Technotes #2 – Developer Defined Custom Attributes Registry

Written by: Nick Thompson, Pablo Fernicola, & Rick Evans
Date: 3/20/96

Introduction

Custom attributes allow you to attach custom data to your 3D objects, and to store this data along with the object when it is written out to a 3DMF metafile. For more information on the implementation of custom attributes please consult the Addison-Wesley book "3D Graphics Programming with QuickDraw 3D". Also check out the article in Develop issue 26 on custom attributes (available via the World Wide Web at <http://dev.info.apple.com/develop/developtoc.html>).

This technote is the repository of information about developer defined custom attributes. Apple computer is providing this repository as a service to developers to facilitate sharing of 3D information between applications. Inclusion in this technote does not imply any endorsement of any particular custom attribute by Apple Computer Inc. In the future Apple may review the implementations presented here and then explicitly endorse specific custom attributes that are found to be in alignment with the goals and directions of QuickDraw 3D.

Thanks to the following developers for sharing their ideas with the QuickDraw 3D community. As we get more suggestions, we'll add the proposer to this list:

• Robin Lansbert, MicroSpot - developers of 3D World

We'll be providing freebies in the form of tee shirts, CDRoms and the like to the authors of the proposals that we particularly like, so send 'em in!!

To add your custom attributes to the growing list of implementations in the public domain, please contact Apple via AppleLink at DEVSUPPORT, or the authors directly at REVANS or NICKT. (E-Mail can be sent to AppleLink from the internet by appending "@applelink.apple.com" to the AppleLink addresses.) We will need to see an implementation in code (C or C++ please) of your attribute, together with some supporting documentation. The documentation will be added to this technote, and we'll put the implementation with the documentaion on our Web site and on the SDK CD. The information in here is subject to change, so be sure to check this technote on a regular basis.

Please note by submitting your custom attribute for publication in this technote you are placing that the implementation of your attributes in the public domain. This will enable users of other

applications to freely move data between those applications that provide support for your custom attribute/element (for a detailed discussion on the differences between custom attributes and elements, please see the Develop article mentioned above).

Apple defined custom attributes

This section describes some of the custom attributes that engineers in the QuickDraw 3D group and Developer Support have been working on. Use this section and the accompanying sample code (in the file CustomAttribute_Lib.c) as a template for documenting and implementing your own custom attributes. Please include this kind of information (in terms of the level of details for both the description and the implementation of your custom attribute/element) when you submit your attributes to Apple to share with other developers. We'll put the code up on the Web and in various other places to allow other developers to read and write custom attributes in the form you describe.

The "CustomAttribute" simple application (supplied with the QuickDraw 3D SDK, and also available on the Develop Issue 26 CD) makes use of custom attributes. Open the 3DMF files in the accompanying folder to get an idea of how this works.

The code for the following custom attributes is supplied:

name
scale
up vector
forward direction
W3 anchor
W3 inline

The function RegisterAllCustomAttributes() in CustomAttribute_Lib.c registers all the types (and you should also call UnregisterAllCustomAttributes when exiting).

You want to look at the functions AdjustCamera and FindAndDisplayCustomAttributes, both in CustomAttributesSupport.c, for examples on how to use the attributes. How you integrate these attributes in your application is very application specific. Regard this code as an example. In this sample, I deal with the upVector, forwardDirection, and scale when reading the models in. I deal with the name and URL attributes when the user selects the object.

Note that for the upVector and forwardDirection the code is not dealing with the case where the current up direction/view direction is the negation of the model's vectors. The code is based on vector dot and cross products, which produce the same result for vectors that are colinear and for vectors that are colinear, but with opposing directions. See AdjustCamera() for more information.

Description

name

This element contains a string object. It can be attached to any object within the Shape hierarchy. It can also be attached to an attribute set and assigned to a geometry and/or faces. Note that in future releases we will have other subclasses to the String class, allowing you to use non-ASCII characters.

```
Container (
        NameAttribute ( )
        CString ( "1 meter box" )
)
```

scale
This element is of type double. It determines the relation between 1 unit in the model and 1 meter. For example, if 1 unit in my model is the equivalent to 10 meters, the scale should be set to 10. This element should only be attached to groups or geometry objects. If you add objects that have a scale attribute to a group, make sure that the objects are transformed (objects are placed in a group with a transform and then added to the main group) so that the scale for the group is uniform. Traversal by applications should be top down. This means that if the scale element is found on a group, there is no need to traverse the objects within the group.

```
Container (
        ScaleAttribute ( 1.0 )
)
```

up vector
This element is of type TQ3Vector3D. It specifies the up vector for a model.
This element should only be attached to groups or geometry objects. If you encapsulate objects that have up vector elements within a group, make sure that the objects are transformed (objects are placed in a group with a transform and then added to the main group) so that the up vector for the group is uniform. Traversal by applications should be top down. This means that if the upVector element is found on a group, there is no need to traverse the objects within the group.

```
Container (
        UpVector ( 0.0 1.0 0.0)
)
```

forward direction
This element is of type TQ3Vector3D. It specifies the forward direction for a model.
This element should only be attached to groups or geometry objects. If you encapsulate objects that have up vector elements within a group, make sure that the objects are transformed (objects are placed in a group with a transform and then added to the main group) so that the forward direction for the group is uniform. Traversal by applications should be top down. This means that if the forwadDirection element is found on a group, there is no need to traverse the objects within the group.

```
Container (
        ForwardDirection ( 1.0 0.0 0.5)
)
```

W3 Anchor
This element contains an URL (universal resource locator) in the form of an C string (ASCII only), an option field, which can be set to kURLReferenceOptionUseMap (meaning that the application should attach a point to the URL before sending the URL to the server), and a string object to encapsulate the description of the site pointed to by the URL (note that this allows for non-ASCII descriptions in the future).
This element should only be attached to groups or geometry objects. Traversal by applications should be top down. This means that if the element is found on a group, there is no need to traverse the objects within the group.

```
Container (
        W3Anchor ("http://www.info.apple.com" 0)
                CString ( "Apple's home page"
                )
)
```

W3 Inline
This element contains an URL (universal resource locator) in the form of an C string (ASCII only). This element should only be attached to groups or geometry objects. The group or geometry that this element is attached to acts as a proxy for the data pointer to by the URL. This allows the

application to perform the URL data retrieval on a separate thread or in the background, or delay the operation until the user expresses interest on the proxy. Once the URL data is retreived, the data should replace the object that holds the element. Traversal by applications should be top down. This means that if the element is found on a group, there is no need to traverse the objects within the group.

```
Container (
        W3Inline ("http:www.info.apple.com")
)
```

Submission from MicroSpot

Here are all the custom attributes that we use plus all of the standard ones that Apple have already defined. The only thing that I do not know that we are doing right is whether the names entered in the registering of the attributes are correct. I am only documenting the custom attributes that Microspot have added although the source code contains all the Apple ones too.

```
struct THandleRecord

    {

    unsigned long theSize;

    Handle theHandle;

    };
```

All the following Microspot attributes are based around the THandleRecord structure. This basically holds a size or length followed by a handle to the data.

- 'snd ': Holds a handle to a sound which can be played with PlaySnd.
- 'PICT': Holds a handle to a QuickDraw picture which can be drawn with DrawPicture
- 'TEXT': Holds a handle to a block of text. Its length is found using GetHandleSize
- 'moov': Holds a handle to an alias to a movie
- 'desc': Holds a handle to a block of text that holds the description of an object
- 'LOCK': a 32 bit integer whose bits represent the objects locked or constrained state, settings for this bit are as follows:

```
kObjectLockedBit = 1    // the object is completely locked and may

                        // not be modified in any way

kObjectConstrainPosXBit = 2 // position is not allowed to move in X

kObjectConstrainPosYBit = 3

kObjectConstrainPosZBit = 4
```

```
kObjectConstrainSizeXBit = 5 // size is not allowed to grow in X

kObjectConstrainSizeYBit = 6

kObjectConstrainSizeZBit = 7

kObjectConstrainRotateXBit = 8 // angle is not allowed to rotate in X

kObjectConstrainRotateYBit = 9

kObjectConstrainRotateZBit = 10

kObjectPrimitiveBit = 11 // an object or group is a primitive and

                         //may not be ungrouped.
```

Proposed new attributes but not implemented:

The cost attribute may need more thinking about and maybe Apple has some good ideas as to how to add a price to an item. We will need this attribute when we want to have a report of the items in a design with a costing.

- 'alis': A THandleRecord which holds a handle to an alias.
- 'COST': A double which holds the price of an item in a standard currency (Dollar or Pound?? should probably be stored along with the currency type so conversions can take place)

If you have a suggestion for a custom attribute type, or comments, or implementations for suggested attributes, please contact us and we'll add them to this technote.

Proposals for Collaboration

This section is for stimulating discussion regarding ideas for future Custom Attributes. If you would like to facilitate a collaborative development effort on custom attributes, get in touch with us and we'll add a section here.

AREF (Architectural Exchange Format)

InterStudio, a European developer based in Italy, are interested in supporting AR.E.F. (Architectural Exchange Format) as custom attributes and would like to work with other developers on this. Their products: Domus.Cad and Nonio C, which support QuickDraw™3D (using both interactive rendering and the 3DMF Metafile support) have been shipping, with QD3D support, since January 2nd 1996. They are working in the architechural modelling field.

They are very interested in using QuickDraw™3D Metafile to export and import topological relationships between objects, user defined attributes and architectural and mechanical attributes, and do this now in their current applications using AR.E.F. They would like to work with other developers in their field, if you'd like to work with them on this as a well defined custom attribute that can be shared and used by many developers please contact them at this address:

Sauro Agostini
Interstudio S.r.l.

Via Borgo Melano 27
I-51100 Pistoia, Italy
Tel ++39 573 31307 Fax ++39 573 23039
Internet sauro@softeam.it

Agreement on an implementation for this would be a big plus for users, and would really add value to 3DMF.

Other Sources of Information

if you are a registered developer, contact Developer Support DEVSUPPORT@applelink.apple.com.

Also, you may want to check out www.info.apple.com/qd3d for the latest info. Last, we monitor the newsgroup comp.sys.mac.graphics and sometimes answer questions there.