PANDA-glGo manual

A 3D and 2D Goban, SGF editor, client for IGS-PandaNet and interface for GNU Go.

Peter Strempel

Coordinated by R31

Table of Contents

Introduction

PANDA-glGo is a 3D and 2D Goban, client for IGS-PandaNet and interface for GNU Go. glGo is available for Windows, Mac OS X and Linux.

glGo supports playing with a GTP engine like GNU Go.

glGo provides loading and saving of SGF game files and support for loading the UGF (PandaEgg), XML (Jago) and Ishi file formats. Standalone ugf2sgf and ishi2sgf converters are available.

You can use glGo to play and observe games online on IGS-PandaNet.

Utilizing the wxWidgets library allows native compilation under Windows, Linux and Mac OS X. This addresses the common resistance of many people against Java applications I have noticed.

The usage of OpenGL allows a high quality goban display in 3D, which greatly enhanced the plain 2D display in gGo. It is possible to rotate and move the goban in three dimensions, which is an unique feature within current Go clients.

The SDL library offers a way to render a fast and pretty 2D goban as alternative to the OpenGL 3D display.

glGo is written in C++ and Python.

For further informations and updates please visit the glGo homepage.

Chapter 1. Requirements

The goban display is available in two modes: 3D based on OpenGL and 2D based on SDL. The SDL 2D board is pretty much looking the same as the gGo/Java board. Some users might not have a graphic card with OpenGL support (but those must be very old, any consumer card sold after 1998 supports OpenGL) or prefer the classic 2D view, so the 2D mode might be an alternative. The default view is the 2D board.

If you have a proper 3D graphic card, make sure you have some recent driver from your hardware manufacturer installed. The default drivers that come with Windows do *NOT* work well.

The most common consumer cards should be from NVidia or ATI, all of them will work properly with glGo if you have a NVidia or ATI driver installed. Be careful with NVidia, there will be two drivers available for "NVidia Riva TNT" card in the Windows driver selection: One from Microsoft and one from NVidia. Use the NVidia one, the Microsoft driver is useless.

If you run Linux and don't have a hardware driver installed, the Mesa libraries might work, but will be inferior to a real driver. There are both NVidia and ATI drivers for Linux available (my old NVidia TNT 1 card works fine with the latest NVidia/Linux driver).

The OpenGL mode won't work well without a hardware driver installed. It will be slow and ugly. I am aware this is a serious limitation to the program. Most gamers have those drivers up to date, but the average IGS user is no computer gamer and certainly will have the default Microsoft Windows drivers installed, which won't work with glGo.

The glGo sound system is based on the OpenAL or SDL mixer libraries. Both are available and can be selected from the preferences dialog. OpenAL is the default system and basically superior. Both OpenAL and SDL_mixer runtime support are provided with the glGo Windows installer. On Linux you need to install OpenAL and/or SDL mixer from your distribution, it is not included in the glGo installer.

The SDL board requires the SDL, SDL_image, SDL_ttf and SDL_gfx libraries. They are all included in the Windows installer, on Windows you don't need anything additional. On Linux you require libsdl, libsdl_ttf and libsdl_image, which skip with all major Linux distributions. It does not make sense to include the SDL runtime libraries on Linux as they are all available with your distribution already.

glGo requires the Python runtime library which is included in the Windows installer. Linux users please install Python 2.4 from your distribution. To find out if and which Python version is installed type "python" in a terminal, the version is shown there.

On Mac OS X 10.3 "Panther" glGo runs out of the box. The current status of glGo on OS X 10.4 "Tiger" is unknown to me until I upgraded to Tiger myself.

Chapter 2. Installation

Table of Contents

Windows

For Windows there is a common installer. Double-click on the downloaded glGo-xxx.exe file and follow the instructions.

You need OpenGL libraries installed, they are included in the standard installation of Windows 98 and newer (not recommended) or come with your hardware driver. See section Requirements for details.

The glGo installer includes the redistributable OpenAL library published by Creative.

Tip

If you have trouble getting OpenAL sound working, you can start glGo from a DOS window with: **glGo --sdl** to use SDL sound, or **glGo --nosound** to turn sound off.

The installer includes the required SDL, Plib and Python runtime libraries.

If you want to play with GNU Go, get one of the many Windows GNU Go binaries and drop gnugo.exe into the glGo installation folder. You can download a ready-to-run GNU Go for Windows on the glGo webpage, the GNU Go homepage and from many other sites on the Internet.

Note

If you are running Windows 98 and have trouble with text scrolling in the IGS textareas, replace C:\Program Files\glGo\share\resource.xrs with resource.xrs_win98 found in the same directory. Also try changing the setting for "Force scrolling" in the preferences, see IGS tab.

Linux

Install the .rpm or .deb files like you would install any of these packages: **rpm -i glGo-xxx.rpm** or **dpkg -i glGo-xxx.deb**. If you use the .tar.gz archive, unpack it somewhere and run the glGo.install script. To uninstall, run glGo.remove.

Basically it does not matter which of the installers you use. Take the one most fitting for your distribution. They all install exactly the same files.

The user configuration is found in $HOME/.glGo. Please remove this directory manually after uninstalling.

In any case you require OpenGL libraries installed, they are installed together with your hardware driver. In worst case, install the Mesa libraries for software-only rendering, but this is not recommended

glGo on Linux uses the GTK-2 toolkit. This requires a couple of libraries available. They should be installed together with GTK-2 from your distribution. If you use the Gnome desktop, you already have all of these libraries.

You need the following SDL libraries: libsdl, libsdl_ttf and libsdl_image. They are all available with all major Linux distributions. If you don't have them already installed, get them from your distribution. In case try **ldd glGo** to check the dependencies.

You need the Python 2.4 runtime library installed (shared library). Python should be included in almost every Linux distribution.

For the sound system you need either the OpenAL or the SDL_mixer runtime libraries installed. They should be included in every Linux distribution. Unlike Windows, OpenAL is not included in the Linux glGo installer, as it is to be preferred to use your Linux distribution version. If neither libopenal nor libSDL_mixer are available, glGo should still run, but without sound output.

Here are the most important dependencies. Nothing special, but you might need to install some library from your distribution.

- libgtk 2.0
- libjpeg
- libpng 1.2
- libz
- libGL
- libGLU
- libSDL 1.2
- libSDL_image 1.2
- libSDL_ttf 2.0
- libfreetype
- libpython 2.4
- libpango 1.0
- libfontconfig

Optional for sound output:

- libopenal
- libSDL_mixer

If you want to play with GNU Go, please install it from your distribution, it is included in almost any, or compile it yourself. Make sure the gnugo binary is found in your PATH environment, so glGo will find it. Common locations are /usr/local/bin, /usr/bin or /usr/games/bin.

If you installed glGo in an unusual directory, you can use the “-s” commandline option to point to the shared data directory. Example: **glGo -s /home/foobar/myprogs/coolapps/glGo/share**. Another possibility is to set the environment variable “GLGO_SHARED_PATH”, for example in bash: **export GLGO_SHARED_PATH=/home/foobar/myprogs/coolapps/glGo/share**. The “-s” option has higher priority than the environment variable. If neither is given glGo will search for the shared directory in the folder the binary is located, /usr/share/games, /usr, /usr/local, /usr/local/share/games, /opt, $HOME, $HOME/glGo and $HOME/.local/share/glGo. If no shared directory is found, glGo will fail to start. If you keep the default directories in the installation and don't move files around later, you do not need to worry about this.

glGo support libraries (libalsound.so, libsdlsound.so and libsgfparser.so) are located in /usr/lib/games/glGo. If you really need to move this directory around, you need to tell glGo the location of these libraries. You can do this by setting the environment LD_LIBRARY_PATH to the new directory.

Autopackage

Additionally to rpm, deb and tar.gz format, glGo for Linux is available as alternative installer using Autopackage, a new multi-distribution installation system which does not rely on the distribution specific formats. Unlike rpm, deb and the tar.gz installer, Autopackage offers the possibility to install glGo as user without root access.

Execute the **glGo-xxxx.package** file in a terminal or from Nautilus/Konqueror. Depending on your current environment, a GTK, Qt or ncurses frontend will start. If this is the first autopackage you are installing on your Linux system, it will require some additional libraries. This is an automated process and only needed once. When the autopackage system has been installed, upgrading existing packages or installing new ones won't require this procedure again.

Autopackage will prompt you for the root password. If you provide the password, glGo will install into /usr. If not, glGo is installed into $HOME/.local. These are the default locations and can be overwritten using the --prefix parameter. See "glGo-xxxx.package --help" for details.

The installer will check your system for the availability of required libraries. If any are missing, please install them from your Linux distribution.

Note

Autopackage gotchas

When installing into userspace, the glGo binary will be in $HOME/.local/bin, which is by default not included into the PATH, so starting glGo would fail. Autopackage automatically appends a line to .bashrc, which adds $HOME/.local/bin to your PATH. However, on some Linux distributions .bashrc is *not* read when logging into your desktop from gdm or kdm. Hence, glGo will start from the terminal, but fail when starting from the menu entry or a desktop icon. The workaround I use on my system is to copy the line autopackage appended to .bashrc to a new file .gnomerc, which will be read when Gnome starts up. There should be some similar trick with KDE or other desktops.

To deinstall glGo, use **package remove glGo**.

Macintosh

glGo requires Mac OS X 10.3 "Panther". It does not run on OS X 10.2 and earlier.

To install glGo, mount the disk image you downloaded in Finder and copy the glGo icon to your Applications folder. If you also want the Playermanager, copy the Playermanager icon to your Applications folder as well. The Playermanager is optional and not required to run glGo. That's it!

To uninstall glGo move the glGo and Playermanager icons from the Applications folder into the Trash. You can manually delete the directory /Users/yourname/.glGo to delete the glGo configuration.

If you want to play with GNU Go, you can download a ready-to-run GNU Go for Macintosh from the glGo webpage.

What is installed where

Windows

This assumes you are using an English Windows. On my German Windows 2000 the folder names are slightly different, but generally the locations are the same.

- C:\Program Files\glGo contains the complete installation. There are subfolders containing the documentation, shared data like images, resources and translations, and the HTML help files.
- C:\Documents and Settings\username\glGo contains the configuration file glGo.rc and the logfile glGo.log. If you uninstall glGo, please delete this folder manually. Windows 9x does not know about a Home directory and writes these files into the Windows system directory.

  Tip
  If glGo fails to find the home directory and tries to write into the Windows directory, an error message will appear if this is done by a non-administrator account, as such an account cannot write into the system directory. Pointing the HOME variable to a writable folder will help.

- Some registry entries are automatically generated by the installer for the Windows software installation support. They will be completely removed during the uninstall process.

Note
To uninstall glGo, please use the provided uninstaller to clean up the registry entry properly.

Linux

- The glGo binary and the python scripts are located in /usr/games.
- /usr/share/games/glGo contains data and other shared files glGo needs at runtime.
- /usr/lib/games/glGo contains the shared sound libraries.
- /usr/share/doc/glGo contains the documentation.
- $HOME/.glGo contains the user configuration and temporary cached files.

Note

When uninstalling glGo, the user configuration directory $HOME/.glGo will be not removed, please delete it manually.

Macintosh

- glGo and Playermanager are usually located in the Applications folder.
- /Users/yourname/.glGo contains the user configuration and temporary cached files.

Note

I know this is an unusual place to store configuration files on the Mac. Usually those files should be in /Users/yourname/Library/Preferences. I plan to adjust the Mac version accordingly in a later release, but this is not really mission critical at the moment.

Chapter 3. Usage

Table of Contents

Mouse control

Playing and Editing

If you are in the SGF editor, things work like in gGo or any other SGF editor. Left-click will play stones in alternating colors. You can switch into “Edit mode” by selecting one of the SGF marker icons in the Sidebar. Depending on which mark you have selected, a left-click will place a mark and a right-click remove an existing mark. If you select stone editing mode, left-click will add a black and right-click a white stone. To remove a stone, simple click on a placed stone and it will be gone. To return to “Play mode” again select either the white or black stone icon. You can also switch the turn with this icon, if you need to add for example two white moves.

When playing a move in “Play mode” there is a shift modifier available. Usually, when you play a move that already exists as son of the current move, glGo will instead navigate to this move and not create a new node as branch of the current move. However, occasionally you might want to force the creation of a new branch, then hold down the **Shift** key while clicking the move. If the current move has no son yet, then the shift modifier is ignored.

Searching

To search for a move at a certain position, **Alt**-click or **Control**-click on the board and if there is a move played at this spot *within the main branch*, the game will go to this node.

Note

Alt-click and **Control**-click do the same. Both are available because some Linux windowmanagers already use **Alt**-click so the click would never arrive in glGo.

3D controls

The board can be rotated, shifted and zoomed using the mouse: Hold down the **Control** key and move the mouse to rotate, hold down the **Shift** key and move the mouse to shift the board. Use the mousewheel and hold down the **Shift** button to zoom the board, hold down the **Control** button to change the fovy angle.

Mousewheel

The mousewheel can be used to navigate forward or backward in a game. If the right button is pressed while the wheel is rotated, the game will cycle through the variations of the current move, if any are available.

Keyboard control

The keyboard is used to rotate, shift and zoom the 3D board and to navigate through the game.

The following table explains all supported key combinations.

Cursor right	Next move
Cursor left	Previous move
Cursor up	Next variation
Cursor down	Previous variation
Home	First move
End	Last move
Page Up	Start of variation
Page Down	Next variation
Insert (Mac: **Help**)	Go back to main branch
Control+**cursor right/left**	Rotate the board on x axis
Control+**cursor up/down**	Rotate the board on z axis
Alt+**cursor right/left**	Rotate board on y axis
Shift+**cursor right/left/up/down**	Move board right/left/up/down
Numpad plus/minus	Zoom board in/out
Plus/Minus	Change fovy angle
Backspace	Reset view

Note

This table contains all keys for the OpenGL 3D board. The 2D board has the same keys for navigation, but of course lack the rotating and zooming functions.

How to use own images and sounds

You find the sounds in the share directory within the glGo installation. On Linux this is by default /usr/share/games/glGo, on Windows C:\Program Files\glGo\share (or wherever you installed glGo). On Mac you need to right-click on the glGo icon in your Applications folder, select "Show package contents" and move to the SharedSupport folder.

In this directory you find the sound files and a file data.dat which contains the default images.

Using own sound files is pretty easy, just replace the existing sound files with your own. You need to use exactly the filenames. To replace the stone sound, you need to replace "stone.wav", etc. The sounds must be in .wav format. The current sounds were taken from CGoban2 with the permission of the author. If you have a cool sound, please send it to me if you want to share it!

To use own images, you need to create a directory data/ within the share directory and drop the images there. glGo will first search real existing files, and if they are not found use the defaults in data.dat. The images must have a certain name, format and size:

The goban kaya background	kaya.jpg	512x512
The 3D white stone texture	white_tex.jpg	64x64
The 3D white last-move marker	mark_white.jpg	64x64
The 3D black last-move marker	mark_black.jpg	64x64
The table background (2D+3D)	table.png	128x128
The SDL white stone	hyuga1.png - hyuga8.png	49x49
The SDL black stone	blk.png	49x49

Example: To replace the kaya goban background, copy an image file kaya.jpg of size 512x512 pixels into the following location:

- Windows: C:\Program Files\glGo\share\data\kaya.jpg
- Linux: /usr/share/games/glGo/data/kaya.jpg
- Mac: /Applications/glGo.app/Contents/SharedSupport/data/kaya.jpg

If you use another image format or another size, it might work or not. Probably not. The OpenGL images *must* have a size of a power of two. SDL doesn't care about.

All images are automatically resized. The table background is made from 6x6 tiles with each 128x128 pixels size.

To change the font used by the 2D board, replace FreeSans.ttf with a TrueType font file of your choice. The font does not need to be FreeSans, just the file must be named "FreeSans.ttf".

You can also use an own font file for OpenGL board coordinates and text markers by replacing the coords_font.txf file in the share directory. If you want to create those font files yourself, have a look at the gentexfont program within the GLUT distribution which allows to convert a X-server font to a .txf file. I have no idea if there is a way to create these font files on Windows. You can find them on the net, the Plib example code has a couple of those font files.

The tree display

glGo offers a tree display to get a visualization of the variations in a game, much similar to the tree in CGoban2, which I personally consider the best tree available, but that's a matter of taste. To open the tree window, select Tree in the boards View menu. glGo will remember this setting and always show the tree if you open the SGF editor. However, it will never show the tree by default for IGS or GNU Go

games, but you can still open the tree manually. I assume the tree is not too interesting for real games. glGo will also remember the size and position of the tree window.

The tree supports folding and move navigation. The most obvious way to access these action is by right-clicking on a node to open a popup menu. The popup will display the available actions. If you right-click on some empty space in the tree window, the popup will display global configuration options instead of the actions associated with one node. Currently there is only one global option available: You can select a different background color.

You can perform the same actions in the popup menu by clicking on nodes or the intersections. This should be more convenient to use.

- Click on a node to navigate to this move.
- Click on a horizontal connector (the blue line) between two nodes to fold or unfold.
- Click on a connector between a node and its branches will fold or unfold all branches after their first move (so you keep seeing the start of the variation)
- Double-click on a connector between a node and its branches will completely hide or show all variations.

If a branch is folded, the tree will still display the blue connector line, so you can see where folded moves are available. If you hide all variations of a node, it will draw a small vertical connector line to indicate this node has branches.

When adding moves or navigating in the board, the tree will automatically scroll to the correct position.

Chapter 4. Playing on IGS-Pandanet

Table of Contents

Login to IGS-Pandanet

If you don't have an account on IGS-Pandanet yet, registration is quick and easy. To configure glGo to use your account, open the IGS account dialog in menu "Connection", "Setup account". Add a new entry and enter the name and password. If you have more than one accounts listed, select the one you wish to use for the next login. When done, close the dialog and you are ready to connect to IGS-Pandanet by clicking the button showing a small world ball in the toolbar of the IGS main window.

Without an own account, you can login as guest. This is the default setting in glGo, so if you never enter you account name and password in glGo, you will enter as guest account only. As guest, you can observe games but not play yourself, so registering a real account is highly recommanded.

Observing a game

Click on the "Games" button in the toolbar of the main IGS window to display a list of all currently ongoing games. To observe a game, double-click on a line, or right-click to open a popup menu which also allows to start observing. You can watch several games at the same time.

The games list can be sorted by clicking on the column headers in the table.

To unobserve a game, simply close the board window.

Tip

When observing multiple games, the stone sounds coming from several boards might get annoying. You can turn sound off for individual boards by clicking on the sound toggle button in the boards toolbar.

Playing a game

Click on the "Players" button in the toolbar of the main IGS window to open a list of all currently online players. To play a game you need to select a proper opponent and send him a match request. After selecting the chosen player in the table and then the "Match" button a dialog will open where the game

conditions like board size and time settings can be entered. When satisfied with these conditions, click "Match" in the dialog to finally send your request. An alternative way to open the match dialog on a selected player is to right-click on a line in the table and then select "Match" from the popup menu.

Finding an opponent

Finding a proper opponent from a list of 2.000 players online can be a difficult task. glGo offers some assistance to limit the displayed players.

- Rank range

  Define an upper and lower rank. If you are a 2d player, you might try something like setting a range from 4d to 2k or similar. This greatly reduces the number of shown players in the table.

- Available toggle

  Show only players which are currently open to receive a match request. This means those players did not set themselves closed for accepting matches and are currently not playing in another game.

- Friends toggle

  Show only your Friends. You can mark players as friend by right-clicking on the line and select "Status" - "Friend" from the popup or using the Playermanager.

Receiving match requests

Unless you set yourself being "Not open for playing", once another player sends you a match request a dialog will open and inform you about the proposed game settings. If you agree to the settings, click "Accept" and the game will start. Otherwise click "Decline."

In the toolbar of the IGS main window there is a dropdown box which defines the behaviour for being on the receiving end of match requests.

- Not open for playing

  Nobody can send you a match request. You can send a match request yourself, but this will automatically switch your status to "Open for playing".

- Open for playing

  You can receive match requests.

- Looking for game

  This is similar to "Open for playing", but indicates you really want to play a game now. This increases the chance someone else asks you for a game greatly. Some clients flag these players in a special way.

glGo displays these flags in the player table in the first column. "X" means "Not open for playing", "!" means "Looking for game" and neither "X" nor "!" means "Open for playing".

Seeking a game

“Seek” is the easiest and fastest way to play a game. Click the Seek button in the toolbar of the main IGS window and a new dialog will open which allows to configure the game conditions:

- Time

  Select one of the available time settings which suits your taste best.

- Board size

  Select the desired board size. Usually this is 19x19 for most games.

- Max handicap

  Select the largest acceptable handicap you are willing to give to a weaker opponent or to take from a stronger opponent. Seek will always try to find an opponent of closest rank, but if none is available accepting a handicap will improve chances to get a game quickly.

Once the conditions are set, click "Ok" to enter your seek request. IGS-Pandanet keeps a pool of players who entered their seek request, and will select the opponent for you which fits best: Play with the same time setting, on the same board size and within the maximum handicap. When an opponent is found, the game will automatically start. If no game starts after several minutes, you might consider increasing the maximum handicap so seek can select from a bigger number of available players.

While your seek request is active, there will be a small dialog showing the configured game settings. It is possible to adjust the conditions from this dialog if you are unhappy with the previously selected ones or want to improve your chances to find an opponent by increasing the maximum handicap.

You can abort your seek request anytime by clicking the Cancel button. When a game starts, the seek request is automatically closed.

Time systems

There are two different time systems available on IGS-Pandanet: Pandanet style and Byoyomi style.

Pandanet style

This time system is also known as Canadian time. Each player starts with an initial main time. Once the initial time has run out, a player gets a fixed time for a certain number of stones to play, usually 25 stones. When those 25 stones have been played, the clock will be reset to the original value.

For example "1/10" (often seen in game annoucements in Shouts) means 1 minute initial time plus 10 minutes for 25 stones. First each player gets 1 minute main time, afterwards the clock shows 10:00 (25) meaning there are 10 minutes available to play 25 moves. Now it will go down like 09:42 (24), 09:28 (23)

etc. with each move. Once the 25th move has been played, the clock is reset to 10:00 (25). This means, you can never run out of time when you play the total of 25 stones within 10 minutes.

This style has been used on IGS-Pandanet for many years and is most common. Every client supports this time system.

Byoyomi style

Byoyomi style is quite commonly used among Japanese players and in professional tournaments.

Each player starts with an initial main time and then playes one move in a given period. After the main time has run out, you get for example 30 seconds for one move. The clocks show this as "00:30 Byo". If you play your move within 30 seconds, the clock is reset to 00:30 again.

Additionally to the byoyomi time there can be overtime periods. For example you get 5 overtime periods of each 10 seconds. Then when your byoyomi time runs out, you receive an extra 10 seconds to play this move instead of losing the game instantly. However, you get this extra shot only 5 times. When your 30 seconds byoyomi time runs out without playing your move (without overtime you'd lose now), the clock switches to 00:10 (4) meaning you are using your first of total 5 overtime periods and have 4 more periods left. If you play your move now within these 10 seconds, your clock resets to the 30 seconds byoyomi time again. The next time you don't play within 30 seconds, you use your second overtime period: 00:10 (3) - you are using your second period of total 5 and now have 3 left. If you used all your overtime periods up and again don't play within the 30 seconds byoyomi, you lose the game.

Basically Overtime can be understood as extra time allowing to exceed byoyomi time, but you get this extra time only a couple of times.

Overtime is optional, if overtime is defined as zero, no overtime is used.

Note

The Byoyomi time style was recently introduced to IGS-Pandanet and is as of writing this only supported by PANDA-Egg and glGo. This means if your opponents client does not support Byoyomi time, you can not use this system and need to use Pandanet (Canadian) time instead.

When glGo sends a match request to another player, it automatically detects if the client of that player supports Byoyomi style and presents you with a different match dialog allowing to select the time style. If the opponents client does not support Byoyomi style, glGo shows the old match dialog which only allows Pandanet style.

The PANDA-Egg client uses the term "Koryo time" instead of "Overtime".

Handicap games

In games created by the old "match" command (supported by all clients), handicap is setup *after* the game starts. Blacks first "move" is to setup the handicap stones. To do so, click the "Handicap" button in the glGo sidebar and enter the amount of handicap you wish to have. As alternative, you can enter the command "handicap 4" in the terminal window to get 4 stones for example. Only the black player can

setup the handicap. If White disagrees with the handicap, he can Undo the first black move, which was the handicap setup.

When both players are using a client which supports the new "nmatch" command (PANDA-Egg and glGo as of writing this), the handicap can be defined *before* the game starts in the match dialog. You don't need to ask your opponent which client he uses, as glGo will automatically detect if your opponents client supports the "nmatch" command and then show you the advanced match dialog.

Chapter 5. Playing with GNU Go

Table of Contents

PANDA-glGo supports connecting to a GTP engine to play with a computer program. Currently the only Go playing computer program known to me which supports GTP is GNU Go. If you are on Windows, download GNU Go from the glGo webpage or get a version from the GNU Go homepage and copy gnugo.exe into the glGo installation folder If you are on Linux, install GNU Go from your distribution or download it from the GNU Go webpage.

If the gnugo.exe file is in the same folder as the glGo executable, glGo won't have trouble finding the file. If it does, it will complain and offer the user a way to find the right file. You can also tell glGo the path to GNU Go in the preferences dialog.

You can change the GTP engine and arguments in the preferences dialog. The arguments must include "--mode gtp" for GNU Go.

Using the GTP engine

If you have the GNU Go binary in place as described above, you can select "Play with GNU Go" from the glGo start window. If GNU Go is not found, glGo will notify you. You can setup the game parameters and select which color to play. Once you hit Ok in this dialog, the game will start. Resuming games is supported, select a SGF game you want to continue playing in the GTP setup dialog.

After two passes, the "Pass" button in the sidebar changes to "Score", which allows you to manually score the game with the glGo built-in scoring tool.

GNU Go 3.6 supports resigning the game. When GNU Go resigns, glGo will notify you and the game ends.

Tip

Open the GTP console and type **estimate_score** to let GNU Go give an estimated result.

Clocks are not yet supported.

GTP console

You can open a GTP console and examine or overwrite the GNU Go commands manually. Please consult the GTP documentation for a list of available commands. You do not need this if you just want to play.

Tip

The About dialog in the Help menu of the GTP console will show you which GNU Go version you are connected to.

Closing the GTP connection

To quit a GNU Go session, select Disconnect GNU Go from the GNU Go menu. The connection to GNU Go will be closed, the GNU Go process terminated and the board will return to normal mode again.

Another way to quit a current GNU Go session is to simply close the board window in which you played the game.

Using the GNU Go score estimation

You can use GNU Go to estimate the score of a game. To do this, select Guess score in the Edit menu of a board window. This will first save the game to a temporary file. The file location depends on your operating system. Then it will start a GNU Go process with something like this: “gnugo --score estimate --quiet -L 254 -l /tmp/glGoxfCodg”. This would estimate the score at move 254 in the given temporary SGF file.

The result will be displayed in a dialog once GNU Go has estimated the score. GNU Go might need some time for this, especially if the game is still in the middle game.

Chapter 6. Options

Table of Contents

Global application settings can be customized in the preferences dialog, found in the Settings menu. The following options are currently available:

Global settings

- Language

  Define which language glGo will use. You should restart glGo after changing the language.

  "System default" will select the language depending on your current locale. If the translation is available, it will be used. If not, the default language is English.

- Board type

  Here you can select which Board type to use. Available are a 3D board using OpenGL or a 2D board using SDL. The 2D board is the default view. All new opened boards will use this type. If you accessed the dialog from a board window, this current board will not change it's type.

- Enable tooltips

  If you dislike tooltips, uncheck this to disable them globally.

- Use old SGF parser

  glGo comes with two different parsers for SGF files. The old version is slower but more stable and reliable. The new version is faster but might have problems with big and partly invalid files. It's a good idea to use the new parser because it's faster, but if you encounter problems loading a SGF file, try loading it using the old parser.

- Confirm deleting nodes

  If this is enabled, glGo will ask you for confirmation before deleting a node in the SGF editor.

- Playing moves

  Select if you want to play your moves with a single or a double click. This only applies to real games on IGS or with GNU Go, not to the SGF editor.

Sound settings

- Global sound

  If unchecked, all sounds are turned off. If you see this checkbox disabled but didn't turn it off yourself, for some reason the sound libraries failed to load. Please check the glGo logfile.

- Sound type

  You can select the OpenAL or SDL Mixer sound system here. On Windows both give about equal sound quality. On Linux I found the SDL Mixer better than OpenAL. However, this might depend very much on your sound hardware and driver, so you should just try yourself which system gives you a better result.

  On Mac OS X there is currently only SDL Mixer available.

  
  Note
  Switching from one sound system to the other might or might not work. If it does not work (glGo will tell you), exit and restart glGo, then return to the preferences dialog and enable sound again - it is disabled if the sound system change failed. Most probably the new sound system should be available now. If things still fail, start glGo from the command line and configure the sound system via the **--openal** or **--sdl** parameters.

IGS-PandaNet settings

- Show shouts in terminal

  When enabled, shouts will be displayed both in the shouts frame and in the main IGS terminal. When disabled, they will only be shown in the shouts frame.

- Skip guests in player list

  When enabled, guest accounts will not be shown in the player list.

- Display info dialogs

  When disabled, informations about observed games like results, adjourning etc. will not be shown in a dialog. These dialogs might be occasionally a bit annoying when observing multiple games, so this option offers you to turn these message boxes off. For own games (well, once they are implemented...) this option has no effect.

- Ignore rank limit for friends

  If this option is enabled, the current rank limit in the player table (for example 1d-3k or something) is ignored when you use the "Friends" filter. The idea is, you probably want to see all your friends, from 9p-NR, if you use the Friends button. However, this requires some additional work as the player list needs to be reread. This is the behavior known from gGo.

If disabled, the rank limit will also be applied when you turn on the Friends filter. This is the old behavior of glGo. It is faster as the player list does not need to be reloaded from IGS.

- Use commands in sidebar

  This allows typing command shortcuts in the say/kibitz input of the sidebar. Syntax is **#command**, **. text** and **; text**. Additionally incoming tells and channel yells will be forwarded to the game window.

  Some examples for sending tells: **#tell tweet Hi** or **. Game 123 is interesting**. Examples for sending yells: **#yell Hello.** or **; Cool game!**. Any command will work, so **#exit** will indeed send "exit" to IGS.

  Anything you type which does not start with the "#", "." or ";" letter will be treated as if this option were disabled: Depending on the game type of this window it is sent as "say" (own games) or "kibitz" (observed games).

  Tells and yells you send this way are added to a tell window, if any is open, and to the channels window. Incoming tells and channel messages will be shown in the say/kibitz textarea using the IGS notation, for example: *tweet*: Hello.

  This feature is meant for advanced users who know the IGS command syntax and want to use commands while watching or playing a game without switching to another window.

- Timer

  If enabled, glGo will send alternating the “user” and “games” commands every five minutes to IGS to prevent the server logging you out due to idle time early. Additionally the information is used to refresh the available players and games. However, this feature won't keep you online for an infinite time, when you are you idle too long, the timer will stop. But you should be online long enough to observe complete games without worrying about an early disconnection.

  This feature has one disadvantage: Your idle time in the player list will never go above 5 minutes, so to other users you appear to be present while you might be not. Keep that in mind when using this feature.

  Enabling or disabling the timer takes effect on the next connection to IGS.

- Sound for chats

  If enabled, you will be notified about incoming tells with a sound. Every chat window has an own button to enable or disable sound for this chat session individually. So if you enabled sound for chats, you can still disable it for individual chats, or vice versa.

- Sound for match requests

  When enabled you will be notified with a sound when you receive a match request.

- Force scrolling

  If enabled, glGo will force the various IGS textareas (terminal, tells, shouts, channels etc.) to jump to the last position when a new line was appended. On Windows this gives better results, on Macintosh worse results and on Linux it doesn't seem to matter. You can try changing this option if you have trouble with scrolling.

  On Windows 98 scrolling does not work well, replace C:\Program Files\glGo\share\resource.xrs with resource.xrs_win98 found in the same directory.

- Update frequency for player and games lists

  You can configure the frequency in seconds for the periodic automatic updates of the games and player tables. Unlike calling "Refresh", which will reread the complete lists using "user" respectively "games" commands, the periodic updates will merge the existing information with the messages coming from "toggle quiet false". You will notice that new games and players won't show entries in all columns, this is because these "quiet false" messages from IGS don't include the full information, so glGo cannot show what it does not know. This is particular useful if you are looking for a game, as players who just started a game will vanish from the "Available" filtered list, so you don't need to hit "Refresh" constantly.

  It is a good idea to set the frequencies in a way the players and games updates won't overlap as for example in 60 seconds and 120 seconds. The default values are 60 seconds for player updates and 130 seconds for games updates.

  To disable this feature, set the update frequency to 0 (zero) or do **toggle quiet false** on IGS to stop receiving the update messages.

- Time warning

  Here you can configure time warning for your own games. If your byoyomi time runs below the warning threshold, the clock will blink to notify you. Additionally you can configure if you want to get a warning sound.

  To disable time warning set the threshold to zero.

  There are two values available for the warning threshold: One for Pandanet style games (this is the old IGS system, also known as Canadian) and another for Byoyomi style games. The defaults are 30 seconds for Pandanet style and 10 seconds for Byoyomi style.

- Autosave

  You can let glGo automatically save all finished own and/or observed games by enabling the autosave options. You should define a target directory where the SGF files are saved to. For example, I have a directory IGS/own and IGS/observed to keep own and observed games separated. You could save both into the same directory if you want.

This feature is meant for interaction with the SGF indexing of the Playermanager to collect games associated with certain players you might be interested in.

Tip
When a game is autosaved, glGo will automatically add it to the Playermanager game index.

GNU Go settings

- Path to GNU Go

  Select the location of GNU Go on your computer.

  On Windows this would be something like C:\Program Files\gnugo\gnugo.exe, depending where you copied the GNU Go program to.

  On Linux you can keep "gnugo" if the command is available in your PATH setting, otherwise give the exact location like for example /usr/local/bin/gnugo.

- GNU Go parameters

  These parameters are sent to GNU Go or any other GTP program if you are not using GNU Go. For GNU Go it must contain "--mode gtp", otherwise GNU Go won't use the GTP mode to talk with glGo. The default parameters are "--mode gtp --quiet". More available parameters are listed in the GNU Go documentation.

See also: Playing with GNU Go

Chapter 7. OpenGL Information

Table of Contents

Textures

Textures are used for the stones and the goban. The white stones are overlayed with an image for a slight grain effect. The goban will always have a wooden texture, switching this off getting an orange board makes little sense. The stone textures can be disabled, resulting in plain black and white stones. This will improve performance, but the overall quality drops significantly. Probably in future versions disabled textures might be removed. Textures don't hurt the performance much, and they add quite a lot to the overall scene.

Textures quality

The texture quality can be set to low and high. When zooming in, the difference should be obvious. Low quality textures will appear blocky. When running in hardware mode, high quality textures do not drop performance much, so it is a good idea to use them. However, when running in software mode high quality textures are a speed killer, so for software mode it is highly recommended to switch to low quality. When the board is not zoomed in closely, the difference is not too noticeable.

Light

Reflections

Light causes the reflection effect on the black stones, which look quite similar to the stones in the Java version of gGo. When rotating the board, the reflection will slide over the stone surface relative to the light source, which is placed slightly to the upper left. Think about a lamp shining just over your head.

Shadows

Shadows significantly add to the overall 3D scene. The shadows are drawn relative to the light source. Basically they are just another light effect, but can be toggled off independent from light to improve performance in software mode while keeping the reflection effect. Shadows require light to be enabled.

You might notice that the shadows do not move relative to the light source when you rotate the board. I tried to implement this feature, but the shadows would mostly look quite weird. I am sacrificing some realism here for better quality.

Blending

Blending is an important effect. All antialias effects (see below) depend on blending enabled. Blending means that the alpha value of each pixel will be calculated when drawing the scene. When you take the border of a stone or a grid line, the pixels close to the wooden board will have a lower alpha value, so the borderline will not appear blocky, instead the edge of the black grid lines or the stones will be smooth. When blending is disabled, all antialias effects will not work. Blending is a recommended feature, however it can be quite performance costly in software mode. You need to find out if trading the performance for quality is worth it. When running in hardware mode, blending is very fast and should be enabled.

Antialiasing

You might notice the grid lines and stone edges can get blocky, which is called aliasing. The effect which prevents this is called antialiasing. glGo supports two antialias modes, a third is accessible via the configuration of your 3D driver.

Two simple but quite effective antialias settings affect the drawing of lines and stones only. This are the line and stone antialias settings in glGo. You find them in the View menu.

Tip
Blending must be enabled, else antialiasing will show no effect.

Line antialiasing

This will affect the grid on the goban and smooth the lines when rotating the board. If turned off, the lines will appear blocky. The quality of this effect depends on the capability of your graphic card.

Stone antialiasing

This will affect the board of stones to appear smooth. The effect should be obvious when toggling this on and off. This is a recommended feature which does not affect performance much and even is fast in software mode, so there is little reason to turn this off.

Scene antialiasing

Scene antialiasing will improve not only the lines and stones but the complete display. To enable this feature, turn on the Antialiasing scene menu item in glGo. You can configure if low or high quality should be used. This scene antialiasing effect can be very slow on older hardware. It only runs in hardware

mode on very new cards (hardware supported accumulation buffer, to be specific), so this is quite slow when it is not supported by the card, as the CPU has to do the work. If you have a fast computer, on low quality setting it might still be performant enough. But with lots of stones on the board and on high quality settings this eats CPU power for breakfast unless your card has the above mentioned hardware accumulation buffer. I think Geforce4 has it, but not sure.

Modern graphic cards have a feature called Multisampling, which will also smooth the overall scene. glGo will automatically turn multisampling on if it is supported by the hardware.

Tip

You need to enable full screen antialiasing (FSAA) in your 3D driver. You can access this in the 3D driver configuration dialog, for NVidia cards (sorry, no idea about the ATI and Matrox drivers) there is a slider which allows the FSAA detail setting. On Linux you can control this feature by setting an environment variable, see the documentation of your driver. Unfortunately this configuration is not accessible from the application itself. FSAA is supported by most recent graphic cards.

You might need to play a little with the settings and your 3D driver configuration to find good settings which meet your needs for quality and performance.

Tip

On a medium to low-end computer you should be Ok if you have a recent 3D driver installed (see section requirements) and enable Blending, Line and Stone antialias, Textures and Light. Keep the Scene antialias in glGo disabled. If your card supports it, enable FSAA in the driver configuration. On a high-end computer give the glGo Scene antialiasing a try and see how much it drops the performance.

Fast rendering

While rotating the board using the cursor keys, to be exact while one of the cursor keys is pressed down, some of the above mentioned display improvements are temporary disabled to speed up the rotating. While rotating the whole scene has to be redrawn quite often, so every unnecessary feature will slow this down. And while the picture is moving, not the best quality is required.

Once you release the key, the final scene will be drawn with all settings enabled again. This should make rotating more efficient while the quality drop should not be very noticeable during the movement. If you have a fast computer, you can probably keep this disabled. Try and see if this makes a significant performance difference. Once you stop rotating or moving the board, this option does nothing.

Stone quality

Stone quality defines from how many triangles the wireframe of a stone is built. Using more triangles obviously looks better and is slower. High means a stone is created from 512 triangles, low means from 128 triangles. Why triangles instead of quads? Because most graphiccards can draw triangles much faster.

Any values above 512 don't improve the look anymore but get very slow, so "High" definately means good enough.

Marks

glGo supports a mark for the last move and a couple of common SGF marks like square, triangles, letters etc.

There are two markers available for the last move mark:

Simple marks

Simple marks appear very similar to the Java version. A plain circle is drawn on the stone - actually it is drawn inside the stone to avoid odd effects when rotating the board.

The SGF marks like square, triangle, circle and cross marks are always drawn as simple marks.

Multitexture marks

Multitextures are an extension to OpenGL and is only available when running in hardware mode. Most graphic cards should support multitextures. Multitextures means, a second texture can be laid over the stone. The white stones already have the grain texture, multitexture allows to add a second texture with a mark. This makes the marks look realistic, as if painted over the stone, and not floating above or inside the stone as the simple marks do. I intentionally made a mark texture which looks "as if painted", instead of the usual plain circle or cross. I think this looks pretty realistic. There is currently one problem with these marks, when zooming in closely and rotating, the marks slide over the stone. This has to be improved.

Note

glGo will detect if your graphic card supports multitextures and enable this feature. If it is not supported, multitexture marks are automatically disabled.

Background

Select between a background image or a color. You can customize the background image by using an own image file.

Chapter 8. SDL Information

Table of Contents

SDL is a library for writing 2D and 3D crossplatform games and applications. glGo uses SDL for the 2D board display, the SDL Mixer sound system as alternative to OpenAL and a couple of supportive tasks like loading and rescaling images. The SDL 3D functions are not directly used.

The 2D SDL board offers an alternative display for users who prefer the classic top-down 2D view like known from other clients or have trouble getting the 3D display working. The 2D boards looks very similar to the gGo Java board, which is intentional, and is very fast even on old computers.

Font

There are two types of fonts available for the SDL board: A fixed 8x8 font and scaled true-type fonts. The true-type fonts look much better but render slightly slower than the fixed font. The fixed font is displayed very fast and has a good quality, but 8x8 can get too small if the board window is large. The scaled font will always have an adjusted size so the letters of SGF marks or the coordinates will fit to the stone and board size.

Background image

The 2D board display options dialog allows to select the background from a default image - the green table -, an own image or a simple color. To select an own image, click the "Select image" button and point the file selector to your image. The following formats will be accepted: BMP, PNG, JPG and GIF. The images do not need to meet the same requirements as the OpenGL files, like a square size of a power of two. Square images will work best, but other sizes will work as well. If the image is smaller than the board window, it will be automatically tiled. Selecting very large images is not recommended.

SDL Mixer Sound System

The SDL Mixer sound system can be used instead of OpenAL. On Windows OpenAL should produce a better sound quality, on Linux it very much depends on your sound hardware. You find more informations about OpenAL in the next chapter.

Chapter 9. OpenAL Sound System

glGo uses the OpenAL sound library published by Creative and NVidia. I admit it appears somewhat weird to include a full fledged sound library just to play some simple sound files. But there are some good reasons to do so.

The standard sound playing mechanism on Windows is pretty bad. It is impossible to ensure a good sound quality if the computer is busy with something else, like a running GNU Go calculating the next move in the background. So using plain Win32 API functions to play a sound is not a solution.

The situation on Linux is similar. Without the usage of an additional library it is not possible to easily access the various Linux sound devices like Alsa, Esd, Artsd, OSS etc.

High quality sound is available on Windows via the DirectSound layer, which is a part of DirectX. However, directly accessing DirectSound would mean a Windows-only solution. High quality sound on Linux is available via the native OSS kernel modules or Alsa. OpenAL serves as platform independant intermediate layer to access the native sound devices. OpenAL will use DirectSound on Windows and whatever sound device can be found or configured on Linux. This is a good solution as the application itself does not need to bother about the platform dependant low level tasks, OpenAL will take care of that.

So the usage of OpenAL for playing some simple sound is certainly overkill but justified concerning the aim of crossplatform availability of glGo.

OpenAL is already included in the Windows glGo installer. On Linux you need to install OpenAL from your distribution, it is not included in the glGo Linux version installer. Please also see the chapters Requirements and Installation of this manual.

There are commandline options to disable global sound (if you have trouble with esd or artsd on Linux) and to select the sound system (if you have trouble doing that in the preferences menu at runtime). Use **glGo -n** to disable sound, **glGo --openal** to select the OpenAL sound system and **glGo --sdl** to select the SDL Mixer sound system. These options will overwrite the saved configuration.

Tip

On Linux OpenAL will by default use the OSS kernel sound device. As alternative you can tell it to try alsa, artsd or esd by creating a file ~/.openalrc with a line like for example: “(define devices '(alsa esd native))”. This will try alsa first, esd next and OSS last. Unlike gGo/Java I managed to get xmms and glGo sound at the same time using OpenAL on esd.

OpenAL is currently not available in the glGo OS X version.

Chapter 10. The player database

Table of Contents

The playerdatabase is an extension implemented in Python and used as a sort of "plugin" (embedded python, to be exact) within glGo. It offers the friends/bozo list known from the Java gGo, the possibility to edit and assign custom flags and write a comments about a player. Additionally directories can be scanned for SGF files and all games of a certain player be looked up in the playermanager. A graph and win/loss statistics are created from this games collection.

Friends/Bozo list

There are several ways to access and edit the player database. From within glGo you can assign friend or bozo status to a player from a dialog, the playertable popup or the playerinfo dialog - just like in the Java gGo client. I do not plan to add much more to glGo itself. Instead there is a seperate application which will allow to edit and access the database in more detail, which is written in Python. Both glGo and the standalone GUI access the same database, you can edit it with the GUI while running glGo without problem as glGo will automatically detect if the database has been changed. There is also a simple Python commandline interface for the database, which might be useful for the terminal fans out there.

Flags and comment

The GUI allows you to define customized flags, like "Escaper" or "Cool dude", and then assign these flags to a player both in glGo and the playermanager. Go to "Edit flags" in the playermanager to define up to 5 flags. You will then see checkboxes for each custom flag in the playerinfo dialogs of glGo and Playermanager. Please note a flag is stored as its number (1-5) in a player, so if you for example assign flag #1 as "Escaper" and set flag #1 in a player, then change flag #1 to "Cool dude", all players formerly flagged as "Escaper" are now "Cool dudes". You can use the filter choicebox in the first playermanager tab to get all players with certain flags set listed if you want to do some cleanups.

You can write a short comment about each player, this feature can be accessed from both glGo and Playermanager.

Cleaning up the player database will remove all players which are not friend or bozo, have no flags and no comment set.

Games index

If you use the autosave feature in glGo (or have a game collection on your disk from whatever source), you can create an index of SGF games within a directory with the "Scan directory" menuitem in the playermanager. The SGF game headers are parsed and the information written in to an index file. When

you open the player info dialog in playermanager, you get a list of games with this player. Double-clicking on a table row opens the SGF in glGo if the glGo localserver mechanism is enabled.

When an own or observed game is autosaved in glGo, it will automatically be added to the Playermanager game index.

Some statistics are gathered from this games collection and displayed in the playerinfo and match dialogs in glGo. There are three values: "Games played", "My wins", "My losses." These show the total number of games you played with this certain opponent and the number of your wins and losses.

You can open a graph showing rank progress over time in the Playermanager. The x axis shows the number of days since the oldest SGF file found, the y axis the rank progression. You need to have a couple of games available for one player to get a graph.

The Cleanup games menuitem will remove all game entries from the index where no SGF file was found. This is useful if you manually deleted SGF files from a previously scanned directory.

The SGF index is in no way as sophisticated as Kombilo. The idea is simply to find all games a certain player has played. For more extensive studying I strongly recommand the usage of Kombilo, excellent program.

Installation notes

Both the Python GUI and commandline applications are bundled in the glGo installation. On Windows there are executables in the glGo installation folder and a link to the Playermanager GUI in the Start menu. On Linux you find the Python scripts in /usr/games.

These scripts included in the Linux version require Python and wxPython installed (wxPython for the GUI, commandline runs without wxPython). Most Linux users will have Python (you need it anyways to run glGo). Get wxPython from your distribution if you want to use the Playermanager GUI. You need wxPython 2.4 or above. Due to various differences between wxPython 2.4 and 2.5/2.6 the behaviour of the rank graph will not be the same. Using wxPython 2.5 or above is recommended.

The Windows glGo installer includes standalone executables which do not require Python or wxPython installed.

On Mac OS X Panther, which already comes with a full Python installation, the Playermanager will run out of the box.

Chapter 11. SGF, UGF, XML and Ishi support

Table of Contents

Supported formats

The default file format for loading and saving games is SGF FF[4], as specified here. glGo supports reading older SGF formats including the long format which is used in some old games available on the Internet. SGF files saved by glGo follow the SGF FF[4] specification and should be valid SGF files. If you stumble over a file created by glGo which does not meet the specification, please notify me so I can fix the problem.

There are two SGF parsers available at the moment: A parser written in Python and a native parser. You can switch between both in the preferences dialog, where "old parser" means the slower but more stable and mature Python code. The new native parser is much faster (loads Kogos Joseki dictionary in ~2 seconds), but will need a bit more testing and finetuning.

Note

This setting is temporary, the Python code will be removed once the new parser is solid.

glGo supports reading - but not writing - the UGF format which is commonly used in Japan and the default format of PandaEgg, the Jago XML format and the Ishi format. Writing UGF or Ishi files is not supported, writing XML might be implemented somewhen in the future.

The UGF parser works well with PandaNet mail magazine files.

The UGF and XML parsers are "real parsers", they read and load games directly from the UGF or XML files. Ishi files are first silently converted into SGF, and then glGo loads the SGF data. The Ishi format is very similar to SGF, so a lossless conversion is possible.

Converting file formats

A simple way to convert files from other formats into SGF is to load the game into glGo and save it again as SGF. However, this will become a tiresome process if you need to convert more than a few files. To help with this, glGo includes an utility to convert a batch of files from UGF or Ishi to SGF.

On Windows you can right-click on an UGF or Ishi file in Windows Explorer and select "Convert to SGF" from the popup menu. To operate on multiple files, look into the glGo installation directory, by default C:\Program Files\glGo. The converters are ugf2sgf.exe and ishi2sgf.exe. Drag and drop a single game file or a complete directory over the executable, and you will find the new .sgf files in the same location of the original .ugf or .ishi files. Finally, you can use those two utilities from a DOS command shell.

On Linux there are "ugf2sgf.py" and "ishi2sgf.py" commands which you can use from a terminal. Operation is available on single or multiple files and directories. See "ugf2sgf.py -h" or "ishi2sgf.py -h" how to use the utilities, or check the glGo SGFTools webpage for a more detailed explanation.

Chapter 12. Translations

glGo is available in English, German, Dutch, Spanish, Chinese and Czech. Thanks to Maarten Derksen for the Dutch translation, to Alberto Hernando for the Spanish version, to Exaos Lee for translating glGo into Chinese and to Chejnik for the Czech version.

If you are interested and want to offer your help to translate glGo into your language somewhen in the future, please drop me a mail.

Japanese, Chinese and Korean characters are supported in glGo out of the box on Mac OS X. On Linux and Windows you might need to install the required fonts and adjust the system locale.

Chapter 13. Known problems

Generally problems and bugs can be expected. If it crashes, don't panic. Please try to reproduce the problem and report the bug to me.

Text scrolling in the various IGS output windows is strange on Windows 98. To fix this, I included a special Windows 98 compatible resource file. Replace C:\Program Files\glGo\share\resource.xrs with resource.xrs_win98 found in the same directory. This changes all textareas to be more Win98-friendly. Maybe the installer should do this automatically, but as I lack a proper Windows 98 test system I am currently unsure about this. Also try changing the "Force scrolling" option in the IGS preferences.

When opening a board on Windows, sometimes there are display artifacts. Once the window got a bit resized or moved, the artifacts are gone. Another good idea is to enable "Autohide starter" in the preferences, this seems to solve the issue as well. This does not happen on Linux. I don't know the reason for this.

If you run a GNU Go score estimation and kill the gnugo process via task manager or killall, glGo will crash.

The captures algorithm has trouble with Kos. When loading SGF files this could produce some weird results. With observed IGS games it should be ok as then glGo won't calculate captures itself but instead use the provided list sent from IGS.

On Linux, toggling the toolbar creates some artifacts; resize the window manually a little to fix this. I don't know why this happens, it's ok on Windows.

Automatch does not display a dialog. glGo will automatically select nmatch if available, which is far superior to automatch.

The localhost server mechanism on Windows is a potential security risk if you run Windows without firewall. However, if you do such, you have much more serious things to worry about than someone hacking your Go client.

Mac note: The new SGF parser on Mac has a tendency to crash on big files like Kogo. So on Mac the old SGF parser is the default setting. You can change it in the preferences dialog. On normal files (Kogo just isn't normal) the new parser works fine.

Note

If you encounter trouble, please don't hesitate to tell me about it. What I don't know I cannot solve.

Chapter 14. Copyright

Table of Contents

Authors

PANDA-glGo is written by Peter Strempel. The artwork was created by Tweet.

Copyright and licence

Third-party software

PANDA-glGo uses several libraries which are either Free Software or released under a royality-free licence. My respect and credits go to the authors of these packages which are overall of excellent quality.

- This software is based on the OpenGL library.
- This software is based on the wxWidgets library.
- This software uses the SDL, SDL_image, SDL_ttf, SDL_mixer and SDL_gfx libraries, which are licenced under the LGPL.
- This software is based in part on the work of the Independent JPEG Group.
- The sound system is based on the OpenAL library. OpenAL is licenced under the LGPL.
- The Linux version is based on the GTK+ library which is licenced under the LGPL.
- This software is based on the Plib library. Plib is licenced under the LGPL.
- This software uses the zziplib library. zziplib is licenced under the LGPL.

  For details about the LGPL license see the file LGPL.txt in this distribution.

- glGo uses the regex library written by Henry Spencer.
- glGo uses Python.
- glGo uses Pyrex.

- The Windows installation package is created with the Nullsoft Scriptable Install System.
- The Windows Python executables are created using py2exe.
- The Linux installers are created using EPM and Autopackage.