Adaptive Augmented Reality Using Context Markup and Style Maps

Erick Mendez[1]
Graz University of Technology
Institute for Computer Graphics and Vision
Inffeldgasse 16a, Graz, 8010, Austria

Dieter Schmalstieg[2]
Graz University of Technology
Institute for Computer Graphics and Vision
Inffeldgasse 16a, Graz, 8010, Austria

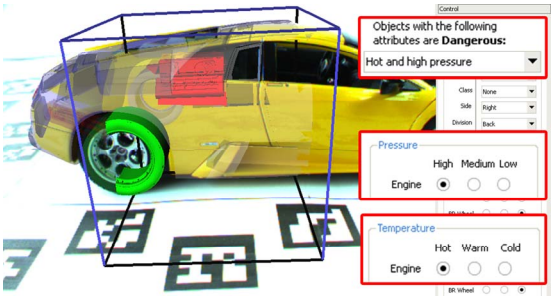

Figure 1 In this example the user selects a predefined mapping of context markup to styles and changes the rendering styles of the objects in the scene accordingly.

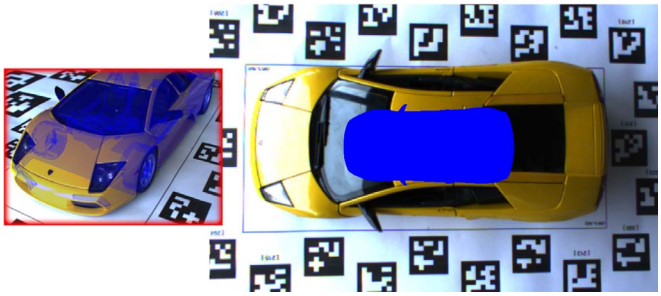

Figure 2 The virtual object's context markup is derived from its expected error. The resulting rendering (blue) is shrunk to guarantee that the overlay falls within the registered area.

Abstract

Augmented Reality (AR) enables users to visualize synthetic information overlaid on top of real imagery. Such visualization may be achieved by tools that distort, filter or enhance the explored information. However, little to no work has focused on the separation of style definitions and their mapping to scene objects. We target this separation based on a context rich scenegraph. Our research allows the definition of visualization styles independent on the data to be visualized.

CR Categories and Subject Descriptors: H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities; I.3.6 [Methodology and Techniques]: Interaction techniques, Graphics data structures and data types.

Additional Keywords: Object overlay and spatial layout techniques, real-time rendering, interaction techniques for MR/AR.

1 Introduction and Related Work

A frequent goal of Augmented Reality (AR) is to display information linked to the real world. High density of such information leads to display clutter, so information filtering techniques have been investigated as a remedy. The goal of information filtering is usually dynamically changing. For instance, a tired tourist's interest may change from cultural landmarks to public transportation and the route back to the hotel. Traditionally, application design demands a decision on the range of possible objects of interest and their respective styles beforehand. If a scenegraph based implementation is used, the application code must be tightly coupled with the scenegraph data structure so that procedural calls can modify the rendering styles of the right portions of the scenegraph.

Ideally, application behavior leading to the definition of the styles, the selection of styles and the objects to which the styles are applied should be separate concerns. The application code should only control the mapping of styles to objects fitting a particular characterization, without touching the objects directly. Our goal is a framework that targets this separation based on context markup of the scenegraph. Our work enables AR application designers to define visualization styles, independent of the underlying data.

Many approaches exist that let a user choose a technique and then apply it uniformly to a particular dataset. For example, the original magic lens [2] applies a style uniformly to everything contained in the lens. Context Sensitive Magic Lenses (CSML) [5] extends magic lenses with object styles, but does not allow the mapping of these styles to be dynamic.

Rule based visualization systems address the selection of style or content by dynamically applying a set of user-defined rules. A well-known work on information filtering for AR was introduced in [3], which uses spatial proximities to compute relative object importance given user properties such as tasks sets. This type of rule-based approaches are complementary to our work in the sense that they can easily be adapted to output style mappings or context attributes rather than directly modifying scene objects.

2 Context and Mapping define Style

All of the previously discussed techniques require a prior knowledge of the scene objects to be visualized, or have to perform expensive searches for objects. This limits the usability for large scenes and complex sets of categories. The technique presented here does not have these shortcomings since its complexity is shifted away from the application and towards the data itself. The key to achieve this lies on two simple techniques: First, the scene should be composed of textual attributes in addition to geometrical. We call this, *Context Markup*. Second, it should also contain mappings that assign rendering styles to objects in the scene. We call these *style maps* and *style templates* respectively.

A *context family* of objects is defined as all objects that have the same context attributes. However, these objects do not have to belong to the same part of the scenegraph hierarchy – they can be scattered throughout the scenegraph and still be jointly affected. The family membership of an object is the result of an aggregation of context attributes encountered while traversing the path from the scenegraph's root node to the object itself. This mechanism

[1] mendez@icg.tugraz.at
[2] schmalstieg@icg.tugraz.at

works exactly like other traversal states such as transformations, where these are aggregated depending on their position in the graph. We call the total set of context attributes of a node, its *context markup*.

Geometrical information is encoded in *Styled subgraphs*. A particular *styled subgraph* is member of several *context families* at the same time, which become known during the actual traversal of the subgraph itself. These family memberships depend on the partial or total context attributes present during traversal. This allows members of a particular family to be in hierarchically disconnected areas of the scenegraph. Our system can, therefore, affect objects without a detailed knowledge of the scenegraph arrangement.

A *style template* is a system resource given in the form of a named rendering style subgraph. This typically is composed of nodes controlling the visual appearance, such as material definitions, textures, or even transformations. *Style templates* are used to influence the appearance of *styled subgraphs*. During the rendering traversal every *styled subgraph* is preceded by the first *style template* mapped to one of its *context families*. The mechanism to achieve this is similar to that of [5].

The mapping of a *context family* to a *style template* is determined by another system resource, the *style map*. Every entry in the *style map* assigns a context family to a *style template*. The *style map* is composed of individual nodes, each defining one mapping arranged hierarchically as part of the scenegraph. During traversal of the scenegraph, every *styled subgraph* checks the mappings. Then, depending on whether its own context markup is mapped, it fetches the appropriate rendering style subgraph defined by the template (see Figure 3). This check depends only on the number of available mappings.

The mappings function like cascading stylesheets, since every *styled subgraph* will progressively search for the first mapping that matches one of its *context families*. *Styled subgraphs* may come from legacy data bases such as those from the Geographical Information Systems (GIS) community.

Tracking uncertainties like those described by Machado et al. [4] may be used as *context markup*, for example. The *style templates*, however, are usually defined by application designers since visual styles and naming conventions may carry a semantic meaning. Style maps in turn may come from a range of possible sources such as high complexity rule sets that consider user properties and tasks [3].

3 Adaptive Augmented Reality Scenarios

Consider an example application where the user has to repair a particular piece of a car. These pieces have been enriched with contextual information such as whether they are interior or exterior, seats or engine, and so on. This application allows users to mark a family of objects for repairing (in green) based on their contextual attributes. At the same time, dangerous objects may also be highlighted so that the user may be aware of potentially problematic situations (in red). Additionally we partially display the car's body depending on the position of the viewer. By doing this we filter distracting artifacts that may needlessly increase the depth complexity of the scene.

Figure 1 shows a screenshot of this application. The designer of which does not need to specify which objects are, for example, dangerous because such semantic interpretation of dangerous objects may not be known during design time. The mapping of styles and specific context families happens during the execution of the application. These mappings require a combination of context attributes, and a template name to which this context family will be mapped to. The mappings are not predefined during the application implementation but are explicitly selected by the user. In this case the car engine matches the given set of attributes (enlarged in a box) and is rendered accordingly. In this application the three style templates are in use providing different information: visual depth cueing, highlight problematic pieces and draw the attention to focus objects. This means that three tasks run concurrently: spatial context, danger awareness and priority targets. These three tasks are highly reminiscent to those of [3].

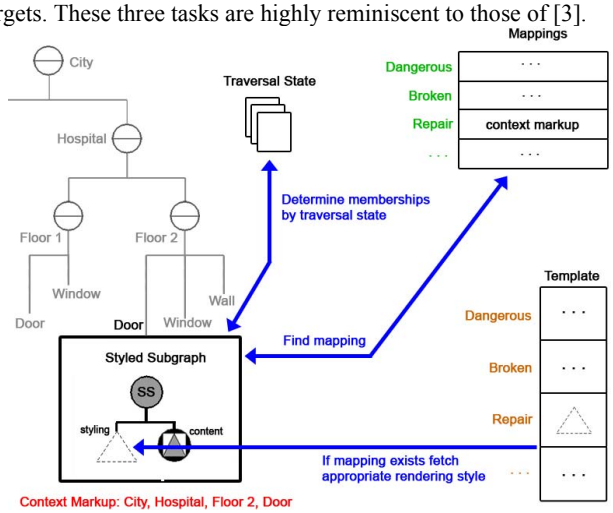

Figure 3 During traversal every styled subgraph checks for the first mapping that matches its context markup and then precedes the rendering style to its content.

Another potential application of such context markup is to scale the object depending on its tracking uncertainty. The resulting scaled rendering may be used to identify "screen real estate" regions similar to those used by Bell et al. [1] for annotation. More powerful labeling techniques may be obtained by having a real estate region defined not only as a function of the object's shape, but as a mixture of its shape, tracking uncertainty and contextual markup. Figure 2 shows an example of this technique, the object's markup is obtained from the expected error of the fiducial tracking (higher when closer to the normal of the marker).

4 Conclusion and Future Work

We have presented a framework that allows the separation of styles from the objects in the scene, based in contextual information. This makes it easier to design complex content, a requirement not unique to AR but very relevant if one wants to incorporate legacy data sources. We are now interested in implementing rule-sets that will output style mappings to our system. Additionally, a better formalization of our separation of styles and mapping is already being defined for a further publication. The fact that application designers let users influence the styles of objects implicitly instead of explicitly will reduce the amount of interaction required. This may translate into simpler user interfaces, which play a prominent role in AR.

REFERENCES

[1] Bell, B. et al, "View management for virtual and augmented reality," Proc. UIST '01 (CHI Letters, vol. 3, no. 2), pp. 101-110.

[2] Bier E. et al, "Toolglass and Magic Lenses: the see-through interface," In proceedings SIGGRAPH 1993, pp. 73-80

[3] Julier S. et al, Information filtering for mobile augmented reality. In Proc. ISAR 2000, pp 3-11

[4] Machado E. et al, "OSGAR: A Scenegraph with Uncertain Transformations" In Proceedings ISMAR04, pp 6-15

[5] Mendez E., et al, "Interactive Context-Driven Visualisation Tools for Augmented Reality," In Proceedings of ISMAR 2006, pp. 209-218