The File Format for SmartGo Books

Anders Kierulf

July 10, 2013

This document describes the file format used to include Go-specific data and diagrams in SmartGo Books, an app for iPad and iPhone that offers digital books about the game of Go.

History

SmartGo Kifu for the iPad pioneered a new way of presenting annotated SGF. Instead of forcing users to replay the game move by move, its book view breaks the game into diagrams and shows comments next to diagrams. Readers are used to such diagrams from Go books, and can often grasp a diagram at a glance. Unlike printed Go books or PDF documents, readers can replay the moves inside the diagram to explore the sequence in detail, and expand a partial diagram to see it in the context of the whole board.

The SmartGo Books app takes this a step further, offering complete electronic Go books. At first, those books were created with extensions of SGF. SGF has always been able to associate comments with moves; the SGF Kifu extensions added richer comments and more sophisticated flow of diagrams and text.

Experience with laying out twenty Go books using SGF Kifu highlighted the mismatch between the tree structure of SGF and the more linear structure of a book. Forcing a linear layout into SGF made the process of creating books more cumbersome and error prone than necessary.

This new format for SmartGo Books more closely mirrors the structure of a book, significantly improves performance of the app, and is more extensible for the future. It started out as an XML representation (closely resembling HTML for some aspects), but that turned out to be too hard to edit. It ended up as a simpler text-based format, taking cues from text-to-HTML formats like Textile and Markdown, and inheriting properties from SGF.

This file format is work in progress, and will evolve over time, but it has already proven itself in a shipping product that includes more than sixty Go books with a total of over 20,000 figures, 22,000 diagrams, and 3,500 problems.

Goals

The goals for this file format were as follows:

- Capture the structure of Go books while providing for flexible layout.

- Make it easy to edit and proofread Go books.
- Make it possible for nontechnical people to create Go books. In particular, simple text should just be represented as text.
- Make it easy to get existing Go content into electronic book form.
- Make it easy to share diagrams between multiple language editions.

This file format is specifically designed for interactive Go books; it's not intended for storing Go games or as a replacement for SGF.

Example

A short example will demonstrate the main points of the format.

```
::book(#sample) title="Sample Title" author="Some Body"

::chapter

::h1 break=none #title#

::s1 #author#

::img url="http://www.smartgo.com/img/books_ipad.png"

Some text describing this book.

::chapter(#contents)

::h2 Table of Contents

::p href=ch1 Link to First Chapter

::p href=dia2 Link to Diagram 2

::chapter(#ch1)

::h2 First Chapter

Start of a new chapter with ID ch1.

::go mv="D4 F4 H4 K4 M4 O4 Q4"

Simple sequence of moves, starting from the empty board, Black playing first,
alternating players. Normally, Go data contains all the moves of a game.

Paragraphs are separated by an empty line. Line breaks matter:
This is one line.
This is another line.

::fig at=1 to=3 A=D6

This figure shows moves 1 to 3, with the letter A shown on the board at D6.

::dia at=2 mv=F6H6 ca="Dia. 1"
```

```
Diagram with caption __Dia. 1__ showing the position before move 2 in the
game, with an alternate sequence for moves 2 and 3.

::h3 Minor heading

::fig at=4 to=7 vw=A1T10

Figure showing moves 4 to 7 of the Go data. Only a partial view of the board
is shown.

::dia(#dia2) at=5 mv=M6O6Q6 ca="Dia. 2" vw=A1T10

__Dia. 2__ shows a sequence of three moves starting at move 5. They'll be
numbered 1 to 3. It has an ID so it can be referenced later.

This paragraph contains an embedded diagram. <dia at=2 mv=O7Q7>When you tap
here, you'll see an alternate sequence.</dia>

::dia base=dia2 at=2 mv=O8Q8 ca_en="Dia. 3 English" ca_ja="Dia. 3 Japanese"
vw=A1T10

__Dia. 3__ is a variation based on Dia. 2. At the second move in the
variation, it shows two alternate moves. Japanese and English will show
different captions.

::dia base=none ab=C3C4D5 aw=D3D4 pl=w mv=E5 a=E6 vw=A1G8
ca="Self-contained diagram"

::p align=center Some centered text with --- em dash, **bold**, __italic__,
**bold __italic__**, and __italic **bold**__.

Go Books can contain translations, and the user can select to have the book
presented in a specific language or multiple languages.
<a href="de">Tap here to switch to German.</a>
<a href="en">Tap here to switch back to English.</a>

::de Dieser Paragraph wird nur auf Deutsch angezeigt.

::en This paragraph is only shown when English is selected.

This example explored just a few features of this format. For more
information check the <a href="http://www.smartgo.com">SmartGo web site</a>.
```

Book structure and layout

The format for SmartGo Books specifies a linear flow of content: text, diagrams, and images. The overall structure of a book:

- Each file contains one book. The file extension should be `.gobook`.
- A book consists of chapters.
- Each chapter consists of headings, paragraphs, diagrams, images, and Go data.
- Go data consists of board positions and move sequences played at those positions.

- Diagrams are based on Go data. Go data can be separate or can be part of a diagram.
- The paragraphs following a diagram describe that diagram.
- Diagrams can be embedded inside a paragraph, and can be shown by the user on demand. These are called inline diagrams.

Layout

The structure of paragraphs and diagrams determines what will be shown, and it gives clues for the layout, but it doesn't specify exactly how everything will be laid out. The layout depends on device capabilities and orientation, as well as user settings such as font, font size, and multi-column layout. Where possible, diagrams will be placed near associated text, but there is no guarantee that a text will be visible at the same time as its associated diagram.

File structure and elements

The file is divided into elements; each element can have various attributes and a value.

In addition to the specific attributes listed for each element type, each element can have an id and languages:

- **`id:`** The ID of an element is used to refer to that element, e.g. a specific heading or diagram. IDs should be unique per chapter, as referenced IDs will first be matched in the current chapter, then in the whole book if no match is found.
- **`language:`** The language or languages of this element (see details below).

None of the attributes listed for an element are required. The default type if not specified is text.

Element, attribute, and ID names are case sensitive. Element and attribute names all start with an ASCII letter and contain only ASCII letters and digits, plus underscores for languages. ID names consists of Unicode text and can contain underscore, dash, and period, but no other symbols. ID names can't consist entirely of language tags.

Text and lines

SmartGo Books files are text files, in utf-8 encoding. Accepted line endings are Unix/Mac style (LF) or Windows style (CRLF).

Elements are separated from each other by an empty line. Lines containing only spaces or tabs are treated as empty lines. (Thus line breaks within paragraphs indicate line breaks in the text; empty lines indicate paragraph breaks.)

Languages

A book can include multiple translations. The reader can select which language or languages to see; only elements that are language-neutral or that are tagged with one of

those languages will be shown. The language setting is inherited from the parent element.

In practice, figures and diagrams will be shared among all translations, while each text paragraph will be marked with a specific language.

Each language is identified by a two-letter code: **`cn`** (Simplified Chinese), **`de`** (German), **`en`** (English, default), **`es`** (Spanish), **`fr`** (French), **`it`** (Italian), **`ja`** (Japanese), **`ko`** (Korean), **`po`** (Polish; not **`pl`** due to conflict with **pl**ayer), **`pt`** (Portuguese), **`ru`** (Russian), **`tw`** (Traditional Chinese), and **`ui`** (the current user interface language).

Multiple languages can be separated by an underscore. For example, a **`book`** element for English and Japanese would be specified as **`book_en_ja`**. Some attributes can also be language-specific.

Note that the language doesn't really specify the language of the text, but rather the book language for which a specific element should be displayed. However, standard text will be inserted using that language.

Elements and attributes

Proper elements start with a double colon, the element name (with optional languages appended using underscores and optional #ID in parenthesis), followed by attributes in the form `attributeName=attributeValue`, followed by the value of the element. No spaces are allowed around the equal sign separating attribute and attribute value. For example:

```
::h2_de(#someID) attr1=simpleValue attr2="a b c" Table of contents
```

Attributes are either `attr=valueWithoutSpaces`, or `attr="value with spaces"`. Quotes are necessary when the value contains spaces or special characters (specifically, any character other than letters, digits, and underscore). Within quotes, backslash and quote characters need to be preceded by a backslash. Tabs can be added using `\t`.

If an element doesn't start with a double colon, it is assumed to be a paragraph without any ID or attributes.

Go coordinates

Some of the XML attributes contain lists of coordinates. All coordinates are expressed in standard coordinates (`A1` .. `T19`, letter 'I' skipped), with `A1` as the bottom left corner. In lists of coordinates, an optional space can be used to separate the coordinates.

For lists of moves, '`Pass`' and '`Tenuki`' are allowed instead of a point. They differ in that `A1PassB1` will be numbered as 1, 2, whereas `A1TenukiB1` will be numbered as 1, 3, with '2: elsewhere' shown in the list of moves. Pass is useful if you are interested in showing how many moves it takes one player to capture a group rather than a move sequence of both players.

Any program writing this format should always write standard uppercase coordinates. While reading gobook files, lowercase standard coordinates as well as

lowercase coordinates in SGF format should be accepted in attributes (but not in paragraphs).

Book

The **`book`** element must be the first element in the file, and must only occur once. The **`id`** of the book should be unique; in a library of books, it would typically be the SKU (stock-keeping unit) of the book, e.g. SmartGo Books uses `sg0006_ki_invin` for “Invincible” from Kiseido and `sg0069_ss_wrk5` for “Workshop Lectures vol. 5” from Slate & Shell.

Book attributes

- **`format:`** (integer) The version of the file format (default: 0). This will be incremented if there are substantial changes to the format that are not backwards-compatible.
- **`revision:`** (integer) The revision of this book (default: 0). Incrementing the revision when the content of the book gets updated allows apps to reload the book as needed.
- **`title, subtitle, author, publisher, copyright:`** (language-specific texts) The main title, subtitle, authors, publisher, and copyright of the book. These texts can be language-specific, e.g. you can specify both **`title_en`** and **`title_ja`** to provide different titles in English and Japanese.
- **`cover:`** The URL of the cover image. Default: The **`id`** of the book with `_cover.jpg` appended.
- **`fullWidth:`** (0 or 1) Usually diagrams are shown slightly smaller than figures. Setting this forces all diagrams in this book to be shown at full width (default: 0), unless overridden by the chapter or diagram setting.

Chapter

The chapters are represented by **`chapter`** elements. Each chapter contains Go data, paragraphs, diagrams, and images.

Note that chapters are a grouping mechanism, making it easy for e.g. a table of contents to refer to a chapter, and can allow the user interface to provide ways to navigate by chapter; chapters don’t add anything to the content flow.

The ID for the table of contents chapter should be `"contents"` so that the app can provide navigation to that chapter.

Chapter attributes

- **`fullWidth:`** (0 or 1) Same as for the **`book`** element, but applied to this chapter.

Go data

Go data is represented by **`go`** elements. Go data contains a starting position, an optional sequence of moves, and optional game information. Go data can be based on other Go data.

Note that Go data does not add any text or visuals to the content flow.

Board position attributes

- **`sz:`** (integer) Board size (square boards only). Defaults to 19.
- **`ab`**, **`aw:`** (point list) List of black or white stones to add to the position.
- **`pl:`** (`"b"` | `"w"`) Whose turn it is to play at that position (default: black).
- **`koPoint:`** (point) Point that is forbidden because a ko just got played.

Instead of explicitly describing the board position, a **`go`** element can also reference another Go data or diagram element and get its data from the position at a specific move.

- **`base:`** (`ID` | `"none"`) ID of the **`go`** or diagram element that this element is based on. This inherits attributes like board size from the base data. The default is the most recent Go data in the current chapter (see below for special defaults for diagrams); set to `none` to not base the diagram on any go data. **Note:** The base needs to be defined before it's used, no forward references.
- **`at:`** (integer) Move number (default: starting move number of the base, or `1` if no base). Designates the position **before** that move gets played in the sequence; for example, for a sequence of 3 moves, `at=2` specifies the position after move `1`, `at=4` specifies the position at the end of the sequence. This also updates whose turn it is to play. The move number is in the numbering scheme of base. Note that passes in the move sequence are not counted.
- **`ae:`** (point list) Stones to remove from this position. **Note:** This is only intended for diagrams where the author wants to show the position as it would have been if some earlier moves had not been played. Do not use it to create arbitrary positions from other positions.

If a position is specified using **`at`**, the **`ab`** and **`aw`** properties can be used to add stones to that position. Again, this should only be used if it reflects the author's intent.

Move sequence attributes

The move sequence can be either specified explicitly with **`mv`**, or use **`at`** in combination with **`to`** to designate a range of moves in the data this element is based on. (The two ways of specifying a sequence are mutually exclusive.)

- **`mv:`** (list of: point | `"Pass"` | `"Tenuki"`) Sequence of moves, alternating players, starting with the player given by **`pl`**.

- **`to:`** (integer) Specifies a sequence of moves from the base. This range is inclusive: for example, `at=3 to=5` means moves 3, 4 and 5 are included. Use the same number for **`to`** and **`at`** to include a single move.

Game info attributes

The game info associated with Go data is not added to the content flow, but may be shown by the app. Any game info to appear as text in the book should be added directly using regular paragraphs.

- **`pb`** (black player), **`br`** (black rank), **`wp`** (white player), **`wr`** (white rank), **`km`** (komi), **`ha`** (handicap), **`re`** (result), **`tm`** (time limit), **`dt`** (date), **`pc`** (place), **`ev`** (event), **`ro`** (round), **`us`** (user), **`an`** (commentary), **`cp`** (copyright), **`so`** (source): Like the corresponding properties in SGF.

SGF

To make it easy to create books based on existing data, Go data can be supplied as SGF. This is strictly for input; any files written in SmartGo Books format should use the properties described above.

- **`sgf:`** Text that will be interpreted as SGF. The following SGF properties will be translated to the equivalent Go data properties: SZ, AB, AW, PL, B, W, and the game info properties. Only the main line of the game is used, and any other properties like comments, markings, and diagrams are ignored.

Display Elements

Diagrams, paragraphs, and images constitute display elements that share a few common attributes.

- **`align:`** (`"left"` | `"center"` | `"right"`) Alignment of the diagram, paragraph, or image. Default is `"left"` for paragraphs and `"center"` for diagrams and images. (Justified text is not supported because it usually doesn't look good at the narrow column widths that can be shown on mobile devices.)
- **`href:`** ID of an element within this book to navigate to when the user taps this element, a language to switch to, or a normal http link for going to a web site. All other interactions with this element are disabled. (Within a paragraph, use standard `<a>` links. See the section on paragraph links for examples.)
- **`break:`** (`"page"` | `"column"` | `"line"` | `"none"`) Start this element at a new page or column. For paragraphs, `line` forces it to start on a new line (ends text wrapping around diagrams). Default is `page` for **`h1`** and **`h2`**, `column` for **`h3`**, and `none` for other paragraphs and elements.

Diagrams

A diagram describes a specific board position and move sequence to be shown as an image. In interactive media, the move sequence can be animated.

Diagrams are an extension of Go data, and can contain anything that Go data can contain. Unlike Go data, diagrams are part of the content flow, and can contain markup to be shown in the diagram.

Diagrams are expressed by **dia**, **fig**, **var**, and **prb** elements. The four elements are identical except for their defaults, reflecting their intended usage:

fig: Main figures of a game, based on a Go game given as Go data, with move numbers reflecting the moves in the game, and the move numbers listed below the figure. Shown at full size.

dia: Diagrams showing alternate move sequences. Move numbering starts at 1. Usually shown at a slightly smaller scale than the figures.

var: Variation based on another diagram. By default, continues the move numbering of that diagram. Based on the most recent diagram rather than Go data.

prb: Diagrams showing a problem diagram. Problems are shown at full width, and move input will react differently, giving feedback on correct or wrong solutions.

element	base	number	full width	showFromTo	showToPlay
fig	Go data	at	1	1	0
dia	Go data	1	0	0	0
var	diagram	at	0	0	0
prb	Go data	1	1	0	1

Diagram attributes

- **mn:** (integer) Starting move number for numbering the sequence (default **at** for **fig** and **var**, 1 for **dia** and **prb**).
- **vw:** (point point) Rectangular section of the board to be shown (default is full view).
- **a, b, ..., z, A, B, ..., Z:** (point list) Mark board points with a single letter. For example, `A=K5 b=C4C5` will mark K5 with a capital A and C4 and C5 with a lowercase b.
- **A1, A2, ..., A19, B1, ..., T1, ..., T19:** Show text at the point with that coordinate. For example, `B5=23` would show 23 at point B5.
- **tr** (triangle), **ma** (cross), **sq** (square), **rg** (diamond), **cr** (circle), **tb** (black area), **tw** (white area), **ln** (line), **ar** (arrow): (point list) List of points for markup on board, like in SGF. For lines and arrows, consecutive pairs of points define the end points.

- **`width:`** (`"full"` | `"dia"` | `"half"` | `"third"`) Usually diagrams are shown slightly smaller than figures. Setting this to `full` forces diagrams to be shown at full width. The **`fullWidth`** attribute sets the default for a whole book or chapter. Two `half` or three `third` width diagrams can fit side-by-side. (Currently, the alignment of half- and third-width diagrams needs to be set.)

Caption attributes

- **`ca:`** (language-specific texts) The caption to show below the diagram. Simply **`ca`** is language neutral; specify **`ca_en`**, **`ca_ja`**, **`ca_de`** for language-specific captions.
- **`showToPlay:`** (0 | 1) Whether to add the current player in the caption (default 0, except 1 for **`prb`**).
- **`showFromTo:`** (0 | 1) Whether to show move numbers in caption (default 1 for **`fig`** with no caption, 0 for **`dia`**, **`var`**, **`prb`**, and **`fig`** with caption). If there is no move or moves sequence, no move numbers are added.
- **`gaps:`** (text | `"auto"` | `"none"`) Moves that can not be shown as numbered stones on the board (because they're on top of other moves or captured stones) are mentioned below the diagram, in a separate caption line. The default is `auto`, causing the program to generate an appropriate text like "12 at 7", "ko: 5, 8", or "9: connects". If that is not good enough, just specify the text to show. Specify `none` to not list move gaps.

Problem attributes

The following attributes only apply to diagrams marked as **`prb`**.

- **`correct:`** (move sequences) Move sequences that are correct to play in this position. Default: If the problem is based on Go data that includes a move sequence, and this diagram doesn't show that move sequence, then assume that move sequence is correct.
- **`wrong:`** (move sequences) Move sequences that show what happens when you play the wrong move.

For both right and wrong answers, multiple move sequences can be separated by semicolons. Each move in the move sequence can be followed by text in parenthesis that will be shown when that move has been played. For example:

```
::prb sz=9 pl=b correct="E1E2D1(White is dead.);E1D1E2(Seki.)"
```

The value of a **`prb`** diagram can contain text that will be shown at the start of the problem. While playing through the problem interactively, that text will be replaced by any text specified in **`correct`** or **`wrong`**. The text for the user's move will be shown only briefly when answered by a computer move.

Inline diagrams

A diagram may be embedded inside a paragraph. In that case, it needs to be delimited using XML syntax.

Inline diagrams are based on the diagram referenced in the containing paragraph, thus if no base is set, they refer to the most recent diagram or Go data in the flow. For example:

```
See what happens <dia at=2 mv=C4B5D6>if Black plays 2 at 3</dia> instead.
```

In this example, the text “if Black plays 2 at 3” can be tapped to show a diagram with three moves starting at move 2 in the diagram associated with this paragraph. The details of how the diagram is shown will vary on different devices.

The **`vw`** attribute is inherited from the diagram the inline diagram is based on.

Restriction: Inline diagrams can’t be used as a base for other diagrams.

Paragraphs

The elements **`p`**, **`h1`**, **`h2`**, **`h3`**, **`h4`** and **`h5`** are interpreted like in HTML; **`s1`** and **`s2`** are used for subtitles. Regular paragraphs (**`p`**) are the default for any text that doesn’t specify an element type. Paragraphs are by default associated with the most recent diagram or Go data in the flow.

Paragraphs for particular languages can simply use the language tag as the element. For example, **`de`** is equivalent to **`p_de`** to denote a German paragraph.

The headings are customarily used as follows:

`h1:` Main title of the book. Large font, centered. Starts a new page.

`h2:` Chapter title. Medium large font, centered. Starts a new page.

`h3:` Section title. Font slightly larger than normal, bold, left-aligned. Starts a new column.

`h4:` Minor section title. Same font size as text, bold, left-aligned. Does not start a new page or column.

`h5:` Minor title. Doesn’t affect the font; does not start a new page or column.

`s1, s2:` Subtitles to h1 and h2. Slightly smaller, centered, extra space above and below.

Paragraph style

The **`style`** attribute (`"normal"` | `"block"` | `"hanging"` | `"indent"` | `"bullet"` | `"text"`) is used to format paragraphs on a high level. The style is inherited until the end of the chapter, until the next heading (except `text`), or until the next paragraph with style `normal`.

- `block:` Indents the paragraph from both left and right by a standard amount. This can also be set using the **`leftIndent`** and **`rightIndent`** attributes.
- `hanging:` Indents all lines of the paragraph except the first by a standard amount. This can also be set using the **`hangingIndent`** attribute.

- `indent:` Indents the paragraph by the standard amount.
- `bullet:` Looks for tabs in the text of each consecutive bulleted paragraph, finds the maximum amount of indentation, and creates a hanging indent to match. Note that this does not add any bullet characters, it just uses the text up to the tab as a bullet.
- `text:` Separates paragraphs by indenting the first line (by 1 em) instead of adding extra leading. This is more appropriate for long passages of text rather than a mix of text and diagrams. Note that the first paragraph after a heading or diagram is not indented. Unlike the other styles, this style is not terminated at headings.
- `normal:` Normal paragraph used to reset inherited styles.

Standard indentation is two em (1 em = current font size). For `block`, `hanging`, and `indent` styles, if the paragraph starts with tabs, the paragraph is indented by the number of tabs times the standard indentation.

Paragraph attributes

- **`keepWithNext:`** (0 | 1) Keep this with next paragraph if possible. Default is 0, except 1 for titles (**`h1`**, **`h2`**, **`h3`**, **`h4`**, **`h5`**) and subtitles (**`s1`**, **`s2`**).
- **`hint:`** (0 | 1) This paragraph is a hint that should be hidden on first reading; some way for the user to uncover the hint is needed.
- **`ref:`** `(ID | "none")` ID of the diagram that this paragraph is describing. Can be `none`, in which case this paragraph is not associated with any diagram. (Default is the most recent diagram in the flow; the reference is empty at the beginning of a new chapter.)
- **`paraSpacing:`** (integer) The space before and after this paragraph, in 1/10 em. This is inherited until the next chapter. Can be overridden by **`spaceBefore`** and **`spaceAfter`** for specific paragraphs. The default depends on the style: for **`bullet`** and **`hanging`**, the spacing is about half the regular paragraph spacing.
- **`spaceBefore, spaceAfter:`** (integer) The space before or after the paragraph, in 1/10 em. The space between paragraphs is the maximum of the space after one paragraph and the space before the next paragraph. Headings have default settings for these attributes. To add an empty line of space before a paragraph, use `spaceBefore=8` (the default setting for paragraphs separated by empty space).
- **`firstIndent, hangingIndent, rightIndent:`** (integer) The indentation of the first line, the remaining lines, or the right side, in 1/10 em.
- **`leftIndent:`** (integer) Sets **`firstIndent`** and **`hangingIndent`**.
- **`blockIndent:`** (integer) Sets **`firstIndent`**, **`hangingIndent`**, and **`rightIndent`**.
- **`box:`** (0 | 1) Draw a box around this paragraph (default 0). If consecutive paragraphs are marked this way, a single box will be drawn around all of them. (Restriction: This currently only works well for text that is also using blockIndent.)

Paragraph value

The value of a paragraph is the text in the paragraph. However, a paragraph may also contain `<dia>` elements (where the value of the diagram becomes tappable text). For more on text formatting see below.

As paragraphs are delimited by empty lines, line breaks can be added directly in the paragraph. Line breaks can also be added inline using `\n`, `<br>` or `<br />`.

Rich text in paragraphs

Text can include markup and abbreviations that will be displayed in a richer way.

Text style

****bold text**:** Any text surrounded by double asterisks will be shown in **bold**.

__italic text__: Any text surrounded by double underscores will be shown in *italic*.

Bold and italic can be nested, e.g. *__italic text with *****bold italic*****__* or ****bold text with ***__bold italic__*******. However, styles can't span multiple paragraphs.

Symbols

Go books often use special symbols, some of which are not in Unicode. You can insert the following symbols into the text:

- `^T` → triangle: △ (**`tr`** attribute).
- `^D` → diamond: ◇ (**`rg`** attribute).
- `^0` → black or white stone (shown as small circle when not followed by b or w).
- `^S` → square: □ (**`sq`** attribute).
- `^X` → cross mark: ✕ (**`ma`** attribute).

Each of these can be followed by a lowercase `b` or `w` to more specifically refer to a black or white stone with such a mark. When possible, these should be shown that way in the text, e.g. `^Tb` should be shown as a black circle with a triangle on it. (SmartGo Books restriction: Diamonds are not shown in circle yet, nor cross on black stone.)

Coordinates

A1 to T19 refer to coordinates on the board. If there is an associated diagram, there must be a letter, number, or symbol at that point to replace this coordinate. For example, "invade at F8" will be changed to "invade at A" if F8 on the board is marked as A in the diagram. (If there is no currently active diagram, i.e. `ref=none`, no coordinate substitution takes place.)

There is no exception for coordinates B1 to B19. Informal game comments sometimes use B1, W2, B3 to refer to moves Black 1, White 2, Black 3, but in the context of SmartGo Books, these will always be interpreted as coordinates.

Typography

Text in paragraphs can use simple typography; SmartGo Books improves the typography to make it easier to read and more like a high-quality book:

- Straight quotation marks (single & double) → curly quotation marks. (These may vary by user language.)
- Double hyphens → en dash.
- Triple hyphens → em dash.
- Triple periods → ellipsis.
- Double spaces → single space. (Triple spaces are not changed.)
- Hyphen between two digits → en dash.
- Hyphen between two spaces or at end of paragraph → en dash.

Early implementations also converted (c) to ©, but that could conflict with actual text, so use the Unicode copyright symbol directly in the text instead.

Links

Tappable links can be embedded in the text using HTML syntax:

```
<a href="http://www.smartgo.com">SmartGo web site</a>
<a href="mailto:support@smartgo.com">Contact support.</a>
```

You can link directly to a given chapter, paragraph, or diagram by linking to the ID of that element. The ID is searched in the current chapter first; if not found, it's searched in the whole book. If an ID is not found, the link should be disabled and e.g. shown in gray (this can be used in the table of contents for a book sample).

```
<a href="chapter2">Tap here to go to chapter 2.</a>
```

If the link consists of a language tag, the book language is switched to that language:

```
<a href="ja">Tap here to switch to Japanese.</a>
```

Image bullets

Images can be used as bullets. Like other bulleted paragraphs, the image is measured and consecutive bulleted paragraphs are indented by the same amount. For example:

```
::p style=bullet <img url="image.png" href="http://smartgo.com"></img>\tSome
paragraph text.
```

Standard Text

A number of specific text strings within hash tags will be replaced by text in the language of the current paragraph. For example, `#btp#` will be replaced by “Black to play” if the element is marked as English, by “A Noir de jouer” if it’s marked as French, and by the corresponding text in the current user interface language if it’s marked as language neutral.

The main purpose of these standard texts is to define language-independent Go problems. The following texts are currently defined; `^` needs to be replaced by `b` for Black or `w` for White:

```
#^tp#  : “^ to play.”
#^tn#  : “^ to respond.”
#^tk#  : “^ to kill.”
#^tl#  : “^ to live.”
#^tc#  : “^ to capture.”
#^tf#  : “^ to fight.”
#^tu#  : “^ to cut.”
#^to#  : “^ to connect.”
#^tx#  : “^ to extend.”
#^tr#  : “^ to reduce.”
#^ti#  : “^ to invade.”
#^ta#  : “^ to attack.”
#^td#  : “^ to defend.”
#^te#  : “^ to escape.”
#^ts#  : “^ to save the marked stones.”
#^tw#  : “^ to win.”
#^twf# : “^ to win the fight.”
#^twd# : “^ to draw or win.”
#^tgk# : “^ to get a ko.”
#^tgr# : “^ to get the best result.”
#^tpe# : “^ to play the best endgame.”
#^tms# : “^ to make shape.”

#^do#  : “What can ^ do?”
#^wi#  : “^ wins.”

#^he#  : “^ has escaped.”
#^hnk# : “^ has no ko threat.”

#^id#  : “^ is dead.”
#^ia#  : “^ is alive.”
#^ias# : “^ is alive in sente.”
#^iddk#: “^ is dead in double ko.”
#^iadk#: “^ is alive in double ko.”

#fe#   : “False eye.”
#ko#   : “Ko.”
#ok#   : “Only ko, not good enough.”
#os#   : “Only seki, not good enough.”
#se#   : “Seki.”
#sn#   : “Snapback.”
#prb#  : “Problem” (note: no period)
```

The special strings `#app#` and `#ver#` get replaced by the name and version number of the current app. The strings `#title#`, `#subtitle#`, `#author#`, `#publisher#` and

`#copyright#` can be used to insert the corresponding attribute from the book element. (Restriction: This is currently limited to single-language books, as the text gets replaced before choosing the display language.)

Standard texts with a language tag appended get replaced with the string in that specific language (e.g. `#wtms_de#` or `#btw_ja#`).

SmartGo specific: Numbers within hash tags (e.g. `#123#`) get replaced by the resource string of that number.

Images

Images can be inserted in the flow using the **`img`** element:

```
::img url="http://smartgo.com/img/icon_smartgo_kifu.png" scale=0.75
```

Image attributes

- **`url:`** The image to insert. This can be an http link to an external image, or just the file name of an image that the app knows about. Default: The cover image of the book.
- **`scale:`** (floating-point number) Normally, the image will be drawn either at its natural size, or filling the width of the column, whichever is smaller. This would be the default (1.0). A smaller value of scale can be used to reduce the image size.
- **`ca:`** Caption (as mentioned above for diagrams).
- **`border:`** (0 | 1) Whether to draw a dark thin border around the image. Defaults to 1 for jpg images, otherwise 0.
- **`shadow:`** (0 | 1) Whether to draw a shadow behind the image. Defaults to 1 for jpg images, otherwise 0.
- **`fill:`** (0 | 1) Whether to draw a dark fill color behind the image. Defaults to 0.
- **`width:`** (`"full"` | `"half"` | `"third"`) Sets the image at a specific width instead of full width (like diagrams). Scale is computed off that width.

Buy book button

A button for buying the current book can be inserted using the **`buy`** element. The interpretation of this element is app-specific. For example, in SmartGo Books, this may show up as BUY BOOK if the book is available for purchase, and as PURCHASED if the book has already been bought.

Attributes

- **`sku:`** The SKU of the book to buy. Default: ID of this book.

Include standard chapters

Text used in several books can be included with the `::include` statement:

```
::include url="about.gobook"
```

This includes all the elements from the given url as if it was directly inserted in the text. (This is currently limited to `"about.gobook"`, `"buy.gobook"`, and `"kiseido.gobook"`, which are standard sections included with SmartGo Books.)

Attributes

- **`url:`** The text to insert. For now, this can only be a file name that SmartGo Books knows about.

Comments

Comments are elements that will be read and written, but don't change the book in any way. Comments have a value but no attributes. Some examples:

```
::c This is a comment.

::c ::include url="about.gobook" This include is commented out.

::c attr=value ERROR: comments can't have attributes.
```

Conventions for writing gobook format

Apps that write gobook format should adhere to the following conventions to minimize formatting changes that make it harder to review differences:

- Exactly one empty line before all elements, except two empty lines before headings (**`h1`** - **`h5`**) and **`include`**, and three empty lines before **`chapter`**.
- Write attributes in the same order as they were read. Preserve whether an attribute starts on a new line or not.
- Always use quotes around **`ca`** and **`url`** attributes, even if not strictly required.
- New line before all **`book`** attributes and game info attributes.
- Always write **`pl`** attribute for go data that specifies a position.

Possible future additions

- Diagram attribute **`showCoords`** to show the coordinate system around a board. In general, coordinate display should be an option left to the user, but in some cases the writer might want to turn on coordinate display for specific diagrams.
- Allow a whole diagram to be duplicated inline by reference (**`clone`** attribute).

- Book and chapter attribute **`shortTitle`** to avoid truncation of book and chapter titles (e.g. in navigation controls).
- Diagram attribute **`header`** to add a header above the board, in addition to the caption below.
- More control over text: Smaller/larger fonts, small caps, etc.
- More general default settings for book and chapter. (The **`fullWidth`** attribute captures the most common case for now.)
- Diagram attribute **`hideInReplay`** (point list) to specify board marks to be hidden during diagram replay.
- Allow katakana for labels on the board.
- Vertical writing for Japanese.
- Allow full answer trees in problem diagrams instead of move sequences.
- Allow game info from the current Go data to be shown in the text using #pb#, #br#, #wp#, #wr#, #km#, #ha#, #re#, #tm#, #dt#, #pc#, #ev#, #ro#, #us#, #an#, #cp#, #so#.

Feedback welcome

Thanks for great feedback from everybody who has worked with early versions of this format, in particular John Mifsud and Richard Hunter.

This is work in progress. Please send your thoughts to anders@smartgo.com.

SGF definition

http://www.red-bean.com/sgf/