Graide Manual

for Graide v. 1.0

Graide is an integrated development environment for use in developing Graphite-enabled fonts. Graide stands for **GRA**phite **I**ntegrated **D**evelopment **E**nvironment.

Graide allows you to edit source code, build the font, manage a suite of test cases, and debug.

This manual assumes that you are familiar with Graphite, the GDL programming language, and concepts such as:

- TrueType fonts
- Passes and rules
- Glyph attributes and GDL's built-in glyph attributes
- Slots, slot attributes, and GDL's built-in slot attributes
- Substitution
- Positioning
- Attachment points
- Font features and languages
- #included files
- Graphite's collision avoidance algorithm

Getting started

To create a Graphite font, there are two main files that are needed:

- a GDL program
- a TrueType font

In addition, Graide requires a configuration file.

The first thing to do when you start Graide is to choose the **Project | New Project…** menu option. The program will ask you to either choose an existing configuration (.cfg) file or specify the name and location of a new one. Once you have does this, Graide brings up a dialog allowing you to indicate the main GDL source code file and the font to which you want to add Graphite support.

You will also be asked to specify the name of a tests file in which to store your suite of tests, which for a new project will very likely be a brand new file. A suite of tests is not absolutely required but is something you will normally want for a Graide project.

The –p option can be used to start Graide using an existing configuration file:

```
graide –p myfont.cfg
```

Techie note: The configuration file stores the two basic file names plus any other configuration options that you choose as you work with Graide. The configuration file is human-readable, so you can edit the file by hand if you so desire (but if you edit the file while Graide is running, it will be overwritten). See

the appendix below entitled “Graide configuration file format” for a complete description of the file options.

Configuring the project

The Configuration dialog provides a way to change the above files as well as set other project options. To open the dialog, click the Configure Project button (wrench icon) in the lower right-hand corner of the Graide window.

General Section

***Font File**:* indicates the TrueType font that will be used for both the input and output of the build process.

GDL File: the name of the main file GDL file that is passed to the grcompiler program. It may #include other files.

Tests File: the main file containing the suite of tests. This files uses an XML format. The Tests Tab provies a way to specify extra tests files for your test suite.

Default RTL: check this box if you would like your tests to be automatically set up to use a right-to-left directionality—for instance when working with a font for Arabic or Hebrew script.

Build Section

The Build section provides options for alternate ways of building the font.

Auto-generated GDL File: Specifying this file causes Graide to run a process to auto-generate a portion of the GDL code, using a tool called *makegdl*. The auto-generated file will `#include` the main GDL file that is specified in the **GDL File** control; then the auto-generated file is the one passed directly to the Graphite compiler. (Without the auto-generated file, the main GDL file is the one used as an argument when calling the compiler.)

There are several kinds of code that can be generated:

- Glyph definitions, based on the glyphs in the font
- Attachment point glyph attributes, using a attachment point database file in an XML format
- Attachment rules

The following options are enabled when an auto-generated GDL file is specified:

> ***Make-GDL Command:*** to auto-generate GDL, Graide uses a tool called *makegdl*. This control allows you to specify parameters to this tool or indicate an alternate tool.

Attachment Point Database: this XML file contains attachment point information from which glyph attributes are auto-generated.

AP Database is Read-Only: this option is currently unavailable; the AP database file is always read-only.

Attachment Point Positioning Pass: if set to something other than None, a set of rules will be output into the given position pass that perform attachment.

These options are available with or without using *makegdl*:

Compiler Executable: this indicates the executable file to be run to compile the font. On Windows, for instance, this would normally be a version of grcompiler.exe, but it could be a batch file or Linux shell script.

Ignore Warnings: this control lists which warnings should be ignored by the compiler. The default warnings to be ignored are 510 and 3521. To produce all the warnings, enter "none" in this control.

User Interface Section

Editor Font: this is the name of the font to use to display text in the File Edit tabs (upper right-hand section of the Graide window). It needs to be a properly installed font.

Editor Font Point Size: this is the point size of the font to use in the File Edit tabs.

Tab Stop Width: this indicates how a tab character will be displayed in the File Edit tabs. You may want to set this to the equivalent of 2 or 4 space characters in the selected font at the given point size, depending on your preference.

Font Glyph Pixel Size: this indicates the size of the font to use for displaying the glyphs in **Rendered Output Region** and in the **Font, Passes**, and **Rules Tabs**.

Attachment Glyph Pixel Size: this indicates the font size to use for glyphs in the **Attach Tab**.

Waterfall Sizes: this indices the set of point sizes to use for the **Waterfall** display (accessed by the button just above the Rendered Output Region).

Display Character Entities: this option causes test sequence characters with Unicode values above 127 (U+007F) to be converted to use the \uXXXX format.

Display Kerning Edges: this option produces a visual indication of the edges of adjacent glyphs when using the automatic collision avoidance mechanism for kerning. The edges are only visible within the **Collisions Tab** for a collision-fixing pass and only at the points in the pass where kerning is being performed.

Editing source code

The **Edit Pane**, the upper right-hand region of the window, contains a series of tabs, each of which can contain a source code file (or other text file) for editing. Files can be opened using the **File | Open File…** menu item or the **Open File** button in the upper right-hand corner of the window. The **Save Files** button saves all the open files, and the **File | Save File** menu item saves the file current being edited.

If there are too many tabs to be shown along the top of the Edit Pane, arrow buttons will be displayed in the upper right-hand corner to scroll among them.

The **Project | Find in Files…** menu option (also accessed via Shift-Ctrl-F) will allow you to find a string in all of the open files. The instances are shown in the **Find Tab** in the lower-right pane.

The User Interface section of the Configuration dialog (described above) includes controls to set the font, font size, and tab stop width for the File Edit tabs.

Building the font

Clicking the **Build button** (gear icon) in the upper right corner of the Graide window, or choosing the **File | Build** menu option, will run the build process. The default process saves any changes to open files, calls *grcompiler* with the target GDL file and font, and replaces the font with the newly-built version.

If other options have been specified in the Configuration dialog, such a using *makegdl* and/or an alternate compiler program, these will be taken into account during the build.

Errors and warnings resulting from compiling the GDL program will be shown in the **Errors tab** in the lower right-hand pane. If there are errors, the tab will be shown automatically. If there are only warnings, you will need to click on the tab to display the list of warnings.

Examining your font and glyphs

There are several tabs that are useful for examining the glyphs in your font.

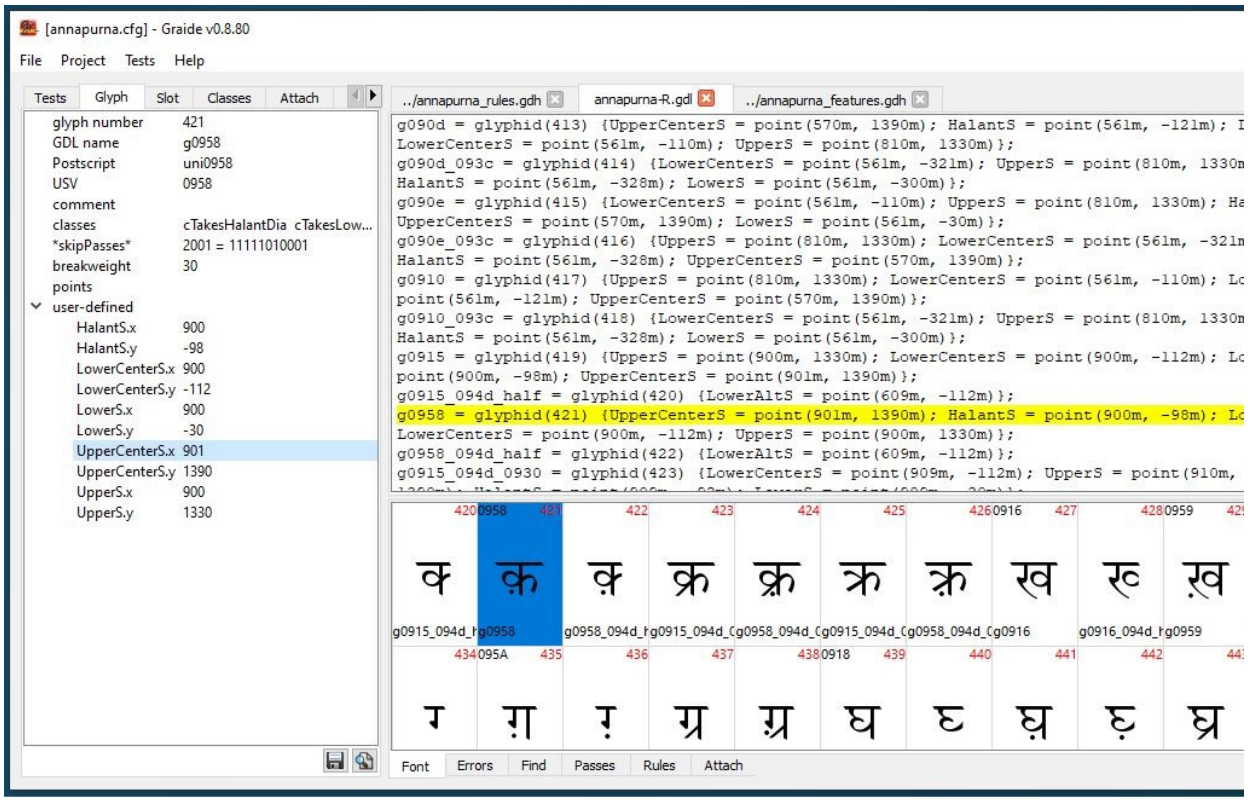
The **Font Tab** in the lower right pane of the screen shows all the glyphs in your font. The number in the upper left of a cell shows the USV (Unicode code point), the number in the upper right shows the glyph ID number, and the string at the bottom is the name of the glyph as defined in the GDL code.

The size of the glyphs in the Font Tab is controlled by the Font Glyph Pixel Size option in the User Interface section of the Configuration dialog.

Double-clicking on a glyph will bring up the **Glyph Tab** in the left-hand side of the Graide window. The tab shows various pieces of data associated with the glyph including all the defined glyph attributes.

In the Glyph Tab, double-clicking on *glyph number*, *GDL name*, *Postscript*, or *USV* will bring up the source code in the File Edit pane where the glyph is defined, with the line of code highlighted in yellow. Double-clicking on a glyph attribute will show the line of the code where the attribute is set. (Often this is the same as where the glyph is defined.) Double-clicking on the classes will bring up a list of classes in a separate scrollable dialog.

(Note: an infelicity in the program is that occasionally the File Edit pane will not accurately scroll the source code to the highlighted line. When that occurs simply double-click again.)



Glyphs Tab and Font Tab

If you have the name of a glyph selected in a File Edit tab, clicking the Show Glyph button (magnifying glass) at the bottom of the Glyph Tab will cause the information for that glyph to appear.

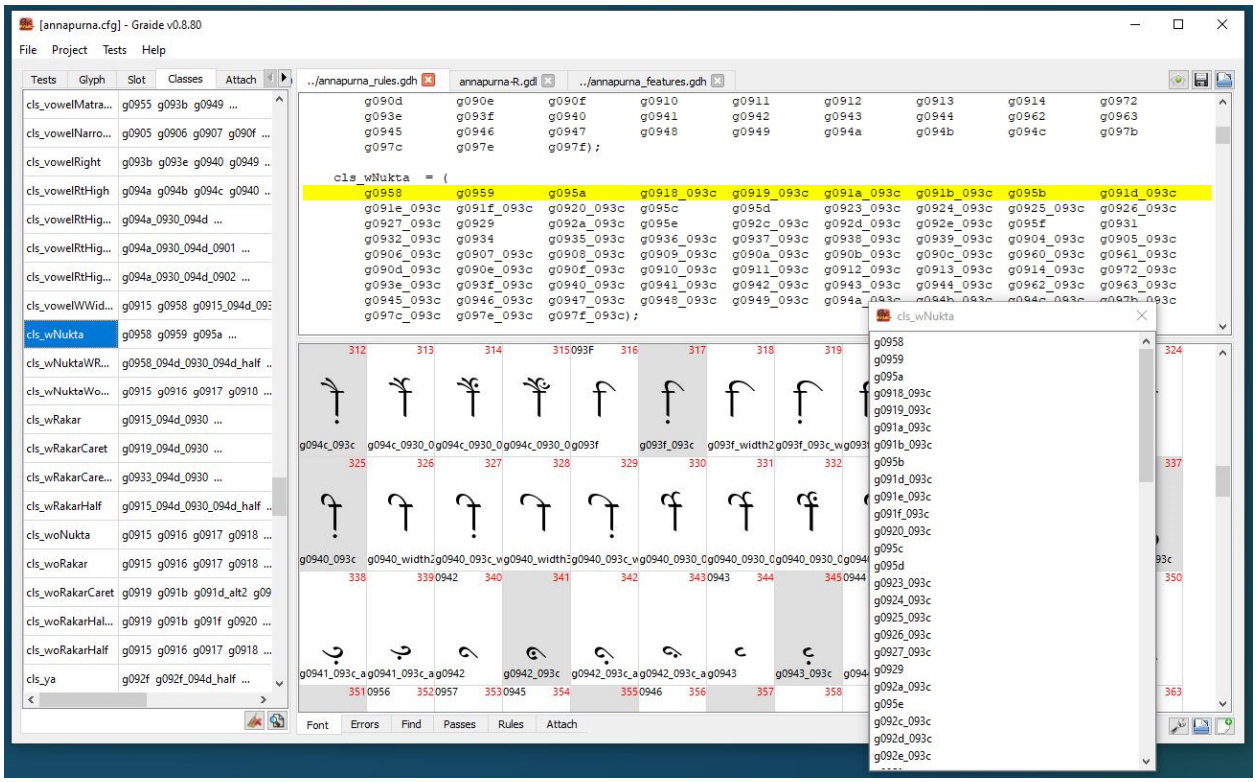
The “octaboxes” (polygons of three to eight sides) that are used to estimate the shape of the glyph for the purposes of collision avoidance are listed at the bottom of the Glyph Tab.

The **Classes Tab** in the left-hand pane shows all the glyph classes defined in the GDL code. Selecting a class will highlight (in gray) all the members of that class in the Font Tab.

Double-clicking the name of the class will show the line of code where the class is defined in the File Edit pane. Double-clicking the list of glyphs will bring up a scrollable list in a separate dialog.

At the bottom of the Classes Tab are two buttons. The left-most one clears the highlights in the Font Tab for the selected class.

The right-most button (magnifying glass) is a class-search operation. If the name of a class is selected in the File Edit pane, clicking that button will scroll the Classes Tab to that class name and highlight it. Double-clicking the button will also find the class definition in the source code and scroll to it.



Classes Tab and class members pop-up dialog

Running a single test

Even without setting up an entire suite of tests, it is possible to run a single test. The bottom two regions of the **Test Tab** are used for this. You can enter text in the **Test Data Control** (the next-to-bottom region), click the **Render button** (arrow), and see the result in the **Rendered Output Region** below.

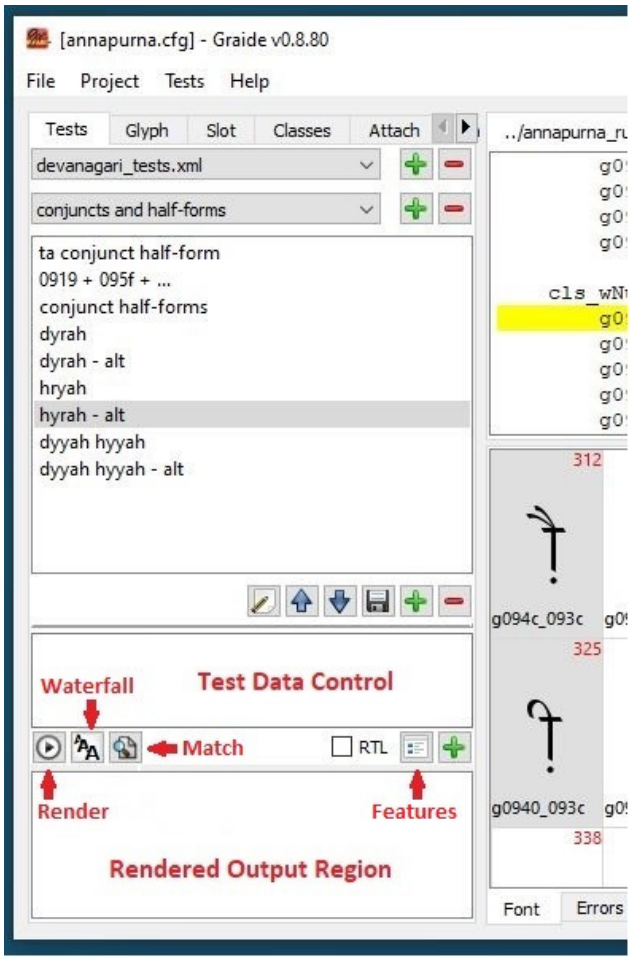
The button to the right of the Render button is the **Waterfall button** (AAA) which brings up the **Waterfall** display showing the result at different point sizes. The various sizes used for the Waterfall can be set in the Configuration dialog under User Interface.

The **Match button** (magnifying glass icon) can be used to easily search for test items that match the test string. The match mechanism is discussed in more detail below.

If you are working with a font supporting a right-to-left script such as Arabic or Hebrew, you will probably want to check the **RTL checkbox**. This option will be turned on automatically if you clicked the **Default RTL** option the Configuration's General section. (You may want to turn this option *off* when creating a test involving numbers in a right-to-left script where numbers are written left to right.)

To the right of the RTL checkbox is the **Features button**. This allows you to indicate which Graphite font features should be turned on for your test, as well as the expansion for justification—see "Setting features for a test" below.

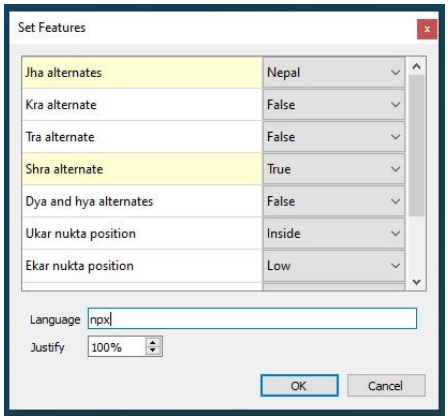
Double-clicking the Rendered Output Region outputs a graphic image to your computer. The image is saved in the current working directory in a file named something like *graide_image_1.png*.



Rendering controls

In the rendered output, you can click on a glyph and see the detailed information for it in the **Glyph Tab**, described below under "Examining your fonts and glyphs." (Note that you need to carefully click on the black "ink" of the glyph; clicking on the white area around the glyph will have no effect.) The bounding area of the selected glyph is shown in blue.

Setting features for a test



Set Features dialog

The **Set Features dialog** shows a list of the Graphite font features with the corresponding setting on the right. The drop-down control for the feature will allow you to choose a different setting for that feature.

Whenever a feature is set to something other than the default value, the feature name will shown on a yellow background.

If you enter a three-character language code for which your font has features associated, the features will be set to the appropriate values for that language. After doing so, you can then override feature settings using the drop-down controls, which will be indicated by the yellow background.

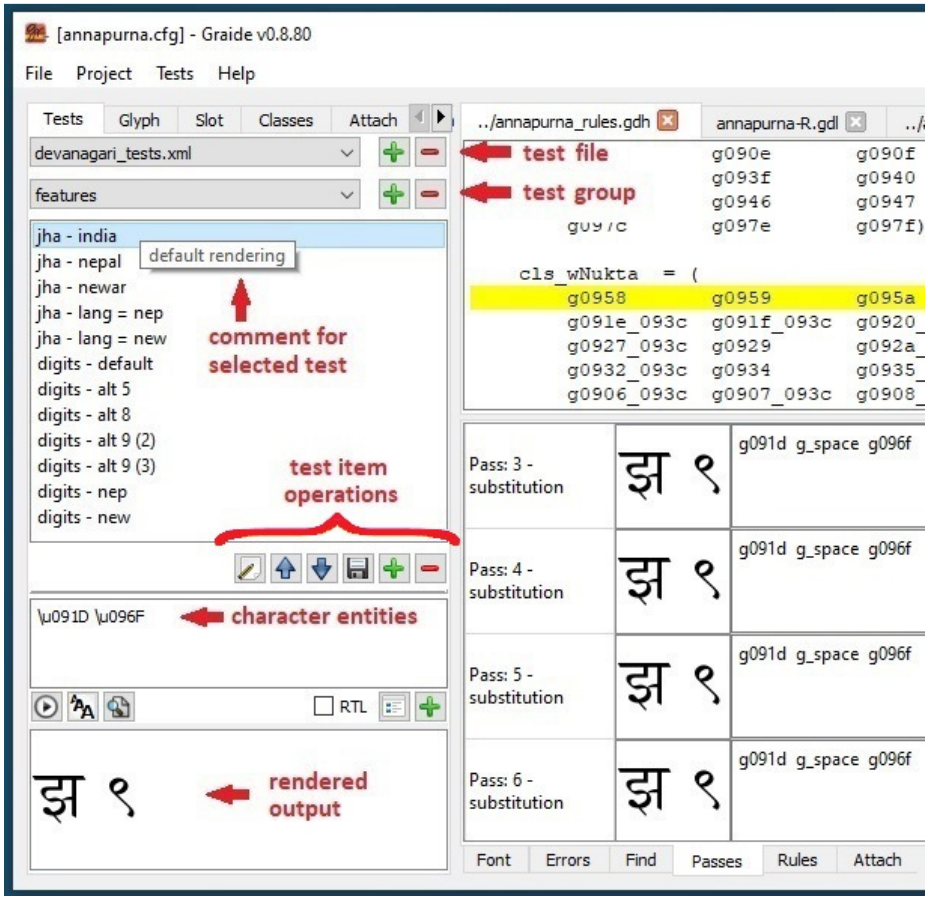
If you have no features defined for your font, the list of features will be empty.

The **Justify control** indicates the percentage by which the text should be expanded or contracted, as would occur to achieve justification. It has an effect only if there are justification passes in your font; it cannot be used for a general “tracking” or text compression effect.

Setting up a test suite

The **Tests Tab** in the left pane of the Graide window allows you to create a suite of tests and see the Graphite rendering for each.

Each Graide project normally includes one test-suite file, but you can have more. The top-most control in the Tests Tab allows selecting a tests file and + and - buttons for adding or deleting them.



Test list controls

(There is currently no user interface mechanism provided to reorder test-suite files, but this can be done by editing the configuration file directly. The test files are listed in the `[data]` section. However, the main test-suite file indicated by the testsfile variable in the `[main]` section of the configuration file is always listed first.)

Each test file can be organized into multiple groups of tests. The file is initialized with a group called "main." Groups can be added and deleted using the + and - buttons to the right of the group control. (Again, there is currently no mechanism for reordering the groups. This can be done by editing the tests XML file directly.)

The region below the group control is the **Test Item List**, and the buttons below the list are available for editing, reordering, saving, adding and deleting them.

Creating a test item

Create a new test item by clicking the Add Test button (green plus sign) under the test list.

Each test item has a name that is displayed in the Test Item List, the text that will be rendered as part of the test, and a comment. The comment will be showed when the mouse hovers over the name in the Test Item List.

The text of the test item can include all kinds of Unicode characters, but it may be convenient to use the \uXXXX syntax if there is any concern about the characters being displayed legibly. Checking the **Display Character Entities** option (in the Configuration tool under User Interface) will automatically convert any Unicode characters above U+007F to use the \uXXXX format.

Test items can be marked with a color which will shown as the background in the Test Item List.

A test item can have features, a language, and a justification factor applied to it. The **Features button** brings up a dialog identical to the one described above in "Setting features for a test."

Rendering a test item

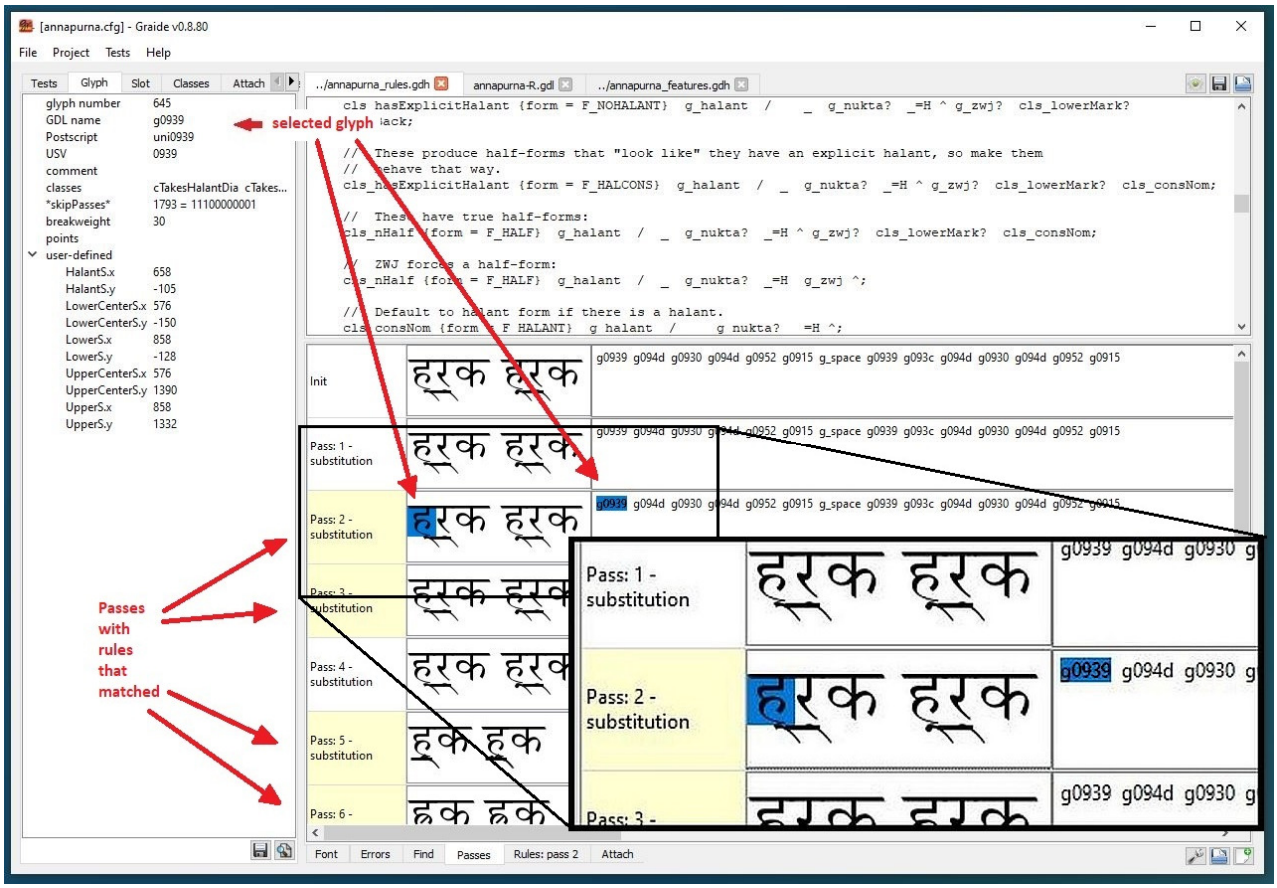
A test item is run simply by clicking on it. You can also use the arrow keys to move up and down the list, rendering each item in turn.

Debugging

Two tabs in the lower-right pane are useful for debugging your GDL code.

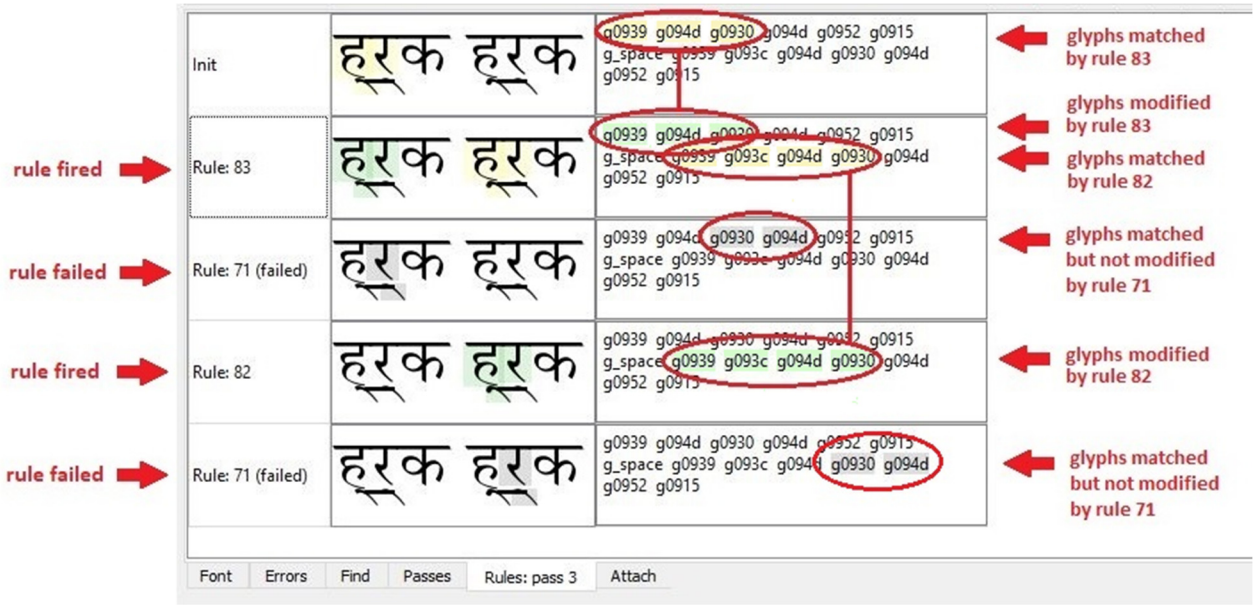
When you run a test (either by clicking a test item or by entering the test data directly in the controls), the **Passes Tab** is populated with the Graphite passes that were run during the rendering process. A yellow highlight in the left-hand column indicates passes in which rules were matched (whether they fired or not). A beige highlight indicates collision-fixing passes that made adjustments.

The center column shows the result of the rendering after that pass, and the right-hand column shows a list of glyphs that are in the output of the pass. Clicking on a glyph will highlight the glyph name and vice versa.



Passes tab

Double-clicking on a pass will cause the pane to switch to the **Rules Tab**, with a list of rules that were matched during the pass. The label of the tab indicates which pass was selected. Double-clicking on a rule will cause the source code for that rule to be displayed and highlighted in the File Edit pane.



Rules Tab

Glyph names in the Rules Tab are colored to indicate how they were affected by rules. Green-highlighted glyphs were changed by the given rule, and the yellow highlight shows the glyph in the input to the rule, which is either the initial pass input or the output of a previous rule. (Normally it will be the immediately previous rule, but it may be an earlier rule if some intervening rules did not fire.) It is possible for a glyph to be changed by subsequent rules, which means it will be highlighted in yellow-green. A gray highlight indicates that the rule was matched but the constraint failed and so the rule was not fired.

Clicking on a glyph image or glyph name will update the Glyph Tab and Slot Tab with information for that glyph.

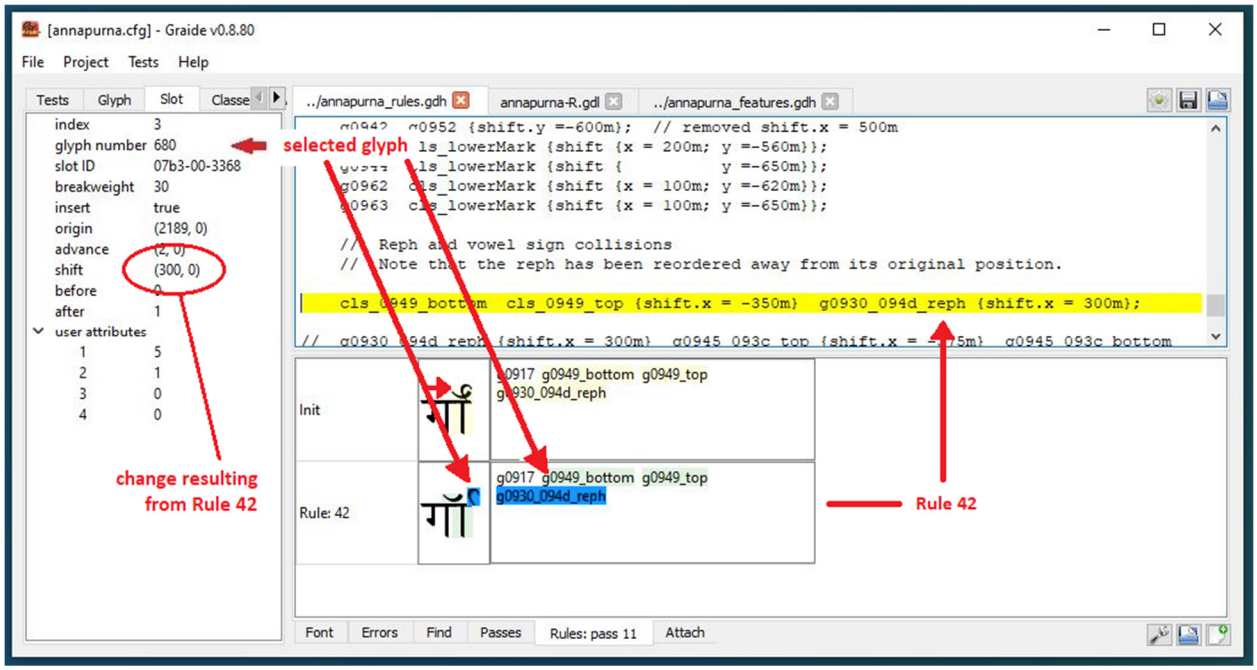
Double-clicking on a rule will show the rule in the source code highlighted in yellow.

The **Slot Tab** in the left-hand column shows various bits of data that are used during the rendering process. The *index* value indicates the (zero-based) index of the slot in the glyph stream, and *glyph number* show the glyph ID of the glyph in the font.

The *slot ID* is an arbitrary string that can be used to keep track of the relationships among glyphs. When an attachment is performed, the *parent slot* shows the slot ID of the slot to which the glyph is attached, and *parent offset* shows the offset of the attached glyph relative to the parent.

The *origin* shows the position of the glyph related to the left-hand origin of the string.

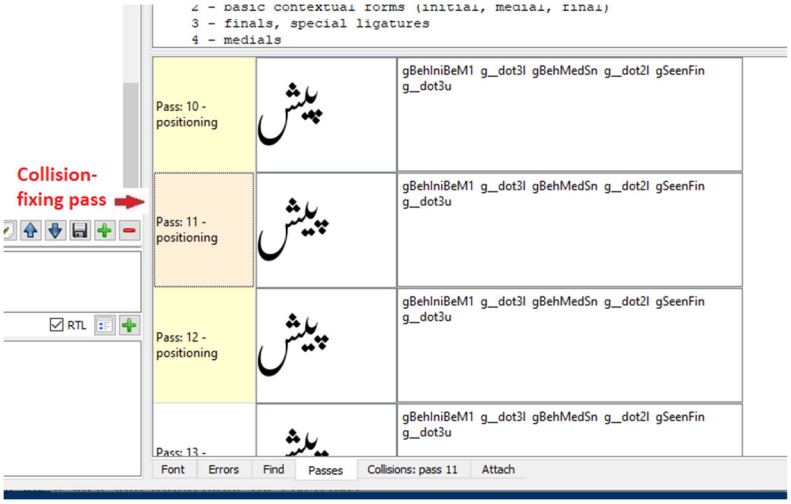
The other items in the Slots Tab correspond directly to GDL slot attributes.



Slot Tab

Collision avoidance

When the pass in question is a collision-fixing pass, it is highlighted in beige in the Passes Tab.



Collision-fixing pass in the Passes Tab

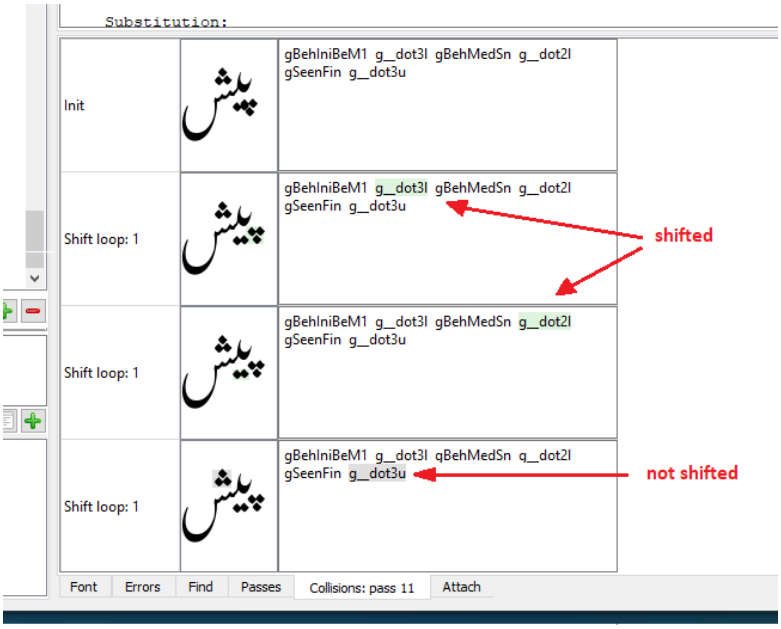
Double-clicking on it the Rules Tab create a **Collisions Tab**. Instead of, or in addition to, rules, the tab is populated with the effects of the steps in the collision avoidance algorithm. (Unlike rules, double-clicking on these rows in the tab has no effect.)

For debugging purposes, it can be convenient to separate collision-avoidance passes from those that set rules; this is what is shown in the images.

The octaboxes that are used to estimate the shape of the glyph for the purposes of collision avoidance are listed at the bottom of the Glyph Tab.

Shifting

In the Collisions Tab, glyphs that have been shifted are highlighted in green, and glyphs that did not need to be moved are marked in gray. Glyphs that remain in a state of collision due to a lack of acceptable adjustment are marked yellow.

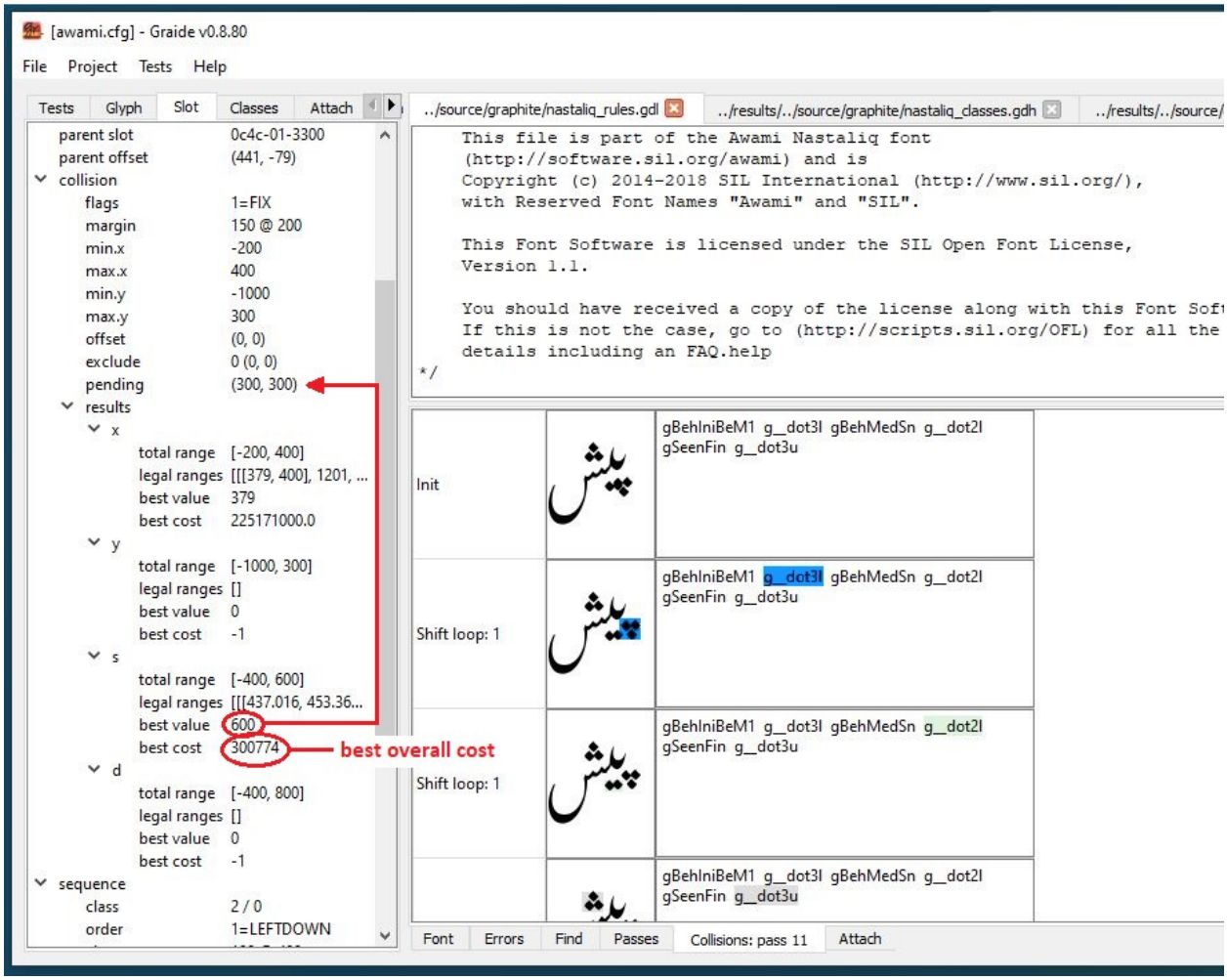


Collisions Pass

Clicking on the highlighted glyph will, as expected, display the slot attributes in the Slot Tab. In addition to the slot attributes, the collision section shows the results of testing the target glyph for collision along four axes, where x = horizontal, y = vertical, s (sum) = diagonal with positive slope, and d (difference) = diagonal with negative slope:

- *total range* – the possible range in which the glyph could be shifted
- *legal ranges* – the possible shifts and associated costs (squared cost, multiplied cost, and fixed cost for the zone)
- *best value* – the best (lowest cost) shift found along the given axis
- *best cost* – the cost of the best shift found along the given axis; -1 means no shift is possible

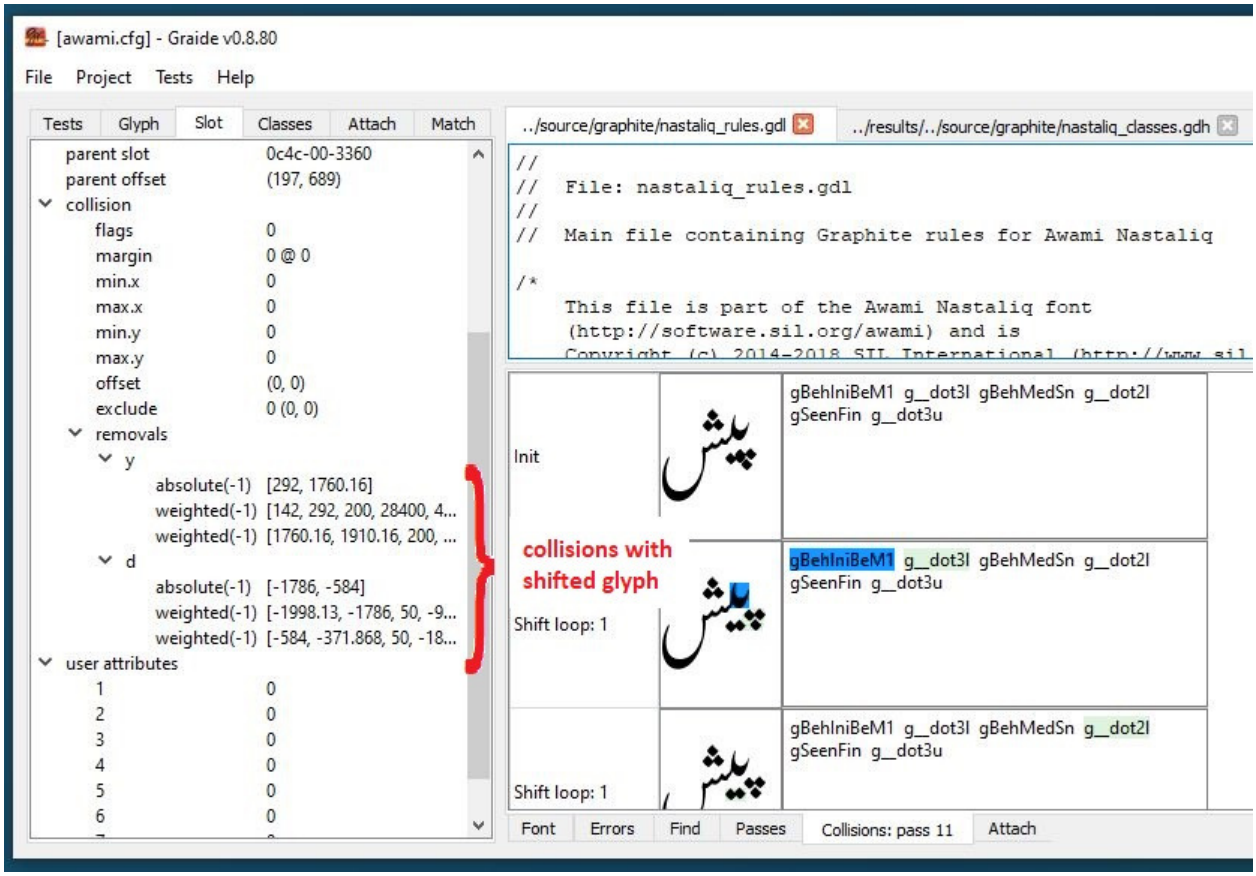
The *pending* value should reflect the lowest-cost value from all four axes. For diagonal axes, the best value is divided by two, and half is assigned to the each of the x and y directions.



Collision-fixing pass with shifted glyph selected

Clicking on a neighboring glyph (one that is *not* highlighted) will display information about collisions between it and the target glyph, if any. These are shown under “removals,” which means that they are ranges of movement prohibited to the target glyph due to possible collision with the neighboring glyph. A removal labeled “absolute” means it involves a true collision, while one labeled “weighted” means

that it would involve either an incursion into a margin that we want to preserve or a violation of the sequencing rules that have been defined. Consult the engineer for more details. The index displayed (e.g., “weighted(3)”) indicates which sub-octabox contains the conflict, with -1 indicating the full octabox.



Collision-fixing pass with neighboring glyph selected

If the `showextraattrs` option is set in the configuration file under the `[ui]` section, extra detail will be included in these values. Contact the engineering team for more information.

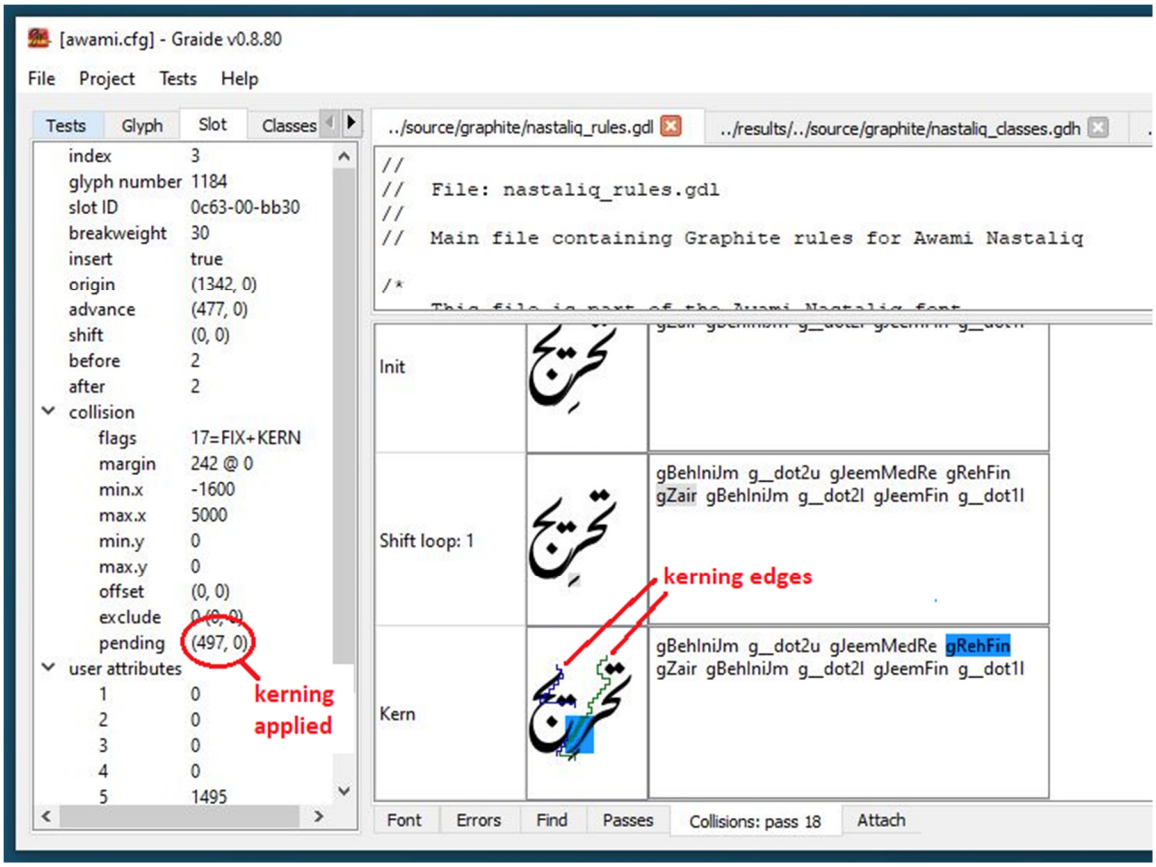
If the `collision.exclude` attribute is set, the exclusion glyph is displayed in the rendered image.

Kerning

Kerning is performed as the last stage of the collision avoidance algorithm, so the kerning adjustments (if any) are shown at the bottom of the Collision Tab.

If the Display Kerning Edges option is checked (Configuration dialog, User Interface), the Collisions Tab will draw what the Graphite engine considers to be the edges, including the margin, of the glyphs that are being examined for kerning.

The *pending* value shows the amount of kerning that will be applied.



Collision-fixing pass showing kerning

Searching for glyph patterns

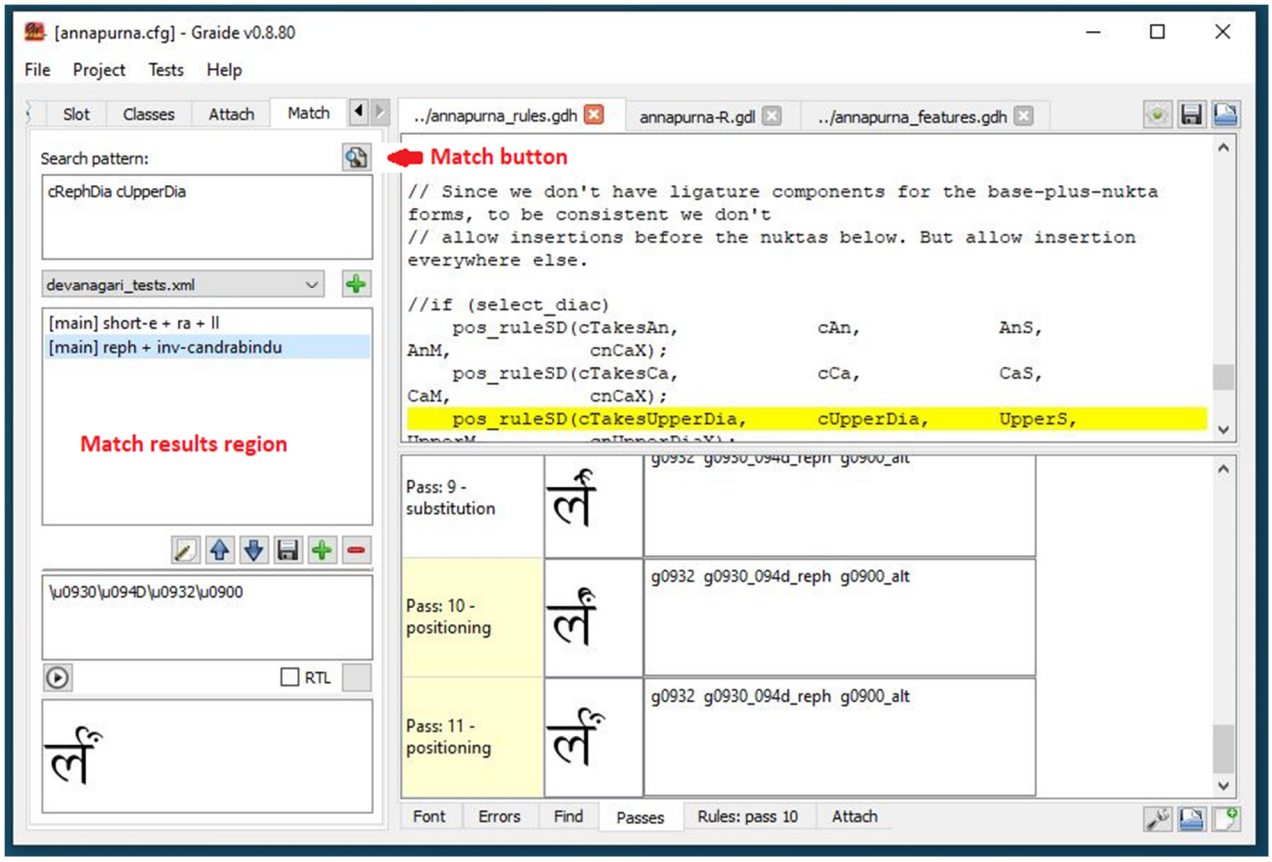
The **Match Tab** can be used to find examples of glyph sequences in your test suite. You can type a sequence of glyph names or classes in the Search Pattern region and click the Search button (magnifying glass) and test items whose final rendering includes the glyph or sequence will be shown in the Test Item List area below.

Note that this search is targeted at the glyphs in the final rendered result, not the original imput. Any reordering that happens during the rendering process can affect the matches.

You can output just the results of a group of these test items to a file using the Save button.

The ANY class is built into Graphite, and you can use it as a wildcard to match any glyph. Currently there is no mechanism to handle optional items in the match pattern.

Note: there is currently a bug such that if you have added a new test to the test suite, the match operation will not find it until you leave the program and reenter.



Match Tab

Setting attachment points visually

The visual attachment mechanism is not completely functional at this time. Here we document the aspects that are working well enough to be useful.

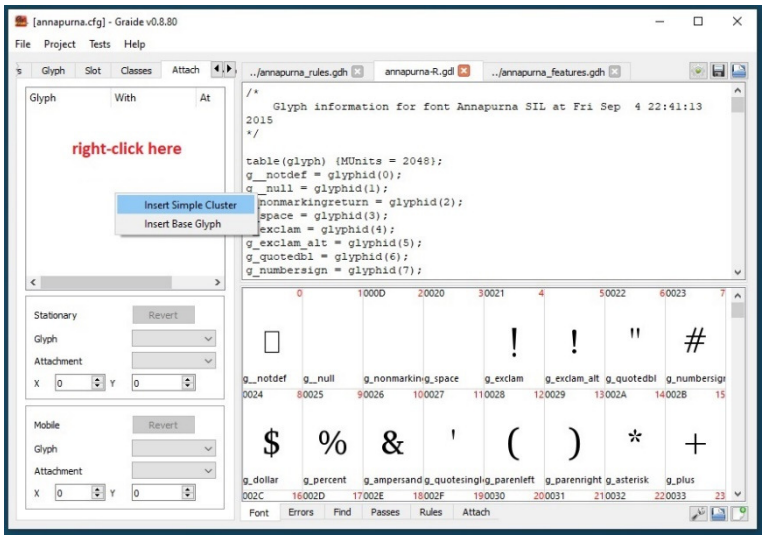
The visual attachment mechanism uses the two **Attach Tabs** in the left-hand pane and the lower right-hand pane. It currently handles pairs of glyphs for which attachment points have been defined, allowing you to see the visual relationship of the attached glyphs and experiment with making adjustments.

Pairs of corresponding attachment points are considered to be those that are identical with an “S” (stationary) or “M” (mobile) appended, for instance:

- `upperS` and `upperM`
- `nuktaS` and `nuktaM`
- `diacS` and `diacM`

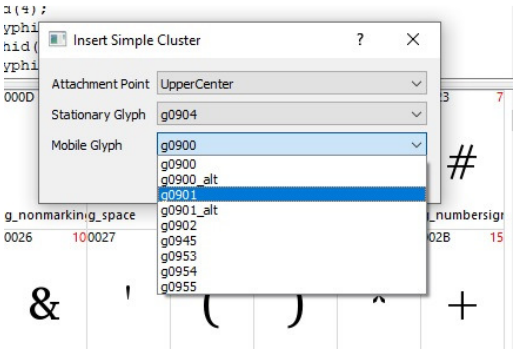
Groups of glyphs that have these kinds of corresponding attachment points defined can be used by this mechanism.

Right-click on the empty area in the left-hand Attach Tab and choose **Insert Simple Cluster**.



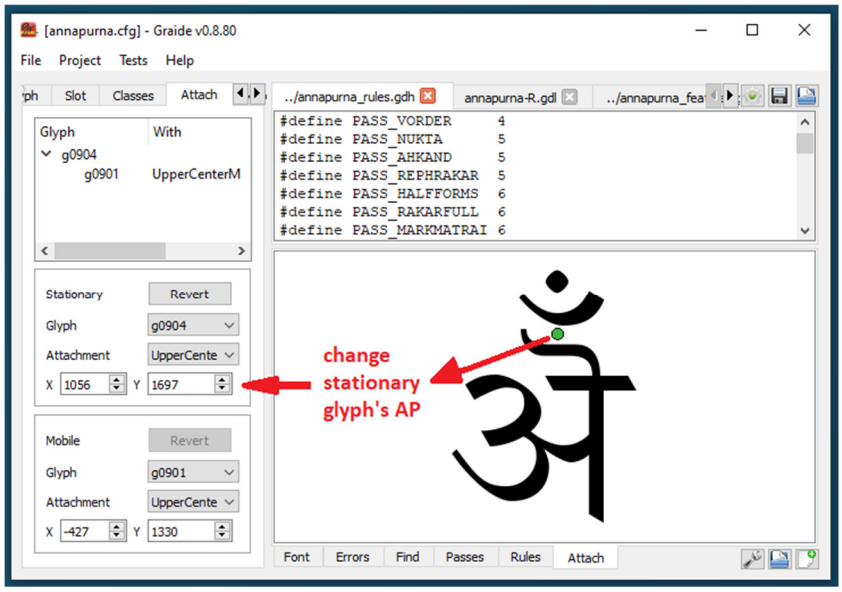
Attach Tab – creating a simple cluster

Then choose an attachment point pair you would like to work with. The lower drop-down controls become populated with glyphs that use that attachment point.

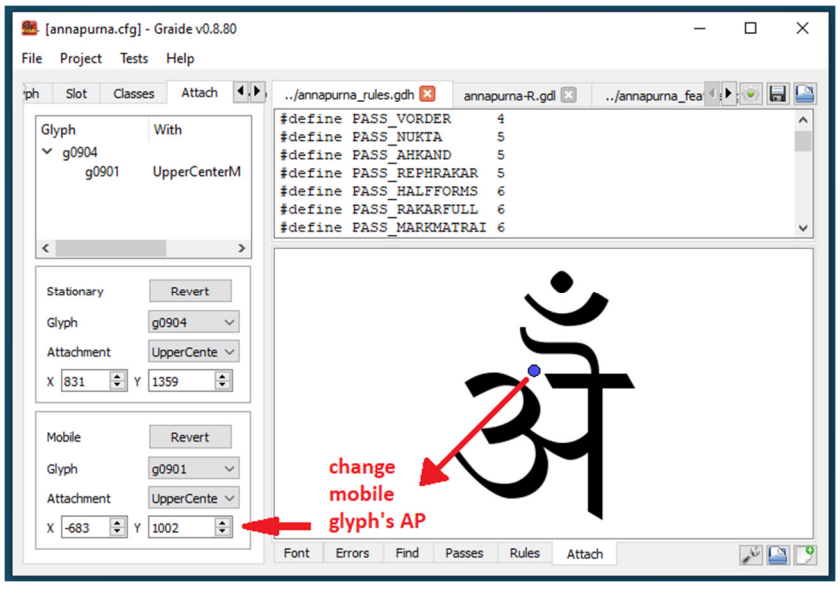


Clicking OK brings up the selected glyphs in the lower Attach Tab. The attachment point values are shown in the associated controls.

Dragging the “mobile” glyph (e.g., the diacritic) will adjust the attachment points for the “stationary” glyph (base). This is indicated by the green dot.



Holding down the Shift key while dragging will cause the mobile attachment point to change, while the stationary one stays fixed. This is indicated by the blue dot. You can also manipulate the spin controls to change the values.



Currently there is no way to automatically update the source code to match your changes. You will need to copy the values shown in the controls into your source code.

Appendix: Graide configuration file format

The Graide configuration file controls the various options in Graide. Although it is read and manipulated by the program, it uses the INI format which very human readable and thus can be edited by the user.

The INI format organizes options in groups. In Graide the sections and options are listed below.

[main]

ap – XML files storing attachment point data for the glyphs. Optional.

defaultrtl – 1 = any new tests created will be assumed to the right-to-left. Default = 0.

font – the .ttf file which is used for rendering and which will be modified by build commands. Required; no default.

testsfile – an .xml file where the main suite of tests will be stored. It is possible to have more than one file of tests, in which case they are listed in *testfiles*. Required; no default.

[build]

apronly – indicates that the AP file is read-only, i.e., that modifying the attachment points in Graide will not overwrite the AP file. Currently this is the only permissible setting.

attpass – the number of the positioning pass into which auto-generated attachment rules will be placed, if any. If omitted, no attachment rules will be generated.

gdlfile – the main GDL file containing source code; it may include other files using the #include mechanism. Required; no default.

grcexecutable – the file name of the Graphite compiler executable file. If omitted, Graide will file the program in the current directory or on the path.

ignorewarnings – a list of comma-separated numbers which indicate the warnings that should not be reported by the Graphite compiler.

makegdlcmd – batch file or shell script containing instructions on how to auto-generate a GDL file that contains basic glyph information.

makegdlfile – name of GDL code file, generally with a .gdl extension, to be ouput by the *makegdlcmd* script. Required if *usemakegdl* = 1.

tweakconstraint – GDL code snippet that is inserted into an `if (…)` statement as the constraint of rules generated by a tweak item.

tweakgdlfile – auto-generated XML files where GDL rules corresponding to tweak items are to be placed.

tweakpass – the number of the positioning pass where auto-generated tweak rules will be placed, if any.

tweakxmlfile – file of where tweak items are stored.

usemakegdl – 1 = when building, use a makegdl script; 0 = simply run the Graphite compiler.

[ui]

advanced – 1 = display additional technical information. Currently this option adds details about the collision avoidance process in the Slot Tab.

attglyphsize – pixel size of glyphs in the lower right Attach Tab.

editorfont – name of the font to use in the Edit Pane. This must be an installed font.

entities – 1 = automatically transform character data above U+00FF to use the syntax \uXXXX.

kernedges – 1 = display the edges of the glyphs that are being considered in the kerning steps of collision avoidance passes.

size – pixel size of glyphs in the font window and the results, pass, and rule panes. Default = 40.

tabstop – the number of pixels the Edit Pane should use for a tab character. A good choice is something like the width of 4 characters in the font specified by *editorfont* and *textsize*. Default = 40.

textsize – the point size of the font to use in the Edit Pane. Default = 10.

tweakglyphsize – pixel size of glyphs in the Tweak Tab.

waterfall – a list of comma-separated numbers indicating the point sizes to show in the waterful dialog.

[window]

mainwidth – width of the Graide window when it was last closed.

mainheight – width of the Graide window when it was last closed.

hsplitter – positon of the splitter control bar between the edit pane and the lower right pane, when Graide was last closed.

vsplitter – position of the splitter control bar to the right of the left-hand vertical pane, when Graide was last closed

openfiles – list of files that are shown in the File Edit tabs.

[data]

testfiles – optional list of multiple test files, including the one indicated in *testsfile***.**

currenttest – used when opening the program to reselect the test was previously selected when the program was last closed; syntax is <file>.<testgroup>.<test>. The indices are zero-based.