Combining ARToolKit with Scene Graph Libraries

Michael Haller
University of Applied Science at Hagenberg (MTD)
haller@fh-hagenberg.at

Werner Hartmann
Institute for Applied Knowledge Processing (FAW), University of Linz
hartmann@faw.uni-linz.ac.at

Thomas Luckeneder
Institute for Applied Knowledge Processing (FAW), University of Linz
luckeneder@faw.uni-linz.ac.at

Jürgen Zauner
University of Applied Science at Hagenberg (MTD
jzauner@fh-hagenberg.at

1. Introduction

Many Augmented and Mixed Reality applications are based on two libraries: OpenGL is used for rendering and ARToolKit [1] is used for marker recognition. The ARToolKit library is great for rapid prototyping of AR/MR applications. The library is very easy to use and it hides the complexity of marker recognition.

In the AMIRE [2] (Authoring Mixed Reality) project, a European founded AR project, we follow up the aim of ARToolKit consistently: the AMIRE approach is to offer well-established gems (software solutions) and components for a faster prototyping of AR/MR applications. Each content user should be able to develop his/her own AR/MR application without any computer graphics skills. Therefore, one of the primary goals of AMIRE is to find well-established solutions of current AR/MR applications. One of the solution is the ARToolKit library, which is used in numberless AR/MR applications. But which library should be used for rendering? Can we use a high-level graphics API together with ARToolKit? Which graphic library would be the best for further AR/MR applications? Should developers use Direct3D/OpenGL or should we propose a high-level graphics API, like Open Inventor [5], OpenGL Performer [3], OpenSG [4], or Open SceneGraph [6]?

High-level graphics APIs have a number of advantages as opposed to low-level graphics APIs. They include:

- A set of loaders (e.g. model and texture loaders)
- A scenegraph concept
- Modern object oriented design
- High performance (optimized rendering, view frustum culling, small object culling, Level of Detail nodes, etc.)

One problem still remains: How difficult is the usage of ARToolKit, originally based on OpenGL, in combination with a high-level graphics API like OpenSG?
For the AMIRE project we tested two different APIs: Open SG and Open SceneGraph.

2. Integration in a Scene Graph Library

To integrate ARToolKit with a Scene Node Library at least the following three tasks have to be performed:

- **Initialising the Camera –** ARToolKit delivers a matrix that describes the intrinsic parameters of the video camera. This matrix has to be applied to the projection matrix of the virtual camera.
- **Transforming the Object –** The transformation matrix that describes distance and orientation offset between a marker and the camera must be applied to the virtual object.
- **Displaying the Video –** The video image has to be applied to the rendered image as background.

To improve the quality of the integration work, also the following can be considered:

- **“Real Occluders” –** It is often desired that real objects should occlude virtual ones to get a more realistic application. To enable this, rudimentary models of the real environment have to be built. These models can be used to modify the depth information of the rendering system to generate occlusion effects.

3. Implementation

3.1. Video and Tracking Abstraction Layer

To make the integration of ARToolKit into high-level graphics frameworks as OpenSG and Open SceneGraph as easy as possible we have decided to create an object-oriented abstraction of video capturing and object tracking functionality. We have used this abstraction layer only to integrate the object tracking and overlaying of live video images into the high-level framework. So, the integration of different tracking technologies is minimised to the implementation of a specific subclass. In our case we have integrated the ARToolKit video capturing, object recognition and tracking abilities.

3.2. OpenSG experience

OpenSG is a library that provides multi-threading safety. This makes it easy to use this library in a multi-thread environment. But it also complicates it to extend and modify this library. Every OpenSG node class is derived from a FieldContainer class and has one or more Fields. These Fields contain all data of a Node. To build a new OpenSG node the Fields of this node have to be described with a tool called fscEdit. This tool generates an XML description and source code to manage the Fields of the node. The generated source code has to be combined with the source code that represents the functionality of the node. Thus, it makes it very hard to extend OpenSG. For this reason we tried to integrate ARToolKit into OpenSG without extending OpenSG. The first integration step, “initialising the camera” was fairly easy, because OpenSG provides a class called MatrixCamera. This MatrixCamera contains two Matrix Fields, one for the projection matrix and one for the modelview matrix. Thus, setting the projection matrix of a MatrixCamera with the projection matrix of ARToolKit initialized the OpenSG camera. The next step, “transforming the object” was done by adding a transformation node at the root of the scene graph and by filling with the modelview matrix of ARToolKit. The visualization of the video was a little bit more difficult, because this necessitated an extension of OpenSG. OpenSG uses so called background objects to clear the screen before rendering. For displaying the video we had to create a new background class that did not only clear the screen, but it also displayed the video data. We called that background "VideoWallBackground". Due to the fact that a background object does not need its own fields, the implementation of this class was done without using fscEdit. Figure 1 shows a virtual object that is rendered by OpenSG and placed by ARToolKit. The realization of the "real occluders" was done by first rendering the occluders, then clearing the screen with the VideoWallBackground, (this overwrote the screen with the videoimage, but preserved the depth information) and finally drawing the virtual object. Figure 2 shows the effect of a real occlusion. Figure 3 depicts the real occluder.

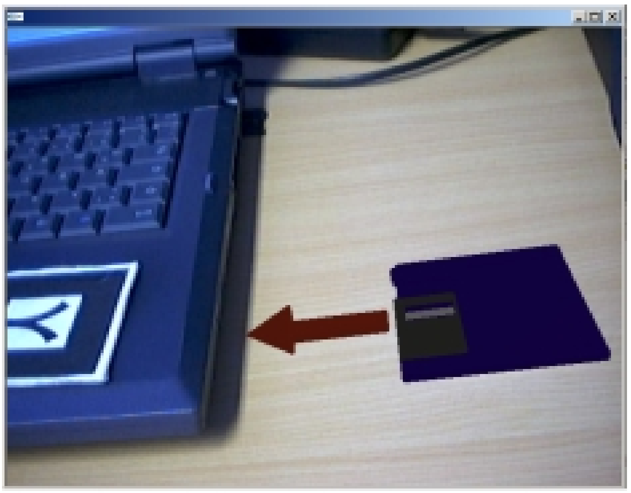
Figure 1: An OpenSG rendered disk, placed by ARToolKit.

3.3. Open SceneGraph experience

Open SceneGraph is much easier to extend than OpenSG, because OpenSG supports thread safety by creating copies of the objects as opposed to Open SceneGraph. Open SceneGraph is also very similar to OpenGL Performer [3], which is a well-known high performance 3D rendering toolkit for developers of real-time, multiprocessed, interactive graphics applications. This similarity makes it much easier to understand how Open SceneGraph works. We have extended the classes GLUT viewer class to integrate the video capturing, object tracking and overlaying into the display method of a MR viewer. To provide occluding real objects we implemented a MR scene view extending the scene view class of Open SceneGraph. First we display the occluding geometry of the real objects, then we overwrite the color buffer with the video image without manipulating the depth buffer and at last we display the objects recognized by our ARToolKit based object-tracker.

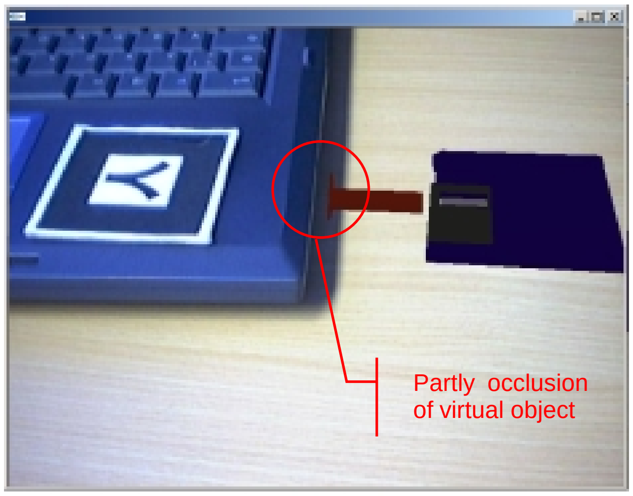

Figure 2 The virtual disk, partly occluded by the real notebook.

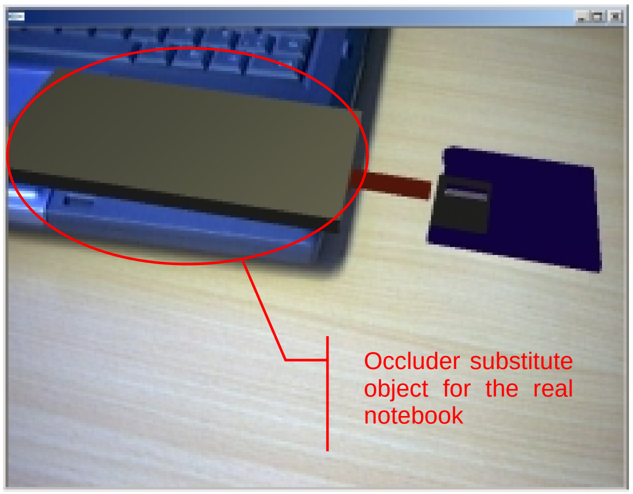

Figure 3: Virtual substitute for real notebook.

4. References

[1] Kato H., Billinghurst M., Blanding B., May R., *ARToolKit*, Technical Report, Hiroshima City University, December, 1999.
[2] http://www.amire.net
[3] http://www.sgi.com/developers/devtools/apis/performer.html
[4] http://www.opensg.org
[5] http://www.sgi.com/software/inventor
[6] http://www.openscenegraph.org