9B.10 CURRENTLY AVAILABLE AND SUPPORTED GRAPHICS METAFILE FORMATS AND THE CHALLENGES THEY PRESENT FOR USE IN FORECASTING APPLICATIONS

James Ramer*
NOAA Earth System Research Laboratory in collaboration with
the Cooperative Institute for Research in the Atmosphere
Colorado State University,
Fort Collins, Colorado.

1. INTRODUCTION

Since the late 1990's the National Weather Service's Advanced Weather Interactive Processing System (AWIPS) has been the primary tool used by National Weather Service (NWS) forecasters. At the National Oceanic and Atmospheric Administration's Earth Systems Research Laboratory in Boulder, Colorado, the Global Systems Division (GSD) has contributed a large part of the software for AWIPS (Jones et al. 2003). GSD continues to contribute to the AWIPS effort by assisting with troubleshooting and maintenence for the operational system, and by developing software to support new applications and data sets for AWIPS. Furthermore, GSD conducts research in new forecasting techniques and new information processing technologies. To support this effort, the Advanced Linux Prototype System (ALPS) has been developed.

The ALPS system (Grote et al. 2005) is kept backward compatible, supporting all current operational AWIPS capabilities, but providing an environment where GSD developers can prototype new capabilities free from the rigid constraints of operational configuration management.

Here we describe a prototype application developed in ALPS, which allows any viewable AWIPS graphic overlay to be exported as a file in the Keyhole Markup Language (KML) format. The KML format is a primary means of displaying geographic data in Google™ Earth (Google 2007). This exercise has been a valuable learning experience regarding the current state of graphics metafile technologies, plus GSD has already had many years of experience with older graphics metafile technologies, acquired while developing AWIPS and its precursors. Combined, this has lead to some useful insights about the future of graphics metafile technologies in forecasting applications.

* *Corresponding author address*: James Ramer, NOAA/GSD4, 325 Broadway, Boulder CO 80305; e-mail: james.e.ramer@noaa.gov.

2. DEFINITION OF SOME TERMS.

This discussion includes terminology that could be ambiguous if not clarified. These concepts are undoubtedly familiar to many who work with computer graphics in geoscience applications, but others may use different terminology to represent these concepts.

By *graphic overlay*, we generally refer to the entire set of like displayable items covering some geographic area at a given time for a coherent hydrometeorological data set. An example would be a set of contours and their labels for a specific level, time, parameter and field from a specific numerical model. A graphic overlay is mostly transparent, making it possible to meaningfully interpret multiple graphic overlays mapped to the same area simultaneously. An *image overlay*, in contrast, such as one might create from satellite data, is often fully opaque over its entire coverage area. The KML export utility described here only exports graphic overlays.

By *global coordinates*, we refer to coordinates that map to the display differently depending on the zoom and pan state (the pan state means where the zoom is centered). In this discussion, global coordinates for plan-view grapics are latitude and longitude coordinates. For graphics that are not plan-view, global coordinates would usually be defined in terms of other parameters.

By *frame coordinates*, we refer to coordinates that always map to the same location on the display regardless of the zoom or pan state. Frame coordinates are often used to display items that are legend type information.

One example of a use for *offset coordinates* is drawing a station model. In the station model example, the global coordinates of the station are supplied as the *offset origin*, and the individual items within the station model are drawn with *offset coordinates*, which are pixel offsets from that origin. This is how one can specify that zooming in does not cause the station model itself to get larger. *Unrotated offset coordinates* mean that the positive y direction in offset space is up on the screen, and *rotated offset coordinates* mean that the positive y direction in offset space is north. When one uses an offset that is the sum of a rotated and an unrotated offset, we refer to that as a *hybrid offset coordinate.* When a text string is plotted that has its location specified using a rotated offset, but the font is such that the character orientation and text baseline are defined as being rectilinear with raw pixel coordinates, then one is in effect using a hybrid offset.

We refer to the process of adding detail as one zooms in as *progressive disclosure*.

3. PROTOTYPE KML EXPORT APPLICATION.

The version of KML this work is based on is 2.1, and this part of the discussion assumes the reader has some knowledge of KML. A good reference for KML 2.1 can be found at:

http://code.google.com/kml/documentation/kml_tags_21.html

The KML that is output for this prototype export application uses a very limited set of KML features. Any drawable item that is positioned with offset coordinates, plus any text item positioned with global coordinates, is implemented by *<Icon>* elements embedded within *<Placemark>* elements. While *<Icon>* elements have a raster image as their core data structure, a raster image used in this manner is mostly transparent, because it is used to represent features that are really vector graphic in nature, rather than raster image in nature. In an attempt to maintain a consistent relationship between the pixels in the icons and pixels on the Google Earth display, an associated *<scale>* value is applied. To date, it has been impossible to make this correspondence completely consistent, even though icons are often artifically squared off in order to make the response to values for the scale more predictable. In the exported KML, rotated offset addressed items are given a *<heading>* value of zero; unrotated offset addressed items have no *<heading>* tag. All drawable items plotted in frame coordinates are implemented by a single large mostly transparent *<ScreenOverlay>* element. All other drawable items are implemented with *<LineString>* elements embedded within *<Placemark>* elements. Progressive disclosure is implemented by applying *<Region>* elements to the *<Placemark>* elements.

Each image that is used to make up the various text and offset addressed items, plus the image used for frame-addressed items, all end up in their own separate .png file. At one level, this result creates a complex file management problem. However, Google Earth supports loading files with a .kmz extension directly, which are zipped KML files. Furthermore, any other desired associated data files, such as the .png files mentioned, can be included in a .kmz file. Therefore, the final result of the export operation is a single .kmz file for each graphic overlay. In the rest of this section we discuss some figures that show the results of this effort.

Figures 1 and 2 show an export of a contour overlay, and an example of progressive disclosure being invoked. Both are the result of loading the exact same file into the Google Earth display, but the zoomed-in version shows more contours and labels. In figure 2, note that the contour that would be valued 510 is not visible, even though that would be consistent with the prevailing contour interval. Zooming in further does reveal that contour. Ideally, it should be possible to make the progressive disclosure work such that the contour interval is always consistent, but this goal has been unattainable so far.

Figure 1. Export of a 50-hPa height contour from a numerical model, valid 06UTC 09 November 2007.

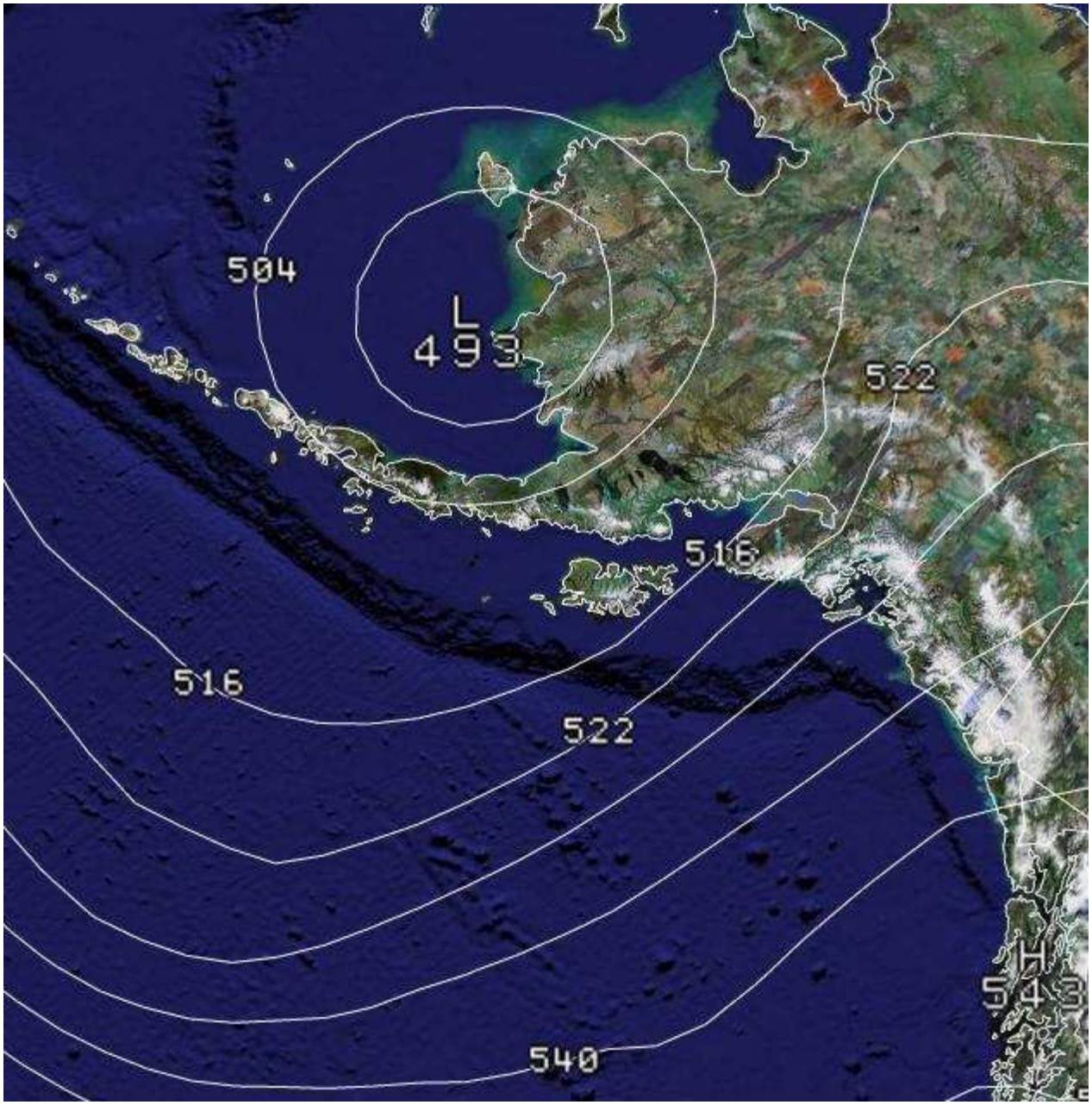

Figure 2. Same export as figure 1, but zoomed in to view more contours.

Figure 3. Export of a METAR plot valid 15 UTC 15 October 2007.

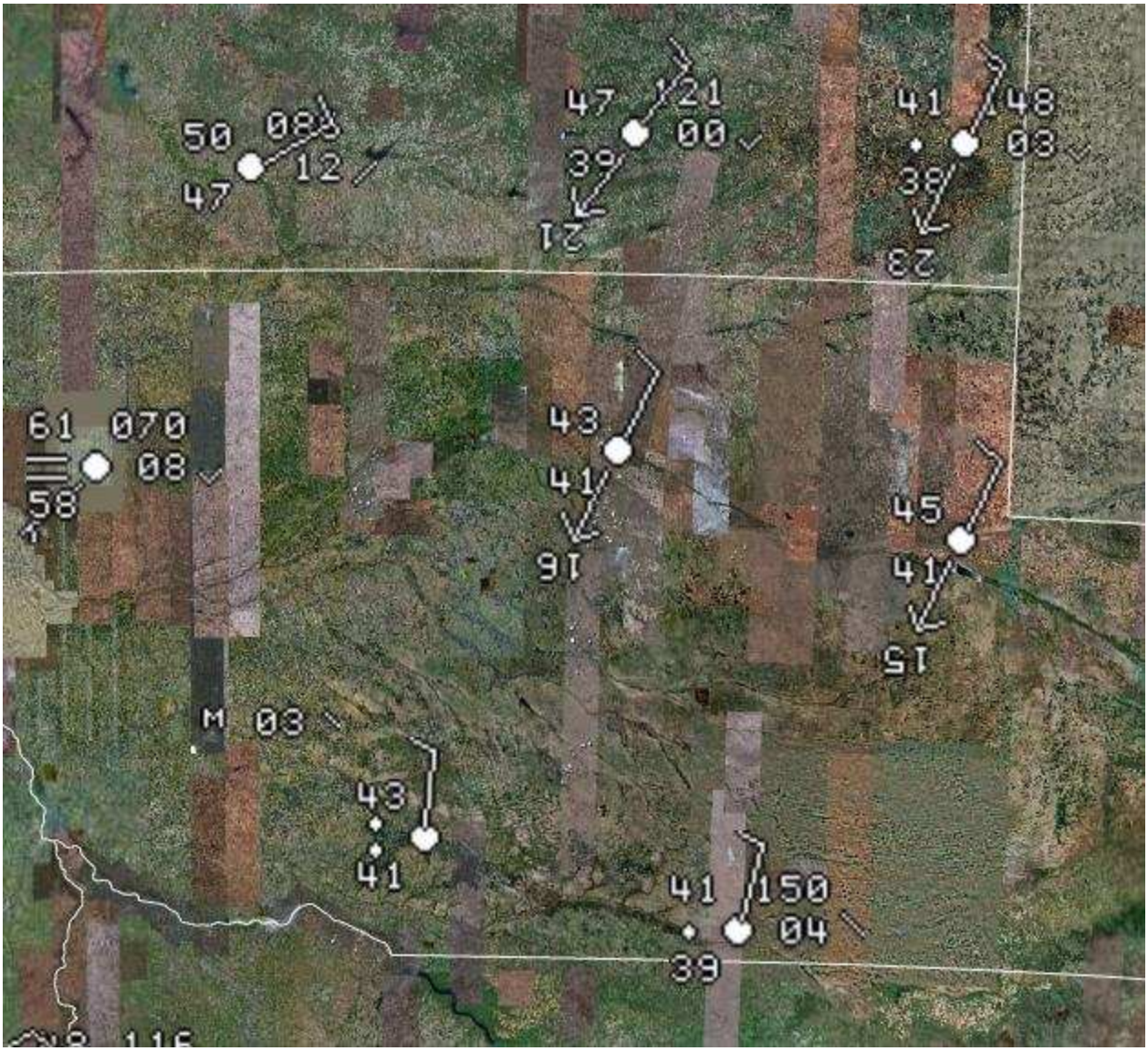

Figure 4. Same export as Figure 3, but zoomed in and rotated such that south is up.

Figure 5. A weekly drought monitor graphic valid 25 October 2007.

Figures 3 and 4 show the export of a METAR plot. Both are the same file, but in figure 4 the display has been zoomed in and rotated such that south is up. Note that the station models stay the same size even as the display is zoomed, and that additional stations have been added. Furthermore, note the station near the middle with a gust speed of 16. In the south-up version, the 16 gets turned upside down and now looks like 91; a meteorologically significant difference, one might argue. The reason this occurs is that no means has yet been discovered to support hybrid offsets in KML.

Figure 5 is a screen shot of a weekly drought monitor graphic. This shows an example of frame coordinates being used; the key displayed in the upper left will stay there regardless of how one zooms or pans. In figures 1 through 4, AWIPS has the grids and metar observations needed to create those displays decoded and stored locally. In contrast, AWIPS does not locally store the information about drought conditions needed to create the display in figure 5. This display comes to AWIPS over NOAAPORT (Kahn and Baer 1992) as a graphics metafile. This illustrates another good application for displays based on graphics metafiles; when the data the display is based upon is highly specialized information. Even though there is no code in the AWIPS system that knows anything about drought conditions per se, an AWIPS user is still able to view a meaningful display of drought conditions and overlay it with other information.

4. REVIEW OF SOME EXISTING GRAPHICS METAFILE TECHNOLOGIES

During the years that GSD and its precursor organizations have been doing research into meterological displays and forecasting techniques, we have had experience with many different graphics metafile formats. Back when the Automation of Fields Operations and Services (AFOS) system was the primary tool used by NWS forecasters, very little gridded numerical model output was made available to forecasters. Nearly all displays of model data were sent to offices over AFOS (Klein 1976) using a metafile format called Universal Transmission Format (UTF). UTF (EMWIN 1990) is a highly compact binary format with a small to moderate amount of complexity. UTF has some support for progressive disclosure, though the implementation is far from ideal. There is support for offset addressing, but one needs to apply knowledge of what is being plotted to know when to use which type of offset. All coordinates in UTF are simple integer coordinates, so the coordinate system is not self describing, and there is no support for frame coordinates.

In the AWIPS era, the vast majority of model output is sent to NWS offices over NOAAPORT as grids, but graphics metafiles are still used to transmit manually produced graphics. The metafile format currently used for this is the Red Book (OFCM 1994) format. At one level, the Red Book format is just a fancy wrapper around UTF. There are additional features in the Red Book specification that go beyond UTF, but the stream of graphics being sent over NOAAPORT for the most part only uses the UTF-like features. While only integer coordinates are supported, the Red Book format does contain some information describing the coordinate system, but this information is often difficult to interpret reliably. Other than that, the limitations of Red Book are similar to those of UTF, except that Red Book is not as simple or as compact as UTF.

Leading up to the creation of the AWIPS system, several AWIPS pre-prototype systems were developed. One of these was the second Denver AWIPS Requirements and Risk Reduction Evaluation (DARE-II) system. DARE-II (Bullock and Walts 1991) was developed by the Forecast Systems Laboratory (FSL), the immediate precursor organization to GSD. During the development of DARE-II, a homegrown graphics metafile format was defined called DARE Graphics Metafile (DGM). DGM filled a need for a highly compact, simple graphics metafile that had robust support for progressive disclosure and offset addressing. For the purpose of displaying graphic overlays, the main DARE-II display was a DGM rendering engine. In front of the DARE-II display was a set of product generation processes that would turn various data sets into viewable pictures encoded in DGM. DGM continues to be used in the AWIPS era. There are several displays in AWIPS created by decoding text on the fly in a separate process, and these use DGM as an intermediate for shipping finished graphical information to the main AWIPS display. Also, DGM is used to ship drawing

instructions for graphics overlays from the server to the client for the FX-Net (Wang and Madine 1998) system. DGM is limited in that it does not distinguish between different kinds of offset addressing and has less than optimal support for using latitude and longitude as global coordinates.

Besides these formats, GSD's precursor organizations have had experience in the past with several other graphics metafile formats. All these other formats are binary, and are either obsolete or do not support progressive disclosure and offset addressing. One thing common to all formats discussed so far is that they are binary. These highly compact binary formats were developed at a time when available disk space, communications bandwidth, and processing power were much less, and file compression technologies were less advanced, than they are now. Currently, graphics metafile technologies seem to be universally evolving toward ASCII formats, such as KML or Scalable Vector Graphics (SVG). Both KML and SVG (W3C 2003) are based on the Extensible Markup Language (XML), described in WC3 2000 and numerous other places.

As has been demonstrated, KML currently has no direct support for offset addressing, but KML can be coerced into behaving as if it does to some extent. This implementation is not ideal; using raster image capabilities to implement what are essentially vector graphics display items seems very much like forcing a square peg into a round hole. In a way, this is understandable, given that KML uses the same geometry model as the Geography Markup Language (GML). GML (OGC 2007) is meant to describe the geographic characterstics of real world objects, not to describe how to represent them visually (Lake 2005). As such, KML is in some ways still better suited for representing actual physical objects, like a building, rather than abstract visual objects, like a station model, an isopleth, or a cold front.

SVG, on the other hand, is meant to describe how to visualize things. However, unlike KML, it has no native support for using latitude and longitude as global coordinates. As such, if one has two SVG metafiles generated in different projections, it may be difficult to visually integrate (overlay) them and still retain the original character of the displays. Nor does SVG offer native support for progressive disclosure. Using non-standard extensions, one could of course implement these features in SVG, as one could do with some of the harder-to-emulate features in KML. However, doing things in this manner would negate many of the advantages of using an off-the-shelf standard.

5. SOME RESULTING IDEAS ABOUT GRAPHIC METAFILES IN FORECASTING APPLICATIONS

The preceding review of existing graphic metafile technologies leads to two conclusions that are the driving force behind why this paper was written. One, the current metafile format being used operationally by the NWS to distribute manually produced graphics to the forecast offices is obsolete and should be replaced. If Red Book truly supported all the desired features, one might decide to leave well enough alone, but this is not the case. Furthermore, the Red Book standard is difficult interpret; as such there is a reluctance to rewrite software that deals with it, even though much of that software is getting very old and difficult to maintain. Thus, Red Book should be replaced with a format that is consistent with the direction in which modern graphics metafile technologies are evolving. Two, no currently supported standard fully implements all the features that are desired in a graphics metafile format meant for forecasting applications. This second conclusion is somewhat more tenuous than the first, due to the fact that the preceding review of currently available formats was not comprehensive. There could be an existing graphics metafile standard that our research has yet to discover which has all the desired functionality. The best information we have says that nobody at the NWS or at Raytheon (the current lead AWIPS contractor) is addressing this issue at the present time.

Metafile technology seems to be universally moving towards XML-based formats. XML-based formats have the advantage that they are endian independent and to an extent human readable. Furthermore, it is easy to include ancillary information along with the raw display instructions, without having to update the logic of a display parser to understand the contents of that ancillary information. Thus, whatever replaces Red Book should probably be XML-based.

Replacing the Red Book format would involve not just the NWS but also the Office of the Federal Coordinator for Meteorology (OFCM) and perhaps other agencies. Should the OFCM (OFCM 2007) and its cooperating agencies go to the effort of replacing Red Book, functionality should not be sacrificed just so adherence to a standard can be claimed. The ideal scenario would be if an undiscovered yet fully functional standard existed that could be adopted. Alternatively, it may be possible to get the appropriate governance committee for some existing standard to accept the additional functionality needed by the NWS. Barring that, the NWS should work with the OFCM to develop a new format for communicating manually produced graphics to the forecast offices. If such a format were developed, it should be kept as simple as possible and still support the desired features; this would both ease the implementation and speed acceptance in the wider meteorological community. If an existing standard is adopted, the set of functionality used for graphics sent over NOAAPORT should still be kept as narrow as possible to ease the transition from Red Book to the new standard. One reason that simplicity is an issue is that NOAAPORT is not just internal to NWS or NOAA; it is accessible to a much wider community of users.

Whatever new standard is adopted or developed, the default implementation should always have a one-to-one correspondence between physical graphics metafiles and graphical overlays. It is not a bad thing to be able to implement a single graphic overlay distributed over several physical files if that is convenient for an individual implementation. However, it is undesirable if one is forced to always implement a single graphic overlay in a distributed fashion. If it were not for the fact that it is possible to bundle many files into a single .kmz file, the KML exporter discussed here would not be practical. Because the output from the KML exporter is internally segregated into a large number of files, this is still far from an ideal approach.

Finally, we comment on describing abstract meteorological objects in graphics metafiles. By this we mean, for example, supplying the location and type of a front, as opposed to supplying raw drawing instructions for the frontal surface, pips, and other features that are typically used to visualize a front. There are times when it is appropriate to include abstract meteorological objects in a graphics metafile. However, it is the wrong approach to force a client to interpret such objects when all they want to do is view a picture. This is problematic for two reasons. First, it greatly increases the complexity of any display interpreter for the metafile. Second, it relies on an assumption that the average client who wants to view the contents of the metafile understands how to visualize such objects just as well as the creator of the metafile. As mentioned, this does not mean that one should never include abstract meteorological objects in a graphics metafile. What this does mean is that one needs to redundantly specify such objects, supplying both the information that defines the object and the drawing commands needed to view the object. Thus, someone who cares about the abstract objects can parse and use them; someone who only wants to view the picture can just implement the simpler set of drawing commands. Fortunately, XML lends itself very well to support this approach for modeling abstract meteorological objects.

6. MINIMUM SET OF FUNCTIONALITY REQUIRED IN A GRAPHICS METAFILE STANDARD FOR FORECASTING APPLICATIONS

What follows is a bare minimum set of capabilities we believe are needed in a graphics metafile standard meant to support forecasting applications. Some of this is restating things previously discussed.

- Ability to define what the global coordinate system for the graphical overlay is. At a minimum, the available global coordinate systems should include latitude/longitude and arbitrary Cartesian. It would be reasonable to make latitude/longitude/height available as a global coordinate system, but it is probably not essential.
- Ability to specify whether the coordinates associated with drawing commands are in global coordinates, frame coordinates, unrotated offset coordinates, rotated offset coordinates, or hybrid offset coordinates.
- Ability to draw unlinked vectors.
- Ability to draw linked vectors.

- Ability to draw filled polygons. These filled polygons would not have borders; those would be drawn with a separate linked-vector operation if needed. The fill pattern should be implemented as a two-dimensional binary stipple of arbitrary size. Named fill patterns are not essential because they complicate creating an interpreter, but it would be reasonable to say that not specifying a pattern would default to solid. Pixels identified in the stipple as unactivated would have nothing happen to them unless an optional background color were supplied.
- Ability to attach interactive displayable items to a polygon. These items would display only when some sort of cursor interaction occurred within the area of the polygon.
- Ability to specify the offset origin. The location of the offset origin can be specified in either global or frame coordinates. When the offset origin is in frame coordinates or the global coordinates are arbitrary Cartesian, there is no functional difference between the different kinds of offsets.
- Ability to assign progressive disclosure thresholds to drawable items. A progressive disclosure parameter associated with drawing commands for a plan-view grapical overlay should have units of earth distance per pixel. This means larger values for the parameter make the associated drawing commands more visible (viewable at lower zoom factors). Furthermore, it should be possible to indicate whether the value of earth distance per pixel compared to a given threshold is based on the local map scale or the mean map scale for the entire display; this allows one to prevent issues like the missing contour in figure 2. For graphics that are not plan-view, the units of the progressive disclosure parameter will be different but should have the same sense; larger values associated with a displayable item make it more visible.
- Ability to associate an arbitrary 32 bit color (0-255 values for red, green, blue, and opacity) with any drawing command. Support for plain language colors is not essential and complicates creating an interpreter.
- Ability to choose line texture. A variable length texture pattern bitmap is the preferred implementation. Again, support for plain language textures is not essential and complicates creating an interpreter.

Note that no directly supported text functionality is part of this bare minimum set of requirements. If text functionality were made part of the format, then decisions would have to be made whether to specify stroke fonts or bitmapped fonts, whether to have certain default font sets or have all fonts be defined in the metafile, and how to allow for defining new fonts. Once made, these decisions would result in reduced flexibility. With this recommended very minimal set of functionality, graphics creators more comfortable with stroked fonts can draw their characters using offset coordinates, those more comfortable with bitmap fonts can implement them as filled polygons.

If a more varied feature set is desired, then a converter could always be written to take a metafile created based on a more complex format that implements things in this simpler format. This would be the recommended means for implementing things like text functionality, named colors, or named line textures.

7. CONCLUSIONS

A utility that exports AWIPS graphic overlays using KML as a graphics metafile format has been discussed. This effort, combined with GSD's previous experience with graphics metafile formats and a cursory review of some other metafile formats in common usage, has been a worthwhile learning experience. This has lead to some useful insights about the current state of metafile technologies for use in forecasting applications. These insights lead to the conclusion that the current metafile format the NWS uses to distribute manual graphics to the forecast offices, Red Book, is out of date and should be replaced. A proposal has been made for a minimum set of functionality needed in a replacement for the Red Book format.

REFERENCES

Bullock, C.S., and D.S. Walts, 1991: Considerations for the DARE II design. *Seventh Int. Conf. on Interactive Information and Processing Systems for Meteorology, Hydrology, and Oceanography,* New Orleans, LA, Amer. Meteor. Soc., 152-156.

Google, cited 2007: Google Earth main KML page [Available online at http://earth.google.com/apis/kml/.]

Grote, U.H., D. Davis, C. Bullock, and J. Tuell, 2005: An advanced Linux prototype of AWIPS. *21st Int. Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology,* San Diego, CA, Amer. Meteor. Soc., CD-ROM, 8.4.

Lake, R. cited 2005: GML and KML Syntax [Available online at http://geoweb.blog.com/313918/.]

Jones, D.R., C. Bullock, W. Carrigg, C. Dietz, M. dedJardin, E. Mandel, D. Rhine, J. Roe, and M.T. Young, 2003: AWIPS Build 5 in review. *19th Int. Conf. on Interactive Information Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology,* Long Beach, CA, Amer. Meteor. Soc., CD-ROM, 4.1.

Kahn, R., and V. Baer, 1992: Information stream project for AWIPS/NOAAPORT (ISPAN). *Eighth Int. Conf. on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology,* Atlanta, GA, Amer. Meteor. Soc.

Klein, W. H., 1976: The AFOS Program and Future Forecast Applications. *Mon. Wea. Rev.*, **104,** 1494-1504.

EMWIN, cited 1990: Appendix B, Universal Transmission Format [Available online at http://www.weather.gov/emwin/h5v6p2.zip.]

OFCM, cited 1994: Standard Formats for Weather Data Exchange Among Automated Weather Information Systems [Available online at http://www.ofcm.gov/s2/s2.htm.]

OFCM, cited 2007: Office of the Federal Coordinator for Meteorology home page [Available online at http://www.ofcm.gov/]

OGC, cited 2007: OpenGIS Geography Markup Language (GML) Encoding Standard [Available online at ttp://www.opengeospatial.org/standards/gml.]

Wang, N. and S. Madine, 1998: FX-Net: a Java-based internet client interface to the WFO-Advanced workstation. *14th Int. Conf. on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology,* Phoenix, AZ, Amer. Meteor. Soc., 427-429.

W3C, cited 2000: Extensible Markup Language (XML) 1.0 (Second Edition) [Available online at http://www.w3.org/TR/2000/REC-xml-20001006.]

W3C, cited 2003: Scalable Vector Graphics (SVG) 1.1 Specification [Available online at http://www.w3.org/TR/SVG/.]