TRANSCODING BETWEEN THE VIRTUAL DEVICE INTERFACE AND TELIDON STANDARDS

Hannah Newman
Dept. of Communications
Ottawa, Ontario

ABSTRACT

The relationship between the draft ANSI Virtual Device Interface standard and the North American Presentation Level Protocol Syntax is considered. Translations required between the two function sets are specified from which the degree of compatibility can be determined. The mappings preserve the usefulness of each function as much as possible although the precise semantics may be somewhat altered.

Étude des liens existants entre les fichiers standard ANSI exprimant les instructions graphiques de façon indépendante des systèmes graphiques utilisés et Télidon. A partie des transformations requises pour passer de chacun des ensembles de fonctions à l'autre, le niveau de compatibilité entre les deux est délimité. Autant que possible l'intégrité de chaque fonction est conservée lors du passage d'un ensemble de fonctions à l'autre même si la sémantique peut être modifiée.

KEYWORDS: Telidon, Virtual Device Interface, transcoding

1. Introduction

At CMCCS '81, a paper was presented by Newman entitled paper "The Relationship of Telidon and Computer Graphics Standards". It addressed, in general terms, how graphics metafiles relate to Telidon. This current work is a more detailed examination of the relationship between the Telidon and the draft ANSI Virtual Device Interface standards. We will show whether this relationship is sufficiently close to make compatability of the two standards feasible. Also, we consider how the ease of the translation process varies over the categories of functions. Throughout this investigation, we tried to find mappings which preserve the usefulness of a function as much as possible, rather than be concerned with the precise semantics.

Translation between the two function sets has been defined. Because many functions do not necessarily map one-to-one, two-tables have been prepared. The first shows, using a possible VDI function set, a VDI to Telidon mapping and the second illustrates the reverse mapping.

2. Background

Telidon is incorporated as part of a standard being developed by both ANSI* and CSA+. The standard is known as the North American Presentation Level Protocol Syntax (NAPLPS). The NAPLPS function set used in the mapping is from [2].

The ANSI X3H3 Technical Committee for Computer Graphics is developing standards for the Virtual Device Interface (VDI) and the Virtual Device Metafile (VDM). At the time of this writing, the VDM draft standard is almost completed and the VDI draft is getting underway. It is planned that the VDM and VDI will be closely related.

Although we are studying the relationship of VDI and NAPLPS, the actual element set used in the mappings is from the VDM standard [1]. There will be some VDI functionality that will not be explicitly included in these mappings. However, the goals of this paper can still be satisfied since the VDM and VDI function sets will have a large degree of commonality. For the most part, the VDM is representative of the primitives, attributes, text and control functions that will be found in the VDI standard.

3. The Translation Process

The mapping has been visualized as a specification for a hypothetical translation mechanism between the two standards. One

*American National Standards Institute
+Canadian Standards Association

issue in developing such a mechanism is how much internal state is required. This is an indicator of the degree of consonance between the models of the two standards. The internal state is required to keep track of functions of one interface which are specified in a different manner or in a different sequence than the other. For example, a model attribute of one interface may be represented as a functional parameter of the other. There are notes in the tables which show the states that need to be maintained in order to do the translation.

4. Conclusion

From the two mappings, we can conclude that there is at least one potentially useful relationship between the standards, specifically that NAPLPS serves as a lower level interface to the VDI.

Since the primitive and primitive attribute elements of the VDM would be expected to be those in the VDI standard, a mapping of the VDI primitive and primitive attribute elements to NAPLPS would be reasonable. It should be possible, maybe even not very difficult, to build a VDI implementation whose device-dependent interface would be a stream of NAPLPS codes. This VDI would treat the NAPLPS as a physical device. Any NAPLPS compatible terminal could be attached to such a VDI implementation. A useful example of such an implementation would be in a system used to interactively generate frames for a videotex database. The operator, interacting with a paint program, would send high level graphic commands down through a hierarchy of interfaces ending up in the VDI driver. The driver would translate the resultant image into NAPLPS codes for preview on a NAPLPS terminal and storage in the database.

5. Acknowledgements

Many thanks are due to Bruce Cohen, who made substantial contributions to the mappings.

6.References

[1] Draft Proposed American National Standard for the Virtual Device Metafile, X3H3/82-33R6.

[2] CSA, Preliminary Standard, T500-1982 Videotex/Teletext Presentation Level Protocol Syntax (North American PLPS).

VDM to NAPLPS mapping

In this mapping, the actual bit sequence is shown to as detailed a level as is feasible. The character period '.' in the bit's position within a data byte indicates that a function value, not related to what is being translated, needs to be specified in order to maintain the proper state of the NAPLPS. The highest order bit of the 7-bit data byte is not shown. It always equals 1.

Control Elements	
Begin Metafile	not a VDI function so no equivalent necessary
End Metafile	not a VDI function so no equivalent necessary
Begin Picture	not a VDI function so no equivalent necessary
End Picture	not a VDI function so no equivalent necessary
VDC Extent	no equivalent
Clip Rectangle	no equivalent
Clip Indicator	no equivalent
Precisions	DOMAIN, .xxxyy, logical pel (reset the logical pel size. xxx is the value of multi-length operands and yy is the value of single-length operands).
Index	Maps to single-length operand.
Enumerated	Maps to single-length operand.
Colour	Maps to multi-length operand.
Colour index	Maps to single-length operand.
Real Coordinates	No equivalent
Integer Coordinates	Maps to multi-length operand.
Single integer value	Maps to multi-length operand.
Single real value	No equivalent
Message	no equivalent, can be accommodated outside the NAPLPS.
Application Data	no equivalent, can be accommodated outside the NAPLPS
Character Set Index	The appropriate escape sequences from ISO 2022 are generated.

Primitive Elements	
Polyline	Sequence of [Set & Line(absolute)]
Polymarker (note 2)	TEXT, ..., 10...., (Set cursor style to crosshairs so character is centred within character box), Char Field dimension (set to match the indicated marker size), Sequence of [POINT SET (absolute, invisible), SI, marker char, SO], TEXT (reset char field dimensions). Text size, rotation, char path, inter-char spacing, inter-row spacing and move parameters values have to be maintained as part of internal state of the mapping.

Cell Array	FIELD with corners [(xmin,ymin), (xmax-xmin, ymax-ymin)], DOMAIN, ..., set logical pel size, INCR.POINT, DOMAIN, ..., reset logical pel size. The current logical pel size, single- and multi-value lengths and dimensionality must be maintained as part of internal state of the mapping.
Polygon (Note 1)	SET & POLYGON (outline) if interior style = off. SET & POLYGON (filled) if interior style ≠ off.
Circle (Note 1)	SET & ARC (outline) if interior style = off. SET & ARC (filled) if interior style ≠ off. Circle can be defined using SET & ARC by specifying the two end points of the diameter of the circle.
Arc (Note 1)	SET & ARC (outline)
Arc_close (Note 1)	SET & ARC (outline), SET & LINE (absolute) if mode is chord and interior style is off. SET & ARC (filled) if mode is chord and interior style is other than off. SET & ARC (outline), SET & POLYGON (outline) if mode is pie and interior style is off. SET & ARC (filled), SET & POLYGON (filled) if mode is pie and interior style ≠ off.
Text (note 2)	SET POINT (absolute, invisible), SI, text chars , SO if text precision = string. Sequence of [POINT SET (absolute, invisible), SI, single text char , SO] if text precision = character.

Attributes

Fill, Marker, Line and Text Colour	If Colour Specification = direct: SET COLOUR with operand = green, red, blue colour value. If Colour Specification = index: SELECT COLOUR with 1 colour index operand (colour mode 1).
Colour Table	The current colour must be maintained in the internal state, SELECT COLOUR with 1 operand (start table entry), sequence of [SET COLOUR with 1 operand], reset current colour.
Line Width (note 3)	DOMAIN, ..., set logical pel size. Precision for single- and multi-value operands and dimensionality must be maintained in the internal state.
Line style	TEXTURE, ...xx (xx corresponds to the line style. 00 - solid, 01 - dotted, 10 - dashed and 11 - dotted dashed). Texture pattern, highlight must be maintained as part of the internal state of the mapping.

Marker Type	no equivalent. See Polymarker.
Marker Size	no equivalent (tied to character dimensions)
Interior Style Perimeter visibility = off	No action is taken if interior style = off. TEXTURE, 000.0. (when interior style = solid). TEXTURE, xxx.0. (for hatch styles, the value of xxx is one of [001,010,100,101,110,111] depending on the value of Hatch Index). TEXTURE, xxx... (when interior style = pattern, the value of xxx is one of [100,101,110,111]). See Pattern Index for definition of patterns. Highlight, line texture must be maintained in the internal state.
Interior Style (Note 8) Perimeter visibility = on	Same as above except bit 2 = 1, i.e. xxx.1.
Hatch Index	The hatch style pointed to by hatch index is defined as: TEXTURE, 001... (hatch style = vertical). TEXTURE, 010... (hatch style = horizontal). TEXTURE, xxx... (All other hatch styles are defined by specifying one of the definable texture masks in TEXTURE where the value of xxx is one of [100, 101, 110, 111]). The states of highlight and line texture must be maintained internally.
Pattern Index (note 5)	The pattern texture pointed to by Pattern Index is defined as: TEXTURE, xxx... (All pattern styles are defined by specifying one of the definable texture masks in TEXTURE where the value of xxx is one of [100, 101, 110, 111]. The style is defined via DEF TEXTURE and the mask size of TEXTURE). The states of highlight and line texture must be maintained internally.
Character Height	TEXT, ..., ..., Char Field dimension. Text size, rotation, char path, inter-char spacing, cursor style, inter-row spacing and move parameters values have to be maintained as part of the internal state of the mapping.
Text Alignment (note 6)	No equivalent
Text Precision	Only string and character precision can be supported using TEXT. Stroke precision can be supported if the text has been converted to strokes.
Character Up Vector (note 7)	TEXT, ...xx, ... (where the value of xx corresponds to the 0°, 90°, 180° and 270°). Text size, char path, inter-char spacing, inter-row spacing, cursor style and move parameters values have to be maintained as part of the internal state of the mapping.
Character Expansion Factor	No equivalent

Character Spacing	TEXT, xx...., ... (where xx defines the inter-char spacing).
Font Index (note 4)	no equivalent
Character Path	TEXT, ..xx.., ... (where xx defines the charater path).

Escape Element	
Escape	no equivalent

Note 1: The coordinates in VDM are all absolute. The coordinates of these NAPLPS primitives are relative and suitable translation on these coordinates would have to occur in order to do the mapping.

Note 2: Shift-In (SI) and Shift-Out(SO) are used when the desired character is in the G0 (ASCII) set. When the character is in a set other than G0, the set might have to be first designated before invoking it. Marker characters not in the ASCII set may possibly be defined by a MACRO or DRCS sequence.

Note 3: The logical pel definition causes a line, arc, or boundary of a fillable graphic primitive to vary its width as its slope is changed. This may not be expected by a VDM generator.

Note 4: This function may be achieved by using the code extension techniques in ISO 2022.

Note 5: The DEF TEXTURE command allows only 1 bit deep pixels to be stored in a pattern in NAPLPS, as opposed to the multi-bit colour values in VDM. The display of a pattern in VDM is expected to be the colours to which the stored values currently map, but the NAPLPS will cause it to be in the current drawing colour.

Note 6: The inter-row spacing capabilitiy in text alignment can be accommodated. For all other capabilities of text alignment, there are no equivalents.

Note 7: Only the four orientations orthogonal to the display axes are supported by NAPLPS.

Note 8: The NAPLPS equivalent of perimeter visibility = 'on' in the VDM is having the perimeter drawn in with a solid black line which uses the current logical pel size to determine its width.

NAPLPS TO VDM mapping

Only the functions needed in the translation are specified. There will be an encoding of the VDM in the style of NAPLPS but it has not been finalized at the time of this writing.

Control Functions	
Reset	Begin Picture
Wait	no equivalent

Display Functions	
Point Set (absolute, invisible)	no equivalent
Point Set (absolute, visible)	POLYMARKER
Point Set (relative, invisible)	no equivalent
Point Set (relative visible)	POLYMARKER
Line (absolute) (Note 1)	can be emulated with Set & Line (absolute).
Line (relative) (Notes 1 & 2)	can be emulated with Set & Line (relative).
Set & Line (absolute)	POLYLINE
Set & Line (relative) (Note 2)	POLYLINE
Arc (outlined) (Notes 1 & 2)	can be emulated as Set & Arc (outlined).
Arc (filled) (Notes 1 & 2)	can be emulated as Set & Arc (filled).
Set & Arc (outlined) (Note 2)	ARC
Set & Arc (filled) (Note 2)	ARC_CLOSE with chord option
Rect (outlined) (Notes 1 & 2)	Current INTERIOR STYLE must be maintained, INTERIOR STYLE = off, POLYGON, reset INTERIOR STYLE.
Rect (filled) (Notes 1 & 2)	POLYGON
Set & Rect (outlined) (Note 2)	Current INTERIOR STYLE must be maintained, INTERIOR STYLE = off, POLYGON, reset INTERIOR STYLE.
Set & Rect (filled) (Note 2)	POLYGON
Poly (outlined) (Notes 1 & 2)	Current INTERIOR STYLE must be maintained, INTERIOR STYLE = off, POLYGON, reset INTERIOR STYLE.
Poly (filled) (Notes 1 & 2)	POLYGON

Set & Poly (outline) (Note 2)	Current INTERIOR STYLE is maintained, INTERIOR STYLE = off, POLYGON, reset INTERIOR STYLE.
Set & Poly (filled) (Note 1)	POLYGON
Field	For use with Incr.Point, specification is included with CELL ARRAY. For defining columnated text, there is no equivalent. The concept of input user areas is outside the scope of the VDM and so there is no equivalent for protect/unprotect.
Incr.Point	CELL ARRAY
Incr.Line (Note 3)	Sequence of POLYLINE.
Incr.Poly (Note 3)	POLYGON

Attribute Functions

Domain	
Single-value Length	Must map to COLOUR INDEX, INDEX, ENUMERATED types
Multi-value Length	Maps to REALs, INTEGERs, COLOUR
Dimensionality	DIMENSIONALITY
Logical Pel Size	LINE WIDTH
Blink	no equivalent
Select Colour	When colour mode 0 is defined (no operands include with Select Colour), set COLOUR SPECIFICTION = direct. When colour mode 1 is defined (1 operand included), set COLOUR SPECIFICATION = index and MARKER, LINE, TEXT and FILL COLOURs = index specified in the operand. When colour mode 2 is defined (2 operands), there is no VDM equivalent.
Set Colour	MARKER, LINE, TEXT and FILL COLOURs = colour value of Set Colour if in colour mode 0. COLOUR TABLE loaded with colour value of Set Colour when colour mode = 1.
Texture	
Line texture	LINE STYLE
Texture Pattern (Note 4)	INTERIOR STYLE
Highlight	INTERIOR STYLE with perimeter visibility = on.
Text	
Rotation	CHARACTER UP
Char Path	CHARACTER PATH
Inter-char spacing	INTER-CHARACTER SPACING
Move parameters	No equivalent
Cursor style	No equivalent
Inter-row spacing	TEXT ALIGNMENT with horizontal alignment = continuous and continuous horizontal alignment = inter-row spacing
Character field dimensions	CHARACTER HEIGHT

Text Characters (G sets)	
Primary Set	The character symbols can be generated but the VDM does not allow format effector characters in the text string
Supplementary Set	Once invoked, same as Primary
Mosaic Set	Once invoked, same as Primary

Macros, DRCS and Control Characters	
Macros	No direct equivalent
DRCS	No direct equivalent
C0 Set	No equivalent, however a number of them (principally the format effector characters) affect the position of the text cursor, and so their effect must be recorded in the internal state of the mapping
C1 Set	No equivalent, however most of the commands in this set involve the modification of attribute values, and their effect must be recorded in the internal state of the mapping.

Note 1: The VDM has no concept of current drawing position in the functional description. The NAPLPS functions which use it have no direct equivalent, although translation software can recompute the value of the starting coordinate for the VDM function.

Note 2: Relative addressing may be included in a given VDM encoding, but is not standardized in the functional specification. A mapping is possible between absolutely addressed VDM elements and relatively addressed NAPLPS functions if the translation software computes the absolute address for each element.

Note 3: The computation required for each coordinate in this mapping involves adding the required increment to the current positon, then computing the absolute address from the result, as required.

Note 4: Since the VDM only supports HATCH INDEX for the specification of hatch styles, the hatch texture patterns would have to be downloaded.