Graphite Requirements

Sharon Correll
Version 0.93

This document describes the requirements for the Graphite system.

The following terminology is used to describe the level of importance of various features:

- **Required** ("shall" or "may"): essential for a complete and working system; expected to be implemented in version 1.0 unless otherwise noted.
- **Strongly desired**: system should be designed so as to allow the feature if at all possible; might not be included in version 1.0.
- **Desired**: system should be designed so as to allow the feature if cost is in proportion to the benefit; fairly likely not to be included in version 1.0; future implementation may depend on demand.
- **Low priority**: should not drive the design of the system; almost certainly not included in version 1.0; future implementation depends on difficulty and demand.
- **Not required**: not expected to ever be implemented.

Comments in italics indicate the extent to which the requirements are supported by version 1.0.

1 End-user requirements

1.1 General script requirements

Scripts with the following problematic characteristics shall be implementable by the Graphite system:

1.1.1 Right-to-left order.

1.1.2 Sloping baselines.

1.1.2.1 This feature might not be fully supported version 1.0.

1.1.3 Signficant reordering, as in Indic scripts.

1.1.4 Internal bi-directionality.

1.1.4.1 Non-requirement: handling of extensive bi-directionality.

> Extensive or multi-level bi-directionality is only required between writing systems, and can be handled efficiently by the FieldWorks paragraph layout module. (See non-requirement regarding multiple writing systems below.)

1.1.5 Extensive use of contextual forms.

1.1.5.1 Independent editing of ligature components.

1.1.5.2 Start- and end-of-line contextual forms.

1.1.5.3 Contextualization across line-breaks.

1.1.6 Hanging and centered baselines, as well as Roman.

1.1.6.1 Support for mixed baselines provided for calling application.

1.1.6.2 Non-requirement: support of mixed baselines within a single writing system.

1.1.7 Stacked diacritics.

1.1.8 Kerning.

1.1.9 Vertical text layout.

1.1.9.1 This feature will not be supported in version 1.0.

1.1.10 Contextualization and relative positioning of glyphs of different styles within the same writing system is desirable, but may not be supported in version 1.0.

For example, placing a bold diacritic above a plain base character.

In version 1.0, contextualization is possble between glyphs having different color and underline styles, but not among differing levels of boldness or italicization.

1.1.11 DTP support

1.1.11.1 Full support for DTP-like applications will not be included in version 1.0.

1.1.11.2 Various mechanisms to support justification that will not be included in version 1.0 include: word/phrase spacing, intercharacter spacing, kashidas, glyph decomposition/ductility/stretching/replacement.

1.1.12 Non-requirement: handling of multiple writing systems (that is, inter-writing-system contextualization, positioning, baselines, line-breaking, etc.).

1.1.13 The following capabillities shall be provided by virtue of Graphite serving as the rendering engine for a FieldWorks application:

1.1.13.1 Multiple, drastically different writing systems to be used by a single encoding.

1.1.13.2 Multiple trial versions of similar orthographies.

1.2 Editing support

1.2.1 Graphite shall provide support for split cursors.

1.2.1.1 Graphite shall be able to enable or disable the rendering of split cursors as requested by the application.

Disabling the rendering of split cursors is not supported in version 1.0.

1.2.2 Graphite shall provide support for proper (visual) arrow key behavior.

1.2.3 Non-requirement: it is not required for Graphite to give the application a choice about the following details of how the cursor should be rendered:

1.2.3.1 whether to use a slanted cursor for italic text;

1.2.3.2 various forms of a split cursor.

1.3 Unicode

Graphite shall be Unicode compliant.

1.3.1 Graphite shall be designed so as to encourage the use of standard Unicode character definitions.

1.3.1.1 Specifically, it is not required to allow overriding of normative Unicode character properties (outside of the Private Use Area).

However, this is allowed in version 1.0.

1.3.2 Graphite shall support the use of Unicode's Private Use Area.

1.3.2.1 Graphite shall be able to make use of definitions provided for the Private Use Area present in the FieldWorks Encoding and Rendering sub-system.

1.3.3 Graphite shall support the use of very large character sets, up to the limit imposed by the size of TrueType fonts.

Note, however, that sets larger than about 3000 glyphs may have greater hardware requirements.

1.3.3.1 This feature might not be implemented as part of version 1.0.

However, we don't anticipate anything in the system precluding such use.

1.3.4 Graphite shall support the use of surrogate pairs.

We anticipate that this will be provided automatically by means of the pseudo-glyph and substitution mechanisms.

Note that the use of surrogate pairs will not provide a way to handle a number of glyphs larger than the limit imposed by the size of the font (currently 64K glyphs).

In version 1.0, the Graphite engine can handle surrogate pairs transparently, but the support has not been tested in the Graphite compiler.

1.3.5 Graphite shall support Unicode's embedded directionality markers.

1.3.5.1 This feature might not be implemented in version 1.0.

C.f. 1.1.4.1.

Embedded directionality markers are supported as of WorldPad 2.0.

1.3.6 Non-requirement: support for MCBS encodings and 8-bit encodings is not required.

We expect that the FieldWorks system will provide a means of converting data using MCBS and 8-bit encodings.

1.4 Font files

1.4.1 Graphite engines shall be able to make use of both commercial and SIL-developed fonts.

1.4.2 It is required that there be an easy way to install and uninstall a complete set of files necessary to support a given rendering implementation.

See also the section on Font files under Inter-system requirements.

1.5 Hardware

1.5.1 Graphite shall provide reasonably fast and efficient behavior for scripts that are commonly used in SIL on hardware that meets the requirements for the FieldWorks suite.

1.5.1.1 Such reasonable behavior is required for IPA.

1.5.1.2 Such behavior is strongly desirable for Greek.

1.5.1.3 Such behavior is desirable for Hebrew (without cantillation marks).

1.5.2 Graphite shall provide reasonably fast and efficient behavior for complex scripts on contemporary high-end hardware.

The remaining requirements apply to script engineers and application developers.

2 Transformation process requirements

The Graphite system shall provide a mechanism whereby text in an underlying encoding is transformed into the corresponding form suitable for visual display, is rendered on an output device, and (on an editable medium) support is provided for editing.

2.1 Unicode and glyph IDs

Graphite shall be able to access any glyph in the available font.

2.1.1 Glyphs shall be accessible by Unicode value, Postscript name, glyph ID, and/or codepoint-plus-codepage values.

2.1.2 Distinct Unicode values that are mapped to the same glyph ID by a font's *cmap* can be either treated as equivalent or distinct.

This alludes to pseudo-glyphs and the auto-pseudo option.

2.1.3 Unicode values that are not mapped to any glyph in the *cmap* can be used within the substitution process.

Note, however, that only glyphs in the font can actually be rendered.

2.2 Substitution

The system shall provide a means whereby underlying characters or sequences of them are replaced by glyphs suitable for visual display.

2.2.1 The substitution process may involve multiple passes.

2.2.2 It shall be possible to insert, delete, and reorder glyphs as well as make substitutions.

2.2.3 It shall be possible to specify and make use of correspondences between classes of glyphs.

2.2.4 It shall be possible to make substitutions, reorderings, insertions, or deletions based on context.

2.3 Positioning

The Graphite system shall provide a means whereby surface glyphs are positioned in specific ways relative to neighboring glyphs.

Note that neighboring glyphs may not be adjacent in the glyph stream.

2.3.1 The positioning process may involve multiple passes.

2.3.2 It shall be possible to position based on context.

2.3.3 Glyphs may be shifted horizontally or vertically without affecting the screen location of following glyphs in the input stream.

For instance, if the normal positioning of three characters is:

A B C

shifting B to the left will result in:

AB C

That is, the normal screen location of C is maintained, resulting in more space between B and C.

2.3.4 Glyphs may be kerned horizontally or vertically, with neighboring glyphs adjusted appropriately.

For instance, if the normal positioning of three characters is:

A B C

kerning B to the left will result in:

AB C

That is, C is moved toward the left as well, in order to maintain the normal spacing between B and C.

Vertical kerning involves progressive shifting of the baseline.

2.3.5 Glyphs may positioned relative to each other using attachment points.

2.3.5.1 Attachment points may be specificed in coordinates relative to the em-grid of the glyph.

2.3.5.2 It is strongly desirable that attachment points which are in proximity to actual glyph outline points be moved by an amount congruent with the movement of the outline point caused by hinting.

This is partially supported in version 1.0.

2.3.5.3 Attachment points may be specified in terms of points used to define the shape of the glyph in the font file.

2.3.5.3.1 Attachment points may be specified in terms of the number of a single-point path used to define the shape of the glyph in the font file.

2.3.6 The positioning process shall provide access to glyph metrics from the font file, including:

2.3.6.1 the glyph's left side bearing;

2.3.6.2 the glyph's advance width;

2.3.6.3 the glyph's bounding box coordinates;

2.3.6.4 the ascent of the font; and

2.3.6.5 the glyph's advance height and vertical side bearing (not included in version 1.0).

These will be needed to implement vertical text layout.

2.3.6.6 It is desirable that the units of em-square on which the metrics are based be arbitrarily scalable.

This allows the units to be specified in terms of permillages, or in terms of the units used in the font, or, if desired, in terms of some other arbitrary scale.

2.3.7 The positioning process shall provide access to metrics for positioned sequences of glyphs treated as a unit.

This feature has been implemented but not well tested in version 1.0.

2.3.8 It shall be possible to group two or more glyphs and position them as a single unit.

For instance, if you attach a wide diacritic to a narrow base character, the entire unit should be positioned with respect to the combined metrics of the two glyphs so that the diacritic does not collide with neighboring glyphs.

2.4 Line-breaking

2.4.1 Graphite shall be able to function as a line-break engine.

2.4.1.1 The line-break table may be multi-pass.

2.4.1.2 It shall be possible to specify appropriate line-break locations.

2.4.1.3 The following line-break weights are defined, as needed to support FieldWorks's requirements:

2.4.1.3.1 Word break: always permissible where wrapping is allowed;

2.4.1.3.2 Intra-word break: acceptable but less desirable;

2.4.1.3.3 Letter break, where hyphen is not permitted: needed for pathologically narrow columns; and

2.4.1.3.4 Clipped break: last resort when a single glyph will not fit in the available width.

2.4.2 It shall be possible to perform substitution with regard to line-breaks.

I.e., start- and end-of-line contextual forms.

2.4.2.1 It shall be possible to perform substitution with regard to the weights of line-breaks.

2.4.3 It shall be possible to substitute and reorder glyphs regardless of line-breaks in the vicinity of the reordering.

E.g., Indic reordering over an optional line-break.

2.5 Bi-directionality

2.5.1 The Graphite system shall be able to render writing systems with internal bi-directionality.

However, extensive ranges of bi-directional text may not be rendered efficiently.

2.5.2 It is required to be able to make substitutions based on either underlying or surface adjacencies.

2.5.2.1 Support of more than one of these approaches is low priority and may not be supported in version 1.0.

2.5.3 It is required to be able to handle multiple levels of bi-directionality within a single writing system.

2.5.3.1 This feature might not be supported in version 1.0.

In version 1.0, the maximum level of bidirectionality supported is left-to-right numbers within a right-to-left Arabic-like script.

2.6 Cursor placement

2.6.1 It shall be possible to specify correspondences between underlying characters and surface glyphs for any kind of substitution, reordering, insertion, deletion, or positioning.

2.6.1.1 The system shall provide consistent, although not necessarily ideal, default behavior for any of the above transformations.

2.6.1.2 It shall be possible for multiple, possibly non-adjacent, surface glyphs to correspond to a single underlying character.

2.6.2 Given a screen location within a range of rendered glyphs, Graphite shall be able to determine the corresponding position in the underlying character sequence, along with an affinity for the previous or following character.

> Specifically, if the screen location is near the boundary between two or more glyphs that correspond to non-adjacent characters in the underlying text, Graphite shall be able to determine a single, unambiguous underlying position.

2.6.3 Given a position in an underlying character sequence, Graphite shall be able to determine the appropriate physical location of the corresponding visual insertion point, and draw it.

2.6.3.1 Graphite shall be able to render two visual insertion points (one preceding, one following) per underlying character position, if necessary.

2.6.4 It shall be possible to prohibit an insertion point between two contiguous glyphs.

2.6.5 It shall be possible to allow insertion in the middle of a single glyph that represents multiple underlying characters (ligature).

2.6.6 It shall be possible to provide information to the calling application regarding the appropriate effect of the arrow keys.

2.6.6.1 It shall be possible to inform the calling application of the underlying position corresponding to the physical edges of the text.

> This would support Home and End key behavior.

2.6.7 Non-requirement: support for the Backspace and Delete keys is a function of the keyboarding component, and is not required for Graphite.

2.7 Behavioral alternatives (“features”)

The Graphite system shall provide a means whereby the transformation process may be parameterized by the calling application, resulting in alternative rendering behaviors.

2.7.1 These alternatives may be arbitrarily general to the point of defining an entirely different transformation process.

2.7.2 Information about the defined features for a given process shall be available to calling applications in the following forms:

2.7.2.1 A form suitable for storing in the encoded form of the textual data.

2.7.2.2 A form suitable for creating a user interface to edit data, including multilingual support.

3 Inter-system requirements

3.1 Calling application requirements

3.1.1 The Graphite system shall support the interface defined for FieldWorks Encoding and Rendering Service subsystem. Specifically, the calling application shall be able to supply at least the following information to the Graphite engine:

3.1.1.1 Indication of the start and end of the writing system (contextual) range;

3.1.1.2 Previous segment, if any;

3.1.1.3 Overall paragraph direction;

We might not need this, but if we do, the calling application supplies it.

3.1.1.4 Indication of how to find line-breaks (ie, in the form of a line-break engine);

The line-break engine may be Graphite itself.

3.1.1.5 Types of permissible line-break—preferred and worst-case;

3.1.1.6 Start- and end-of-line indicators;

3.1.1.7 Physical space available in which to render;

3.1.1.8 Styles imposed by outer view;

3.1.1.9 Whether to render a split cursor;

3.1.1.10 A device context on which to draw;

3.1.1.11 The current font;

3.1.1.12 Depth of direction embedding.

3.1.2 The text data supplied by the calling application shall support at least the interface defined for the FieldWorks Text Services subsystem.

3.1.3 Any text data making use of Graphite “features” mechanism shall be encoded according to the pertinent specifications.

3.1.4 Graphite shall be capable of rendering a range of text in a single writing system that is about a page in length with good performance.

3.2 Support for calling applications

3.2.1 Graphite shall be able to indicate whether it is possible to render according to a given rendering behavior using a given font.

3.2.1.1 Graphite shall be able to perform rendering for an appropriate given rendering behavior ID / font pair.

Rendering behavior IDs are not supported in version 1.0.

3.2.1.2 Graphite shall be able to determine all the rendering behaviors that a given font supports.

Rendering behavior IDs are not supported in version 1.0.

3.2.2 Graphite shall be able to to supply the following to the calling application:

3.2.2.1 details about a text segment’s metrics, including height, width, ascent, baseline ascent, and overhang;

3.2.2.2 information regarding the physical location of selections;

3.2.2.3 information that will enable relative positioning of mixed baselines; support for mixed baselines may not be included in version 1.0.

Specific handling of mixed baselines is not supported in version 1.0.

3.2.3 Graphite shall provide support for encoding Graphite features into textual data.

Specifically, the calling application should be able to give Graphite a set of features and values and receive back a magic value to store in the encoded text.

3.2.4 The Graphite engine shall report to the calling application information regarding errors that occurred during the rendering process.

3.2.5 Graphite shall provide support for multilingual user interfaces in the calling application.

3.2.6 Graphite shall be able to provide calling applications with documentation regarding the writing system implemented by a engine control file, such as version number, ?

3.3 Unicode

See section under General User Requirements.

3.4 Font files

3.4.1 Graphite engines shall be able to make use of both commercial and SIL-developed fonts.

Specifically, Graphite needs to be used in conjunction with font files that we are not allowed to edit, particularly due to copyright restrictions.

3.4.1.1 It is desirable that the engine control file format be compatible with the TrueType file format.

This would allow us to embed our tables inside of the font files themselves if that proves useful.

3.4.2 It is required that there be an easy way to install and uninstall a complete set of files necessary to support a given writing system implementation.

The set of files might include those necessary for keyboarding, e.g., Keyman files.

Graphite installation packaging is not supported in version 1.0.

3.4.2.1 It is required that there be a straightforward way to create such an installation package.

3.4.3 Graphite shall support the following standard Windows styling mechanisms: bold, italic, color, background color, baseline changes, and point size changes.

Baseline and point size changes are needed to support superscripting and subscripting.

3.4.3.1 Strike-through and underlining are desirable; shadow and outline are low priority.

The above features are not supported in version 1.0.

3.4.4 Support of substitution and positioning across such style boundaries is desirable but may not be supported in version 1.0.

This may prove to be more achievable using fonts that are carefully designed to work together, specifically where the glyph IDs are identical for corresponding glyphs and attachments points are designed in a parallel fashion.

3.4.5 Graphite shall be designed so as to facilitate reuse of a script description for a group of font files with relatively small differences among them.

For instance, by means of the #include directive.

3.4.6 Non-requirement: it is not required to be able to deal with writing systems where all glyphs for a particular style are not available in a single.

3.4.7 Graphite shall support use of TrueType fonts.

3.4.7.1 Non-requirement: Graphite is not required to support extensions to TrueType (such as GX and OpenType extensions), Postscript fonts, or raster fonts.

3.5 Operating system requirements

3.5.1 The requirements in this document are intended to apply to the following Windows operating systems at the indicated level of priority:

3.5.1.1 Windows 95: required.

3.5.1.2 Windows 98 and later: required.

3.5.1.3 Windows NT 5 and later: required.

3.5.1.4 Windows NT 4: desirable.

3.5.1.5 Windows CE: low priority.

3.5.1.6 Windows NT 3: not required.

3.5.1.7 Windows 3.1: not required.

3.5.2 It is desirable that Graphite be as portable as possible to other operating systems.

4 Miscellaneous requirements

4.1 Time, space, and hardware

See the end-user requirements section.

4.2 Version compatibility

4.2.1 It is desirable that successive versions of the Graphite compiler be able to compile earlier versions of transformation specifications.

4.2.2 It is desirable that successive versions of the Graphite engine be able to run earlier versions of compiled tables.

4.2.3 It is required that Graphite engines be able to recognized incompatible compiled tables, including those that were generated by later version of the compiler.

4.3 Error handling

4.3.1 The Graphite engine shall perform gracefully, giving the best output possible, in all *but* the following error conditions:

4.3.1.1 Invalid arguments were provided from the calling application.

4.3.1.2 The system is out of memory.

In other words, system behavior will degrade gracefully in the following situations: a glyph ID not found in the font file, one half of a surrogate pair, looping occurring in the transformation process, undefined attributes, invalid attribute values, etc.

The “best output possible” may be as degenerate as a series of boxes. In the case of a missing or obsolete font file, the system may try to render using some fall-back font. In the case of a corrupted engine control file, the system may perform a simple rendering without running the transformation process.

4.3.2 The Graphite engine shall report to the calling application information regarding errors that occurred during the rendering process.

5 Version History

0.1—19-Feb-1999: first draft by Sharon Correll.
0.2—14-Mar-1999: a few changes based on comments by John Thomson and Alan Ward.
0.21—14-Mar-1999: includes comments from Martin Hosken and Bob Hallissy.
0.22—19-Mar-1999: following teleconference on 18-Mar-99.
0.23—22-Mar-1999: a few more comments from John Thomson.
0.24—25-Mar-1999: changes after teleconference.
0.25—31-Mar-1999: inserted comments from AW, MH, JT, BH; changes from teleconference on 1-Apr-99.
0.26—8-April-1999: changes from teleconference.
0.27—14-April-1999: changes from teleconference on 13-April.
0.28—18-May-1999: changes from meeting.
0.9—26-May-1999: changes from meeting on 25-May.
0.91—21-Mar-2000: changed WinRend to Graphite
0.92—13-Feb-2002: added annotations for version 0.8 SH, updated year and version GL.
0.93—7-Sept-2004: added annotations for version 1.0.

6 File Name

Graphite_Req_Annot_1_0.doc