

Using igt-edit.

To load the program, run igt-edit.exe in Windows. There are a few support DLLs which must be located in the same directory as the executable. You do not necessarily have to run the program with the 'current directory' set to the location of the program. That is, you can create a shortcut to the program which specifies a different 'current directory.' But note that the configuration settings file 'app-settings.xml' will be created in the current directory. So if you later start the program with a different current directory, your saved settings will not be loaded.

Settings file

A few administrative settings are automatically stored in the **app-settings.xml** file. Most of the settings can be changed via a corresponding item on the "Options" menu in the program. The app-settings.xml file is a human-readable XML file, so you can inspect it to see what it contains, or to manually tweak settings. At this point, the following settings are saved:

"Escape key exits" – If selected, pressing the 'escape' key on the keyboard will exit the program. If 'Save on exit' is also selected, then all of your current IGT files will be saved. The default value is 'false' (escape key does not exit the program).

"Save on exit" – If selected, all of your current IGT files will be saved when exiting the program. This includes clicking on the close icon, selecting exit from the menu, or via pressing the escape key (if enabled). The default value is 'false' (Files are not saved upon exiting the program.)

"Reload Last session" – If selected, the IGT files you had open previously will be re-opened when the program starts. If possible, the specific IGT instance you were working on will also be re-selected. The default value is 'false' (Files are not reloaded when the program starts). To support this option, the settings file contains the list of files that you used in your last session.

"Reset app settings" – This menu command deletes the app-settings.xml file, with the result that default settings will be loaded the next time the program starts.

Also retained in the settings file is whether the overall application "window" was **maximized** on the screen or not. The previous state is restored the next time you run the app.

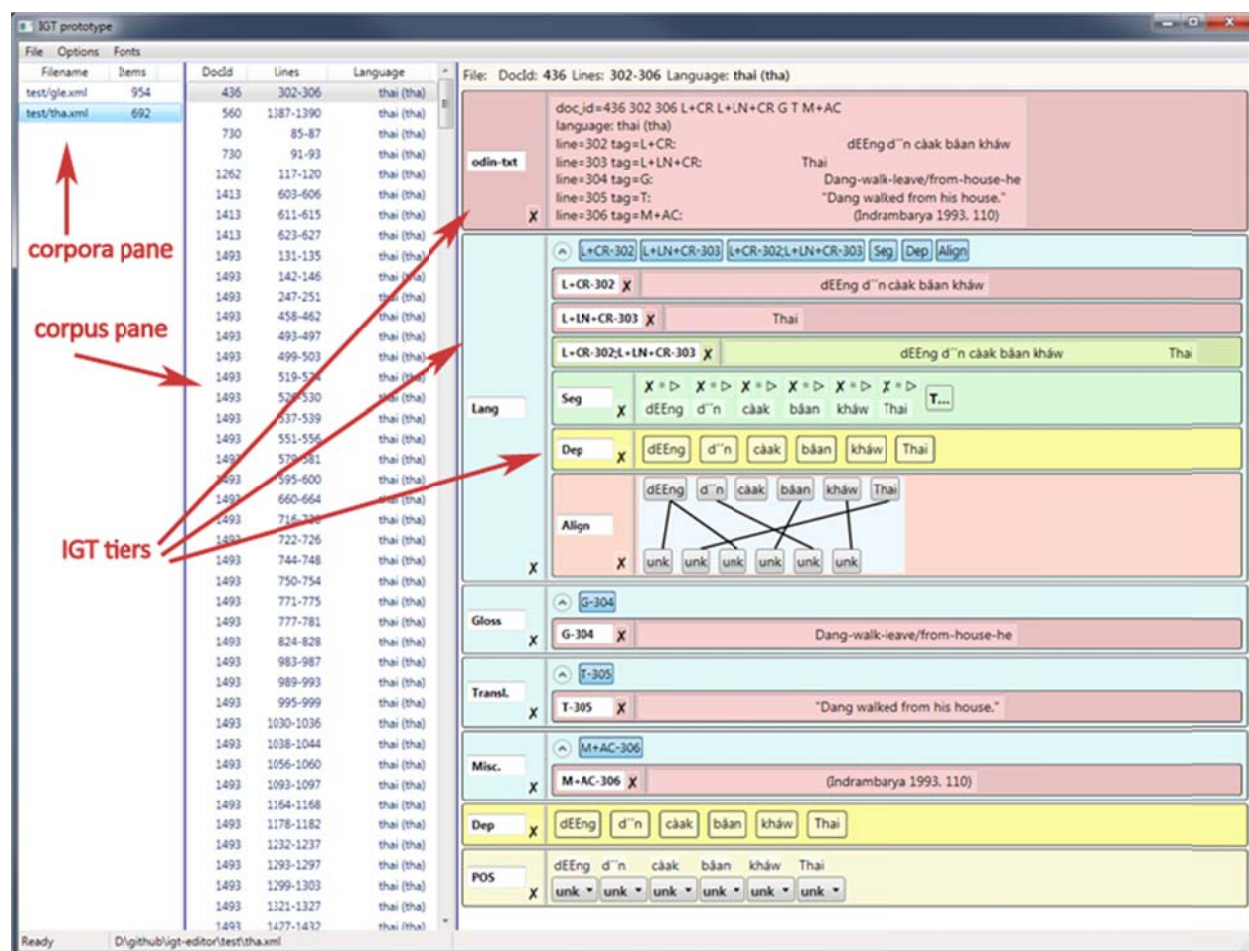
Corpora pane

The leftmost pane of the app shows the IGT files (each an "IGT corpus") that are loaded. You can have zero, one, or more IGT corpus files loaded. Selecting an IGT corpus file here causes all of the IGT instances it contains to be displayed in the Corpus pane. You can resize (or fully hide) the corpora pane using the mouse 'grip' along its right-hand vertical edge. You can move amongst the IGT corpus files using the 'up' and 'down' keyboard arrows. ***Pressing the 'right' keyboard arrow navigates to the corpus pane.***

Corpus pane

To the right of the corpora list is the Corpus pane, which lists each of the IGT instances available for editing in the currently selected IGT corpus. Select an IGT instance using the 'up' and 'down' keyboard keys. The contents of the selected IGT instance are displayed in the main editing area of the application.

Pressing the 'left' keyboard arrow navigates to the corpora pane.



File management

The editor program saves all of the loaded IGT instances to XAML-igt format. This is a human readable, XML-based format which represents the tier structure as defined, including nesting within group tiers, tier ordering, tier type "naming" and all segmentation, dependency, alignment, and part-of-speech (POS) annotations. To save all of the loaded IGT instances, select "Save all" from the file menu. You will be prompted for a directory in which all the files will be saved. The default directory for this is the last directory from which a XAML-igt file was loaded.

The program uses fail-safe saving to prevent loss of data. This means that, when saving files, a new temporary file is written first, before deleting the prior data. Only when the file is successfully written in its entirety is the temporary file copied over the old data. Therefore, if the program exits unexpectedly, only the current editing data is at risk of loss, and this is true even if the program is saving files.

There is also an option to abandon all of your changes from the current editing session. Use this if you made editing errors that you would prefer to revert entirely. To exit the program without writing any of your changes to XAML-igt files, select “Exit without saving” from the File menu.

If editing XAML-igt files manually, you will note that every entity in the file has a unique identifier string. This facilitates forward references when serializing the program’s object graph. If you are editing the XAML manually, it is not necessary to use the 128-bit numeric/hexadecimal format. In fact it may not be necessary to give any identifying attribute to any XML entity which is not referred to elsewhere in the file. But just to be safe, when editing the XAML-igt files manually, you should assign your own unique identifiers to any newly created entities, following any format for unique identifiers that you find convenient (these must use letters a-z, digits 0-9, and the underscore).

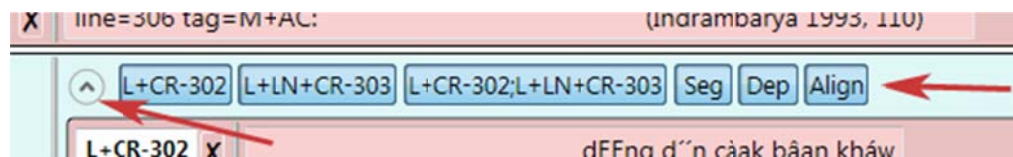
IGT editing concepts

The editing of IGT instances is based on two concepts: Tiers and Parts. A Tier represents one or more lines of text. There are three basic types of tiers:

Simple **“text” tiers** represent one line of text which has a designated or “well-known” logical function. Basic tiers can also represent raw source text possibly spanning multiple lines of a source document. In either case, text tiers represent text as a whole, and thus serve as the basis to which the “standoff annotations” of all other tiers must ultimately refer.

Group tiers can be used to arbitrarily group other tiers together. In this way you can nest tiers with related logical functions together. To move a tier into a group, you can drag and drop it. To move a tier out of its group into its parent (or parent group), right click the tier and select “Promote.”

Note that for certain operations, Tiers can only “see” other tiers in their group *plus* tiers in parent groups. For example, in the “Align with...” command, tiers which are nested below peers of the current tier’s parent are not eligible for alignment unless they (or the current tier) are first promoted.



Group tiers can be collapsed by clicking on the circled arrow near the top of the group tier header. This header area also contains “toggle” buttons which allow any of the tiers which are nested in the group to be temporarily hidden. To hide or unhide a tier that is inside a tier group, click on its respective button in the tier group header. Hiding or unhiding state is not saved in the XAML file.

Parts tiers are tiers that refer to segmented parts. The most basic parts tier is the segmentation tier, which describes an arbitrary standoff segmentation of a text tier. Other types of parts tiers refer to these segmented parts. These tiers include the POS tag tier (which assigns POS tags to segmented parts), dependency tier (which defines dependency relationships between segmented parts), and alignment tier (which defines alignment relationships between two sets of segmented parts).

Tier Type


Every tier has a free-form area to the left where you can type the tier type, or give the tier an identifying name, such as ‘Lang’, ‘Gloss’, etc. The **igt-convert** program pre-assigns some of this information based on the converted IGT instances. Manipulating tiers sometimes places suggested information into this field. It can be changed at will. There are many possible uses for this field. You can use this to name a group of “cleaned” tiers, to keep them all together and/or mark them as having been cleaned, for example.


Tier ordering

In addition to changing tier nesting with the “Promote” command (discussed above), tiers can be dragged to rearrange their ordering. You can drag a tier to a new position—either within its current group (if any)—or in an another tier group, or into the top level of the IGT. Note that if you drag the last tier out of a group, such that the group has no tiers left, that group will be permanently deleted.


Text tiers

There are two types of text tier.


The **basic text tier**  can show a single line of source text, or a section of multi-line source text. Currently basic text tiers cannot be entered in the program; they are already present in the IGT instances of the XAML file(s) which are loaded.

The “**multi-text**” tier  is assembled from other source text tiers by horizontal (adjacent) concatenation. The other source tiers can be basic text tiers or other multi-text tiers. To create a multi-text tier, select “Join with” on the context menu for an eligible tier. Currently, this user interface only lets you create multi-text tiers which concatenate two other text tiers, but the process can be repeated to create longer concatenations. Alternatively, a multi-text tier with more than two parts can be created by directly editing the XAML file.

Segmenting a text tier

Either type of text tier can be segmented (tokenized, or “word-broken”). This is the first step to take on raw data, in preparation for alignment, POS tagging, or assigning dependency structure. When you segment a tier, you create a **segmentation tier** , which refers to the source tier by standoff position. There are two ways to perform tokenization.

You can right click on a text tier and select “**Auto tokenize.**” This will create a new segmentation tier based on the whitespace in the specified text tier.

Alternatively, you can tokenize manually. Currently, this method only works with basic  (not multi) text tiers. In this method, you drag the cursor over exactly the text for which you want to create a token. Notice that a position indicator shows the extent of your dragging while this is in progress. When you release the mouse, the selected section is added as a “part” to the active segmentation line. If there is no active segmentation line, a new one will be created. Repeat the dragging process, and tokens will be

created in the desired order. To delete an erroneous token, click on the “X” button for the part that you don’t want.

Working with segmented parts

Once you have a segmentation tier containing tokenized “parts,” you can perform the following operations to finalize and clean the tokenization.

Add a new part: to add a new part which does not come from any source tier, click the button marked **T....**

Reorder parts: to change the order of parts, click and hold the black dot which appears with each part. This is a drag handle. Drag the part to the left or right to the desired position. **Note:** *it is possible to drag parts to other tiers using this method. They will be removed from their source tier and placed into the target tier. But this may lead to disorganized workflow, so use this feature with caution. One time when this is useful is to restore or recreate a part that was deleted, because dragging over source text to create a new part may not place that part into the desired tier.*

Delete parts: as mentioned above, to delete an unwanted part, click on its ‘X’. You can add the part again by dragging over text in a text tier.


Group parts together: To group together parts, first place them next to each other in the desired sequence (see ‘reorder parts’ above). Then, on the leftmost part, click the rightwards arrow to join it to the part to its right.

Completed segmentation tiers can be used as the basis for generating the additional tier types: POS tagging tiers, Alignment tiers, and dependency tiers. These will be discussed next. In general, creating these is initiated by right-clicking on an eligible tier. The available options will be displayed in a context menu.


Part-of-speech (POS) tier

A part of speech tier lets you assign parts of speech to each part. Currently, parts of speech are selected from a list of pre-configured options built into the igt-editor program.

Dependencies tier

The dependencies tier  lets you assign a tree-like dependency structure between the segmented parts from another segmented tier. Each dependent can have zero or one head. To set the head, right-click on the dependent and select an eligible head. Eligible heads are those that would not create a cycle in the tree. Only after a dependency has been established, the dependency type can be set, also by right-clicking and selecting a dependency type. Currently dependency types are selected from a hierarchy of pre-configured options built into the igt-editor program.

Alignment tier

The alignment tier  lets you establish the alignment of parts between two segmented tiers. Many-to-many alignments are supported.

1. After creating the alignment tier from two eligible tiers, you will see an upper and lower row representing the parts from the source and target tiers.
2. When you click an item in the upper row, it remains depressed (click again to unselect, or simply select a different item from the upper row) and the program is waiting for you to select an item in the lower row that this part aligns with.
3. Once an item in the bottom row is clicked, the alignment is set and a line connects the two aligned parts.
4. To undo this alignment, repeat the procedure, first selecting the upper row part, and then the part from the lower row that it is aligned to. The alignment will be removed.

Bug reports

Bug reports for the **igt-editor** and **igt-convert** program should be sent to gslayden@uw.edu. Please include detailed information on the exact steps for reproducing the problem.

Source code is located on the github repository **glenn-slayden/igt-editor**. The program can be built with Visual Studio 2013 Professional, which is available as a free download to UW students through the www.dreamspark.com website.