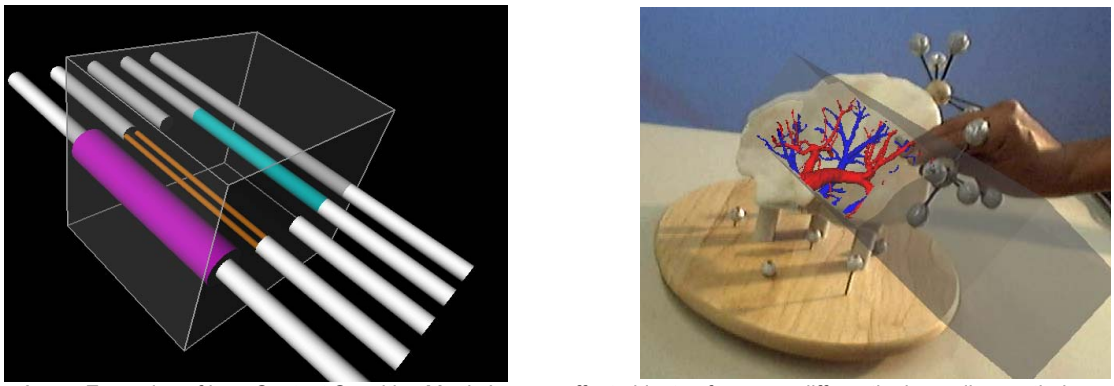

Figure 1 Examples of how Context Sensitive Magic Lenses affect objects of a scene differently depending on their context. On these two images the lenses are shown as semi transparent. On the left image rendering styles are changed or transformations are added to the objects depending on their contextual information and the intersection with the lens. The right image shows a lens that renders differently the vessel trees of a liver depending on their type and the lens intersection.

ABSTRACT

In this article we present an interaction tool, based on the Magic Lenses technique, that allows a 3D scene to be affected dynamically given contextual information, for example, to support information filtering. We show how elements of a scene graph are grouped by context in addition to hierarchically, and, how this enables us to locally modify their rendering styles. This research has two major contributions, the use of context sensitivity with 3D Magic Lenses in a scene graph and the implementation of multiple volumetric 3D Magic Lenses for Augmented Reality setups. We have developed our tool for the Studierstube framework which allows us doing rapid prototyping of Virtual and Augmented Reality applications. Some application directions are shown throughout the paper. We compare our work with other methods, highlight strengths and weaknesses and finally discuss research directions for our work.

CR Categories

H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities; H.5.2 [User Interfaces]: Interaction styles; I.3.6 [Methodology and Techniques]: Interaction techniques; I.3.7 [Three-Dimensional Graphics and Realism]: Virtual reality.

Keywords

Interaction techniques for MR/AR; Real-time rendering

1 INTRODUCTION

Visualization and interaction tools are important topics in Augmented Reality (AR) research. The goal of AR displays is usually to present abstract or otherwise hidden information linked to the real world. Frequently, this additional information may cause a visual overload in the user's field of view. As a relief, information filtering techniques have been investigated [10]. The general idea of information filtering is to modify the appearance of virtual scene objects based on user-defined context parameters, typically fading out uninteresting scene objects to reduce display clutter.

While [10] applies the filtering globally to all scene objects, Magic Lenses are filters that modify the presentation of scene objects in a locally bounded area. Magic Lenses can be used to reveal hidden information, to enhance data of interest, or to suppress distracting information. They were first introduced as a user interface tool in 2D [3], and later extended to 3D [17]. While the effect of a lens is locally bounded, it is applied globally to all scene objects.

The effect of multiple lenses can be aggregated by overlapping multiple lenses, but this does not allow applying the effects only to certain individual scene objects or groups of scene objects. For example, a composite effect of enlarging all scene objects of type A by 10% while rendering all scene objects of type B semi-transparently cannot be composed from a lens that enlarges everything by 10% and a lens that renders everything semi-transparently. Moreover, overlapping 3D lenses have, up to now,

* mendez@icg.tu-graz.ac.at
† kalkofen@icg.tu-graz.ac.at
‡ schmalstieg@icg.tu-graz.ac.at

only been shown in screen space, not in object space, which further limits the types of effects that can be produced.

In this paper, we introduce the notion of 3D context sensitive Magic Lenses (CSML), which overcome these deficiencies. Every scene object's rendering style is defined as a function of an arbitrary set of user-defined context parameters. These parameters are contained in a group with other arbitrary information, such as object type or membership in a certain branch of the scene graph. We permit multiple overlapping Magic Lenses in both screen and object space, in both convex and concave arrangements. Position and extent of the Magic Lenses, the type and arrangement of objects in the scene graph, and the mapping from context parameters to rendering style are completely independent, allowing to mix and match tools from this toolset even at runtime – the only requirement is that all used components agree on a common set of context parameter descriptors.

For example, as shown in Figure 1 (left), objects have been assigned to different context families and their rendering styles change depending to which family they belong and the intersection with the lens. Another example is in Figure 1 (right) where a liver model is presented. Notice that the portions of the vessel trees that intersect the lens are rendered in either blue or red depending on whether it is an artery or a vein.

In general terms, the main contribution of context sensitive Magic Lenses is that they provide powerful information filtering for arbitrary complex scenes, without most of the limiting assumptions of previous work in that area. Specifically, these lenses are completely general in terms of their shape, number, and effect on the visualization. Our main interest lies on outdoor AR applications, which draw from a rich database of virtual objects associated with real world coordinates or entities, and can change dynamically. Therefore, hard-coded visualization behaviors are not a satisfactory option. Although designed for AR setups, the same Magic Lens techniques can be applied to purely virtual scenes.

2 Related Work

2.1. Information Filtering and Context

Researchers from Columbia University and NRL [10] studied techniques for information filtering in AR environments. They based their work on the focus-nimbus technique described by Benford and Fahlen [2]. One of the motivations for this work is the overload of information common to AR setups. An example depicting the possible position of snipers in an urban environment was used to demonstrate the advantages of this technique. Their filtering technique is applied on a global scale to the entirety of the scene and, as far as it can be seen from the article, it cannot be localized.

2.2. 3D Magic Lenses Rendering

Viega et al. developed flat and volumetric Magic Lenses for 3D environments [17]. An interesting discussion takes place in the rendering technique of flat lenses, and how multiple lenses can be combined. However, overlapping volumetric lenses are not discussed.

An elegant algorithm for the rendering of 3D Magic Lenses was presented by Ropinski and Hinrichs [15]. This technique uses multi-pass rendering to achieve the effect of a Magic Lens. This algorithm requires a dual depth buffer, which is not available in today's hardware. Everitt [6] proposed using the shadow buffer as a solution. In contrast, we use a variant of the algorithm in [15] implemented using Cg fragment shaders.

2.3. Magic Lenses as Interaction Tools

Looser et al. [12] presented an interesting work that mixed the use of lenses and semantic information. The interaction techniques discussed in their paper are magnification, object selection and information filtering. The notion of "semantic zooming" is interesting, but very little information is provided on how the objects are enriched with this semantic information. It seems that the used lens will only determine the area where the object can be rendered, but not its rendering style.

Methods for decoupling rendering styles from cutaway objects were researched by Diepstraten et al. [5]. Particularly interesting in this work are the drawing styles, such as sawtooth-shaped cutaway geometries, and the fact that basic semantic information is given to the objects (classified as inside and outside). Cutaway geometries are coupled with the objects to be cut, which limits the objects to be convex.

Wang et al. [18] from Stony Brook University described a framework for volumetric lenses, mostly focusing in volumetric data. This tool allowed them to apply a number of free-style lenses, which, conveniently emphasized specifics parts of the visualized data without loosing the overall context. It must be noted that this work uses context as the overall visual relationship of the displayed data with its surroundings. Arbitrary subsets of data in the same set (such as bones or tissue in an MRI scan) cannot be treated differently.

2.4. X-Ray Vision

A number of articles have studied the rules of correctly displaying X-Ray vision, many of them dealing with techniques to properly provide depth information [9], [10]. Some developments, like the present, have focused on interactive tools for displaying X-Ray vision. In 2004 work was carried out at UCSB for the interactive display of X-Ray vision [1]. A number of interesting techniques were presented, some of which we considered during the development of our current work.

Later, the same group at UCSB presented some encouraging research for cutaway views, and how to give better depth information to the user [4]. These cutaway views have been used in 3D environments before by [5], and [7], this last research also studied other kinds of techniques for presenting occluded information such as ghosting or object removal. Many of these techniques can also be achieved by the work presented in this paper, and in Section 5 we will show some application examples.

3 Background: Context Sensitive Scene Graph

Context-sensitive traversal of scene graphs was introduced in a previous publication by our group [14]. The Magic Lens techniques presented in this paper builds on this work, so its important points are summarized here for convenience.

A set of context parameters is maintained throughout the traversal of a scene graph. This allows parameterizing and repurposing sub graphs in various ways. The context parameters are maintained as part of the state of the scene graph traversal, which makes them independent of the scene graph structure. The context parameters are modeled as an associative array of key-value pairs, where the values are either strings or pointers to sub graphs.

By using pointers as parameters, such template sub graphs can be inserted multiple times during the traversal. In contrast to a conventional directed acyclic graph structure, the binding of child nodes to their parents happens very late, during the traversal itself, so the nodes can be changed for each traversal and provide a very flexible way of assembling complex scene graphs. Transparent

caching of the traversal outcome in display lists ensures that rendering performance is not adversely affected.

The context sensitive scene graph traversal shifts the complexity of managing multiple representations from the application code to the scene graph itself. Rather than writing application code to modify the scene graph or change rendering parameters based on user interaction, the application only needs to change the context parameters for high level control of the visual effects.

The scene graph will adapt the visual appearance of its contained objects to dynamically changing requirements, and, even compose sub graphs on the fly. A particular visual representation is simply an instance of a template sub graph combined with a specific choice of context parameters. Combinations of content and visual interpretation are created during traversal only, which is the actual moment in time they are required.

By combining the context sensitive traversal with the multi-pass rendering technique described in [15], Magic Lenses can be used to define context parameters which are used to influence the appearance of objects on a pixel by pixel basis.

4 Algorithm and Implementation

The CSML tools described in this paper were implemented as an extension to the *Studierstube* [16] framework, which uses Coin[§], a re-implementation of Open Inventor [19].

As was mentioned in Section 2.2 we use a modified version of the algorithm by Ropinski. This algorithm, however, was targeted to only one Magic Lens. Therefore, we had to further extend it to allow multiple intersecting lenses. Although we have designed an algorithm for multiple Magic lenses, given the power that a context sensitive scene-graph brings, many of the applications can be solved by the use of only one Magic Lens. Therefore, we have split our algorithm into two forms, one that deals with a single Magic Lens and one that accepts multiple lenses. Our current implementation adapts automatically between the two algorithms depending on the number of lenses encountered in the graph.

4.1. Background: Ropinski's Algorithm

Ropinski's algorithm [15] consists of three rendering passes:

- First, it renders only fragments behind and next to the lens
- Second, it renders only fragments inside the lens
- Third, it renders only fragments in front and next to the lens

These steps mean that every object that wants to be affected by the lens needs to be rendered three times. Passes 1 and 2 require two depth tests for distinguishing the fragments falling behind and inside the lens, respectively. In the second pass the style changes are applied.

4.2. Algorithm Single Magic Lens

Instead of using a shadow or stencil buffer, we rely on Cg fragment programs [13] combined with floating point textures to overcome the lack of two depth tests. This means that the Magic Lens will have one texture associated that stores its depth information. In subsequent rendering passes this texture will be used by all other objects to determine if they fall inside the lens or not. A more formal description of the algorithm follows.

Let T be the texture that will hold the depth information of the Magic Lens L. We denote C as a string representing our context information and O as a group of 3D objects. Then let $S_j=\{O,C\}$ the j^{th} group composed of a set of objects and their context information, where $j=\{0..m\}$.

When L is encountered in the graph:

- First, we render it such that the depth values of its back faces are stored in the red channel of T
- Next, we render it such that the depth values of its front faces are stored in the green channel of T

Then, for every S_j encountered in the graph:

- The first pass uses the depth information stored in T to render only those fragments lying behind and next to L
- The second pass uses the depth information stored in T to render only those fragments lying inside L
- The third pass uses the depth information stored in T to render only those fragments lying in front and next to L

4.3. Algorithm Multiple Magic Lenses

In this extended algorithm, one texture is used per lens to store its depth information. In subsequent rendering passes, these textures are used by all other objects to determine if they fall inside a lens or not. A more formal description of the algorithm follows.

Let T_i be the texture that will hold the depth information of the Magic Lens L_i, and let D be a common texture that will hold the back depth values of all L_i, where $i=\{1..n\}$. We denote C as a string representing our context information and O as a group of 3D objects. Then let $S_j=\{O,C\}$ the j^{th} group composed of a set of objects and their context information, where $j=\{0..m\}$.

For every L_i encountered in the graph:

- First, we render it such that the depth values of its back faces are stored in the red channel of T_i
- Next, we render it such that the depth values of its front faces are stored in the green channel of T_i
- Then we render it such that the depth values of its back faces are stored in the common texture D such that farthest back faces are kept and all others are ignored

Then, for every S_j encountered in the graph:

- The first pass uses the depth information stored in the common texture D to render only those fragments lying behind and next to all L_i
- The following n passes use the depth information stored in every T_i to render only those fragments lying inside L_i
- The last pass renders the entire object relying only in the traditional depth test

It must be noted that these algorithms do not consider any contextual information attached to the objects. This is because contextual information is added as a part of a sub graph of the scene during traversal, as will be explained in the following section. A special situation occurs when we define a behavior for intersecting lenses, for this we need to check all the possible intersections of the lenses, and add those as extra rendering passes. Additionally, we must also check that every object renders inside a lens only, without rendering on intersecting regions. For this mechanism the last part of our algorithm for object rendering is modified as follows.

[§] http://www.coin3D.org

For every *Sj* encountered in the graph:

- The first pass uses the depth information stored in the common texture *D* to render only those fragments lying behind and next to all *Li*
- The following *n* passes use the depth information stored in every *Ti* to render only those fragments lying inside *Li* ignoring lenses intersections
- The following $\left(\sum_{k=2}^{n-1}\binom{n}{k}\right)+1$ passes use the depth information in every *Ti* to render only those fragments lying on every possible lens intersection. Here, *k* is the number of possible lenses involved in an intersection, and *n* is the number of lenses in the graph
- The last pass renders the entire object relying only on the traditional depth test.

This solution creates a high number of rendering passes, in particular if no information about which lenses should be combined is available. However, in most practical cases, the expected combination of lenses will be known. Our current implementation handles multiple lenses, but the default case is that the effect of intersections depends on the order of the lenses in the scene graph (the first lens is dominant, see Figure 5). The implementation allows handling specific intersection cases correctly if the user desires it (see Figure 6), but the increased number of rendering passes slows down the rendering. We plan an optimized fully generation intersection method as future work.

4.4. Implementation

We provide the details of the implementation considering multiple Magic Lenses, however, as mentioned before, if only one Magic Lens is encountered in the graph, the algorithm falls back to the design that considers only one lens. Our implementation builds on Coin, an object-oriented scene-graph, which can be easily extended by adding new node classes. Magic Lens and objects enriched with context information are thus an extension to Coin which naturally blends in to Coin's runtime object management system. In particular, Coin's built-in scripting language (the precursor to VRML) automatically supports user-defined nodes and allows convenient prototyping.

A *context family* of objects is defined by all those objects that have the same context information. However, these objects do not have to belong to the same part of the scene graph hierarchy, i.e., they can be scattered throughout the scene-graph and still be jointly affected by a CSML according to the *context family* they belong to. Every *context family* has a unique name and defines a specific rendering style that will be used when fragments of its objects are inside or outside a CSML.

We created three new Coin nodes:

- SoCSMLFamilies, which sets the appropriate rendering style for every *context family*
- SoCSMLLens, which holds the convex shape that defines the region of our Magic Lens
- SoCSMLScene, which holds a group of objects and the *context family* they belong to

In the following we give a more detailed description on these nodes based on their scripting interface.

4.4.1. SoCSMLFamilies

This node allows the definition of the rendering styles for the context families of objects. For example:

```
SoCSMLFamilies {
        Family "alpha"
        lensName [ "RedPainter", "BluePainter" ]
        style [ USE Red, USE Blue ]
}
```

Here, all the pixels of the objects of the family "alpha" will be rendered as Red when they are inside the Magic Lens called "RedPainter", whereas all those pixels of the same family that fall inside "BluePainter" will be rendered as blue. The style definitions referred to by the "USE" keyword are actually references to sub graphs of arbitrary complexity, containing any form of rendering effect possible in Coin.

4.4.2. SoCSMLLens

This node accepts an arbitrary sub graph as a geometric description of the Magic Lens. For example:

```
SoCSMLLens {
        Name "RedPainter"
        Content Sphere { radius 2 }
}
```

Here, a sphere of radius 2 will act as a Magic Lens, the name assigned to this lens is "RedPainter". Notice that the content must be convex, but can be of an arbitrary complexity.

4.4.3. SoCSMLScene

This node accepts an arbitrary sub graph as content and a string that acts as our context information. When scene graph traversal happens, this node builds a scene graph on the fly with the rendering styles defined for this *context family* for the pixels that lie inside and outside the lens. For example:

```
SoCSMLScene {
        Family "alpha"
        Content Cube {}
}
```

Here, a single cube is the only part of our CSML Scene and it has been assigned as part of the "alpha" family. Notice also, that, although here a single shape was assigned as the content, this can actually be a more complex sub graph.

Conceptually, the Magic Lenses affect the rendering style of the objects, but in practice, is the objects, inside a CSML Scene, that determine which family they belong to, which enables them to build the sub graph according to their contextual information. The Magic Lenses in the end only define the regions where the rendering styles are changed.

For efficiency issues, we use frame buffer objects for the sharing of the texture between the Magic Lens and the group of objects with contextual information. To reduce the aliasing artifacts we use floating point textures to store the regions of the lenses. Other techniques such as super sampling may be employed to further reduce the artifacts. Similarly to [15] and [17], our Magic Lens must be convex in image space. However, like in [15], the lenses can be arbitrarily complex and can be combined to form concave assemblies from individual convex shapes.

The multiple passes required by every CSML Scene are achieved by building a sub graph with multiple references to the content sub graph to be rendered. In the particular passes where specific styles must be applied to pixels falling inside the lens, we build the sub graph with the information associated with the

respective *context family*. Figure 2 shows how a CSML Scene is built during traversal. The parameterization of the sub graphs is based on the context sensitivity mechanism outlined in [14]. Notice that the sub graph is referenced multiple times independent of the number of objects inside it. This sub graph is constructed with the sequential traversal of the style sub graph (defined by the CSML Families) and the content sub graph (given to the CSML Scene). The style sub graph precedes the content sub graph and can therefore influence its appearance. In the case that multiple Magic Lenses are in the graph, the same content sub graph will be rendered with different style sub graphs. In this sense, the style sub graph, which is provided separately, can be seen as a *style parameter* to CSML Scene rendering.

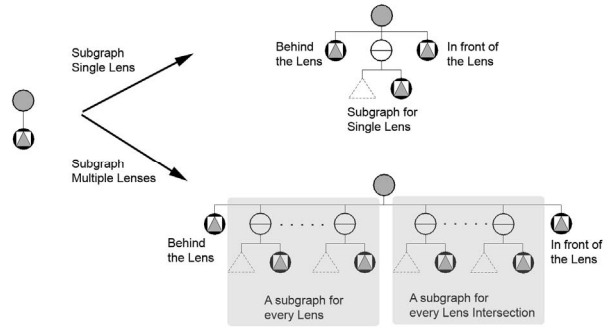

Figure 2 Overview of how a CSML Scene is built from a given content (grey triangle in circle) in the multiple lenses setup. For every lens, the content is traversed once, preceeded by the corresponding style (triangle) of its *context family* with the appropriate defined style. Every style sub graph (dotted triangle) represents the information defined by the *context family*.

It is important to note that the *style parameters* are completely arbitrary sub graphs, and can incorporate any standard or user defined features of Coin. The styles are thus not constrained to simple color or transparency changes.

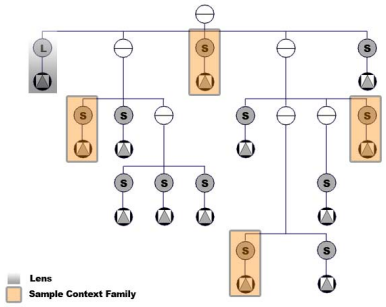

Figure 3 An example scene graph that illustrates how the context families do not need to be hierarchically grouped and yet they are affected by the lens given their context information.

An illustration with a sample scene graph is shown in Figure 3. Here we show how the context families do not need to be hierarchically grouped, and yet, because of context sensitivity, they are correctly affected by the Magic Lens. A 2D conceptual view of how context sensitivity can affect objects in a complex scene is presented in Figure 4. In this illustration those pixels of the objects that fall inside the lens (grey semi transparent square) are rendered with a specific *style parameter*, depending on which *context family* they belong to. The objects in the image are not grouped in sub graphs, but they are grouped by context.

We do not restrict our *style parameters* to contain only appearance modifiers, but they are allowed to contain arbitrary information, which may include, for example, transformations. However, there is an inherent difference between affecting the material properties of an object and, for example, adding transformations. While the appearance state is overwritten every time new material nodes are placed in the graph, transformations, on the other hand, are concatenated (see Figure 5). Therefore, a special treatment must be made when two or more lenses intersect and the *style parameter* for at least one of them was set to add a transformation. Whenever this happens, the CSML Scene sub graph must consider not only objects falling inside a Magic Lens region, but also those falling inside the intersection of two or more Magic Lenses. We have already presented a solution for this problem in Section 4.3 that adds, for every lens intersection, an additional branch to our CSML Scene sub graph treating the objects differently (see Figure 6).

These additional branches must compose their style sub graph from a grouping of the style sub graphs of the involved lenses in the intersection. We detect the intersection of the lenses by checking whether a fragment from the CSML Scene falls inside more than one of the textures of the Magic Lenses at the same time.

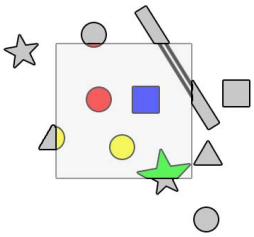

Figure 4 A conceptual 2D view on how a Context Sensitive Magic Lens affects a complex scene. All those fragments intersected by the lens are rendered differently, regardless of their position in the graph, affecting not only the rendering style but all aspects that can be composed as a sub graph.

Transparency always presents issues in interactive applications. Several techniques have been studied to overcome this problem [6]. While our technique can efficiently make use of, for example, alpha blending with one lens in the scene, the behavior turns more complex when multiple lenses come into play. This is a result from that we do not restrain the position of the lenses in our graph and that they can be mixed in the hierarchy of the scene. To solve this problem, we require a small modification of the last rendering pass, to allow the objects in the scene to check whether they are effectively outside every lens area (given by the textures).

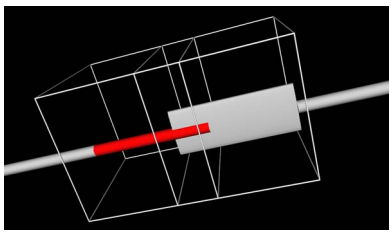

Figure 5 Example of wrong styling of an object in the intersecting area of two Context Sensitive Magic Lenses

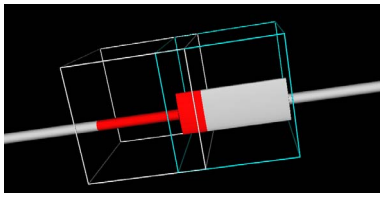

Figure 6 Example of correct styling in the intersecting area of two Context Sensitive Magic Lenses

This technique may decrease the rendering speed only slightly because no extra rendering passes are necessary. However, it can be deactivated by user request. Also, it must be noted that, when this technique is used, the last pass checks the effective outside regions of all the lenses, and consequently, the first rendering pass becomes redundant and can be deactivated (by the removal of the first branch of the sub graph). Since rendering order of the lenses is controlled by the user, this gives us the power to control the order of the *style parameters* in an scenario where multiple lenses intersect. However, when it comes to transparent content inside a lens, an arbitrary lens order forces us to use an order independent rendering strategy such as screen door.

It must be remarked that this transparency issue occurs only when multiple lenses are in the scene graph, and therefore, does not affect the algorithm for a single lens.

5 Results and Application Directions

We have envisioned a number of applications that can be addressed by the use of CSML. Following, some practical examples are given. The images shown throughout this article where generated on a Windows PC with 3GHz CPU and NVidia GeForce 7800GT. As tracking systems, we used ARToolkitPlus[**] and ARTTrack[††]. The geographical data of downtown Graz was extracted from a GE Smallworld database, provided by our partner company Grintec GesmbH. Liver and vessel tree models where taken from a surgery planning system[‡‡].

We distinguish two cases - applications that can be solved by using a single Magic Lens, and applications with special requirements which require multiple Magic Lenses.

5.1.1. Modeling of an X-ray vision tunnel

An obvious use for Magic Lenses is that of X-Ray vision. At UCSB an interaction tool for X-Ray vision was developed by Bane and Höllerer [1]. Two approaches were centrally addressed, volume and room based. In particular the volume approach could be addressed by the use of CSML. The authors describe a tool that creates a virtual tunnel in the field of view of the user. The rendering style of the objects is then affected if they intersect this virtual tunnel.

This work uses the concept of layers, where objects can belong to a "class" which determines whether an object is displayed or not. However, it appears that objects are not affected differently when they intersect the tunnel tool. In other words, the context given to the objects by the use of the layers does not include the intersection with the lens.

We have designed a similar concept of the tunnel tool that makes use of CSML. In Figure 7 we present a conceptual diagram of how the lens is used. One scenario for this concept is that all objects falling inside the lens are made semi-transparent by adding them all to the same *context family*. However, to truly exploit context sensitivity, objects should be affected depending on their *context family*. For example, the outside walls of a building that fall inside the lens will be made semi transparent, whereas interior objects such as furniture are rendered in a standard color (blue), while objects of higher importance such as characters are highlighted (red).

As the camera moves or the X-Ray plane moves, those pixels of the objects falling inside the lens are affected accordingly. In Figure 8 we show a screenshot example of this concept.

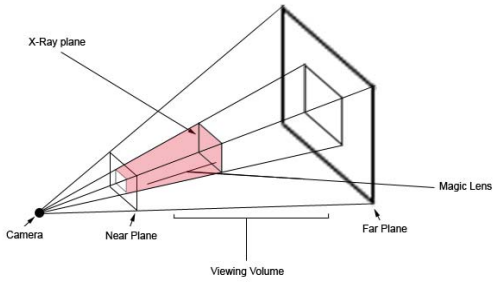

Figure 7 A conceptual diagram of how a single Context Sensitive Magic Lens can be used to mimic an X-Ray vision tunnel.

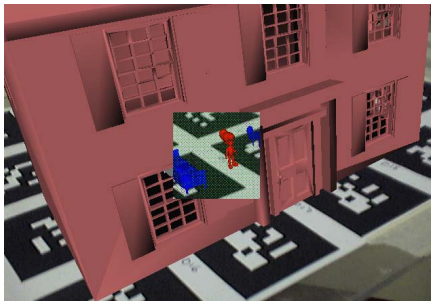

Figure 8 An X-Ray vision tunnel-like technique implemented with Context Sensitive Magic Lenses.

5.1.2. Context Sensitive X-Ray Tool

Naturally, our X-Ray tool does not have to be attached to the viewing frustum and can be instead moved and adjusted interactively by the user. On Figure 1 (right) the user holds a tracked Magic Lens in front of a liver model prop. By intersecting the lens with the liver, the user is able to see the vessel trees inside the liver. In particular, the vessel trees are differently colored, in red or blue, while the parenchyma of the liver, which falls inside the lens, is made transparent.

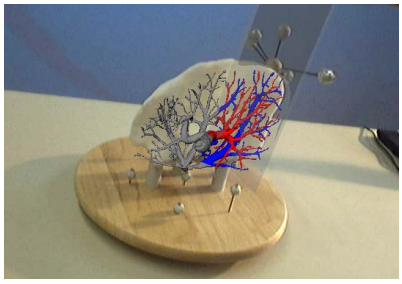

Figure 9 An example of localized visual disambiguation of a liver's vessel trees using Context Sensitive Magic Lenses.

[**] http://studierstube.icg.tu-graz.ac.at/handheld_ar/artoolkitplus.php
[††] http://www.ar-tracking.de/
[‡‡] http://liverplanner.tu-graz.ac.at

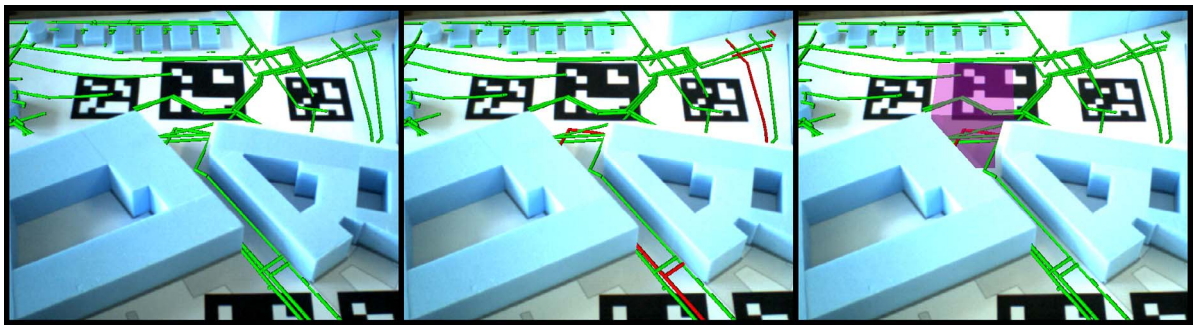

Figure 10 An example of how a Context Sensitive Magic Lens can be used for localized visual disambiguation. On the left most image a scaled model of downtown Graz is overlaid with power lines and gas pipes, as can be seen it is an ambiguous representation. On the centre image, gas pipes have been colored red in the whole scene. On the right most image, the style rendering change has been localized to be inside the lens geometry. This is possible with a context rich scene graph. The blue foam models where created from real geographical data at a scale of 1:250. The power and gas data, similarly, comes from the same real dataset.

5.1.3. Localized Visual Disambiguation

In a dense AR scene, augmentations can often be ambiguous. This is the case of Figure 10 (left) where multiple families of objects with very different semantics are represented in a similar style. Gas and electricity pipes have been rendered on top of the video input. Given the geometrical similarity of both types of objects, it is impossible to tell them apart.

In this case, a common technique to disambiguate these objects is by changing their color (middle image), or some other type of rendering style change. Conventionally, such color-coding is applied to the entire scene. CSML offers the opportunity to selectively apply the disambiguation. Users can place a lens in a specific area (Figure 10 right) where they want to disambiguate objects, without affecting the whole of the scene.

Another example of this application is shown in Figure 9 where two types of vessel trees are overlaid on top of a liver model. These two vessel types are visually similar, but can be locally disambiguated by intersecting them with a Magic Lens and changing their rendering style accordingly.

5.1.4. Cutaway Volumes and Ghosting

Feiner and Seligman discussed a number of techniques to satisfy visibility constrains in dynamic 3D illustrations [7]. Three techniques were mainly discussed: object removal, ghosting, and cutaway. Of these, the last two can also be modeled by CSML.

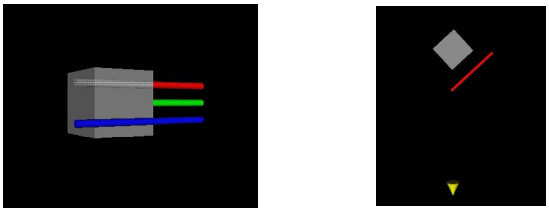

Figure 11 Examples of ghosts, cutaways and occluding objects with Context Sensitive Magic Lenses. On the right a top view of the objects and camera position.

In Figure 11 (left) an example of ghosting and cutaway is shown. We define four families of objects, *non-occludable*, *ghosts*, *cutaways* and *occluding*. Those objects belonging to the family of *ghosts* will be made semi transparent when falling in the line of sight between a *non-occludable* object and the camera.

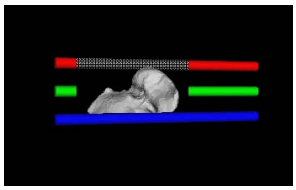

Figure 12 A lens used with the bounding box of a more complex object.

Similarly objects belonging to the family of *cutaways* will not be displayed and *occluding* objects will be shown without any rendering style modification. This can be achieved by dynamically creating a lens that is formed from the back faces of the *non-occludable* object and the near plane of the viewing frustum. This design is similar to the concept of the X-Ray vision tunnel presented before. However, as we mentioned, our lenses must be convex, which limits the shape of the *non-occludable* object. This can be solved by splitting the concave shapes into multiple convex shapes. However, a simpler solution is to create the Magic Lens shape from the back faces of the bounding box of the *non-occludable* object as illustrated in Figure 12, alternatively the convex hull of the object can also be used.

As can be seen, in the cutaway objects the depth information is lost, even though the object is in front of the non-occludable object (Figure 11 right). As a remedy, Coffin and Höllerer [4] proposed the use of Constructive Solid Geometry (CSG) operations. We are considering this for future work.

5.1.5. Information Revealing

As described by Bier et al., two more possibilities of the use of Magic Lenses are to enhance data of interest and to reveal hidden information. These two behaviors can simultaneously be achieved with CSML.

For example, in Figure 13 we show a real model overlaid with 3D representation of buildings, between them, power lines (green) and gas pipes (red) are presented (in this case, for illustration purposes, we use mock-ups). A lens has been placed in the scene intersecting the pipes and the virtual buildings. Those pixels of the buildings that fall inside the lens are cutaway, allowing revealing the information behind them. In turn, the pipes are enhanced with more detailed information on their structure.

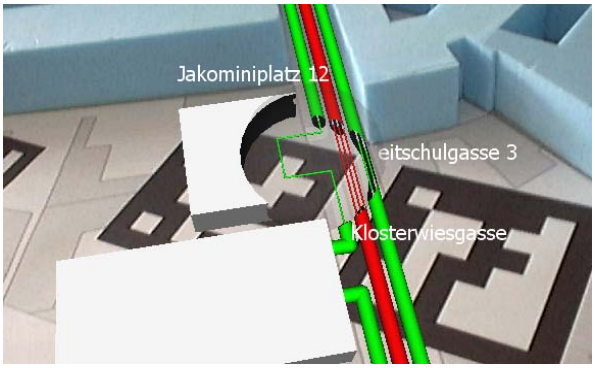

Figure 13 A Context Sensitive Magic Lens is used to enhance the information of some objects (details on pipes structures) and to present hidden information (by object cutaway of the buildings).

5.1.6. Visual Feedback

Object selection tools are amongst the most used interaction techniques in virtual environments. They allow users to manipulate objects in the scene. This is usually done regardless of their distance to the position of the user. One of the problems with selection tools is how to give feedback to users about what objects have been selected, or what parts of an object have been selected.

For example, the aperture selection tool, developed by Forsberg in 1996 [8], allowed the user to select a set of objects in the scene with a "cone-like" object. Many projects have adopted this technique for selection of objects at a certain distance of the user. However, visual feedback about the selection is usually done by displaying a bounding box around the whole selected object, or by rendering the selection tool with a certain transparency.

This limited precision of the selection feedback can be overcome by CSML. By defining the aperture cone as a Magic Lens, the exact portion of the object that is selected by the aperture can be visually highlighted. This technique can of course be combined with the conventional bounding box technique if desired.

5.2. Applications of Multiple Magic Lenses

For many of the AR applications, the use of only one Magic Lens will be sufficient, however, a specific set of applications will require the use of multiple lenses. In this section we outline a few examples in which by using multiple context sensitive 3D Magic Lenses, we are able to apply different rendering styles to a single object depending on which lens it intersects.

5.2.1. Multiple Lenses X-ray tunnel

An extended implementation of the X-ray tunnel technique [1] treats the adjacent areas in the user's viewing frustum differently. A similar assembly is presented in Figure 14 where multiple lenses are attached to the viewing frustum. Every CSML Scene can now define the specific behavior it will present when intersecting each of the three presented lenses. This technique in combination with that described by Feiner and Seligmann for dynamic illustrations [7] can prove to be a powerful way of displaying X-ray vision, as objects intersecting the lenses will change their rendering style given their own contextual information which can be interpreted as Ghosting or Cutaway.

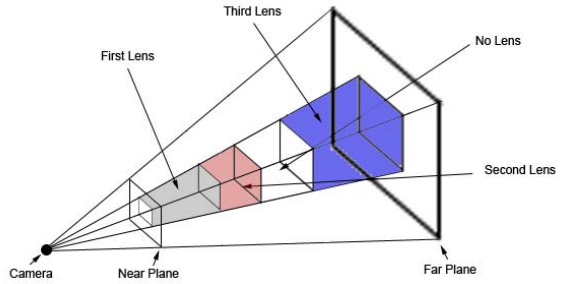

Figure 14 A conceptual diagram on how multiple Context Sensitive Magic Lenses can be used to mimic an X-ray vision tunnel.

5.2.2. Interactive Disambiguation

A single context sensitive 3D Magic Lens can be used to aid the understanding of a scene, by visualizing different parts in easily distinguishable styles. Sometimes, however, the structure of an object assembly is so dense and spatially complex, that, to understand the three dimensional scenario in detail, an interaction with more than one lens will be needed.

For example, in Figure 15, the user wants to explore the location and the geometric relationship of the tumors and of both of the two main vessel trees of a liver. To find the tumors, first the user searches for them by interactively intersecting the entire scene with a lens. This lens renders the tumors in green, while at the same time it renders the vessels in transparent Figure 15(A). Up to here, this shows how a single lens can be used to search for objects in a scene. To furthermore find out about the relationship of the tumors and its surrounding vessels, two more lenses are added that each affect only one of these vessel trees Figure 15(B)(C). By moving the lenses through the scene the user can interactively visualize parts of the vessel trees Figure 15(D). Such an interaction helps in developing a mental map of the geometrical relationship between the vessel trees and the tumors, which in the end gives a better understanding of the 3D scene.

6 Summary and Future Work

The idea of CSML came from working with highly complex AR scenes such as geo-data models, where global information filtering does either too little or too much for the desired effect. The heterogeneity of our complex scene graphs precluded the use of traditional 3D Lenses, because every special case for every object family would have to be hard-coded. As a solution, we have introduced context-sensitive behavior for Magic Lenses to enhance usability and rapid prototyping of interactive 3D environments.

The use of context sensitivity in a scene graph permits dynamic creation of sub graphs during traversal [14]. This mechanism shifts the complexity of multiple rendering styles from the application code to the scene graph data structure. While this makes scene graph design more complex, it can mostly be overcome in larger practical applications by automating the generation of scene graphs through translators and script generators rather than handcrafting the data.

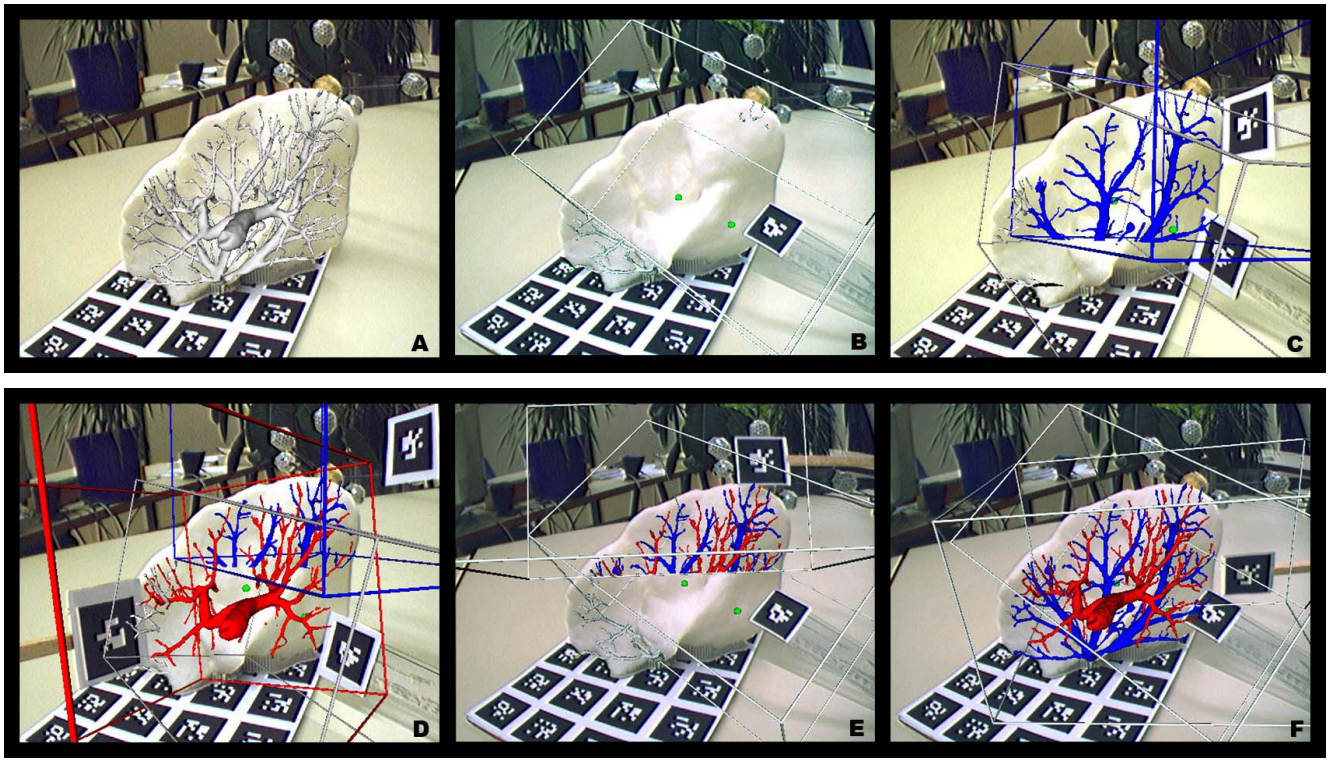

Figure 15 Example of an exploration of a tumor and its surrounding vessels trees, aided by four different context sensitive 3d magic lenses. For a better understanding of the scene, all involved lenses are shown by its silhouette.
A) shows two tumors and the liver's hepatic and portal vessel tree, rendered with the same style applied
B) by intersecting with the first lens, the tumors falling inside the lens' volume are rendered in green, while both vessel trees turn transparent
C) intersections of one of the vessel trees with the second lens will be rendered in blue – notice: the tumors are not affected by the second lens and therefore still rendered in green
D) with the aid of a third lens, we are able to render parts of the liver's second vessel tree in red
E) a fourth lens is substituting the second and third lenses
F) same as (E) but intersecting the entire scene

As was shown in Section 5, a wide variety of applications can be addressed by CSML. In particular, we are pursuing the idea of X-ray vision tool and information revealing.

By allowing a number of Lenses to intersect, we must also define the order of effects applied. As mentioned in Section 4.3 in our current implementation the style applied depends on the combination of the style sub graphs of all intersecting lenses. However, we are currently working on an algorithm to reduce the number of rendering passes. Since this is a closely related problem, we are, at the same time, developing Boolean operations.

To provide the user with better depth information, Coffin et al. [4] suggest the use of CSG operations. However, interactive CSG rendering is itself a multi-pass technique and not trivially integrated with CSML. We intend to address this as future work.

Finally, CSML is freely available under GPL as part of the *Studierstube* framework (http://www.studierstube.org).

Acknowledgements

Our thanks go to Grintec GesmbH for providing data for the localized visual disambiguation example. We thank also Alexander Bornik and Thomas Pock for the segmentation data of the liver and its vessel trees and to Markus Grabner for the invaluable help with CG programming. This research was funded in part by FWF (Y193), FFG (BRIDGE 811000) and the European Union (European Union MRTN-CT-2004-512400).

References

[1] Bane Ryan and Höllerer Tobias, "Interactive tools for virtual x-ray vision in mobile augmented reality," In proceedings International Symposium on Mixed and Augmented Reality, 2004, pp. 231-239

[2] Benford S. and Fahlen L. E., "A Spatial Model of Interaction in Large Virtual Environments," In Proceedings of the Third European Conference on CSCW, 1993, pp. 107

[3] Bier Eric, Stone Maureen, Pier Ken, Buxton William, DeRose Tony, "Toolglass and Magic Lenses: the see-through interface," In proceedings SIGGRAPH 1993, pp. 73-80

[4] Coffin Chris and Höllerer Tobias, "Interactive Perspective Cut-away Views for General 3D Scenes" In Proceedings of 3DUI 2006: The First IEEE Symposium on 3D User Interfaces. Technote, Alexandria, VA, Mar 25–26, 2006

[5] Diepstraten J., Weiskopf D., and Ertl T., "Interactive cutaway illustrations," In Procceedings of Eurographics, 2003, pp. 523-532

[6] Everitt Cass, "Interactive order-independent transparency," Technical report, NVIDIA Corporation, May 2001

[7] Feiner Steven, Seligmann Dorée, "Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations," The Visual Computer 8(5&6) , 1992, pp. 292-302

[8] Forsberg Andrew, Herndon Kenneth, Zeleznik Robert, "Aperture Based Selection for Immersive Virtual Environments," In proceedings ACM Symposium on User Interface Software and Technology 1996, pp. 95-96

[9] Fuhrmann A., Hesina G., Faure F., Gervautz M., “Occlusion in collaborative augmented environments,” Computers & Graphics 23, 1999, pp. 809-819

[10] Furmanski C., Azuma R., Daily M., “Augmented-Reality Visualizations Guided by Cognition: Perceptual Heuristics for Combining Visible and Obscured Information,” International Symposium on Mixed and Augmented Reality, 2002, pp. 215-224

[11] Höllerer T., Feiner S., Hallaway D., Bell B., Lanzagorta M., Brown D., Julier S., Baillot Y., Rosenblum L., “User Interface Management Techniques for Collaborative Mobile Augmented Reality,” In Computers and Graphics 25, 2001, pp. 799-810

[12] Looser J., Billinghurst M., Cockburn A., “Through the looking glass: the use of lenses as an interface tool for Augmented Reality interfaces,” In Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and SouthEast Asia (Graphite 2004). 15-18th June, Singapore, 2004, ACM Press, New York, New York, pp. 204-211

[13] W. Mark, R. S. Glanville, K. Akeley, M. Kilgard. Cg: A system for programming graphics hardware in a C-like language. Proceedings SIGGRAPH 1993.

[14] Reitmayr Gerhard, Schmalstieg Dieter, "Flexible Parameterization of Scene Graphs," In proceedings IEEE Virtual Reality Conference 2005 (VR'05), 2005, pp. 51-58

[15] Ropinski Timo and Hinrichs Klaus, “Real-Time Rendering of 3D Magic Lenses having arbitrary convex Shapes,” In proceedings International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, 2004, pp. 379-386

[16] Schmalstieg, D., Fuhrmann, A., Hesina, G., Szalavari, Z., Encarnação, L. M., Gervautz, M., and Purgathofer, W. (2002). The Studierstube Augmented Reality Project. PRESENCE - Teleoperators and Virtual Environments, 11(1).

[17] Viega John, Conway Matthew, Williams George, Pausch Randy, “3D Magic Lenses,” In proceedings ACM Symposium on User Interface Software and Technology, 1996, pp. 51-58

[18] Wang L., Zhao Y., Mueller K., Kaufman A., “The Magic Volume Lens: An Interactive Focus+Context Technique for Volume Rendering,”, In IEEE Visualization, 2005

[19] Strauss, P. and Carey, R. (1992). An object oriented 3D graphics toolkit. In Proc. SIGGRAPH'92.