Preface

This report describes activities of CGM Open meeting held on September 21, 2003 in Washington DC at the Embassy Suites Hotel.

Table of Contents

1 Meeting Details.................................................................................................................2
1.1 Location and Dates .................................................................................................2
1.2 Meeting....................................................................................................................2
1.3 CGM Open Attendees .............................................................................................2
2 Agenda..........................................................................................................................2
2.1 Committee...............................................................................................................2
3 Output and Action Items.............................................................................................2
4 Agenda discussions.....................................................................................................3
  4.1 CGM Open membership/OASIS.............................................................................3
  4.2 Vendor product web pages.....................................................................................3
  4.3 Interoperability web pages....................................................................................3
  4.4 SVG status ............................................................................................................4
  4.5 XML encoded CGM .............................................................................................4
  4.6 WebCGM DOM.....................................................................................................4
5 Note of appreciation....................................................................................................14
1 Meeting Details

1.1 Location and Dates
Embassy Suites Hotel, Washington DC, September 21, 2003

1.2 Meeting

1.3 CGM Open Attendees
- Dave Cruikshank – Boeing
- Dieter Weidenbruck – ITEDO
- Ulrich Laesche – Ematek
- Benoit Bezaire – Corel Corp
- Franck DuLuc – EADS/Airbus
- Andrew Moorhouse – UK MOD
- Forrest Carpenter – SDI
- Kevin O’Kane – Auto-trol

2 Agenda

2.1 Committee
The items on the agenda of the Committee include:
- CGM Open membership/OASIS
- Vendor product web pages
- Interoperability web pages
- SVG status
- XML encoded CGM
- WebCGM DOM

3 Output and Action Items

<table>
<thead>
<tr>
<th>Item</th>
<th>Who</th>
<th>When</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meeting Minutes</td>
<td>Cruikshank</td>
<td>10/15</td>
<td>Done</td>
</tr>
</tbody>
</table>

Governance Actions

<table>
<thead>
<tr>
<th>Discussion CGM Open with OASIS with Patrick Gannon</th>
<th>Henderson/Weidenbruck/Cruikshank</th>
<th>10/17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circulate changes to bylaws</td>
<td>Henderson</td>
<td>Pending OASIS</td>
</tr>
</tbody>
</table>
4 Agenda discussions

4.1 CGM Open membership/OASIS

CGM Open is in the process of welcoming two new members. The Navy and Arbortext, both of whom are current OASIS members, will notify OASIS of their intent to participate in CGM Open work at their next membership renewal cycle.

OASIS continues to express a desire that CGM Open move from affiliate status to become a member section. Discussions with Patrick Gannon, president and CEO of OASIS, will continue during a telecon scheduled for October 17 with Lofton, Dieter, and Dave.

4.2 Vendor product web pages

All links to the various product ICS pages now resolve correctly. Some products have been reassigned to the correct category. References to “transcoders” has been replaced with “converters” on the web pages.

The proposed changes to the viewer and editor ICS formats are still pending and will be included in the next update of the product information.

4.3 Interoperability web pages

The problem tracking external interface was reviewed. The following comments were captured:
• The product list from the product web pages needs to be inserted into the “tool” pull-down menu
• A window needs to be added to capture tools not listed
• A method needs to be defined to direct responsibility for tool problems reported for which solutions are provided by CGM Open vendors (e.g., Framemaker & Epic)
• Need to add link on page to submit comments to administrator
• Need to add link on CGM Open home page to a “Problem Reporting” page that describes the interoperability project and outlines the process
• Need to develop a proposed time line for problem resolution, based on Franck’s process charts

A proposal was accepted to register “report.cgmopen.org” as the domain name to implement. For “guest” problem reports, Dave volunteered to act as administrator. Once the comments are addressed, CGM Open will do a month’s worth of testing and go live with the service. A proposal to attempt to go live by December 1, 2003 was accepted.

4.4 SVG status
The W3C SVG committee is in the process of developing SVG 1.2, which will be reorganized to contain a “core” functionality section. With the development of this section, it will be possible to write profiles of SVG. Initial work on SVG 1.2 is expected to be complete Q1/Q2 of 2004, with final approval in the Q3/Q4 2004 time frame.

4.5 XML encoded CGM
The membership conducted a lengthy discussion on the potential of an XML encoded specification for WebCGM or CGM. The following comments were noted:
  • Indexing text for searching and updating links dynamically are some use cases of XML encoded CGM
  • Having a DOM in place would address some but not all of the requirements
  • Adding angle brackets to the Clear Text encoding does not address all of the requirements
  • It’s unlikely that there would be much support in the W3C for a second XML encoded 2D graphics format
  • Is it likely that the ISO SC24 committee would be interested in this work?
  • This project would divert resources from WebCGM DOM development

As a result of the discussion it was decided that the work on the WebCGM DOM has priority over an XML encoding. It appears that that an XML encoding of CGM or WebCGM would involve a substantial amount of work from both the vendors and users to develop software and specifications. Based on an analysis of use cases, there does not appear to be much return on investment.

CGM Open recognizes the need to have graphics expressible in XML for the web, but it probably makes more sense to apply resources to define the mapping from WebCGM to SVG for web delivery. If the SVG committee structures the SVG specification so that valid profiles can be written, CGM Open would be interested in contributing to a WebCGM SVG profile.

Dieter and Lofton need to validate this approach with Chris Lilley of the W3C.

4.6 WebCGM DOM
Benoit provided an initial draft of a DOM specification to support WebCGM. This initial work represents a viewer DOM. At some point in the future, it may be extended to an authoring DOM where persistence will be addressed. Additional work need to be done on the DOM to satisfy the
requirements for control of primitive attributes at the picture and APS level defined previously. In
addition, a common interface between the WebCGM APS and metadata in XML companion files needs
to be developed. Benoit volunteered to continue as the DOM editor/developer and will work with the
other vendors to finalize it.

The first working draft of the DOM is included here in the minutes.

-------------------------------------------------------------------------------------------------------------------------------------------

WebCGM DOM (1st Working Draft)

Legend:
Not required in WebCGM DOM

-------------------------------------------------------------------------------------------------------------------------------------------

interface Node {
    // NodeType
    const unsigned short PICBODY_NODE = 1;
    const unsigned short LAYER_NODE = 2;
    const unsigned short GOBJECT_NODE = 3;
    const unsigned short PARA_NODE = 4;
    const unsigned short GDATA_NODE = 5;
    const unsigned short SUBPARA_NODE = 6;

    readonly attribute DOMString nodeName;
    readonly attribute unsigned short nodeType;
    readonly attribute Node parentNode;
    readonly attribute Node firstChild;
    readonly attribute Node lastChild;
    readonly attribute Node previousSibling;
    readonly attribute Node nextSibling;
    readonly attribute Document ownerDocument;
    readonly attribute NodeList childNodes;

    boolean hasChildNodes();
    boolean hasAttributes();

    const unsigned short ELEMENT_NODE = 1;
    const unsigned short ATTRIBUTE_NODE = 2;
    const unsigned short TEXT_NODE = 3;
    const unsigned short CDATA_SECTION_NODE = 4;
    const unsigned short ENTITY_REFERENCE_NODE = 5;
    const unsigned short ENTITY_NODE = 6;
    const unsigned short PROCESSING_INSTRUCTION_NODE = 7;
    const unsigned short COMMENT_NODE = 8;
    const unsigned short DOCUMENT_NODE = 9;
    const unsigned short DOCUMENT_TYPE_NODE = 10;
    const unsigned short DOCUMENT_FRAGMENT_NODE = 11;
    const unsigned short NOTATION_NODE = 12;

    readonly attribute NamedNodeMap attributes;
    attribute DOMString nodeValue; // raises(DOMException) on setting // raises(DOMException) on retrieval

    Node insertBefore(in Node newChild, in Node refChild) raises(DOMException);
    Node replaceChild(in Node newChild, in Node oldChild) raises(DOMException);
    Node removeChild(in Node oldChild) raises(DOMException);
Node appendChild(in Node newChild) raises(DOMException);
Node cloneNode(in boolean deep);
void normalize();
boolean isSupported(in DOMString feature, in DOMString version);
readonly attribute DOMString namespaceURI;
attribute DOMString prefix; // raises(DOMException) on setting
readonly attribute DOMString localName;
);

Definition group NodeType
An integer indicating which type of node this is.

Note: Numeric codes up to 200 are reserved to W3C for possible future use.

Defined Constants
- PICBODY_NODE
  The node is a picbody
- LAYER_NODE
  The node is a layer
- GROBJECT_NODE
  The node is a grobject
- PARA_NODE
  The node is a para
- GDATA_NODE
  The node is a gdata
- SUBPARA_NODE
  The node is a subpara

Attributes
childNodes of type NodeList, readonly
  A NodeList that contains all children of this node. If there are no children, this is a
  NodeList containing no nodes.

firstChild of type Node, readonly
  The first child of this node. If there is no such node, this returns null.

lastChild of type Node, readonly
  The last child of this node. If there is no such node, this returns null.

nextSibling of type Node, readonly
  The node immediately following this node. If there is no such node, this returns null.

nodeName of type DOMString, readonly
  The name of this node, depending on its type; see the table above.

nodeType of type unsigned short, readonly
  A code representing the type of the underlying object, as defined above.

ownerDocument of type Document, readonly
  The Document object associated with this node. This is also the Document object used to
  create new nodes. When this node is a Document or a DocumentType
  which is not used with any Document yet, this is null.

parentNode of type Node, readonly
  The parent of this node. All nodes, except Attr, Document,

previousSibling of type Node, readonly

Page 6 of 14
The node immediately preceding this node. If there is no such node, this returns null.

Methods
hasAttributes
  Returns whether this node (if it is an element) has any attributes.
  Return Value
  boolean true if this node has any attributes, false otherwise.
  No Parameters
  No Exceptions

hasChildNodes
  Returns whether this node has any children.
  Return Value
  boolean true if this node has any children, false otherwise.
  No Parameters
  No Exceptions

interface Document : Node {
  readonly attribute Element documentElement;

  NodeList getElementsByTagName(in DOMString tagname);
  Element getElementById(in DOMString elementId);
  NodeList getElementsByTagNameValue (in DOMString name, in DOMString value);
  NodeList getElementsByURIFragment (in DOMString urifragment);

  readonly attribute DocumentType doctype;
  readonly attribute DOMImplementation implementation;

  Element createElement(in DOMString tagName) //raises(DOMException);
  DocumentFragment createDocumentFragment();
  Text createTextNode(in DOMString data);
  Comment createComment(in DOMString data);
  CDATASection createCDATASection(in DOMString data) //raises(DOMException);
  ProcessingInstruction createProcessingInstruction(in DOMString target, in DOMString data) // raises(DOMException);
  Attr createAttribute(in DOMString name) //raises(DOMException);
  EntityReference createEntityReference(in DOMString name) //raises(DOMException);
  Node importNode(in Node importedNode, in boolean deep) //raises(DOMException);
  Element createElementNS(in DOMString namespaceURI, in DOMString qualifiedName) //raises(DOMException);
  Attr createAttributeNS(in DOMString namespaceURI, in DOMString qualifiedName) //raises(DOMException);
  NodeList getElementsByTagNameNS(in DOMString namespaceURI, in DOMString localName);
}

Attributes
documentElement of type Element, readonly
  This is a convenience attribute that allows direct access to the child node that is the root element of the document.
Methods
getElementById
Returns the Element whose ID is given by elementId. If no such element exists, returns null. Behavior is not defined if more than one element has this ID.

Note: The DOM implementation must have information that says which attributes are of type ID. Attributes with the name "ID" are not of type ID unless so defined. Implementations that do not know whether attributes are of type ID or not are expected to return null.

Parameters
elementId of type DOMString
The unique id value for an element.

Return Value
Element The matching element.

No Exceptions

getElementsByTagName
Returns a NodeList of all the Elements with a given tag name in the order in which they are encountered in a preorder traversal of the Document tree.

Parameters
tagname of type DOMString
The name of the tag to match on. The special value "*" matches all tags.

Return Value
NodeList A new NodeList object containing all the matched Elements

No Exceptions

interface NodeList {
Node item(in unsigned long index);
readonly attribute unsigned long length;
}

Attributes
length of type unsigned long, readonly
The number of nodes in the list. The range of valid child node indices is 0 to length-1 inclusive.

Methods
item
Returns the indexth item in the collection. If index is greater than or equal to the number of nodes in the list, this returns null.

Parameters
index of type unsigned long
Index into the collection.

Return Value
Node
The node at the indexth position in the NodeList, or null if that is not a valid index.

No Exceptions
interface Element : Node {
    readonly attribute DOMString tagName;
    DOMString getAttribute(in DOMString name);
    void setAttribute(in DOMString name, in DOMString value) //raises(DOMException);
    void removeAttribute(in DOMString name) //raises(DOMException);
    NodeList getElementsByTagName(in DOMString name);
    boolean hasAttribute(in DOMString name);
    Attr getAttributeNode(in DOMString name);
    Attr setAttributeNode(in Attr newAttr) //raises(DOMException);
    Attr removeAttributeNode(in Attr oldAttr) //raises(DOMException);
    DOMString getAttributeNS(in DOMString namespaceURI, in DOMString localName);
    void setAttributeNS(in DOMString namespaceURI, in DOMString qualifiedName, in DOMString value) //raises(DOMException);
    void removeAttributeNS(in DOMString namespaceURI, in DOMString localName) //raises(DOMException);
    Attr getAttributeNodeNS(in DOMString namespaceURI, in DOMString localName);
    Attr setAttributeNodeNS(in Attr newAttr) //raises(DOMException);
    NodeList getElementsByTagNameNS(in DOMString namespaceURI, in DOMString localName);
    boolean hasAttributeNS(in DOMString namespaceURI, in DOMString localName);
};

Attributes
tagName of type DOMString, readonly
    The name of the element. For example, in:
    <elementExample id="demo">
        ...
    </elementExample> ,
tagName has the value "elementExample". Note that this is case-preserving in XML, as are all of the operations of the DOM. The HTML DOM returns the tagName of an HTML element in the canonical uppercase form, regardless of the case in the source HTML document.

Methods
getAttribute
    Retrieves an attribute value by name.
    Parameters
        name of type DOMString
    The name of the attribute to retrieve.
    Return Value
        DOMString
    The Attr value as a string, or the empty string if that attribute does not have a specified or default value.
    No Exceptions

elemntsByTagExample
    Returns a NodeList of all descendant Elements with a given tag name, in the order in which they are encountered in a preorder traversal of this Element tree.
    Parameters
        name of type DOMString
    The name of the tag to match on. The special value "*" matches all tags.
    Return Value

Page 9 of 14
NodeList  A list of matching Element  nodes.

No Exceptions

hasAttribute
  Returns true  when an attribute with a given name is specified on this element or has a
default value, false  otherwise.
  Parameters
  name  of type DOMString
  The name of the attribute to look for.
  Return Value
  boolean  true  if an attribute with the given name is specified on this element or has a
default value, false  otherwise.
  No Exceptions

removeAttribute
  Removes an attribute by name. If the removed attribute is known to have a default value, an
attribute immediately appears containing the default value as well as the corresponding
namespace URI, local name, and prefix when applicable.
  Parameters
  name  of type DOMString
  The name of the attribute to remove.
  Exceptions
  DOMException
  NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
  No Return Value

setAttribute
  Adds a new attribute. If an attribute with that name is already present in the element, its value
is changed to be that of the value parameter. This value is a simple string; it is not parsed as it
is being set.
  Parameters
  name  of type DOMString
  The name of the attribute to create or alter.
  value  of type DOMString
  Value to set in string form.
  Exceptions
  DOMException
  INVALID_CHARACTER_ERR: Raised if the specified name
contains an illegal character.
  NO_MODIFICATION_ALLOWED_ERR: Raised if this node
is readonly.
  No Return Value

interface GetWebCGMDocument {
  Document getWebCGMDocument ()  //raises( DOMException );
};

Methods
getWebCGMDocument
  Returns the Document object for the referenced WebCGM document\n  No Parameters

Page 10 of 14
interface Event {
    readonly attribute EventTarget target;
    readonly attribute EventTarget currentTarget;
    readonly attribute DOMString type;

    // PhaseType
    const unsigned short CAPTURING_PHASE = 1;
    const unsigned short AT_TARGET = 2;
    const unsigned short BUBBLING_PHASE = 3;
    readonly attribute unsigned short eventPhase;
    readonly attribute boolean bubbles;
    readonly attribute boolean cancelable;
    readonly attribute DOMTimeStamp timeStamp;
    void stopPropagation();
    void preventDefault();
    void initEvent(in DOMString eventTypeArg,
                   in boolean canBubbleArg,
                   in boolean cancelableArg);
};

Attributes

currentTarget of type EventTarget, readonly
    Used to indicate the EventTarget whose EventListeners are currently being processed. This is particularly useful during capturing and bubbling.

target of type EventTarget, readonly
    Used to indicate the EventTarget to which the event was originally dispatched.

type of type DOMString, readonly
    The name of the event (case-insensitive). The name must be an XML name.
in boolean canBubbleArg,
in boolean cancelableArg,
in views::AbstractView viewArg,
in long detailArg,
in long screenXArg,
in long screenYArg,
in long clientXArg,
in long clientYArg,
in boolean ctrlKeyArg,
in boolean altKeyArg,
in boolean shiftKeyArg,
in boolean metaKeyArg,
in unsigned short buttonArg,
in EventTarget relatedTargetArg);
};

Attributes

altKey of type boolean, readonly
    Used to indicate whether the 'alt' key was depressed during the firing of the event. On some platforms this key may map to an alternative key name.

button of type unsigned short, readonly
    During mouse events caused by the depression or release of a mouse button, button is used to indicate which mouse button changed state. The values for button range from zero to indicate the left button of the mouse, one to indicate the middle button if present, and two to indicate the right button. For mice configured for left handed use in which the button actions are reversed the values are instead read from right to left.

clientX of type long, readonly
    The horizontal coordinate at which the event occurred relative to the DOM implementation's client area.

clientY of type long, readonly
    The vertical coordinate at which the event occurred relative to the DOM implementation's client area.

ctrlKey of type boolean, readonly
    Used to indicate whether the 'ctrl' key was depressed during the firing of the event.

metaKey of type boolean, readonly
    Used to indicate whether the 'meta' key was depressed during the firing of the event. On some platforms this key may map to an alternative key name.

screenX of type long, readonly
    The horizontal coordinate at which the event occurred relative to the origin of the screen coordinate system.

screenY of type long, readonly
    The vertical coordinate at which the event occurred relative to the origin of the screen coordinate system.

shiftKey of type boolean, readonly
    Used to indicate whether the 'shift' key was depressed during the firing of the event.

The different types of Mouse events that can occur are:
The click event occurs when the pointing device button is clicked over an element. A click is defined as a mousedown and mouseup over the same screen location. The sequence of these events is:

mousedown
mouseup
click

If multiple clicks occur at the same screen location, the sequence repeats with the detail attribute incrementing with each repetition. This event is valid for most elements.

Bubbles: Yes
Cancelable: Yes
Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey, metaKey, button,

The mousedown event occurs when the pointing device button is pressed over an element. This event is valid for most elements.

Bubbles: Yes
Cancelable: Yes
Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey, metaKey, button, detail

The mouseup event occurs when the pointing device button is released over an element. This event is valid for most elements.

Bubbles: Yes
Cancelable: Yes
Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey, metaKey, button, detail

The mouseover event occurs when the pointing device is moved onto an element. This event is valid for most elements.

Bubbles: Yes
Cancelable: Yes
Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey, metaKey

The mousemove event occurs when the pointing device is moved while it is over an element. This event is valid for most elements.

Bubbles: Yes
Cancelable: No
Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey, metaKey

The mouseout event occurs when the pointing device is moved away from an element. This event is valid for most elements.

Bubbles: Yes
Cancelable: Yes
Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey, metaKey,

Other possibilities: Key events, mutation events...
Need use cases to determine if they are required.
5 Note of appreciation

CGM Open would like to express our thanks to Dave for allowing his suite to be used for the meeting.