Go typesetting using TEX

Here is a system for typesetting Go games and diagrams using TEX. This system may be used with either plain TEX or LaTEX. It includes the METAFONT sources for a new set of fonts, called GOOE, and a `perl` script called `sgf2dg` which translates files in the common "Smart Go Format" (SGF) files into (plain) TEX.

The numerals on the Go stones in these fonts are instances of the Computer Modern Fonts designed by Donald Knuth and his co-workers as part of the TEX system. The file `romandg.mf` is the same as the file `romand.mf` distributed with TEX, with only trivial modifications to allow the fonts generated from it to be pasted onto Go stones. This file is of course copyrighted by Donald Knuth. The remaining portions of the GOOE/`sgf2dg` system are published under the Gnu Public License, a copy of which is distributed with this system in the file `COPYING`. Although this system is thus free software, we would appreciate acknowledgement if it is used to publish a book.

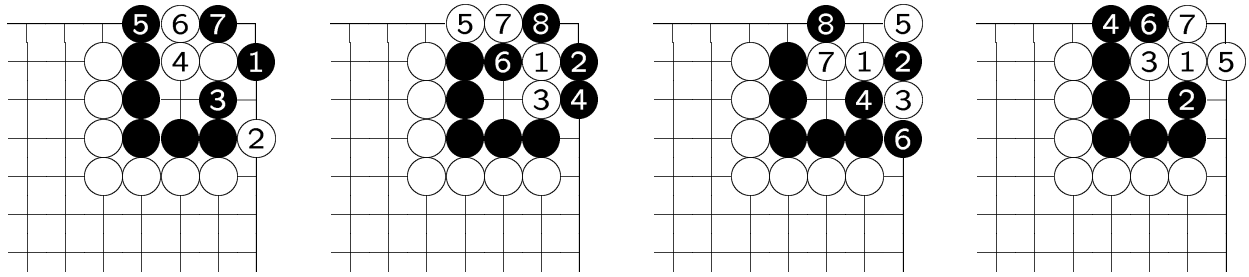
For installation instructions, please see the file INSTALL that is included with this package.

Another set of Go fonts was created by Hanna Kolodziejska around 1990, and revised by Jan van der Steen. Those fonts, together with the latex document style file `go.sty` can be found at CTAN in the directory `fonts/go`, or at the Go ftp sites (such as `ftp://igs.nuri.net` in the `Go/printing` directory) under the name `golatex`. Jan van der Steen's utility `sgf2misc` has the capability of generating LaTEX or postscript files from SGF. It may be obtained from `ftp://igs.nuri.net` in the `Go/prog` directory. Our work is independent of and different from Kolodziejska's and van der Steen's.

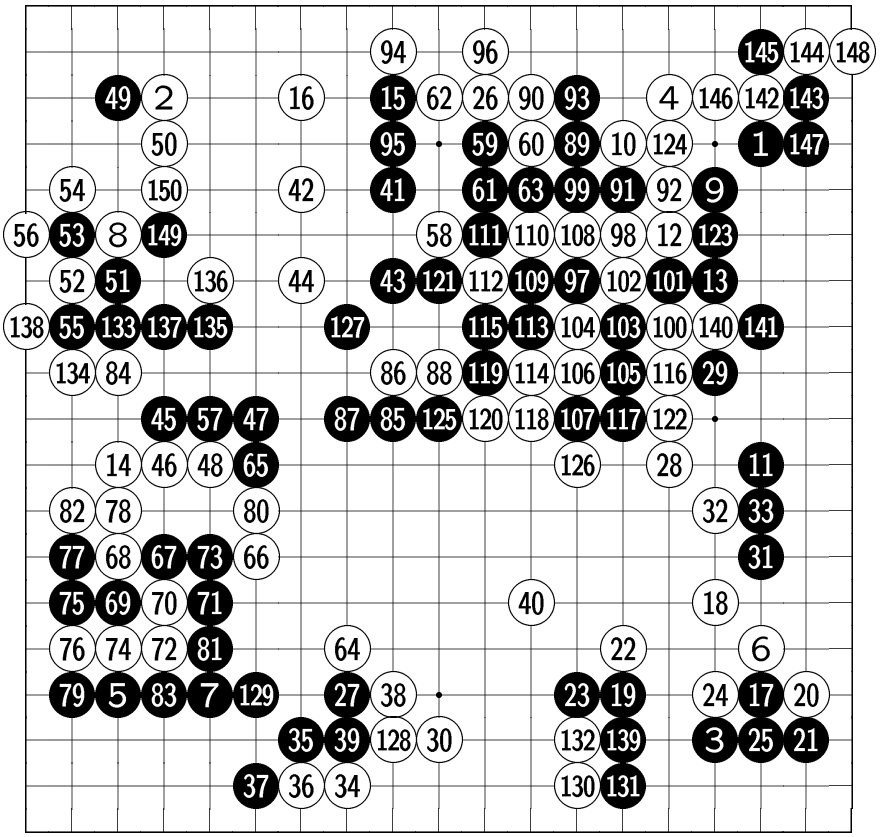
`sgf2dg` may be obtained from `http://match.stanford.edu/bump/go.html` or `ftp://match.stanford.edu/pub/sgf2dg-<version>.tar.gz`. It is also available from the Comprehensive Perl Archive Network (CPAN): go to `http://search.cpan.org/` and search for `sgf2dg`.

TEX is a trademark of the American Mathematical Society. METAFONT is a trademark of Addison Wesley Publishing Company.

Daniel Bump (`bump@math.stanford.edu`)
Reid Augustin (`reid@hellosix.com`), 1997, 1998



Black ❷ is bad.
Black dies.



Left: A classic game with big fonts.

Ito Showa 6-Dan (W)
Shusaku, 4-Dan (B)
October 15, 1844

sgf2dg

Go games are commonly stored in the "Smart Go Format." For example, games played on the internet on IGS or NNGS are stored in this format. Tools such as `xmgt`, `cgoban` or `xgoban` can be used to generate SGF files using a mouse.

The program `sgf2dg` takes a game in Smart Go Format and translates it into a TEX file. As long as you have the GOOE fonts and the macros in `gooemacs.tex` you can `tex` the resulting file, or edit it and incorporate it into a longer document. `sgf2dg` is a `perl` script, and you must have `perl` (version 5.001 or later) installed to run it. If you do not have an up-to-date perl, you can obtain it from the Comprehensive Perl Archive Network (CPAN) at `http://cpan.org` or from `http://language.perl.com/info/software.html`.

Included with GOOE/`sgf2dg` is an SGF file titled `genan-shuwa.sgf`. It is the record of a famous game. The commentary is not intended to be profound but is included to show how `sgf2dg` treats comments. After running

```
$ sgf2dg genan-shuwa.sgf
```

(or `C> perl sgf2dg genan-shuwa` from the DOS prompt under Windows) and

```
$ tex genan-shuwa
```

we obtain files called `genan-shuwa.tex` and `genan-shuwa.dvi`. Here is the *unedited* result of this experiment:

May 16-18, 1842
White: Genan
Black: Shuwa

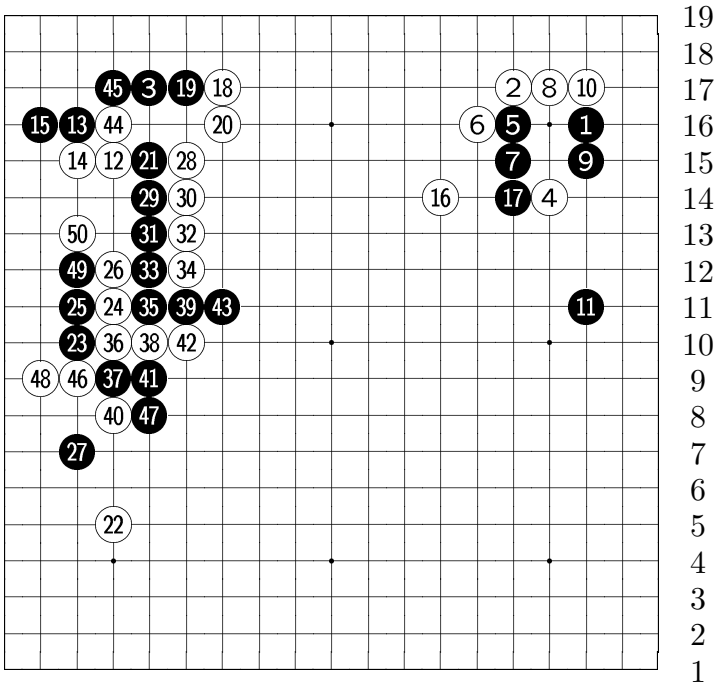


Diagram 1: 1-50

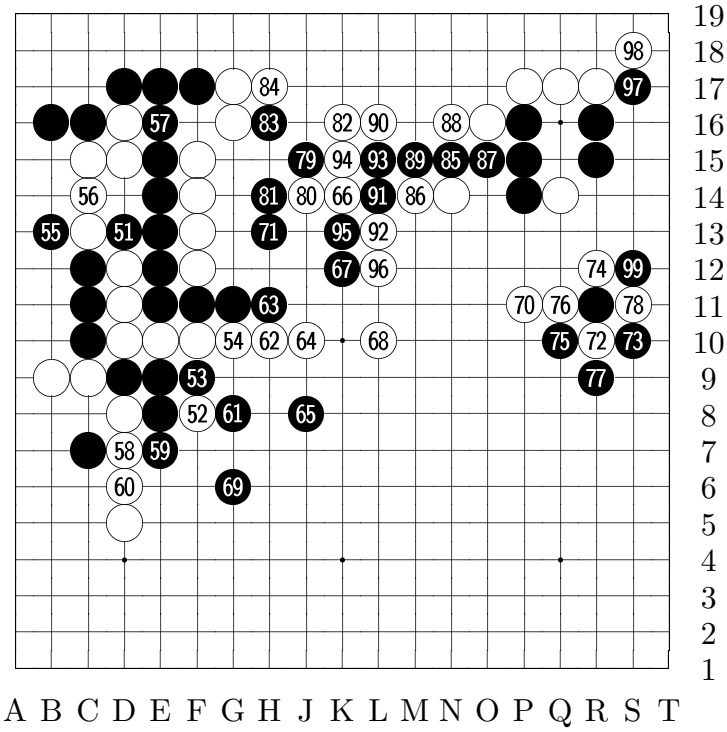


Diagram 2: 51-100

(100) at (72)

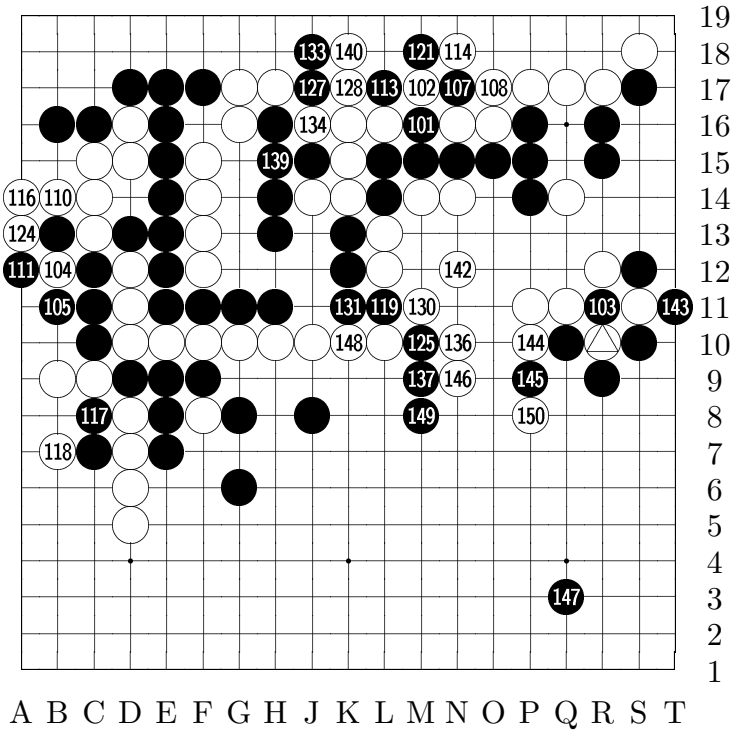


Diagram 3: 101-150

(106), (112), (120), (126), (132), (138) at △

(109), (115), (123), (129), (135), (141) at (103)

(122) at (107)

124: Now the situation becomes complicated beyond the point of mortal understanding, since in addition to the ko on the right, White is trying to revive his dead stones on the left. If they live, White does not need to worry about eyes in the center.

(200) at (186)

165: As is traditional, the captured stone is removed from the board.

188: At long last, White takes the chance to eliminate the ko aji.

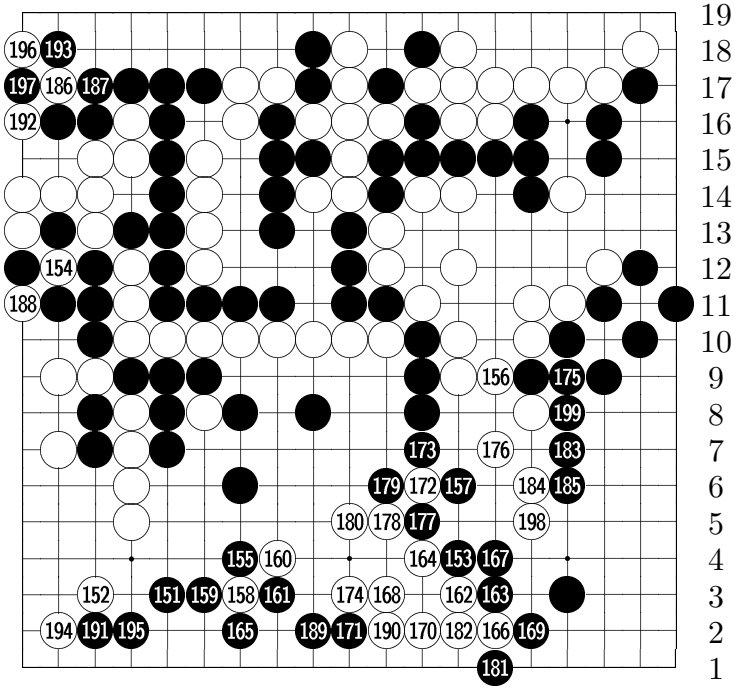


Diagram 4: 151-200

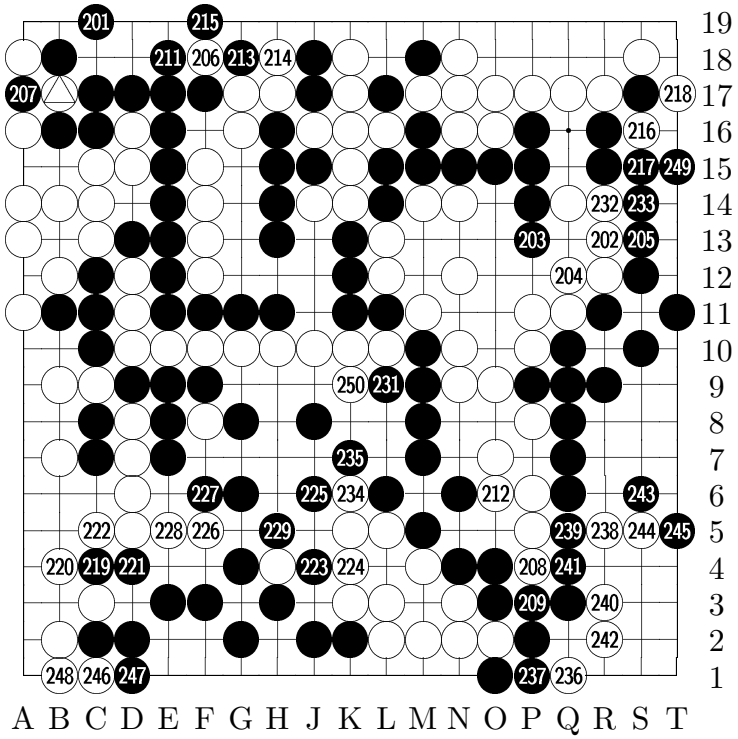
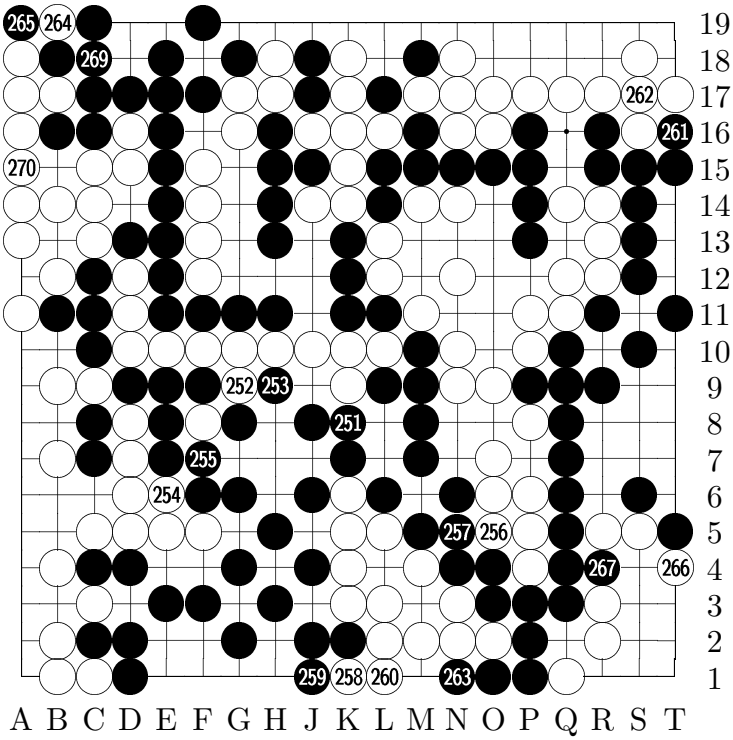


Diagram 5: 201-250

210 at △

230 at 207

245: White is dead in the corner. It is unknown whether Genan knew the invasion was doomed, or whether he made a misreading. According to one theory, Genan knew that he was slightly behind and invaded hoping for a Black mistake. Of course, the failed invasion made Black's lead larger, and after a few more moves, Black won by 6 points.



(268) at (264)

Diagram 6: 251-270

This is the end of the included TEX pasted from `genan-shuwa.tex`.

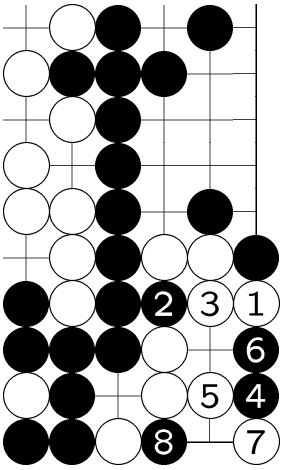


Diagram 7

For completeness, let us explain why White is dead in the corner. The following sequence was not played, and is beyond the reading of most players. It is certain that both players read this out and saw the tesuji of ❽. What is unknown is whether Genan perhaps saw this tesuji *before* he played at 236, and hoped that Shuwa would miss it. If ❽ is used to capture ⑦, then White obtains a ko. Perhaps Genan was aware of this, but felt that he was behind, and hoping for a swindle invaded anyway, then gave up the plan?

Diagram 7 was not generated directly from the game file `genan-shuwa.sgf` since it shows moves not played in the game. These moves are included in `genan-shuwa.sgf` as a *variation*. Most Smart Go Format editors support variations, which are discussed below in a separate section. The variation diagram as produced by `sgf2dg` is labelled "**Variation 1** on Move 245." We did not wish to use it unedited. Rather, we reran `sgf2dg` with the options `-n -break 246,254 -l 14 -t 10 -bigFonts -simple`, producing output which we edited into the current document to make **Diagram 7**.

Command Line Options

We now describe the options which come with `sgf2dg`. First, `sgf2dg -h` (or `sgf2dg -h|more`) prints a help message, then exits. (Under Windows, use the command `perl sgf2dg -h` instead, or more generally `perl sgf2dg` [*options*] [*filename*].)

```
sgf2dg [options] [file.sgf]

 -h | -help                 print this message and exit
 -v | -version              print version number and exit
 -verbose                   print diagnostic information
 -i | -in                   input file name (STDIN for standard input)
 -o | -out                  output file name (STDOUT for standard output)
 -t | -top                  top line in diagram
 -b | -bottom               bottom line in diagram
 -l | -left                 leftmost line in diagram
 -r | -right                rightmost line in diagram
 -crop                      auto-crop diagram
 -break | -breakList        a list of first move in each diagram
 -m | -movesPerDiagram      number of moves per diagram
 -d | -doubleDigits         label stones modulo 100
 -n | -newNumbers           begin each diagram with number 1
 -rv | -relativeVarNums     start each variation from 1
 -av | -absoluteVarNums     move numbers in variation == main line numbers
 -cv | -correlativeVarNums  start main variations from 1
 -rl | -repeatLast          repeat last move as first in next diagram
 -ic | -ignoreComments      ignore SGF comments
 -il | -ignoreLetters       ignore SGF letters
 -im | -ignoreMarks         ignore SGF marks
 -iv | -ignoreVariations    ignore SGF variations
 -ip | -ignorePass          ignore SGF pass moves
 -ia | -ignoreAll           ignore SGF letters, marks, variations, passes
 -firstDiagram              first diagram to print
 -lastDiagram               last diagram to print
 -placeHandi                place handicap stones on board (old style)
 -coords                    print coordinates
 -cs | -coordStyle style    coordinate style:  normal, sgf, or numeric
 -c | -convert | -converter name of a Diagram converter (see below)
 -simple                    use a very simple TeX format
 -mag number                TeX \magnification - default is 1000
 -twoColumn                 use two-column format
 -bigFonts                  use fonts magnified 1.2 times
 -texComments               \, { and } in comments not modified
 -floatControl string       'string' controls float (diagram) placement
```

The help message continues with some more explanation of each option. We will discuss the options in more detail now.

`-verbose` is useful for tracking down sgf2dg and converter decisions about diagram breaks.

The `-i` and `-o` options are not needed with normal usage. The usual usage is `sgf2dg` [*options*] `filename` instead of `sgf2dg` [*options*] `-i filename`. If a file named `filename.sgf` or just `filename` exists, it will create a TEX file called `filename.tex`.

In its default usage `sgf2dg` produces full-board diagrams. You may specify the top, bottom, left and right edges with the `-t`, `-b`, `-l` and `-r` tags. For example, `-l 14 -t 10` produces a diagram in which has its left edge at 14, its right edge at 19 (the default), the top edge at 10, and the bottom edge at 19 (the default). These settings were used in **Diagram 7**.

`-crop` trims empty lines from the diagrams. At least two lines around each stone are not cropped, and if the board edge is within three lines of a stone, it is not cropped.

Generally one does not want an entire game in a single diagram. `-breakList` and `-movesPerDiagram` are mechanisms for controlling the number of moves in the diagrams.

The `-breakList` <*list*> option, where <*list*> is a list of moves separated by commas, will produce a sequence of diagrams broken at the given moves.

If `-movesPerDiagram` <*N*> is used, where <*N*> is an integer, then `sgf2dg` will break the diagram after each <*N*> moves. If neither `-breakList` nor `-movesPerDiagram` is specified, the `-movesPerDiagram` option is understood, with <*N*> set by default to 50. (It can be disabled by setting `-movesPerDiagram 10000`.) On the other hand if `-breakList` is used, then <*N*> defaults to infinity. The options `-breakList` and `-movesPerDiagram` can be used together.

`-doubleDigits` and `-newNumbers` are alternative schemes for keeping the numbers on the stones small. If `-doubleDigits` is set, then the stones are numbered modulo 100, or more precisely, the first move in the diagram is reduced modulo 100 to a number from 1 to 99.

If `-newNumbers` is set, each diagram begins numbering afresh with the numeral 1. Both schemes emulate strategies commonly used in commenting games.

There are circumstances where `sgf2dg` will automatically break a diagram at a point not specified by the user. When a stone is placed on a spot where there was previously a stone (as, for example, in a ko fight), `sgf2dg` marks the first stone if it is unnumbered (i.e. was played on a previous diagram). Then it adds a comment of the form "67 at 55" or "67 at △." In order to prevent ambiguities, there should not be more than one marked stone of the same color referenced in this way on a single diagram. To prevent this, `sgf2dg` may break a diagram at a point not specified by the user.

There are three alternative schemes for numbering in variation diagrams. In the default numbering scheme, each variation is numbered starting with a stone numbered "1." We call this scheme *relative* variation numbering. You can specify it using `-rv` or `-relativeVarNums`, but you don't have to, since it is the default.

In the second scheme, called *absolute* variation numbering, each variation is numbered relative to the beginning of the game. Thus if the variation occurs at move 120, the first stone in the variation diagram is given the number 120. This option is invoked with `-av`

or `-absoluteVarNums`.

In the third scheme, called *correlative* variation numbering, each variation is numbered from the first move to deviate from the main line. Thus all variations beginning with the same move (branching from the main line) are numbered consistently. This option is invoked with the `-cv` or `-correlativeVarNums` option.

`-repeatlast` causes the last move in each diagram to duplicate the first move in the next. This emulates a common style of Go writing.

`-ic` or `ignoreComments` causes SGF comments to be ignored.

`-il` or `ignoreLetters` causes SGF letters to be ignored.

`-im` or `ignoreMarks` causes SGF marks to be ignored.

`-im` or `-ignoreVariations` causes variations to be ignore. (Variations are discussed below in a later section.)

`-ip` or `ignorePass` causes SGF passes to be ignored. If this option is not used, a pass causes a remark in the comments.

`-ia` or `ignoreAll` causes all SGF letters, marks, variations and passes to be ignored.

`-firstDiagram` <*Diagram Number*> allows you to specify the first diagram to print;

`-lastDiagram` <*Diagram Number*> allows you to specify the last diagram to print.

`-placeHandi` should not be necessary except when converting rather old style SGF files. Newer SGF formats require explicit handicap placement using the normal AddBlack (AB) property. If your diagram is missing its handicap stones, this option may fix it.

`-converter` selects the output converter. See the next section for more details on converters. The default converter is `Games::Go::Dg2Tex` which creates source for TEX. Converters get '`Games::Go::Dg2`' prepended, so enter only the part after `Games::Go::Dg2`. The default is thus equivilent to `-converter TeX`.

The remaining options apply only to the `Dg2TeX` converter.

`-simple (Dg2TeX)` avoids TEX complexity, producing the diagram with the comments below it. This option is useful if you intend to edit the TEX and don't want a fancy format forced on you.

`-mag number (Dg2TeX)` changes the default `\magnification` from 1000 to `number`.

`-twoColumn (Dg2TeX)` implements a two-column format, making use of the alternative macros in `gotcmacs.tex`. Two-column formats are used in some Go books and magazines, for example John Power's beautiful *Invincible: The Games of Shusaku*. The Go font is reduced by 1.2 in size so that two 19×19 boards may fit side-by-side on the page. The text font is not reduced.

`-bigFonts (Dg2TeX)` uses `\bgoo` and `\bgoe`, using fonts which are 1.2 times larger.

`-texComments (Dg2TeX)`: Certain characters, when found in comments, are normally remapped as follows:

\	→	$\backslash$
{	→	$\lbrace$
}	→	$\rbrace$
$	→	\$
&	→	\&
#	→	\#
∧	→	$\wedge$
_	→	_
%	→	\%
∼	→	$\sim$
<	→	$<$
>	→	$>$
|	→	$\|$

This is done because either *these characters do not exist in TEX roman font*, so it is impossible to print them without changing fonts, or because the characters have special meaning to TEX (see *The TEX Book* page 38). Instead of using the characters directly, `Dg2TeX` makes plausible substitutions. When `texComments` is specified, the mappings are supressed so you can embed normal TEX source (like `{\bf boldface}`) directly inside the comments.

`-floatControl (Dg2TeX)` is followed by a control string. In normal mode (not `simple` and not `twoColumn`), `Dg2TeX` floats the board diagram to the left or right of the text and if the text is extensive enough, it flows around the diagram. The control string is a sequence of letters that control placement of the diagram float.

Letter	Diagram placement
l	left (text to the right)
r	right
a	alternately on the left and right
any other	random

The first letter is for the first diagram, second letter is for the second diagram, and so on. When there is only one letter left, that letter controls all remaining diagrams. The default control string is '`rx`' which places the first diagram on the right, and the rest are placed randomly.

Output Converters

`sgf2dg` is designed to use plugin output converters. Each converter produces diagrams in a specific output format. The output converters available with the current distribution are:

`Games::Go::Dg2TeX`	TEX source (default)
`Games::Go::Dg2Mp`	MetaPost embedded in TEX (Encapsulated PostScript)
`Games::Go::Dg2ASCII`	simple ASCII diagrams
`Games::Go::Dg2Ps`	PostScript
`Games::Go::Dg2PDF`	Portable Document Format (PDF)
`Games::Go::Dg2Tk`	Perl/Tk NoteBook/Canvas
`Games::Go::Dg2TkPs`	PostScript via Dg2Tk (Dg2Ps is nicer)
`Games::Go::Dg2SL`	Sensei's Library (thanks to Marcel Gruenauer

Converters are quite easy to write - it take just a few hours if you are already conversant with the conversion target. If you would like to create a converter plugin module, the easiest way is to make a copy of an existing converter (Dg2ASCII.pm for example) and modify it. Once it's working, please be sure to send us a copy so we can add it to the distribution.

Converters are always prepended with 'Games::Go::Dg2', so to select the ASCII converter instead of the default TEX converter, use:

```
sgf2dg ...  -converter ASCII ...
```

Converter names are case sensitive.

The default output filename suffix is determined by the converter name: the name is lower-cased to become the suffix, so the ASCII converter produces `<filename>.ascii` from `<filename>.sgf`.

You can select different default converters by changing the name of the `sgf2dg` script (or better, make symbolic links, or copies if your system can't handle links). The converter name is extracted from the name with this regular expression:

```
m/sgf2(.*)/
```

Anything after `sgf2` is assumed to be the name of a converter module. For example, let's create a link to the script:

```
$ cd /usr/local/bin
$ ln -s sgf2dg sgf2Xyz
```

Executing:

```
$ sgf2Xyz foo.sgf [ options ]
```

attempts to use `Games::Go::Dg2Xyz` as the converter. The converter name extracted from the script name is case sensitive.

The three extracted names `tex`, `diagram` and `dg` are treated specially. These three names always attempt to use `Games::Go::Dg2TeX` as the converter.

Converter Options

Converter options are prepended with the converter name so that option `xyz` for converter `Games::Go::Dg2Abc` is written on the command line as:

```
$ sgf2dg ...  -Abc-xyz ...
```

Converter options that take arguments must be quoted so that the shell passes the option and any arguments as a single ARGV. For example, if the `xyz` option for converter `Dg2Abc` takes `foo` and `bar` as additional arguments, the command line would be:

```
$ sgf2dg ...  "-Abc-xyz foo bar" ...
```

or a more realistic example of changing the background color:

```
$ sgf2dg genan-shuwa -converter Tk "-Tk-bg #d2f1b4bc8c8b"
```

Since `sgf2dg` is a super-set replacement for the Sgf2TeX package, TEX holds the default position for converters. Because of this historically priviledged position, the `Dg2TeX` options do not need to be prepended with `-TeX-`.

See the `man` or `perldoc` pages for converter-specific options. eg:

```
perldoc Games::Go::Dg2PDF.
```

Of the converters available with this release, `Dg2ASCII` takes no additional options. `Dg2Tk` and therefore `Dg2TkPs` take no additional options directly, but arguments are assumed to be standard `Tk` and `Tk::Canvas` options and are passed to the `Tk::Canvas` widgets. This explains why the background color example above works.

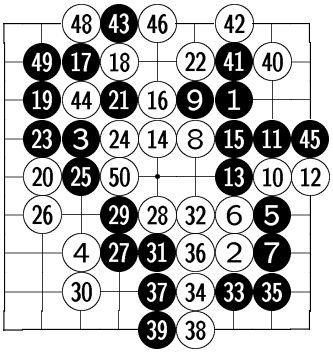
Handling Variations using `sgfsplit`

If your SGF file contains variations, `sgf2dg` will print them and generate labels for them according to a logical scheme.

We also offer an alternative. The `C` program `sgfsplit` will create from the file a sequence of other SGF files each containing one variation. The normal usage is `sgfsplit` [*filename*]. This will create from the file [*filename*] or [*filename*]`.sgf` a sequence of files called [*filename*]`.0.sgf`, [*filename*]`.1.sgf`, etc. The first file contains the main line, the remaining files, variations. Also created is an executable shell script called [*filename*]`.sgf2dg` which contains suggested commands for invoking `sgf2dg`. In particular, breakpoints are specified for each variation. You may wish to edit this file before executing it to fine tune the board sizes. If you invoke `sgfsplit` in the form `sgfsplit` [*filename*] [*extra parameters*] the extra parameters (such as `-simple`) are passed along in [*filename*]`.sgf2dg`.

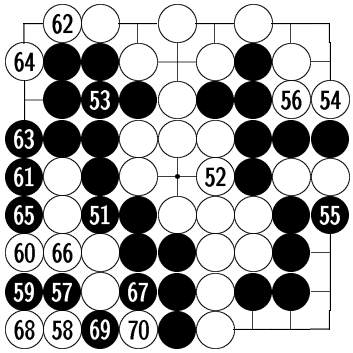
One may ask why `sgfsplit` is needed since `sgf2dg` has built-in support for variations. One possible reason is that by splitting up the file into variational components, one may supply different values of the `-t`, `-b`, `-l` and `-r` parameters. This requires editing the `[filename].sgf2dg` file before executing it.

Small Boards



47 at 21

The most commonly used small board sizes are 9×9 and 13×13. If `sgf2dg` is used on a game of any of these sizes, an appropriate small board is used. The board size is set using the SGF property `SZ`.



The FF4 specification for SGF files allows `SZ[X:Y]` where `X` and `Y` are arbitrary numbers. `sgf2dg` supports this syntax.

TEX Macros

Using the `sgf2dg` translator one need not worry about writing TEX code. Nevertheless with some understanding of how TEX works, GOOE/`sgf2dg` becomes a more powerful system. One method is to use `sgf2dg -simple` to create TEX files, which may then be edited and merged into a larger document. For this, some understanding of how TEX works is useful. (The unedited output of `sgf2dg -simple` is not beautiful—this option is perhaps *only* useful if you intend to edit its output. But if this is your intention, `sgf2dg -simple` will produce a file as uncluttered as possible.)

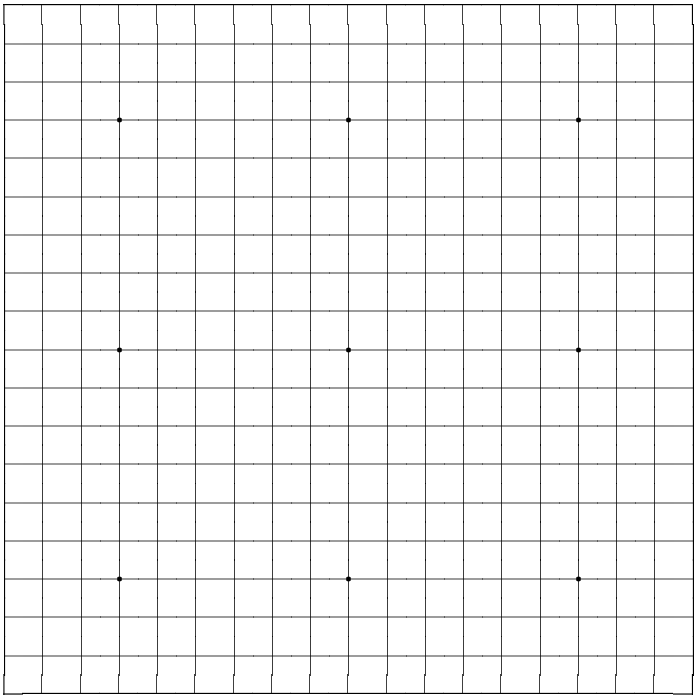
The ultimate source for TEXnical enlightenment is of course *The TEX Book* by Donald Knuth (Addison-Wesley, 1984). We will try to explain a few things which may be helpful to the user.

The file `gooemacs.tex` contains macros needed to typeset Go games. You may either paste these into the beginning of a file, or input them. The most important macros defined in `gooemacs.tex` are `\goo` and `\goe`. These toggle Go modes in which the black stones have odd numbers (`\goo`) or even numbers (`\goe`).

`gotcmacs.tex` is an alternative to `gooemacs.tex`, implementing a two-column format. The macros in `gotcmacs.tex` are used if you run `sgf2dg` in the `-twoColumn` mode.

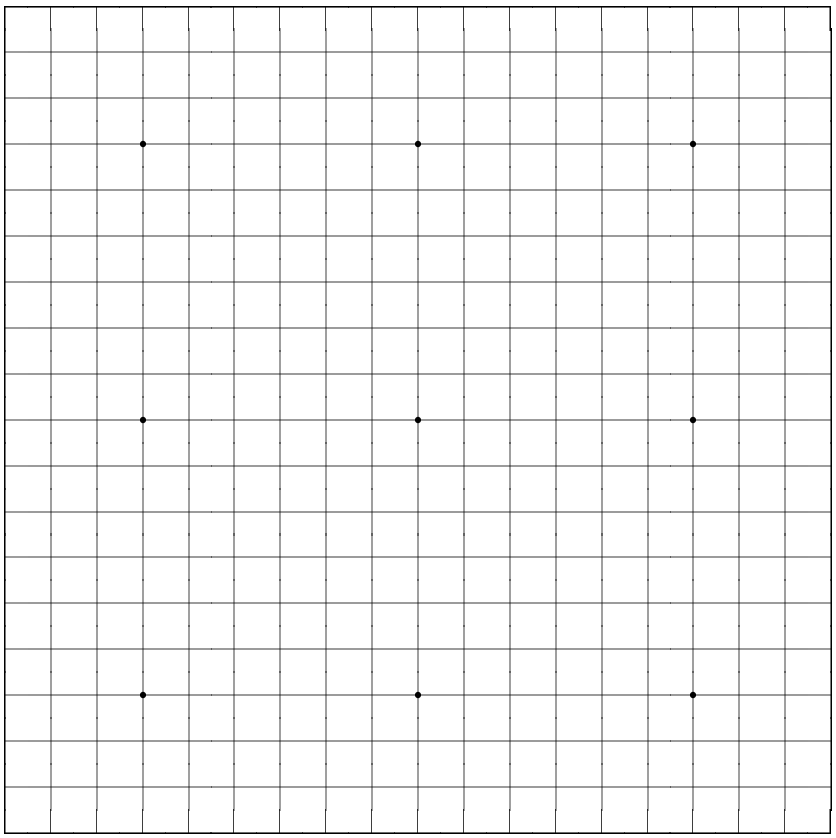
Here is the TEX code needed to produce an empty Go board, with some text alongside, as on the following page:

```
\hbox to \hsize {\vbox{\hsize=228pt\goo
\0??<\0??(\0??(\0??(\0??(\0??(\0??(\0??(\0??(\0??(\0??(\0??(\0??(\0??(\0??(\0??(\0??(\0??(\0??>
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??*\0??+\0??+\0??+\0??+\0??+\0??*\0??+\0??+\0??+\0??+\0??+\0??*\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??*\0??+\0??+\0??+\0??+\0??+\0??*\0??+\0??+\0??+\0??+\0??+\0??*\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??*\0??+\0??+\0??+\0??+\0??+\0??*\0??+\0??+\0??+\0??+\0??+\0??*\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??,\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??.}
\hfil
\vbox to 228pt{\hsize=1.8in\tolerance=10000\vglue6pt
\noindent{\bf Example 1 (left).} Empty Go board.  Blah blah blah etc.
\vfil}}
```



Example 1 (left). Empty Go board. We use the macro `\goo` instead of `\goe` because we intend to fill this with stones in which the black ones are odd numbered. Cut the relevant lines from this file `manual.tex` and use them as a template to which stones may be added as in the succeeding examples.

Example 2 (below). If you want a bigger font within the same document, you may use `\bgoo` and `\bgoe` in place of `\goo` and `\goe` to get the fonts in 14 point.



The TEX code given above for creating an empty board should be parsed as follows:

```
\hbox to \hsize{\vbox{\hsize=228pt\goo[Game Diagram]}
\hfil
\vbox to 228pt{\hsize=1.8in\tolerance=10000\vglue6pt[Text]\vfil}}
```

In TEX, there are two types of "boxes" which can contain typography. A *vertical box* can consist of smaller boxes stacked together vertically, and a *horizontal box* can consist of smaller boxes stacked together horizontally. A horizontal box can be created using the `\hbox` macro, so

`\hbox{`[First Box] [Second Box] ...`}`

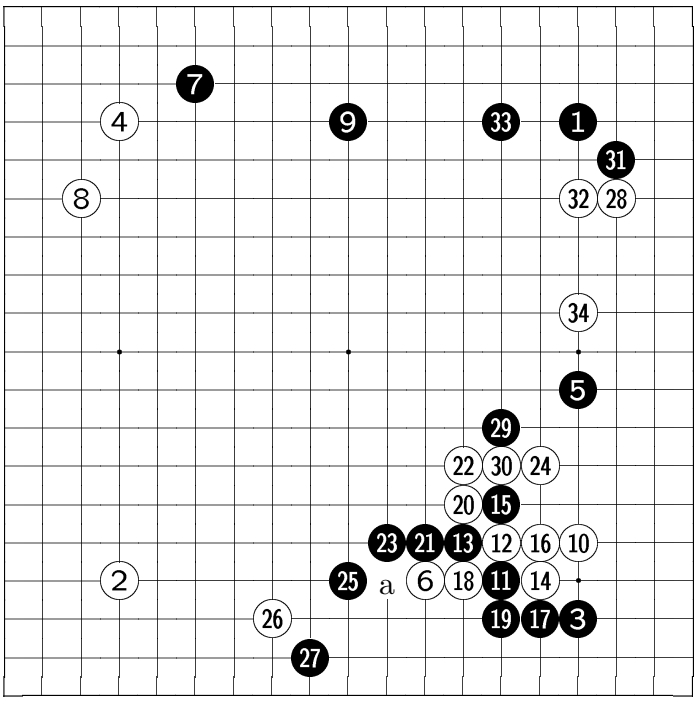
creates a horizontal box containing a series of other boxes. or `\hbox to` ⟨dimension⟩`{...}` creates a horizontal box of specified width. Similarly `\vbox to` ⟨dimension⟩`{...}` makes a vertical box of specified height, and by putting the command `\hsize=`⟨another dimension⟩ inside the braces we can also control the width of the vertical box.

The above diagram consists of a horizontal box containing two vertical boxes, separated by an `\hfil`. The latter is expandable horizontal "glue" or space filling material which can grow horizontally to fill the gaps between the boxes. The first vertical box consists of the game diagram, and since each intersection on the Go board is a box 12 points high and 12 points wide, its dimension is $12 \times 19 = 228$ points. (1 inch equals 72.27 point.)

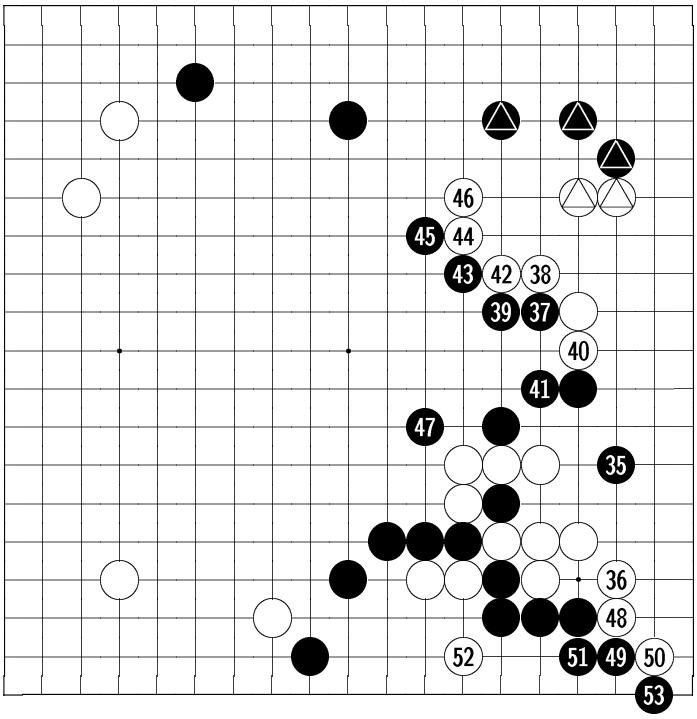
The second vertical box is specified as

`\vbox to 228pt{\hsize=1.8in\vglue6pt` [Text] `\vfil}`.

Thus we've forced it to have height 228 points—to match the known height of the first vertical box—and width 1.8in. The width is chosen somewhat arbitrarily to make a small separation between the boxes. Plain TEX sets the width of a line to 6.5in, but since the first line of `manual.tex` is `\magnification=1200`, the true `hsize` is actually $6.5\text{in} \div 1.2 = 391.462\text{pt}$. Thus the `\hfil` separating the two vertical boxes will expand to fill the gap of $391.462 - 228 - 108.405 = 55.057\text{pt}$ or about .915 inch. Inside the second vertical box we find three items stacked vertically. The first (`\vglue6pt`) consists of inflexible vertical glue filling 6 points; the second consists of the text which goes alongside the diagram; and the third (`\vfil`) consists of flexible vertical glue to fill the space down to the bottom of the box. The text is thus moved down 6 points in the box. This has the effect of lining it up with the top of the go board.

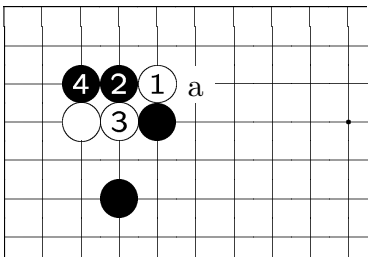


Example 3. We begin by replacing some of the stones `\0??` in the previous file with `\001`, `\002`, ... to obtain numbered Go stones ❶, ②, ... In this example, ㉕ should be at 'a.'



Example 4. Continuing from the above. A normal black stone such as ● is represented in the TEX file as `\- @+`, and a normal white stone such as ○ is represented by `\- !+`. There are a number of special symbols available used for marking the board, including the triangled stones seen in this diagram. See below for a complete list.

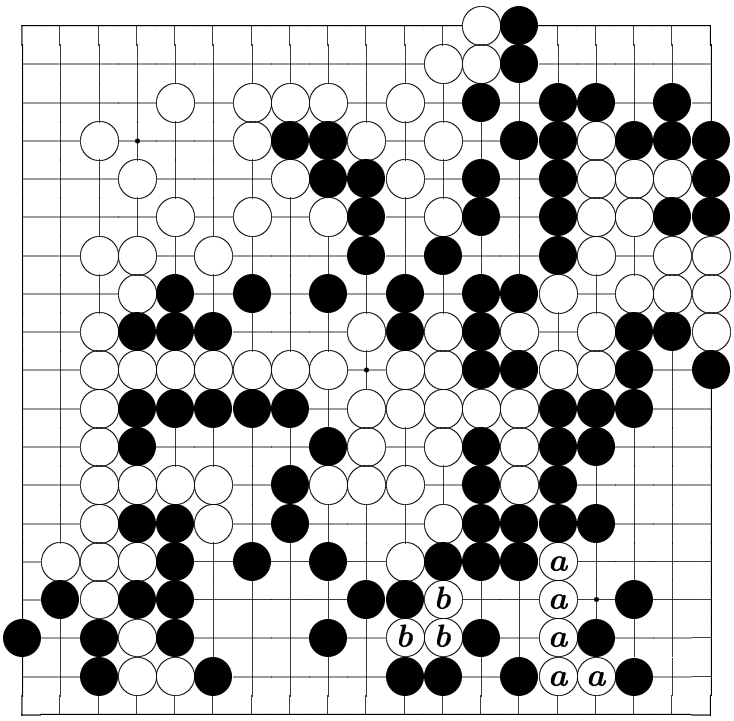
Example 5. Diagrams are handled similarly, except that we only use a portion of the template from Example 1. In this example, since White moves first, we use `\goe` instead of `\goo`. Since the box containing the diagram on the right has height $7 \times 12 = 84$ pt, we give the box containing the text on the left the same height.



Letters on Stones

Beginning with version 3.1 `sgf2dg` supports stones labelled with italic letters. If you have an older version of `sgf2dg` you will have to replace the file `gooemacs.tex`. You will also have to install a few new metafont files, namely `gobl.mf`, `gowl.mf`, `paramsc.mf` and `goxl.mf`. After executing the macro `\gool`, you can use the macros `\401=` through `\426=` to obtain the black labelled stones ***a*** through ***z***, and macros `\501=` through `\526=` to obtain the white labelled stones ***a*** through ***z***.

Thus you can produce diagrams such as this one:



Special Symbols

Special symbols include the characters for making the board (9 types of intersection plus hoshi points), circle, square, triangle, and X marks for empty intersections, and the same four marks for black and white stones. Here is a complete list of the marks, the character assignment in gooegb.mf, and the appropriate `\goo` macro invocation.

Symbol	`\gooegb`	`\goo`	Description
┌	`<`	`\0??<`	Top left corner
┬	`(`	`\0??(`	Top edge
┐	`>`	`\0??>`	Top right corner
├	`[`	`\0??[`	Left edge
┼	`+`	`\0??+`	Middle of the board
┼	`*`	`\0??*`	Hoshi point
┤	`]`	`\0??]`	Right edge
└	`,`	`\0??,`	Bottom left corner
┴	`)`	`\0??)`	Bottom edge
┘	`.`	`\0??.`	Bottom right corner
●	`@`	`\0??@`	Black stone
○	`!`	`\0??!`	White stone
○	`1`	`\0??1`	Circle
◉	`C`	`\0??C`	Black Stone, circled
◎	`c`	`\0??c`	White Stone, circled
□	`2`	`\0??2`	Square
■	`S`	`\0??S`	Black Stone, squared
▣	`s`	`\0??s`	White Stone, squared
△	`3`	`\0??3`	Triangle
◆	`:`	`\0??:`	Black Stone, triangled
◆	`T`	`\0??T`	Black Stone, triangled (copy of :)
◬	`;`	`\0??;`	White Stone, triangled
◬	`t`	`\0??t`	White Stone, triangled (copy of ;)
×	`4`	`\0??4`	X
⊗	`X`	`\0??X`	Black Stone, Xed
⊗	`x`	`\0??x`	White Stone, Xed
•	`-`	`\0??-`	Blob: erase area under label (use white ink)

Overlapping Symbols

TEX provides a method of overlapping font characters so that you can put any font symbol on your board and stones. For example, you may wish to mark a particularly sharp move: ♯.

The `Dg2TeX` converter writes the following three macros into each output file to assist with overlapping: `\goLap#1#2`, `\goWhiteInk#1`, and `\goLapWhite#1#2`. You may copy them and change them to suite your needs.

`\goLap` takes two arguments and draws the second argument on top of the first one. For black symbols on white stones, this is trivial.

Overlapping white symbols onto black stones is more problematic as not all output drivers support changing ink colors. For those that do, the `\goWhiteInk` and `\goLapWhite` macros are useful. `\goWhiteInk` uses `\special color` to change the ink to white while it prints its argument. `\goLapWhite` does the same overlap that `\goLap` does, but it uses `\goLapWhite` when printing the second (overlapping) argument so it can be seen against black stones. Since not all output drivers honor the `\special color`, you will have to experiment to see if your tool works with this kind of overlapping (`pdftex` does not support `\special color` at this time).

Invoking `\goLapWhite` as follows.

```
\textstone{\goLapWhite{\goo \0??@}{$\clubsuit$}}
```

produces this club black move: ●

Drawing white on black tends to suffer from the black overwhelming the white so that a flat black move: ● really requires larger magnification to fully appreciate. Also, any white overlap that spills out onto the empty board will not be visible as it is also white.

The `\textstone` macro is defined in `gooemacs.tex`. It adjusts the stone vertically to the proper position for inclusion in the text.

Font Design Notes

Since you have the METAFONT sources to the GOOE fonts, you are free to change the stones, and their numerals. The principal configurable files are `paramsa.mf` and `paramsb.mf`. If you change the fonts, you will need to reinstall them as described earlier in this manual. If you reinstall the fonts, you should be aware that `MakeTeXPK` generates files with names such as `gooa.360pk` etc. and caches them somewhere in your system. You will need to find and delete these cached font files, or else your changes will not show up.

The numerals on the go stones are instances of the Computer Modern Fonts which were created for TEX by Donald Knuth. The file `romandg.mf` is the same as the file `romand.mf` from the TEX distribution, with only trivial modifications to allow us to paste the numerals onto the stones.

The Computer Modern Font family depends on certain parameters, which we set in the file `paramsb.mf`. From the single METAFONT file `romand.mf`, all of the numerals in the various fonts distributed with TEX are generated, except the italic ones, which use `itald.mf` instead. Only the parameters are changed to obtain these varied fonts.

If you wish to modify the numerals on the stones, you should edit `paramsa.mf` and `paramsb.mf`, but leave `romandg.mf` untouched. To get some idea of the variety of fonts which can be created from `romand.mf`, consult Knuth, *Computer Modern Typefaces*, Volume E in the Computers and Typesetting series, Addison-Wesley 1986. In the section titled "Font Specimens" at the back of the book, the 75 standard typefaces in the Computer Modern family are presented.

In Go typography, such as may be found in any Go book or magazine, one does not find modern numerals, such as these from the Computer Modern font `cmr10`: 0123456789. Instead, one finds simplified *sans serif* fonts. These are more readable in the context of a Go diagram. We obtained good results basing the numerals on the `cmss10` font: `0123456789`. We changed four settings only, namely the font parameters `u` and `fig_height`, which control the width and height of the digits, the parameter `notch_cut`, which we found had to be increased to correct the shape of the 4 when the font was compressed horizontally, and `fine`, increasing which improves the appearance of 2 and possibly other numerals by fattening the arc at its base. Consult the initial section of Knuth, *loc. cit.*, titled "Introduction to the Parameters" for more information about the font parameters. The parameters must satisfy certain inequalities which are described there. In particular, `fine` cannot be increased past `17/36pt` unless `thin_join` is also increased.

Apart from Knuth's parameters in `paramsb.mf`, a number of configurable parameters are set in `paramsa.mf`. The parameters `oneleft`, `twoleft`, etc. control the horizontal spacing of the stones. If `setback` is set to a small nonzero number (probably no larger than 0.1) the stones will be slightly smaller than 12 points in diameter, and a portion of the board will be visible between them. This requires modification to `gooemacs.tex`. The changes needed are documented in the comments to that file.

If the parameter `smudge` is set > 0 in `paramsa.mf`, the numerals on the Black stones are expanded by `smudge` pixels in the vertical direction. This can enhance readability at the cost of some crispness. The default value `smudge=1` optimizes readability on older 300dpi laser printers.

It would be very desirable to have Type 1 versions of these fonts, since these are required for use with `pdftex`.

MetaPost

John Hobby's **MetaPost** is a program adapted from Donald Knuth's METAFONT which produces encapsulated postscript graphics from a text file. MetaPost is standard with many TEX distributions, such as TeTEX, Web2c, MikTEX and NTEX. It may be called `mpost` or `mp` on your system. Look for the manual, called `mpman.ps`.

There are two ways that you can use MetaPost to make Go diagrams. First, you can run `sgf2dg -converter Mp`. For example

```
sgf2dg -converter Mp genan-shuwa
```

will produce two files, called `genan-shuwa.mp` and `genan-shuwa.tex`. Before you run `tex`, you must run MetaPost. We will assume that MetaPost is installed on your system as `mpost` though on some systems it may be installed as `mp`. Now issue the commands

```
mpost genan-shuwa.mp
tex genan-shuwa
dvips -o genan-shuwa.ps genan-shuwa.dvi
```

These will first produces the files `genan-shuwa.1, genan-shuwa.2, ... , genan-shuwa.30`. The second command then produces the file `genan-shuwa.dvi`; finally the `dvips` invocation produces `genan-shuwa.ps`. You can print the postscript file or preview it with ghostview.

The TEX file for this manual includes the METAFONT postscript files as follows:

```
\epsffile{genan-shuwa.1}
```

Note `\input epsf` at the beginning of the `manual.tex`, which includes the `epsf` package. If you want to use `pdftex`, you will have to make pdf versions of these files using `mps2eps` and `epstopdf`, which you may include using the `miniltx` and `graphicx` packages.

Unfortunately you probably cannot successfully preview `genan-shuwa.dvi`. The reason for this is that the files `genan-shuwa.1` etc. are not true encapsulated postscript files—they do not contain the required font information. They work perfectly well with `dvips` but not other utilities such as `xdvi`. This issue is discussed at the web page of Jon Edvardsson, who describes the postscript files produced by MetaPost as *MetaPost postscript*:

```
http://www.ida.liu.se/~joned/download/mps2eps/
```

A utility called `mps2eps` by Jon Edvardsson which cleverly filters a MetaPost postscript file using `dvips` and `tex` to produce a true encapsulated postscript file. A copy of his program is included in the `sgf2mpost` directory of the distribution. If it is installed, then you can issue the shell command

```
for i in `ls genan-shuwa.[1-9]*`; do
echo ¨processing $i¨;
mps2eps $i;
mv $i.eps $i;
done
```

This replaces the MetaPost postscript files by true encapsulated postscript file. If you do this after you run `mpost` and before you run `tex`, then the `.dvi` file that you produce can be previewed using `xdvi`.

SGF2MPOST

Sgf2mpost is a tool for producing Go diagrams from SGF files. It produces first a MetaPost file, which you can run through `mpost`. The go diagrams in this document were prepared using `sgf2mpost`. The particular commands used to prepare this document are contained in `sgf2mpost/Makefile`. For example, Figure 2 is prepared with the commands:

```
sgf2mpost -i vars.sgf -o figure2.mp -r 9 -b 7 -S 0.85 -F cmbx10
mpost figure2.mp
```

This creates a file called `figure2.1`. The sgf2mpost command line options are:

```
-i <input file>
-o <output file>
-s <start move>       first NUMBERED move number or board location
-e <end move>         last move number or board location displayed
-n <num>              first numbered stone gets this numeral
-l <left column>      limit portion of board printed
-r <right column>
-t <top row>
-b <bottom row>
-L <location>:<label> (e.g. -L a:K3) mark the location with this label
-T <location>         mark this stone with a triangle
-S <scale factor>     scaling factor (default 1.0
-F <font name>        (e.g.  ¨cmr8¨) TeX font for numbers
-I <font name>        (e.g.  ¨cmti8¨) TeX font for letters
-B <font name>        TeX font for numbers > 99
-h                    print this message
-d                    print debugging traces
-v                    print version
```

The first two options are mandatory: you must specify the sgf input file and the MetaPost output file. We suggest that you give the output file the suffix `.mp`. Running it through `mpost` will produce an encapsulated postscript file with the same prefix, and the suffix `.1`.

If you have `dvips` (included in most TEX distributions), you may include the encapsulated postscript file in your your TEX document. Start your TEX file with the macro `\input epsf`. Then include the diagram with the command `\epsffile`{*filename*}. Use this document as a model.

You can use any standard TEX font in your diagrams. (Avoid cminch.) We use a variety of different fonts below to demonstrate the possibilities. See Knuth, *Computer Modern Typefaces* for a complete list.

To build `sgf2mpost`, change to the `sgf2mpost` directory and issue the command `make`.

SGF2MPOST Examples

Over the next few pages we will give examples of how `sgf2mpost` can be used. Particularly, we will give examples using different fonts. The `sgf2mpost/Makefile` contains instructions for making the figures in this document. Different fonts are used for different figures. The encapsulated postscript files are made from the `sf2mpost` directory with the command `make eps`. Study the Makefile to see how the different figures were made.

Figure 1 show a game played by GNU Go on NNGS against an opponent who chose to give GNU a 9 stone handicap. GNU Go is black, taking 9 stones.

Figure 1. GNU Go 3.3.12 (B) vs. 9 stone handicap. Moves 1–56.

After 108, W resigned, having lost two corners. Up to move 56, B made a number of strange moves, none of them disasterous.

B 8. Assuming B wants to play in this area, *a* is better.

B 28. B should pull out 24.

B 32. Better to play at 45.

B 34. Capturing is better, since the game move gives W a useful forcing move at 35.

B 38. Good.

W 47. Should be at 48. W does not take the chance B has given him, and does not get a second one.

B 56. Good. After this W collapsed in the upper left. We will examine GNU Go's analysis of this move. GNU Go values this move as worth 35.55 points; the second highest move is 'b' valued 29.00 points, and the third most highly valued move is 'c' worth 22.05 points.

GNU Go's OWL code takes 639 variations to conclude that G16 can be attacked at E14, and 409 variations to conclude that it can be defended at G13. A human player would look at far fewer variations but probably come to the same conclusion. A typical variation considered by GNU Go is given at left.

B 60. Strange shape. Simply extending to 61 would be the ordinary move.

W 63. W should simply live at 65 or 66.

W 71. Unfortunately W does not really have sente.

B 80. Good.

B 90. The top is more urgent.

W 91. W should live at 92.

B 94. Better at 99 for a simple resolution.

B 106. Better at S19 to avoid ko.

W 107. If W plays at S19, he gets a ko.

After 108, W resigns.

Figure 2. An OWL variation.

Figure 3. 57–108.
B 70 connects.

Figure 4. 1-155. 93 at 75, 96 at 90, 99 at 75 and 144 at 31.

Figure 4 shows another game. GNU Go played poorly until move 156, then came to life.

B 26. Must be at B15.

B 42. Even in isolation this is bad. B must connect at K5.

B 60. Bad tenuki.

B 80. Having played well on the bottom, B must continue at L2.

B 92. Pointless.

B 102. Due to an OWL mistake, B thinks the corner is alive. It can of course be attacked at S1. B sees that R1 threatens O1, and due to a missing pattern believes there is another half eye around O5.

B 120 and 126. Pointless moves.

By move 155 W must think he has a won game.

Figure 5. 156–168. B defends the center group by attacking.

Figure 6. 169–186. The 158–159 exchange fills a crucial liberty for W.

Figure 7. 187–202. Ignoring 187, B fights and wins a ko.

Figure 8. 203–218. 207 connects. W resigns.

Figure 9. From another game. After 1, GNU Go will capture one of the four marked strings.

Figure 10. A 13x13 game. GNU Go 3.3.12 is white. Black passes at moves 3 and 5.

No Warranty

With the exception of the files `romandg.mf` and `itallg.mf`, which are part of the Computer Modern Font sources and therefore copyrighted by Donald Knuth, all METAFONT sources and TEX macros included herewith, as well as the computer programs `sgf2dg` and `sgfsplit` are published under the GNU General Public License, version 2 (GPL). A copy of this license is contained in the file `COPYING`. If for some reason your distribution is missing this file, consult the Free Software foundation at:

`http://www.gnu.org/licenses/gpl.html`

or write to them at The Free Software Foundation, Inc., 59 Temple Place — Suite 330, Boston, MA 02111-1307, USA.

The GPL permits free redistribution of software published under its terms, provided the source code to that software, and the source to any published software derived from it is always provided or made available. Software published under the GPL comes with **absolutely no warranty**. Therefore **nothing in this release, including but not limited to the METAFONT sources and the programs `sgf2dg` and `sgfsplit` comes with any warranty whatsoever.**

Test Pattern

The purpose of this page is to confirm that the fonts are working correctly. Testing, ❶, ②, ❸, ...

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225
226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250
251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275
276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325
326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350
351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375
376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	△

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225
226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250
251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275
276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325
326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350
351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375
376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	△

Small Fonts

The purpose of this page is to test the fonts in a smaller size. These fonts are used in the two-column format. They may be smaller than you may ever need. If you need fonts at this size on a 300dpi device, you may wish to consider setting `smudge` to 1 or greater in `paramsa.mf`. (This is the current default.) On the other hand if you have a high resolution device `smudge:=0` might give a crisper font. If you change `paramsa.mf` or `paramsb.mf` you must reinstall the fonts.