

Thank You!

A big "thank you!" goes to everybody, who helped us to create OpenPPL and this document:

- Amstad
- Bob4567
- CakeWalk
- ChuckNeverMuck
- CoolBotMe123
- CrumbleSunday
- Chucksta
- Dr.Kaktuz
- Efu
- Fn101
- Gecko
- JConner
- Johnny
- JokeDoke
- KaeKruse
- Kyo
- Lavezzi
- Mariano268
- MerzkiyTipon
- Morello
- MrRuder
- Narfal
- Nik0
- NPNG
- PastorPoker
- Poker_Dog
- Postenjak
- Qwerty
- Rambam
- Relhby
- R2D2



- Salem
- Satanikut
- Stangelnator
- SteveG1963
- Strikles
- SuperBlak
- TheHighFish
- ThemAndMe
- Trish
- Want2Learn
- Zolp

and to everybody we have forgotten. Without your help it would not have been possible to create this manual.

Please contact us at www.maxinmontreal.com/forums, if you miss your name on the list.

What Is OpenPPL

PPL is an acronym for *Poker Programming Language*, a language to program poker bots. It was invented by <u>Shanky BonusBots</u> to ease the customization of their proprietary bot-logic.

One day an unnamed hero decided to give away his "80% working translator" to the OpenHoldem community. So OpenPPL was born and after a long period of development OpenPPL finally got completely integrated into OpenHoldem. Whereas the first version of OpenPPL needed to be translated to OH-script, OpenHoldem does now natively support plain-text OpenPPL without any translation steps. So OpenPPL now combines the best of both worlds even better: easiness of programming and mighty potential at the tables.

The Advantage Of OpenPPL

Of course OpenPPL supports the main features of PPL, especially:

- an easy English-like programming language, that can be learned in no time.
- a library of several hundred ready-to-use functions for an easy start.

Besides that OpenHoldem & OpenPPL provide some additional cool functionality that you might have missed if you have used other bots in the past:

- knowing the stacksize of every single player.
- the ability to play at any casino you want, provided you create a tablemap for it.
- the ability to develop and debug your bot with tools like ManualMode, PokerAcademy and PokerTH. Did you ever want to simulate A2o at the button or a flushdraw out of position several dozen times until you are satisfied?
- an auto-connector, that handles up to 25 tables at once with less than 1% CPU-overhead. All you have to do: open a table and sit-down.
 OpenHoldem connects automatically (one instance per table) and starts to play.
- ability to use Poker Tracker stats directly in your OpenPPL-code.

But if you are an expert you might want to make use of some advanced features:

- building symbols on your own.
- accessing Openholdem's native symbols and doing hand-range calculations on the fly.

And the best of all:

- you can contribute to the project to make it even better.
- it's all for free. Well nearly free; of course it requires a bit of learning and some work to become a master.

Welcome to the world of open source!

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. < http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License.

Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may

convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

• a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software

interchange.

- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be

resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part

of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the

covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General

Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE

COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it
Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or mo it under the terms of the GNU General Public License as publishe the Free Software Foundation, either version 3 of the License, o (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public Licens along with this program. If not, see http://www.gnu.org/licens

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html>.

Structure Of A PPL-File

If you have used OpenHoldem in the past you will know that you have to answer several questions: Shall I go allin? Shall I raise? Shall I call? And if you answer all questions with no then OpenHoldem will fold. That is one approach and it clearly has some pros. But most poker-playing people and non-programers will find a different approach more easy: *What shall I do in this situation?* And that's exactly how OpenPPL works.

Simple When Conditions With Actions

The most simple way to code a bot consists of a series of conditions followed by actions to be chosen.

WHEN HaveFlushDraw AND AmountToCall < 1/3 PotSize Call FORCE

These conditions are *always evaluated top-down*. Once the first condition is true, the appropriate action will be taken. Always! - so the order of programming matters. Let's assume, that you want to call your flushdraws, but raise to 10bb your nut-flushdraws (expert-strategy 2012). Then you will have to write your commands in the following order:

WHEN HaveNutFlushDraw RaiseTo 10 FORCE WHEN HaveFlushDraw Call FORCE

Do it the other way and your nut-flush-draw would trigger the rule for normal flush-draws. A call would be the result. As a consequence of this top-down-evaluation we recommend you deal with:

- strong hands first, special cases first
- weak hands last, general cases last

The bot simply does not know if one rule is "more special" or "more important" — you have to tell it by your coding order.

If you wonder about the keyword force: it was inherited from Shanky-PPL and means, that it overwrites the default bot (without Force). Though we don't provide a default bot and don't think, that user-defined actions should be ignored if they lack the FORCE, we kept this keyword to stay compatible and because it is nice to read (syntactical sugar).

Open-Ended When Conditions

Programming your bot with when-conditions alone will — in principle --- do the job, but there will be lots of situations that are very similar.

```
WHEN hand\$AT AND StillToAct = 2 AND Raises = 1 AND AmountToCall <= 4 WHEN hand\$AT AND StillToAct = 2 AND Raises = 1 AND AmountToCall > 4 WHEN hand\$AT AND StillToAct = 2 AND Raises = 2 ...
```

Here one part of the condition gets repeated:

```
WHEN hand$AT AND StillToAct = 2
```

For more sophisticated profiles this would be lots of code to write, lots of code to evaluate and a true nightmare to change once you want to improve it. So OpenPPL provides two kinds of conditions: top-level conditions without actions (called "open-ended when-conditions") and simple "when conditions with actions" like explained above.

Once the first open-ended-when-condition is located all following "normal" when-conditions are bound to that condition and only evaluated when the open-ended when-condition is true. So you could rewrite the example above like that:

```
WHEN hand$AT AND StillToAct = 2
   WHEN Raises = 1 AND AmountToCall <= 4 RaisePot FORCE
   WHEN Raises = 1 AND AmountToCall > 4 Fold FORCE
   WHEN Raises = 2 ...
WHEN hand$A9 AND StillToAct = 2
...
```

Each open-ended when-condition is active until the next open-ended whencondition is found. In the example above:

```
WHEN hand$A9 AND StillToAct = 2
```

To terminate all your open-ended when-conditions just write:

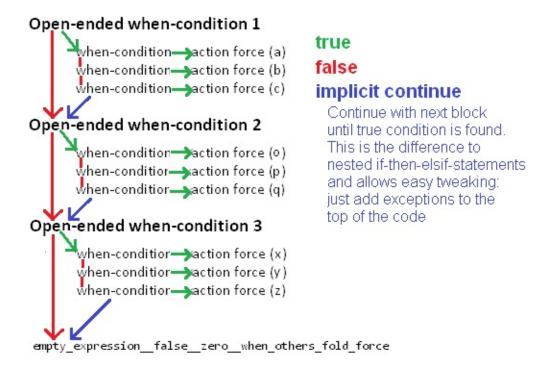
```
When Others
...
When Others Fold Force
```

Coding this way makes your code smaller, more easy to read and more easy to change. However: some people would like to take code-structuring to extremes and use multiple nested open-ended when-conditions like below:

```
WHEN hand$AT
WHEN StillToAct = 2
WHEN Raises = 1
WHEN (AmountToCall <= 4) RaisePot FORCE
WHEN (AmountToCall > 4) Fold FORCE
WHEN Raises = 2
WHEN...
```

In principle this is a good idea, but it does not work. Simply because there is no way to tell, where one open-ended when-condition ends and where the next one starts. So the semantics would be completely undefined (it is in fact not, but it is for sure not what you want). Sure you could argue about indentation, but spaces have no meaning in most programming languages (except good old Fortran 77) and everybody does it differently. So let's restate: There is *at most one level of open-ended when-conditions* (without action), each one bound to a sequence of when-conditions with actions. If you want to structure your code even more (a very good idea!) then we recommend to look at the chapter "Building Symbols On Your Own".

Controlflow of Open-Ended When-Conditions



Structure Of A PPL-File

Once you understand how when-conditions work, programming your first bot becomes easy and straightforward: you just have to provide a sequence of when-conditions for Preflop, Flop, Turn and River. These 4 main code-sections are named f\$preflop..f\$river, because that's how user-defined OpenHoldem symbols get named and from a technical point of view these code-sections are functions.

Unspecified Return Values

People who create complete profiles usually add

WHEN Others
WHEN Others Fold FORCE

to the end of every code-section. But it does not hurt if you don't so. If no condition matches the situation OpenHoldem will automatically continue to evaluate the built-in default-not (Gecko). If no default-bot is present (you may safely delete it) then OpenHoldem will evaluate functions without a return-value to zero, which is also the encoding for false, and also for check/fold.

More Advanced Coding

Coding sequences of when-conditions is very easy and intuitive, however there is one big disadvantage: poker is a somewhat complex game and there are countless situations to consider. So these code-blocks can become rather large — too large for a sane human mind. But of course there is a solution: OpenPPL supports structured coding, namely:

- user-defined hand-lists
- user-defined symbols, i.e. named functions, that get defined once and can be used at multiple places.

Both of them are very useful, but a little bit "advanced" and not standard Shanky-PPL. So we discuss them in later chapters of this manual.

Hand And Board Expressions

Hand expressions

One of the most important decisions of the game happens preflop: shall I play this hand and how? This decision can be coded with the self-explanatory hand-expression, like in the example below:

```
WHEN hand$AA RaiseMax FORCE
WHEN hand$AQSuited OR hand$AJSuited RaiseTo 3 FORCE
WHEN hand$22 OR hand$33 OR hand$44... Call FORCE
```

AK does include both AKs and AKo. So if you want to play suited hands differently you should code them first, as OpenPPL gets evaluated top down. It will stop at the first condition that matches (evaluates to true). So always remember: *strong hands first, exceptions first, bad hands later.*

Coding For Specific Suits

If you want you can also code for specific suits, like an ace of diamonds in your hand. This is rarely necessary, but was used in the past (before real randomization was introduced to Standard PPL) to randomize actions.

WHEN hand\$AcQd Or hand\$JhTs ...
// Randomize your Action, the old way
WHEN hand\$AdT RaiseTo 10 FORCE
WHEN hand\$AT Call FORCE

But be careful with the code below:

WHEN hand\$AKs...

It means: any ace and king of spades, but not AK suited

Board Expressions

A lot of the game also depends on the board cards. There are symbols like "HaveStraightDraw" or "FlushPossibleOnTurn", but for some cases might not be concrete enough. Therefore it is possible to specify board-states similar to the above:

```
// Calling, if any ace is on the board
WHEN board$A Call FORCE
// Raising, if there is a low pair on board
WHEN board$22 OR board$33 OR board$44... RaiseTo 10 FORCE
// Raising, if there are A and T of the same suit
WHEN board$AT SUITED RaisePot FORCE
```

You see, this is pretty straightforward, but sometimes lots of code to write. If you want to create code that is easy to understand, easy to reuse and easy to maintain, then you should encapsulate expressions like the second one in functions (place it within its own function):

```
##f$LowPairOnBoard##
WHEN board$22 OR board$33 OR board$44... RETURN TRUE
```

User-defined variables would also be possible, but they are not nearly as good as functions.

Document generated by <u>eLyXer 1.2.5 (2013-03-10)</u> on 2015-01-04T12:29:51.515000

Actions

The main code sections of Open-PPL consist of when-conditions with actions. They look e.g. like this:

```
##f$preflop##
// Openraising on the button
WHEN (Raises = 0 and Calls = 0 and StillToAct = 2)
    // Normal openraises
    WHEN ((Hand$AA) or (Hand$KK)...) RaiseTo 3 FORCE
    // Pushing according to Sklanky
    WHEN (StackSize < 20 and ((Hand$ ...) OR ...)) RaiseMax FORCE</pre>
```

OpenPPL provides 3 kinds of actions:

Fixed Actions

Examples look like

```
BetMax FORCE
RaiseMax FORCE
Allin FORCE
                   // synonym for BetMax and RaiseMax, but no standa
BetPot FORCE
RaisePot FORCE
BetThreeFourthPot FORCE
RaiseThreeFourthPot FORCE
BetTwoThirdPot FORCE
RaiseTwoThirdPot FORCE
BetHalfPot FORCE
RaiseHalfPot FORCE
BetThirdPot FORCE
RaiseThirdPot FORCE
BetFourthPot FORCE
RaiseFourthPot FORCE
Bet FORCE
                   // min-bet (fixed limit)
Raise FORCE
                   // min-raise (fixed limit)
BetMin FORCE
RaiseMin FORCE
Call FORCE
Check FORCE
Fold FORCE
Beep FORCE
                   // beep, but don't act
```

Contrary to standard PPL OpenPPL does not distinguish bets and raises.

OpenHoldem treats them exactly the same way.

Please note: SitOut is no longer a supported action. In our opinion it is beneficial to separate playing logic and hopper logic. Therefore OpenHoldem provides some hopper-functionality, especially the functions f\$sitin, f\$sitout, f\$leave and f\$close.

Please also note: fixed actions are functions from a technical point of view, therefore they are case-sensitive.

Actions With Fixed Betsize

Examples look like

RaiseTo 3.5 FORCE

or

RaiseBy 2.5 FORCE

RaiseTo specifies your final betsize, RaiseBy specifies the amount you want to add to the aggressors bet-amount.

Actions With Relative Betsize

Examples look like

RaiseBy 70% FORCE

Here the betsize gets measured as a certain percentage of the pot (pot = common pot + all players bets + the amount to call). Potsized bets are always RaiseBy.

Using Expressions For The Bet-Amount

Let's assume you want to raise to 3 bb in an unraised pot, plus 1 bb for each caller. With standard PPL this would result in clumsy code. But with OpenPPL you can simply write:

```
WHEN (Raises = 0) RaiseTo (3 + Calls) FORCE
```

If you want you could take this to extremes and write good code like the following:

```
##f$MyFavouriteBetsizeForDryBoards##
...
WHEN (...) RaiseTo f$MyFavouriteBetsizeForDryBoards FORCE
```

Being strict: using "FORCE"

Every action in OpenPPL has to be terminated by the keyword "FORCE". For standard PPL this keyword was optional. If you left it out, it meant, that the bot continued to evaluate and finally fell back to the default logic. We deviated from that behaviour, as:

- in our opinion it does not make any sense to specify actions that have no effect.
- OpenPPL does not provide a default bot.

Quickfolding bad hands

Most poker-sites provide pre-action-buttons to act before it is your turn. Especially useful is the prefold-button that makes it possible to click bad hands away and forget about them. Being able to click this button with your bot has at least two positive effects: more human-like behaviour and playing more hands per hour due to faster actions . Nothing could be more easy:

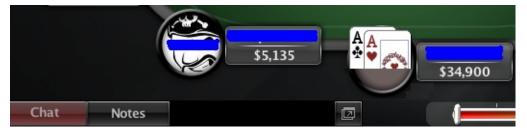
```
##f$prefold##
     WHEN AmountToCall > 0 AND (hand$32 OR hand$43 OR ...) Return
```

Be aware of potentially unstable game-states!

Please note:

• prefold gets evaluated when it is *not your turn*. Therefore it might happen that OpenHoldem takes a screenshot, evaluates and acts while the casino updates its table-display and the game-state is unstable. "Garbage in — garbage out" will happen. Now let's assume you have some super-nitty bot:

```
##f$prefold##
WHEN (NOT ((Hand$AA) OR (Hand$KK))) Fold FORCE
```



"Fold when I don't have a good hand". But here your good hand can not get recognized because your aces get only displayed partially. Your bot would fold pocket aces! To avoid problems like that it is recommend to code more fail-safe like in the first example: "Fold when I have a bad hand". It won't hurt that much if that command failed and worked only on the next heart-beat-cycle.

There are even more consequences of potentially unstable frames:
 OpenPPL does its main calculations when it is your turn. Especially some

more complex symbols which depend on the history of the game get updated only when we have stable frames to guarantee their correctness — RaisesSinceLastPlay or LastCallerPosition are examples. As a consequence these symbols will simply be undefined before your first action preflop. Therefore it is recommended to be extra careful about the GIGO-principle and stick to the most basic symbols only. But this is not too hard for prefold, is it?

Backup actions

It may happen, that a certain action is not available, e.g. you might want to raise by half the pot, but this is currently not possible, e.g. because your opponent made a large raise and half-pot would be less than the minimum. In such situations OpenPPL behaves in the following ways:

- betsize invalid (too less or too much (PotLimit or more than your balance)): betsize gets automatically adapted. This happens also to actions like RaisePot and RaiseHalfPot, that get executed with the f\$betsize-function.
- action can't be executed, because it is not possible, e.g. there might be no raise-button, because you can only call (allin) or fold. Here OpenPPL behaves in the following way:
 - RaisePot if RaiseMax is not possible
 - Raise(Min) if RaisePot or RaiseHalfPot is not possible
 - Call if no Raise is possible
 - Check if no Call is possible
 - Fold if no Check is possible (default behaviour of OpenHoldem).

This deviates from standard PPL again. However we have the following reasons:

- OpenHoldem evaluates its technical functions in the order above (for details please refer to the OpenHoldem Manual).
- we believe, it is better to behave in a more conservative way if an error occurs. Most probably you don't want to push allin if you specify RaiseHalfPot as your desired action. With a min-raise you are probably more happy (or less unhappy).

Document generated by <u>eLyXer 1.2.5 (2013-03-10)</u> on 2015-12-07T18:31:26.437000

Handlists

OpenHoldem supports preflop handlists to simplify preflop coding. You can name handlists what you like, but handlist names need to start with the word "**list**".

After that you can use your custom handlist symbol like this:

```
WHEN (Opponents = 1 AND userManiacFourBetsMe AND list007 RaiseMax F0
```

There is no limit to the number of lists you can define and you can use any name you want. Indeed it is recommended that you choose verbose names that speek for themselves.

Document generated by <u>eLyXer 1.2.5 (2013-03-10)</u> on 2014-12-12T08:54:20.061000

Mathematical Expressions

Of course OpenPPL supports arbitary complex mathematical expressions. As an example let us consider odds and outs.

```
// Calling according to odds and outs
WHEN AmountToCall / (AmountToCall + PotSize) > Outs / CardsLeft Call
```

Off course this example is a bit simplistic. It does not consider other players in the pot, implied odds on future streets, the chance to semi-bluff, etc. But I think you get the point.

The following operators are supported:

Equality Operators

Operator	Meaning	Example	Example explained
=	equal	StillToAct = 2	true, if you are on the button
!=	not equal	Position != First	true, if you are not out of position
<	smaller	AmountToCall < 5	true, if there are less than 5 big blinds to call
>	larger	PotSize > 20	true, if the pot is larger than 20 big blinds
<=	smaller or equal	AmountToCall <= 5	true, if there are less than or equal to 5 big blinds to call
>=	larger or equal	PotSize >= 20	true, if the pot contains 20 or more big blinds

Logical Operators

The logical operators "and", "or", and "not" should be pretty self-explanatory.

Operator Example

Not WHEN (HaveNothing AND OpponentsLeft >= 2 AND NOT

BotIsLastRaiser) Check FORCE

And WHEN (BotIsLastRaiser AND OpponentsLeft = 1 AND Bets = 0

And ... BetHalfPot FORCE)

XOr Meaning: either or, which is true, if exactly one of the operands is

true, but not both

Or WHEN (hand\$AA OR hand\$KK) RaiseMax FORCE

Negation (Not) has highest priority of all operators, thereafter follow And, XOr and OR in decreasing order. So if you want to write an expression like

```
WHEN AmountToCall <= 4 AND (hand$22 OR hand$33...))
```

you have to throw in some extra brackets, otherwise the bot will call with 33 any bet and that is probably not what you want. More complicated expressions sometimes lead to confusion. If you have a problem with that you might want to revisit the basics of mathematical logic and the "Laws of De Morgan".

Arithmetical Operators

OpenPPL also supports basic arithmetic. The usual rules apply of course. The percentage-operator has the same priority like multiplication and division, which is higher than addition, subtraction.

Operator	Meaning	Example
+	addition	
-	subtraction	
*	multiplication	
/	division	
%	Percentage-	WHEN (AmountToCall <= 50% PotSize) Call
	operator	FORCE
Mod	Modulus-operator	

Bitwise Operators (for Experts)

Furthermore OpenPPL supports bitwise operations that work on all single bits of bit-vectors or binary numbers simultaneously. They are useful for very low-level-stuff like detecting which chairs are seated with OpenHoldem's bitwise symbols (playersseatedbits, playersdealtbits, etc.). Most players won't ever need these symbols, so we will only give you a link to a good explanation here: http://en.wikipedia.org/wiki/Bitwise_operation

Operator

BitAnd

BitCount

BitNot

BitOr

BitXOr

User-Defined Variables

Most probably you need some game-history to take your decision; then you will find symbols like NoBettingOnFlop and OpenHoldem's history symbols and they will be useful for a good portion os use-cases, but identifying very special situations afterwards only with the built-in symbols might be hard or even impossible. So wouldn't it be helpful if you could remember what happened in the game? Of course you can - with user-defined variables. Let's take a small example.

User-defined variables need to be prefixed with the word "user", and the word Set when you give them a value.

WHEN FirstCallerPosition = 9 Set user_UTG_Was_Limping

- As you see, there's the **Set** command and after it there is a user-defined variable instead of an action after a condition, but no keyword force.
- Whenever OpenPPL / OpenHoldem sees such a construct it evaluates the when-condition. If the condition is true, OpenHoldem sets the user-variable to true and continues with the evaluation, until it finds a true condition with an action.
- All user-defined variables start with false (0). Once you set them they become true (or 1, which is technically the same).
- User-defined variables keep their value for the current hand and can't be reset back by the user. But when the hand is over they get reset automatically.
- Naming: every user-defined variable starts with the prefix "user", followed by a sequence of characters [a-zA-Z], digits [0-9] and underscores [_].
- Querying the value of a variable is simple: you can use it just like any other symbol as part of an expression:

```
##f$preflop##
WHEN Hand$AA RaiseMax FORCE
WHEN Hand$KK Set userDoesNotKnowWhatToDo
WHEN Hand$QQ Set userStartsToCry
WHEN userDoesNotKnowWhatToDo Call FORCE
WHEN userStartsToCry SitOut FORCE
WHEN Others Fold FORCE
```

Expert Tricks (Memory Symbols)

Do you need variables that can be set to any arbitrary value? Do you need variables that can be reset back? Do you need variables that don't reset automatically, but keep their values till the next hand or even for the complete session? There is a solution for it (of course). Instead of PPL-like user-variables it is also possible to use OpenHoldem's memory-store-command (following example store the decimal number 3,14..):

```
WHEN (...) Set me_st_MySecretVariable_3_141592653
and then use it later with OpenHoldem's memory-recall-command:
WHEN (... me_re_MySecretVariable ...) ...
```

This construct is a little bit more mighty, but also contains some possible pitfalls and requires extra care by the user.

- OpenHoldem's user-defined variables are case sensitive. me_st_x_1 is something different than me_st_X_1.
- No underscores allowed in the name of the variable. Underscores are used to separate name and value. (If you wonder, why underscores are allowed in simple user-variables: because the translator removes them).
- No automatic reset. If you need such a reset, you could do it e.g. before your first action preflop:

```
##f$preflop##
WHEN (BotsActionsOnThisRound = 0)
    WHEN Others Set me_st_MySecretVariable_0
    ...
```

• Table occlusion reset the memory variables.

May the force be with you!

Document generated by <u>eLyXer 1.2.5 (2013-03-10)</u> on 2015-09-29T22:00:19.015000

Building Symbols On Your Own

The philosophy of (standard) PPL is: provide poker-logical symbols, that can be used by any poker-playing person, no matter how much (or less) programming experience they have. Examples are e.g. MaxOpponentStackSize, StartingStackSize

However this approach has some drawbacks: it shifts all work to the developers and limits the users, who might need additional symbols for their bot-logic. Staying with the example above, they might need: SmallBlindStacksize, BigBlindStacksize, UTGStacksize, ..., ButtonStacksize, OpenRaiserStacksize, LimpRaiserStacksize, ThreeBetterStacksize, FlopCheckRaiserStacksize, TurnDonkerStacksize, etc. No matter how busy the development team is, they will never be able to satisfy all needs. Therefore OpenHoldem's (and partially OpenPPLs) philosophy is slightly different: provide technical symbols like balance0..balance9 (the stacksizes for chair0..chair9) and let the user figure out the rest. This way the end-user is far more flexible; however at the cost of a bit more work.

As an example we develop a symbol BigBlindStackSize. As a first step we need to know the chair of the big blind. Then as a second step we will be able to return the stacksize for this chair. To solve the first problem we use the OpenHoldem symbol ac_dealposX which returns the deal-position of chair X. The big blind is (with the very rare exception of a missing small blind) always the second player to be dealt, so we search for a the chair, that got dealt as second player.

```
##f$BigBlindChair##
WHEN ac_dealpos0 = 2 RETURN 0 FORCE
WHEN ac_dealpos1 = 2 RETURN 1 FORCE
WHEN ac_dealpos2 = 2 RETURN 2 FORCE
WHEN ac_dealpos3 = 2 RETURN 3 FORCE
WHEN ac_dealpos4 = 2 RETURN 4 FORCE
WHEN ac_dealpos5 = 2 RETURN 5 FORCE
WHEN ac_dealpos6 = 2 RETURN 6 FORCE
WHEN ac_dealpos7 = 2 RETURN 7 FORCE
WHEN ac_dealpos8 = 2 RETURN 8 FORCE
WHEN ac_dealpos9 = 2 RETURN 9 FORCE
// Other cases should not happen
WHEN Others RETURN -1 FORCE
```

Having this information we can continue with the second step (the dull part). We return the stacksize for the chair of the big blind, making use of OpenHoldem's stacksize symbols balance0..balance9. As you can see it is possible to use OpenHoldem Symbols in your OpenPPL code. Of course - we nearly forgot to mention it.

```
##f$BigBlindStacksize##
WHEN (f$BigBlindChair = 0) RETURN balance0 FORCE
WHEN (f$BigBlindChair = 1) RETURN balance1 FORCE
WHEN (f$BigBlindChair = 2) RETURN balance2 FORCE
WHEN (f$BigBlindChair = 3) RETURN balance3 FORCE
WHEN (f$BigBlindChair = 4) RETURN balance4 FORCE
WHEN (f$BigBlindChair = 5) RETURN balance5 FORCE
WHEN (f$BigBlindChair = 6) RETURN balance6 FORCE
WHEN (f$BigBlindChair = 7) RETURN balance7 FORCE
WHEN (f$BigBlindChair = 8) RETURN balance8 FORCE
WHEN (f$BigBlindChair = 9) RETURN balance9 FORCE
// Other cases should not happen.
// But if you forget about "WHEN Others"
// there always is an implicit "RETURN 0 FORCE".
WHEN Others RETURN 0 FORCE
```

You see: it is not that difficult to extend OpenPPL on your own. The possibilites are nearly endless. For the moment we skipped some details, but creating new symbols is self-explaining: each new symbol starts with a function header, that defines its name. The name of user-defined symbols traditionally begines with f\$. For example ##f\$BigBlindStacksize##. Thereafter follows the function's code, which usually is in a sequence of (optionally open-ended) when-conditions. These when-conditions usually define actions (in the case of f\$preflop... f\$river) or they contain return-statements like in the example above. That's all. OpenPPL is easy.

Advantages of Symbols

In our early versions of the manual we didn't talk about the advantages of structured code (especially functions AKA symbols), because they were too obvious for us. This caused some confusions for newbies. Symbols are great:

- to get understandable and self-documenting code (good naming; http://en.wikipedia.org/wiki/Information_hiding)
- to get reusabel code (named code-snippets)
- to get small code (no code clones)
- to get maintainable code (change and fix one location only)
- to get fast code (because of OpenHoldem's symbol-caching: evaluates only once, use the value often)
- to get readable log-files (because you see all the symbol-names and their values)
- ...

So how often should you make use of symbols? *All day. Everywhere. As much as possible. Enjoy them!*

OpenPPL Symbols

Below you find a list of OpenPPL symbols. They are part of the OpenPPL function-library, which gets loaded automatically when you load OpenHoldem. Most of these symbols work exactly the same way as their PPL counterparts. Only very few differ, mostly for technical reasons, but some of them because we think it is better that way. These differences are documented of course. You will also find some new symbols, that are not part of standard PPL; some PokerTracker and stacksize symbols for example. However: we didn't implement everything that is possible and desireable. More extensions are subject of the next chapter: "Building Symbols On Your Own". Who knows: if you come up with some good code, reasonable naming and a good description your new symbols might become part of a future OpenPPL-release.

Please note that all oPPL symbols are case sensitive

Document generated by <u>eLyXer 1.2.5 (2013-03-10)</u> on 2014-09-23T13:50:40.366000

BettingAction Symbols

Name Meaning

AllOpponentsLeftSittingOut True, if all remaining opponents are

sitting out. This symbol is especially meant for use in

tournaments

Bets The number of bets made by

opponents this betting round, can only be 0 or 1 since the 2nd bet is a

raise

BigBlindSittingOut True, if the big blind is sitting out.

This symbol is especially meant for

use in tournaments

BotCalledBeforeFlop True if we called preflop

BotCalledOnFlop True if we called on the flop
BotCalledOnTurn True if we called on the turn
BotCalledOnRiver True if we called on the river

BotIsLastRaiser True if we have the betting / raising

initiative, i.e we were the last raiser

on the previous round

BotRaisedBeforeFlop True if we raised preflop, can also

be used preflop

BotRaisedOnFlop True if we bet or raised on the flop,

can also be used on the flop

BotRaisedOnTurn True if we bet or raised on the turn,

can also be used on the turn

BotsActionsOnFlop Number of actions where we put

money in the pot on the Flop

BotsActionsOnThisRound Number of actions this betting

round where we put money in the

pot

BotsActionsOnThisRoundIncludingChecks Number of checks + number of

actions where we put money in the

pot

BotsActionsPreflop Number of actions where we put

money in the pot preflop, all-ins are not counted as the game would then

be over

BotCheckedPreflop True if bot checked preflop

BotsLastAction

BotCheckedOnFlop True if bot checked on the flop
BotCheckedOnTurn True if bot checked on the turn
BotCheckedOnRiver True if bot checked on the river

Bot's last action, can be one of the following: None, Beep, Raise, Bet,

Call, or Check

BotsLastPreflopAction Bot's last preflop action, can be one

of the following: None, Beep, Raise, Bet, Call, or Check

BotsLastFlopAction Bot's last flop action, can be one of

the following: None, Beep, Raise,

Bet, Call, or Check

BotsLastTurnAction Bot's last turn action, can be one of

the following: None, Beep, Raise,

Bet, Call, or Check

Calls The number of calls by opponents

on this betting round

CallsSinceLastRaise The number of calls by all

opponents since the last raise by an opponent on the current betting

round

Checks The number of checks made by

opponents this betting round

Folds The number of folds this betting

round

MissingSmallBlind True, if there is no small blind in

this hand, e.g. the player who would

have been SB did bust in the

previous hand.

NoBettingOnFlop True if no bets/raises were made on

the Flop, may also be used on the Flop. Bets by hero are also counted

True if no bets/raises were made on

the Turn, may also be used on the Turn, p. Bets by hero are also

counted

NoVillainBetOrRaisedOnFlop No villan bet or raised on Flop. Bets

by hero are not counted

NoVillainBetOrRaisedOnTurn No villan bet or raised on Turn. p.

NoBettingOnTurn

Bets by hero are not counted

NumberOfOpponentsAllin Number of opponents who raised or

called allin. Range: 0..9. If the amount to call is equal to our stack

size and we are headsUp we consider the villan as being allin.

NumberOfRaisesBeforeFlop The number of raises before the

Flop made by opponents

NumberOfRaisesOnFlop The number of raises on the Flop

made by opponents. Bets don't

count

NumberOfRaisesOnTurn The number of raises on the Turn

made by opponents. Bets don't

count.

OpponentCalledOnFlop An opponent called on Flop, and

did not raise or bet

OpponentCalledOnTurn An opponent called on Turn, and

did not raise or bet

OpponentIsAllin An opponent is all in, there may

still be other players in the hand. If the amount to call is equal to our stack size and we are headsUp we consider the villan as being allin.

Raises The number of raises made by

opponents this betting round. Bets

don't count.

RaisesBeforeFlop True if any opponent raised before

the flop

RaisesOnFlop True if any opponent raised on the

flop. Bets don't count.

RaisesOnTurn True if any opponent raised on the

turn. Bets don't count.

RaisesSinceLastPlay The number of raises since our last

action. Bets don't count

SmallBlindSittingOut True, if the big small is sitting out.

This symbol is especially meant for

use in tournaments

Please note: OpenPPL history counters like Raises and Calls need to get updated exactly once per orbit when we see stable input (i.e. out turn). This update gets executed after autoplayer-actions. Therefore these counters need an active autoplayer to work properly for multiple orbits.

Betsizes And Stacksizes

Standard PPL is a bit restrictive: betsizes, stacksizes and potsizes may appear only on the left or on the right side of comparison operators --- please don't ask us about the exact rules, we forgot them. In OpenPPL you can put them anywhere you like.

WHEN (AmountToCall < Pi * R * R) DANCE FORCE

It's up to you, what and how you code.

Name Meaning Limitations

AmountToCall The amount to call, None

counted in bigblinds

BetSize The number of big blinds None

bet by the last aggressor

BigBlindSize The size of the bigblind, None

usually in dollars

MaxStacksizeOfActiveOpponents The biggest stack size Could be b

(expressed in bblind and EffectiveM not dollars) of all to know ex

opponents currently playing the hand.

EffectiveMaxStacksizeOfActiveOpponents The biggest effective You some

stack size (expressed in chips/mone bblind and not dollars) of return the a all opponents currently how much playing the hand.

MaxOpponentStackSize The biggest stack of all None

opponents (playing and Depends on not playing). measured in your first a

big blinds at the

beginning of the hand

MaxStillToActStackSize The number of big blinds First action

in the stack of the

opponent with the largest stack who has not acted

yet

MinOpponentStackSize The smallest stack of all None

opponents. (playing and Depends on not playing) measured in your first a

big blinds at the

beginning of the hand

MinStillToActStackSize The biggest stack of the First action

opponents behind you (including SB and BB), measured in big blinds

OpponentStacksizeHeadsUp Stack size of opponent

when headsup. Please read Limitations

OpponentS when not v you have to

then make

None

PotSize The current pot, None

including all players bets, counted in big blinds. PotSize returns effective pot size you are participating in.

StackSize Our current balance,

counted in big blinds

StackSizeAtBeginningOfCurrentRound Our Stack Size at the Valid on ev

beginning of current betting round (StackSize

+

TotalInvestedThisRound)

StackUnknown Returns true if the

userchair is unknown,

false otherwise. Introduced for compatibility with

Shanky-PPL

StartingChips Our balance at the start None.

of a session, counted in "dollars" This symbol is especially meant for

MTTs.

StartingStackSize Our balance at the start None

of a session, counted in Depends of big blinds

big blinds your first a
TotalInvested The money put into the None

pot in this hand, counted

in big blinds

TotalInvestedThisRound Amount Invested in None

current betting round

Board Symbols

If a set is on board, "PairOnBoard" is also true; full houses do also count as sets and pairs, made flushes as flushdraws, etc. One of many reasons why you should code strong hands first.

Name Meaning

SecondTopFlopCardPairedOnRiver Synonym for

Second Top Flop Card Paired On River

SecondTopFlopCardPairedOnTurn Synonym for

Second Top Flop Card Paired On Turn

AcePresentOnFlop An ace is present on Flop

FlushOnBoard The entire board are the same suit

FlushPossible A flush is possible on the current board

(3 or more of 1 suit)

FlushPossibleOnFlop The entire Flop is one suit

FlushPossibleOnTurn A flush is possible on the Turn

FourCardsToWheelOnBoard True if a wheel can be made by using

only one hole card, i.e. if 4 cards of

A2345 are on the board

FourOfOneSuitOnTurn True if only 1 suit is/was present on the

board on the Turn

FullHouseOnBoard There is a full house on the board

HighCardOfCommonStraigh Returns the value of the highest card of a

shared straight. Especially meant to decide if we have the shared nuts (Ace)

or if we can beat the board

KingPresentOnFlop True, if at least one of the Flop-cards is a

king

LowCardsOnBoard The number of cards with the rank of 8

or lower (ace is counted as low).

Duplicates of one rank are not counted

MoreThanOneStraightPossibleOnFlop More than one straight is possible on the

Flop

MoreThanOneStraightPossibleOnTurn There is/was more than one way to make

a straight on the Turn

NutsOnBoard True if the best possible hand is on the

onboard

OneCardFlushPossible The board contains 4 or 5 cards of the

same colour

OneCardStraightFlushPossible Only one holecard is needed to make a

straightflush

OneCardStraightPossible Only one hole card is needed to make a

straight

OneCardStraightPossibleOnTurn A one card straight is/was possible on

the Turn

Only1OneCardStraightPossible only one straight can be made using only

one hole card with the current board

cards

OnlyOneStraightPossible Only one straight possible

OvercardsOnBoard The number of common cards that are

higher than the highest card in our hand

PairOnBoard There are at least 2 cards of the same

rank on the board

PairOnFlop A pair is present on the Flop

PairOnTurn The board has a pair on the Turn or on

the Flop)

QuadsOnBoard There are quads on the board QueenPresentOnFlop A queen is present on the Flop

RiverCardIsOvercardToBoard The River card is the highest ranked

common card

possible at the River

SecondTopFlopCardPairedOnRiver True, if the 2nd highest Flop card paired

on the River. If the Flop is paired it will be the lowest card. If all ranks are equal

quads at the River will make this

function true.

SecondTopFlopCardPairedOnTurn True, if the 2nd highest Flop card paired

on the Turn. If the Flop is paired it will

be the lowest card. If all ranks are equal quads at the Turn will make this function

true.

StraightFlushOnBoard Straight flush is on board StraightFlushPossible Straight flush is possible

StraightFlushPossibleByOthers A straight flush can be made by an

opponent with regards to our cards

StraightOnBoard The board contains a straight

StraightPossible Straight is possible

StraightPossibleOnFlop A stright is possible on the Flop StraightPossibleOnTurn A stright is possible on the Turn

SuitsOnBoard The number of different suits on board.

Always 0 Preflop.

SuitsOnFlop The number of different suits on the

Flop. Always 0 Preflop.

SuitsOnRiver The number of different suits on the

River. Always 0 Preflop, at the Flop and

the Turn.

SuitsOnTurn The number of different suits on the

Turn. Always 0 Preflop and at the Flop.

ThreeCardStraightOnBoard There are at least three connected cards

on the board

TopFlopCardPairedOnRiver The card with the highest rank on the

Flop paired on the River

TopFlopCardPairedOnTurn The card with the highest rank on the

Flop paired on the Turn

TripsOnBoard At least three cards of the same rank are

present on the board

TripsOnBoardOnTurn At least three cards of the same rank is

present on the Turn

TurnCardIsOvercardToBoard The Turn card is the highest ranked

common card

TurnCardPaired The card that was dealt on the Turn

paired on the River

TwoOfOneSuitPresentOnFlop True, if the Flop has / had at least 2

cards of the same suit.

TwoPairOnBoard True, if the board contains two pairs

TwoPairOnBoardOnTurn True, if the board contained two pairs on

Turn

UncoordinatedFlop True, if the Flop contains/contained no

pair on board, no possible flush, three different suits, no possible straight and no opponent could have 7 or more outs

to a straight

WheelPossible True, if a straight with A2345 is possible

HandStrength Symbols

If a set is on board, "PairOnBoard" is also true; full houses do also count as sets and pairs, made flushes as flushdraws, etc. Therefore it is a strongly recommended to code made hands first (and strong made hands at the very beginning), then weaker holding with positive potential and weaker draws at the very last.

Name Meaning

HadOverpairOnFlop Our pocketpair is/was of higher rank than

the highest ranked flop card

HadOverPairOnTurn Our pocketpair is/was of higher rank than

the highest ranked turn card

HadPairOnFlop
We had a pair on the flop
We had a pair on the turn
HadSetOnFlop
We had a Set on flop
HadSetOnTurn
We had a Set on turn

HadTopPairOnFlop
HadTopPairOnTurn
HadTwoPairOnFlop
HadSecondOverPairOnFlop
We had 2nd over pair on flop
HadSecondTopPairOnFlop
We had 2nd over pair on flop
We had 2nd over pair on flop
We had 2nd over pair on flop

HaveSecondBestKicker There is only one card that is better than

our current kicker

HaveSecondBestKickerOrBetter We have the best or second best kicker

HaveSecondBestOverPairOrBetter We have the second best overpair (KK) or

a stronger hand

HaveSecondNutFlush We have the second best flush possible We have the second best flush draw

HaveSecondNutStraight We have the second best straight possible We have a hole pair which is between the highest board card and the 2nd highest

card rank on board

HaveSecondTopPair We have the second highest pair
HaveSecondTopSet We have the second best set
HaveThirdBestKicker We have the third best kicker

HaveThirdBestKickerOrBetter We have the third best kicker or better HaveThirdBestOverPairOrBetter We have the third best overpair (QQ) or a

better hand

HaveThirdNutFlush We have the Third best flush
HaveThirdNutFlushDraw We have the Third best flushdraw

HaveThirdOverPair We have a hole pair which is between the

2nd highest board card and the 3rd highest

card rank on board

HaveThirdTopPair we have the Third highest pair
HaveThirdTopSet We have the Third highest set
HaveFourthNutFlush We have the Fourth highest flush

HaveFourthNutFlushDraw We have the Fourth highest flushdraw

HaveFourthOverPair We have a hole pair which is between the

3rd highest board card and the 4tf highest

card rank on board

HaveFourthTopPair We have the Fourth highest pair
HaveFourthTopSet We Have the Fourth highest set
HaveFifthNutFlush We have the Fifth highest flush

HaveFifthNutFlushDraw We have the Fifth highest flushdraw

HaveFifthOverPair We have a hole pair which is between the

4th highest board card and the 5th highest

card rank on board

HaveFifthTopPair We have the Fifth pair

HaveBackdoorSecondNutFlushDraw We have the 2nd highest backdoor nut

flush draw

HaveBackdoorThirdNutFlushDraw We have the Third highest backdoor nut

flush draw

HaveBackdoorFlushDraw We have a flush, flushdraw or a backdoor

flushdraw. Have BackdoorFlushdraw is

only true, if

• we contribute 2 cards

- or we contribute 1 card to the nuts
- or we contribute 1 card to the 2nd nuts

HaveBackdoorNutFlushDraw HaveBackdoorStraightDraw We have a backdoor nut flush draw True, if we need 2 cards to complete a

straight

HaveBestKicker

We have the best kicker

HaveBestKickerOrBetter

We have the best kickeBestOvr or a better

hand

HaveBestOverPairOrBetter

Have the best overpair (AA) or a better

hand

HaveBottomPair

We have a hole card that is paired with the

lowest card on board

HaveBottomSet HaveBottomTrips We have a set with the lowest board card
We have a trips with the lowest board card

HaveBottomTwoPair

We have bottom two pair

Have Double Gutshot Draw

We have a double gut shot draw

HaveFlush

We have a flush

HaveFlushDraw HaveFullHouse We have a flushdraw
We have a full house

HaveInsideNutStraightDraw

True if the bot has an inside nut straight draw. An inside nut straight draw is

defined as a hand with at least 4 'outs' to a nut straight. Unlike NutStraightDraw, outs

that make a flush possible are not

excluded

HaveInsideStraightDraw

We have an inside straight draw

HaveNothing We have nothing (no pair, overcards,

flushdraw or straightdraw)

HaveNutFlush We have the nut flush

HaveNutFlushCard We have the nut flush card We have the nut flush draw

HaveNuts We have the best hand possible at this

time

HaveNutStraight We have the best possible straight

HaveNutStraightDraw True, if we have a draw to the best

straight; this means: if we hit, no better straight is possible. E.g.87 at a board of 653. A nut straight draw requires "at least 7 outs" according to Shankys definition. Therefore straight draws get discounted if

there is a flush draw possible. Use

HaveUnDiscountedNutStraightDraw if you don't worry about possible flushes.

HaveNutStraightFlush We have the nut straight flush

HaveOpenEndedStraightDraw We have an open ended straight draw

HaveOverPair We have a pocketpair higher than any card

on the board

HavePair We have a pair, a paired board doesen't

count

HavePocketPair We have a pocket-pair like AA or 55

HaveQuads We have quads

HaveRunnerRunnerFlushAtRiver We have Flush and we made it with Turn

and River card

HaveSet We have a set, i.e. three of a kind with a

pair in the hand

HaveStraight We have a straight

HaveStraightDraw We have a straightdraw HaveStraightFlush We have a straight flush

HaveTopNonBoardPairedPair One of our hole cards is the same value as

the highest non-paired card on board

HaveTopPair One of our hole cards is paired with the

highest ranked card on the board

HaveTopSet True, if we have a set with the highest

board card

HaveTopTwoPair The two highest cards on the board are

paired with our hole cards

HaveTopTrips True if we have the best possible trips
HaveTrips We have trips, i.e. three of a kind with a

pair on the board

HaveTwoPair We have two pair, pair on board does not

count

HaveUnderPair We have a pocketpair lower than the

lowest ranked card on board

HaveUnderStraight We have the lower part of a straight

HaveWeakBackdoorStraightDraw These symbolsdescribe hands that are so

weak that we don't consider them as regular holdings, e.g. the single-card idiotend of a straight or a shared flush-draw at

the board.

HaveWeakDoubleGutshotDraw

HaveWeakFlush

HaveWeakFlushDraw

HaveWeakInsideStraightDraw

HaveWeakStraight

Have Weak Straight Draw

KingPresentOnFlop

NutFullHouseOrFourOfAKind

A king is present on the flop

This symbol evaluates the strength of quads and full houses. Top quads are always rated as 1, bottom quads or bestfull house as 2, next best hand as 3 and so on. This symbol does not take straight flushes into account. So it could return 1 even if our hand can be beaten by a

straight flush.

Standard PPL returns 0 if we don't have any quads / FH at all. However we think

this is counterintuitive and causes

problems, as a smaller number means a better hand. OpenPPL returns 999 for that

case.

Overcards The number of hole cards that are

overcards to the board

PairInHand True, if we have a pocketpair

SuitsInHand The number if unique suits in our hand

TopPairRank Rank of the hole card giving you Top Pair

(2-14 where 14 is Ace)

SecondTopPairRank
Rank of our second Top Pair
ThirdTopPairRank
Rank of our third Top Pair
ForthTopPairRank
Rank of our forth Top Pair
FifthTopPairRank
Rank of our fifth Top Pair

TopPairKickerRank Rank of the hole kicker card when you

have Top Pair (2-14 where 14 is Ace)

TripsRank Rank of our Trips

TripsKickerRank Rank of the kicker of our Trips

Other Symbols

l 	I	
Name	Meaning	Limitations
IsFinalTable	We are at the final table of a tournament	Works only with OpenHoldem 2.2.0.+ and at casinos where the final table and normal tables can be visually distinguised. See the tablemap-symbol s\$isfinaltable.
Others	Always true - mainly used for When Others Fold Force	None
Random	Returns a random number in the range [0100]	Gets evaluated new each time it gets used. So be careful if you code sequences of random actions. If you need a random function that stays constant for some time you could use the OpenHoldem symbols • randomheartbeat • randomround • randomround1randomround4 • randomhand But be careful: OpenHoldems random symbols are in the range
		random symbols are in the range [01]
	(I)E	19

Player- and OpponentSymbols

Name Meaning Limitations
HandIsHeadsup True if two people None

compete for this pot. the hand might have been more-handed

before.

LastAggressorActsAfterUs True (returns 1) if the None

last aggressor acts

after us

Opponents The number of REMARK. To know the

opponents that are number of Opponents with currently in the hand.

Does also count should use the formula

players that are allin "Opponents -

(contrary to Shanky NumberOfOpponentsAllin" who count players that

went allin in this

betting round, but not players allin from previous rounds).

OpponentsAtTable The number of None

opponents that were dealt cards this hand

OpponentsLeft Same meaning as None

Opponents but better

naming.

OpponentsOnFlop The number of None

opponents that saw the

Flop

OpponentsWithHigherStack The number of None

opponents that are seated and have higher balance than yourself OpponentsWithLowerStack The number of None

opponents that are seated and have less balance than yourself

TableIsHeadsup True if only two None

people have been dealt (cash-game) and only two people are seated (tournaments, where players who sitout also get dealt). This symbol is especially meant to detect the latest stage of a tournament, contrary to

HandIsHeadsup for

cash-games.

Poker Tracker symbols (Version 3)

Assuming all the prerequisites as described in the configuration manual are met, the following native OpenHoldem symbols will be available to your bot for use in its logic processing. The appropriate use of these symbols in opponent modeling is beyond the scope of this document, but plenty of references can be found on the Internet with a Google search.

All the symbols below are available both for cash games and for tournaments (SNG, MTT). OpenHoldem will automatically adapt the database-queries for your game-type and provide the right stats. All stats can be accessed in 3 different ways:

- for the chair of the aggressor, e.g. pt_icon_raischair. This situation is one of the most common needs for PokerTracker stats.
- for a single opponent *headsup*, e.g. pt icon headsup
- to identify certain players by their preflop-position use the postfixes *smallblind*, *bigblind*, *dealer*, *cutoff*, *user*.
- besides of that you can use *firstcaller*, *lastcaller* and *firstraiser*. They work for the current orbit only.
- by *chair number*, e.g pt_icon0..pt_icon9: this is more complex and less convenient than both methods above, but you get the ability to build any advanced stat on your own as long as you are able to identify the chair of your villain.

Summary:

- You need to use the prefix "pt_".
- Then add the symbol name.
- Finally add one of those postfix:
 - _raischair
 _headsup
 _smallblind
 _bigblind
 _dealer
 _cutoff
 _firstcaller
 - _lastcaller firstraiser
 - dealer
 - user

• the chair number between 0 and 9

General stats

□Symbol

Meaning

icon

Poker Tracker auto-rate icon code

hands

Poker Tracker number of hands that are in the database

vpip

Poker Tracker VP\$IP

pfr

Poker Tracker pre-flop raise

wtsd

Poker Tracker went to showdown

wssd

Poker Tracker won \$ at showdown

aggr_factor

Poker Tracker total aggression

aggr_factor_without_preflop

Poker Tracker total aggression excluding preflop

fold to 3bet

Poker Tracker folded while facing 3bet total

4bet

Poker Tracker Overall 4B

Preflop stats

□Symbol

Meaning

preflop_aggr_factor

Poker Tracker preflop aggression factor

preflop_rfi

Poker Tracker pre-flop raise first in

preflop_3bet

Poker Tracker 3bet preflop

preflop_fold_to_3bet

Poker Tracker folded while facing 3bet preflop

preflop_called_raise Poker Tracker pre-flop called raise preflop_attempt_steal Poker Tracker attempt to steal blinds bigblind_fold_to_steal Poker Tracker folded big blind to steal smallblind_fold_to_steal Poker Tracker folded small blind to steal preflop_3bet_vs_steal Poker Tracker 3bet vs. steal bigblind_3bet_vs_steal Poker Tracker BB 3bet vs. steal smallblind_3bet_vs_steal Poker Tracker SB 3bet vs. steal preflop_4bet Poker Tracker Preflop 4B preflop_fold_to_4bet Poker Tracker Preflop fold to 4B

Flop stats

□Symbol Meaning flop seen Poker Tracker saw flop flop_aggr_factor Poker Tracker flop aggression factor flop cbet Poker Tracker flop cbet flop_fold_to_cbet Poker Tracker folded while facing cbet flop flop_raise_cbet Poker Tracker raise flop cbet flop_fold_to_3bet Poker Tracker folded while facing 3bet flop flop checkraise Poker Tracker flop check-raise flop donkbet

Poker Tracker donk bet flop

Turn stats

□Symbol

Meaning

turn_seen

Poker Tracker saw turn

turn_aggr_factor

Poker Tracker turn aggression factor

turn_cbet

Poker Tracker turn cbet

turn_fold_to_cbet

Poker Tracker folded while facing cbet turn

turn_fold_to_3bet

Poker Tracker folded while facing 3bet turn

turn checkraise

Poker Tracker turn Check-Raise

turn_checkcall

Poker Tracker turn Check-Call

River stats

□Symbol

Meaning

river_seen

Poker Tracker saw river

river_aggr_factor

Poker Tracker river aggression factor

river_fold_to_3bet

Poker Tracker folded while facing 3bet river

river_fold_to_cbet

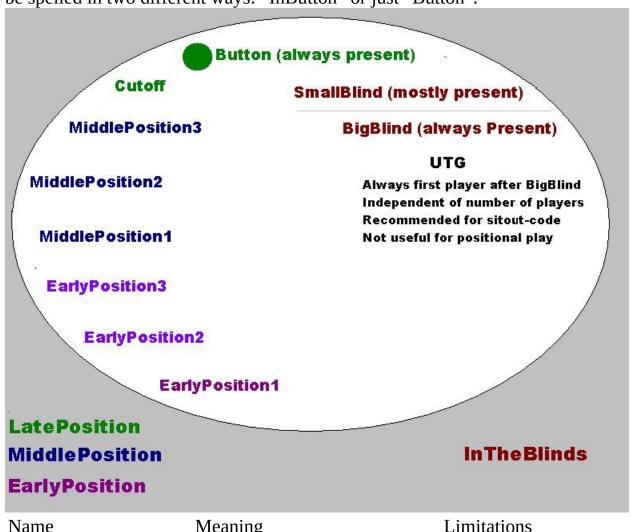
Poker Tracker folded while facing cbet river

river_bet

Poker Tracker river bet

Position Symbols

Good news: contrary to Standard-PPL most position symbols are not restricted to first orbit preflop only. You can use them in later orbits too and also postflop. Positions are always BU — CO — MP3 — MP2 — MP1 — EP3 — EP2 ---EP1 — BB — SB. If there are less than 10 players at the table, then some of the early positions will be missing (EP1 — EP2, etc.). Symbols like "InButton" can be spelled in two different ways: "InButton" or just "Button".



Meaning

Limitations

FirstCallerPosition

The position of the first caller (big blind = 0 (although this can not happen), small blind = 1, then counter-clockwise till

First orbit preflop only

UTG = 9

FirstRaiserPosition The position of the first raiser First orbit preflop only

(big blind = 0, small blind = 1,

then counter-clockwise till

UTG = 9

InBigBlind True, if you are big blind. None

OpenPPL is smart enpugh to detect a missing small blind

(e.g. he busted in a

tournament). In this case the player to the left of the dealer

will be big blind.

InButton True if you are button (last to None

act postflop)

InCutOff True, if you are CutOff (right to None

the button; this position exists only, if the game is at least 4handed, otherwise the symbol

will always be false)

InEarlyPosition True, if you are in one of the None

early positions

InEarlyPosition1 True, if you are in EP1 (left to None

the big blind, right to EP2; this position exists only, if the game is at least 10-handed, otherwise the symbol will always be

false)

InEarlyPosition2 True, if you are in EP2 (this None

position exists only, if the game is at least 9-handed, otherwise the symbol will always be

false)

InEarlyPosition3 True, if you are in EP3 (this None

position exists only, if the game is at least 8-handed, otherwise the symbol will always be

false)

InLatePosition True, if you are either CutOff None

or Button

InMiddlePosition True, if you are in one of the None

middle positions

InMiddlePosition1 True, if you are in MP1 (this None

position exists only, if the game is at least 7-handed, otherwise the symbol will always be

false)

InMiddlePosition2 True, if you are in MP2 (this None

position exists only, if the game is at least 6-handed, otherwise the symbol will always be

false)

InMiddlePosition3 True, if you are in MP3 (this None

position exists only, if the game is at least 5-handed, otherwise the symbol will always be

false)

InTheBlinds

InSmallBlind True, if you are small blind None

True, if you are either small None

blind or big blind

InUTG True, if you are *under the gun* None

(left to the big blind),

independent of the number of players at the table. This

players at the table. This symbols is escpecially useful to

sitout after the last hand of a session (before the next blind), but should not be used for positional play. Better use symbols like *InEarlyPosition1* for selection of your starting

hands.

LastCallerPosition The position of the last caller First orbit preflop only

(big blind = 0, small blind = 1,

then counter-clockwise till

UTG = 9)

LastRaiserPosition The position of the last raiser

(big blind = 0, small blind = 1,

First orbit preflop only

First orbit only, both

Preflop and post Flop

None

then counter-clockwise till

UTG = 9)

Position Our position relative to the

other player, meant especially for postflop play. There are 3

positions at the table:

Position = First

Position = Middle

Position = Last

Middle is everything, that is

neither first nor last.

StillToAct The number of opponents that

have not yet acted in the hand

when it is your turn, i.e. the players behind, including the

blinds.

The Shanky-way to determine your preflop-position, but there

are better symbols, e.g.

InButton, etc., which can also be used in later orbits and in

any betting-round.

Document generated by <u>eLyXer 1.2.5 (2013-03-10)</u> on 2015-12-01T12:34:24.625000

Technical Symbols

While building the OpenPPL library the developers had to create lots of internal supporting functions. Most of them are so technical that they are of no use for the end-user. However some of them might be helpful for people who want to extent the OpenPPL library with their own symbols. A function like SmallBlindChair might for example be useful for the development of PokerTracker symbols like PT_SmallBlind_VPIP.

Symbol	Explanation	Remarks
Chair0StartingStackSize Chair9StartingStackSize	Starting stacksize of Chair N at the beginning of the hand (balance + currentbet). Measured in bets, not in dollars.	Mainly for preflop. In other betting rounds it returns the starting stack at the beginning of that round.
BigBlindCurrentBet, SmallBlindCurrentBet	Current bets of the blind posters, measured in dollars	Mainly used to detect, if the blinds are truely raising or if they are "blind raisers".
ConstCardTwo ConstCardAce	Named card constants to improve readability and maintenability. Useful if you want to access OpenHoldem's card symbols \$\$pr0,	None

ConstBetRoundPreflop, ConstBetRoundFlop, ConstBetRoundTurn, ConstBetRoundRiver	\$\$pr1, \$\$cr0 \$\$cr4. Named constants for the four betting rounds. To be used with OpenHoldem's "betround" symbol	None
CommonCard0Paired CommonCard4Paired	True, if the rank of common card N is equal to the rank of another common card.	Postflop only
MaximumPossibleBetsizeIndollars	Maximum Possible bet size in dollars considering our current bet and balance	
PT_LastCaller_Chair	Last Caller Chair number	None
RankOfSetOnBoard	Rank of the set on board	Valid only, if there are 3 or 4 cards of the same rank, undefined otherwise
\strikeout off\uuline off\uwave offRankOfSpareCardWhenTwoPairOnBoard	\strikeout off\uuline off\uwave offreturns rank of the spare card when two pair on board on the river	River only
RankOfTopPairOnBoard	returns rank of the highest pair on board (true also if set or twopair or fullhouse on board)	Postflop only
SidePot	Pot size (in number of blinds) we are not	None

SidePotInDollars

EarlyPosition1Chair ... MiddlePosition3Chair, CutOffChair, ButtonChair, SmallBlindChair, BigBlindChair

AggressorChair

participating in due to opponent's bet being bigger than the sum of our balance plus currentbet. PotSize returns effective pot size because SidePot amount is subtracted

from it.

Pot size (in dollars) we None are not partecipating in due to opponent's bet being bigger than the sum of our balance plus currentbet

Chair numbers of the None specific players. Could be used e.g. for symbols like PT SmallBlind VPIP

Same meaning as None ac_aggressor, but more reliable. ac aggressor alone is somewhat unreliable, as it gets scraped (by bets). But if we raise and there are super-fast calls behind us (e.g. at PokerAcademy), then we don't have stable frames and soon we will switch to the next betting round, so OpenHoldem might miss the aggressor. This symbol also considers our last

	action and the amount to call to overcome this problem.	5
SmallBlindRaising ButtonRaising	True, if the chair in question is betting or raising. Contrary to OpenHoldem's conventions "blind raisers" are not counted preflop; only if they truely raise.	Meant for preflop, also working postflop, though it counts betters too.

Document generated by <u>eLyXer 1.2.5 (2013-03-10)</u> on 2014-12-12T11:48:15.587000

Differences To Shanky And Old-Style-PPL

If you are a botter who used Shanky-PPL in the past there are some differences you should know about:

Plain Text Only

The deciphering of encypted PPL is a business-secret of Shanky BonusBots. We don't know and will never support this data-format. Our aim is to build an open-source botting-tool and not a marketing-platform for shady salesmen. Therefore OpenPPL is plain text only — for profiles you own legally, or even better: for profiles you build on your own.

No Option Settings

Standard Shanky-PPL supports option settings to configure their built-in-default bot; at least that was the old way to do so, but nowadays most of these settings get overwritten by the profile and have no effect at all. OpenHoldems built-in Gecko-bot does not support any Shanky-opttions; they have no effect and simply get ignored.

```
MakePotSizedPreFlopRaisesWhen = 6
FoldToPre-FlopRaisesForAQAJsKQ = 0FF
FoldPost-FlopBelowTopPairToAnyBetOrRaise = 0FF
FoldToPost-FlopRaisesWithUnpairedBoardsFor = 2
...
custom
preflop
...
```

No Keyword Custom

As no built-in default bot exists the whole bot-logic is "custom". Sure, we could ignore that superfluous keyword, however it is an indication of old-style or Shanky-style (Open)PPL. There will be more things "wrong" for sure, so we prefer to show you a warning as early as possible.

Preflop, Flop, Turn and River Sections

The four main code-sections are functions (technically speaking), therefore they have OH-script-style function-header-syntax:

```
##f$preflop##
    WHEN ... RaiseMax FORCE
    ...
##f$flop##
##f$turn##
##f$river##
```

At least the f\$preflop-section must be present for OpenHoldem to switch to OpenPPL-mode.

Floating Point Numbers

There are some differences between Standard PPL and OpenPPL you should care about — luckily only very few:

• Standard PPL uses integers everywhere, whereas OpenPPL uses floating point numbers. An example: in Standard PPL you could write code like:

```
WHEN Stacksize = 1 AND ... RaiseBy 1 FORCE
WHEN Stacksize = 2 AND
or even
WHEN AmountToCall = 70% Stacksize
```

which will cause troubles with OpenHoldem, because OpenHolden uses real numbers like 31.41. Therefore it is recommended to use inequality-operators, for example like below:

```
WHEN Stacksize < 1.5 AND ... RaiseBy 1 FORCE WHEN Stacksize < 2.5 AND
```

Or you could even use the new approximately-equal-operator:

```
WHEN Stacksize ~~ 3 RaiseMax FORCE
```

• Standard PPL's symbol NutFullHouseOrFourOfAKind uses lower numbers for better full houses or quads, but 0 for no full house at all. This is somewhat inconsistent and OpenPPL uses a high number (999) for that case.

Hand And Board Expressions

OpenHoldem's formula engine uses floating-point-numbers internally and there is no easy way to represent a board of cards as a single floating-point-number. There could even be various different board-expressions at the right side of an equality comparator that are all true.

```
WHEN Board = AT ...
WHEN Board = ATSuited...
```

Therefore we don't support native Shanky-syntax here but use parameterized symbols

```
WHEN board$AT ...
WHEN board$ATSuited ...
```

Cases Matter (Partially)

OpenHoldem is case-sensitive — contrary to standard PPL. Therefore the case used (upper and/or lower) matters. The rules for your OpenPPL code are more user-friendly and less rigid. Three areas where case matters:

• Keywords like "WHEN", FORCE and operators like OR etc. For keywords, three different formats are allowed: all upper-case, all lower-case, or a reasonable mix.

```
WHEN (Hand$AA) RaiseMax FORCE
When (Hand$AA) RaiseMax Force
when (Hand$AA) RaiseMax force
```

For better readability we recommend to use only upper-case for keywords.

• OpenHoldem symbols. Native OpenHoldem symbols are case-sensitive and always lower-case.

```
WHEN (pt_vpip0 < 0.10) Fold FORCE
WHEN (balance0 < 50) RaiseMax FORCE
```

However: you will probably use these symbols very rarely in your code. Probably only to create poker-logical symbols like PT_OpenRaiser_VPIP and BigBlindStackSize.

• OpenPPL-symbols that are part of the library. They are case-sensitive OH-script-functions with mixed upper and lower case letters.

```
WHEN (StillToAct < 2) RAISETO 3 FORCE
```

The naming is pretty intuitive, but the function that generates errormessages is smart enough to look for similar named symbols and will show you a helpful warning if something is wrong.

False Friends

Again: case-sensitivity matters. There are some (only three) symbols with the "same" name that have a different meaning for OH-script and OpenPPL.

Symbol Meaning

bet The minimum bet for the current bet-round, measured

in dollars

Bet The action min-bet (equivalent to min-raise)

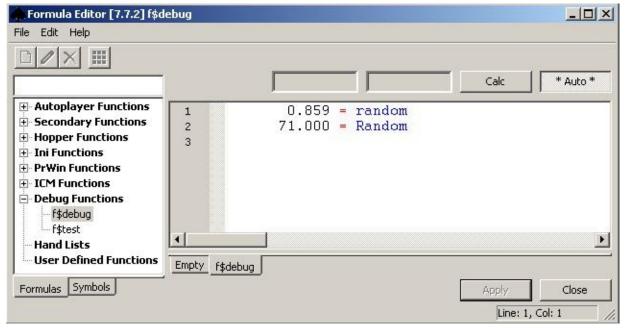
call The amount you need to call, measured in dollars

Call The action "Call".

random Generates numbers in the range [0..1)

Random Generates integer numbers in the range [0..99]

You probably won't need the OH-script symbols, but at least the library of OpenPPL-functions uses them internally, so they have to be there. For the first two cases OpenHoldem's parser can detect if you took an identifier instead of an action and will warn you about that. But at the moment you need to take care about the latter case.



Keyword "Set" For User-Defined Variables

Long story short: Shanky-PPL is an easy, English-like language that is intuitive to use, but it contains some technical flaws. For example, it can't be parsed with a one-token-look-ahead like all other modern programming languages. This complicates the parser and especially the generation of good, helpful errormessages. Therefore we had to deviate at some points a liitle bit. The most conspicuous point is the new keyword "Set" before a user-variable:

WHEN ... Set user_utg_limp_raised_preflop

No Shanky-Style Delay

```
preflop
WHEN ... RaiseBy 3 Delay 5 FORCE
```

A user-defined delay after an action simply does not fit the concept of functions, that traditionally return only a single value. Besides that: we believe that it is very clumsy to have thousands of lines with a fixed delay. OpenHoldem natively supports a solution that is way better suited for this use-case: a f\$delay-functions.

It requires only some (or some dozen) lines of code for perfect randomized delays, depending e.g. on board-texture, betting-actions, etc.

```
##f$delay##
WHEN UncoordinatedFlop AND Random < 10 RETURN 2500 FORCE
WHEN ...</pre>
```

f\$sitout Function

For similar reasons OpenPPL does no longer support a SitOut-command. We prefer to separate the playing logic from hopper-logic. Therefore OpenHoldem has a f\$sitout-function. Furthermore OpenHoldem supports f\$sitin, f\$autopost, f\$leave, f\$close and some more.

```
##f$InUTG##
    WHEN (dealposition = 3) RETURN True FORCE

##f$sitout##
    // issittingin, handsplayed and floppct are OpenHoldem symbols
    // that can be used like any other OpenPPL symbol.
    // We wait until the orbit is finished and sitout
    // before we have to post the big blind again.

WHEN (issittingin
    AND handsplayed > 15
    AND floppct < 0.20
    AND HaveNoCards
    AND InUTG)
    RETURN True Force</pre>
```

RaiseBy And RaiseTo Actions

Let's assume the following situation: you sit in the big blind and had to pay \$10. Everybody folds to the button who raises to \$30. Now it is your turn again and you decide to enter "90" into the raise-box. What does this mean? Well, it depends on the casino.

- at some casinos \$90 will be your final betsize (RaiseTo, betsizeinterpretationmethod = 3)
- other casinos will add \$90 to the \$10 you already posted, so the final betsize will be \$100 (betsizeinterpretationmethod = 2).
- and finally some other casinos will add these \$90 to the \$30 of the last raiser, so the final betsize will be \$120 (RaiseBy, betsizeinterpretationmethod = 1)

How does Shanky handle this Babylonian confusion?

In one case the small blind is counted as part of the pot and in the other it is not. This can vary between poker rooms as well. You just have to experiment to get it where you like and save each profile the way you want it.

(Egor at http://bonusbots.com/support/index.php/topic,7934.msg79372.html#msg79372

However we don't like undefined behaviour and we don't want to keep multiple versions of our bots for different casinos either. Therefore we introduced 2 new commands: RaiseTo and RaiseBy. This way you can clearly specify what behaviour you want. You only have to specify the correct betsizeinterpretationmethod in your tablemap to tell OpenHoldem how your casino behaves and OpenHoldem will care about all the rest.automatically. *That's how it should be.*

Gecko, Our Default Bot

The Gecko story

We once had an old-school-member called Gecko from the Netherlands. Gecko had no experience in the IT-business, but he was a really dedicated guy; a hard worker and fast learner. Gecko worked round the clock, contributed 500 posts per month to the forum, reported bugs Sunday morning 4 am. Over the course of the next three years he rewrote his bot at least 20 times from scratch, each time structuring it a bit better and making it stronger. Hard work pays off and so Gecko finally got a winning bot that made him a little fortune. Gecko traded his well-earned botting-money for six renovated teeths, a twelf-man internet-advertising-company in Pakistan and freedom from the Dutch police.

One day Gecko decided to visit his employees in Pakistan. He took his \$200-car (bought from all the money the Dutch police left him), found the way through the snowy mountains of Austria and sunny Greece, missed Pakistan by some miles but discovered Thailand where he now lives in happiness.

Gecko finally quit botting; but as a true gentleman he generously donated his pokerbot to the OpenHoldem community.

Its playing style

Lots of people used the Gecko-bot as a base for their own pokerlogic. Everybody praised its coding-style and its good play. Gecko plays no-limit big-stack in a tight-aggressive (or maybe semi-loose-aggressive) way. It uses PokerTracker to adapt to its opponents, stealing more against tight blind-posters, value-betting harder against calling-stations, being more cautious if a passive player gives action and more. For example Gecko is prepared to play for full stacks with mid-pair against hyper-maniacs. The Gecko-bot is by far the best bot available to the public we have ever seen (demo-bots) or heard of (commercials). It tries to squeeze the slightest edge in an aggressive manner; so be prepared for some funny swings, hopefully more often in the right direction.

Does Gecko play tournaments?

To be honest: not really. Gecko is designed to play manly bigstack and squeeze the slightest edge in an aggressive manner. But in tournaments chip-EV and money-EV usually differ (except for winner-takes-it-all); tournaments usually get played for survival and they often end with a shortstacked push-fold-phase. Nothing Gecko is really designed for. The Gecko-bot probably needs a little tweak on top of it that tempers its aggression here and there a little bit and cares about ICM.

Using the Gecko-bot

The Gecko-bot is now fully integrated and OpenHoldems default-bot. It sits in the background and waits for situations where your bot-logic does not specify an action. Then Gecko steps in. You can use Gecko in several ways:

- play pure Gecko. Whenever your bot-logic is empty or not loaded or you click "New" then Gecko will care about everything and play all games from the beginning to the end. In other words: forget to load your bot-logic and increase your winnings.
- tweak Gecko. If Gecko satisfies you only 99% or if you want to adapt it to other game-types like tournaments, then you can easily tweak it. Don't worry: you don't have to change anything in Geckos code and possibly ruin it. All you have to do: create a new file with the exceptional situations that you want to play differently and leave the rest unspecified. Then OpenHoldem will play your bot-logic and Gecko will care about all the rest. We call this type of bot-logic a *Gecko-tweak*.
- use Gecko as a starting-point.for your own complete bot. Even if you want to create a bot completely from scratch Gecko will be useful. You can start your bot-logic as a Gecko-tweak and as long as your logic is incomplete Gecko will care about the forgotten situations. So you can for example test your half-made bot as if it was already finished.
- don't use Gecko at all. If you are one of the fortunate old-school-members who already have a complete and better bot, then you don't have to worry that Gecko ruins anything. You can safely delete the Gecko-file from the botlogic folder or just terminate all evaluations with

WHEN Others
WHEN Others Fold FORCE

Debugging Your Code

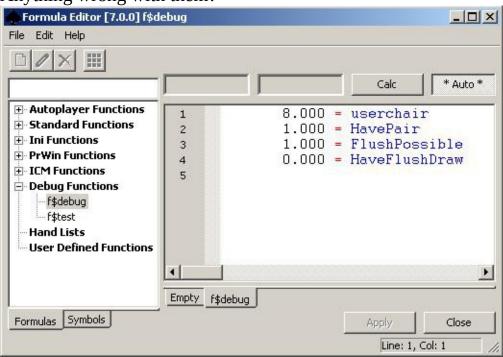
Why debugging? If you code your bot or if you watch it play you will for sure find situations where the play doesn't match your expectations. Sometimes you know immediatelly what you did wrong or at what place of the code you have to look for your error. But sometimes you don't know or it might even seem, that everything looks ok on your side. So you might want to look a little bit deeper - at your code and maybe even at the symbol library. Here in this chapter we will show you some techniques to locate the problem:

Working with the debug-tab

OpenHoldem provides a cool feature called the debug-tab. Just open the formula-editor (for OH-script), switch to the f\$debug-section and then you can enter your expressions. For example:

- = userchair
- = HaveTopPair

Here userchair is a native OpenHoldem symbol and HaveTopPair is an OpenPPL-symbol. So once you have entered the symbols in question click "Apply" to confirm your input and "Auto" to turn the evaluator on. OpenHoldem will evaluate your expressions and show you their values. Anything wrong with them?





Everything is correct with this screenshot. But if something is wrong, there might be several reasons:

- *incorrect input*. Have a look at OpenHoldem's table display: everything ok?
- *a problem with your code (or with the symbol library):* a symbol like for example StillToAct depends on other symbols like dealposition and nchairsdealtleft. So if StillToAct was wrong you might put these symbols into the debug-tab to evaluate further until you find the problem.

Simulating positions

The debug-tab is a wonderful tool — but: the problem happened at a live-table — what can you do? There are two things you could try: Openholdem provides a tool called ManualMode. Here you can set up some situations to simulate your bot-logic. The other possibility: you could shoot so-called replay-frames, i.e. screenshots of the casino-table that get saved to the replay-directory in your bot-folder. Once the session is over you can load these frames with OHReplay.exe and connect OpenHoldem to OHReplay like a normal casino-table. And then you can work with the debug-tab as described above. Please refer to OpenHoldem's manual for a more detailed description of these tools.

Investigating log-files

Not satisfied with the play? You might also look at the autoplayer-log. It is stored in your bot-folder and named e.g. oh_0.log. OpenHoldem uses this file to not only store the action it took, but also the complete evaluation process. First locate your hand (you can identify it e.g. by the time, your cards, the hand-number, etc.). At first the log might look like a complete mess, but actually it is well structured: function names on the very left are higher-level functions; symbols that are more indented got called by these higher-level functions. So what you see below is actually an evaluation-tree:

```
f$flop = -1000001.000 [Line 3/3]
   HaveStraightDraw = 0.000 [Line 2/2740]
      HaveOpenEndedStraightDraw = 0.000 [Line 10/2685]
         HaveStraight = 0.000 [Line 1/2735]
            nstraightfill = 3.000
         rankbits = 24770.000
      HaveDoubleGutshotDraw = 0.000 [Line 15/2286]
   HaveQuads = 0.000 [Line 1/2679]
      isfourofakind = 0.000
   BotsLastAction = -1000008.000 [Line 20/549]
      betround = 2.000
      ConstBetRoundPreflop = 1.000 [Line 1/194]
      me_re_MemBotsLastAction = 0.000
      Fold = -1000001.000 [Line 1/77]
      PrevActionWasAllin = 0.000 [Line 2/492]
         prevaction = 3.000
    PrevActionWasAllin = 0.000 [cached]
   Raise = -1000008.000 [Line 1/134]
   Raise = -1000008.000 [cached]
   Fold = -1000001.000 [cached]
```

Long story short:

- At the very top you see an OpenPPL main-function, here f\$flop
- Below you see functions that get called by f\$flop either directly or indirectly
- Functions that are indented once are directly called by f\$flop. E.g HaveStraightDraw, HaveQuads, BotsLastAction
- Functions that are indented one level deeper are called by the function above them that was indented one level less, E.g. HaveStraightDraw calls

HaveOpenEndedStraightDraw and HaveDoubleGutshotDraw.

- After the = you see the result of the function evaluation
- [Line 10/2685] means: the function returned the result at its tenth line, which is line 2685 in the file.
- [cached] means that the result already got calculated and the cached value gets reused.
- symbols without line information are built-in OpenHoldem symbols.

That's it, basically. We were a bit in a hurry when we wrote this paragraph. Any better explanation is very welcome.

Things To Watch Out For

• *Incorrect hand-reset:* Most symbols depend only on the state of the table (cards, players, etc.) and are quite reliable. However there are some symbols, that depend on previous game-states, like BettingAction-symbols; these symbols reset whenever a new hand starts. Also user-defined variables will lose their value if a hand-reset occurs. These hand-resets can be triggered by certain events, like a changing dealer-chair or disappearing community cards, depending on your hand-reset-method defined in the tablemap. Depending on your settings it might be that an occlusion of the table will cause a hand-reset for OpenHoldem. So watch out for messages like below in your log-file.

If they appear in the middle of a hand, then you should revisit your hand-reset-method.

• *Misread information:* Most problems (and most reported "bugs") are of the from "garbage in - garbage out". If OpenHoldem gets incorrect data from the scraper-engine (e.g. no seated players) then all further calculations by the symbol-engine and the evaluator will return bogus values. The reason is usually simple: a beginner with incorrect system settings or incorrect casino settings. Let us emphasize: Win2000 classic theme and all animations turned off are recommended, but not necessary. In pronciple you can scrape whatever configuration you like. However: if you use somebody else's tablemap, then you *have to* use exactly the same settings as the profile creator. So before you go crazy investigating log-files: have a look at basic OpenHoldem symbols: are they reasonable or plain wrong? And before you sit at a real-money-table observe your bot and OpenHoldem's table-display for some time: does OpenHoldem recognize every game-state correctly? Don't get frightened: *screenscraping is an easy and very flexible approach*, *that works great. But it requires some care from your side!*

Using Flags

OpenHoldem provides a useful set of buttons with a number on it, from 0 to 19 called flags. When an OH flag button is pressed then corresponding symbol (f0..f19) will be true. For example our code could look like the following:

WHEN betround=2 AND (BotRaisedBeforeFlop OR f0) Raise 50% FORCE

The above line is true if the bot raised before the Flop or the flag 0 button has been pressed. This way you could simulate costom conditions in Manual Mode.

Document generated by <u>eLyXer 1.2.5 (2013-03-10)</u> on 2014-12-12T07:42:07.449000

OpenPPL In Practice

If you have read this manual up to this point you have some basic understanding of OpenPPL: syntactical rules, symbols, etc. But the main thing — turning your poker-knowledge into bot-logic --- may look like a different kind of beast, if you have no programming experience at all. So let us take some small examples and see how OpenPPL is used in practice.

Folding Trash

```
##list_of_biggest_trash##
    // Trashy hands.
    // Never to be played, except we are in the blinds
    K8s K7s K6s K5s K4s K3s K2s Q7s...
    K8o ... 32o

##f$preflop##
    // Make life easy and get rid of the greatest trash
    WHEN NOT In BigBlind AND list_of_biggest_trash Fold FORCE
```

Open-Raising On The Button

WHEN StillToAct = 2 AND Calls = 0 AND Raises = 0 WHEN hand\$AA OR Hand\$KK OR ... RaiseTo 3 FORCE

Threebetting A Steal-Raiser

// LastRaiserPosition <= 3 means: CO, BU or SB
WHEN Calls = 0 AND Raises = 1 AND LastRaiserPosition <= 3
 WHEN AmountToCall <= 3 AND list... RaiseTo 9 FORCE</pre>

Contibetting

```
##f$flop##
WHEN Bets = 0 AND BotIsLastRaiser
WHEN Opponents = 1 AND Random < 80 BetHalfPot FORCE
WHEN Opponents = 2 AND Position = Last RaiseBy 66% FORCE</pre>
```

Effective StackSize

##f\$EffectiveStacksize##

// First orbit preflop only, as it uses MaxStillToActStackSize
WHEN StackSize > MaxStillToActStackSize RETURN MaxStillToActStac
WHEN Others RETURN StackSize FORCE

Push/Folding In A SNG

```
// Going into push-fold-mode when stacksizes are small
// But be careful!
// We don't want to push loosely if something is wrong with the stac
// e.g. because the table was occluded, so we check, if it is non-ze
WHEN EffectiveStacksize > 0 AND EffectiveStacksize <= 13
    WHEN Calls = 0 AND Raises = 0 AND (Hand$... RaiseMax FORCE</pre>
```

Detecting A Limp-Raise

```
// First action preflop
WHEN BotsActionsOnThisRound = 0 AND Calls >= 1 AND Raises = 0
    // FirstCallerPosition is limited to first orbit preflop only,
    // so we remember it, making use of advanced memory symbols
    WHEN Others me_st_MemFirstCallerPosition_FirstCallerPosition
    // Then continue with normal bot-logic
    ....

// After that we can detect a limp-raise like that
// (assuming, there are no other raisers in the pot)
WHEN LastRaiserPosition = me_re_MemFirstCallerPosition ...
```

Counting Outs

```
##f$MySimpleOutsCounter##
WHEN (HaveNutFlushDraw AND HaveNutStraightDraw)
    // We already have 9 + 8 - 2 duplicates
    // Plus up to 6 undiscounted outs for Overcards
    // but some may be already counted for the flush
    // (if we want to make it extra good, we should check the co
    // of our hole cards with OpenHoldems symbols $$ps0 and $$ps
    // The others should be discounted a bit
    WHEN (Overcards = 2) RETURN 18 FORCE
    WHEN (Overcards = 1) RETURN 16.5 FORCE
    WHEN (Overcards = 0) RETURN 15 FORCE

WHEN (HaveNutFlushDraw) // AND NOT HaveNutStraightDraw
    WHEN (HaveInsideStraightDraw AND Overcards = 2)...
...
```

Calling According To Odds And Outs

```
##f$CardsLeft##
   WHEN betround = 2 RETURN 47 FORCE
   WHEN betround = 3 RETURN 46 FORCE
   // Drawing at other betrounds doesn't make much sense
   WHEN Others RETURN -1 FORCE

WHEN AmountToCall / (AmountToCall + PotSize) > f$Outs / f$CardsLeft
```

Playing Fit-Or-Fold Multiway

```
##f$HaveStrongDraw##
```

WHEN HaveStraightDraw OR HaveFlushDraw RETURN True FORCE WHEN HaveInsideStraightDraw AND Overcards = 2 RETURN True FORCE

##f\$HaveTopPairOrBetter##

WHEN HaveTopPair RETURN True FORCE WHEN HaveOverPair RETURN True FORCE WHEN HaveBestOverpairOrBetter RETURN True FORCE

##f\$flop##

WHEN Opponents >= 3 AND NOT (f\$HaveTopPairOrBetter OR f\$HaveStrongDr WHEN AmountToCall = 0 Check FORCE

Closing The Tables

```
##f$leave##
WHEN issittingout
    AND elapsedauto > 300    // 5 minutes without action
    AND nopponentsseated < 4
RETURN True FORCE</pre>
```

Now it's up to you. Let the fun begin!

Document generated by <u>eLyXer 1.2.5 (2013-03-10)</u> on 2015-09-29T21:48:00.750000

If you want to buy...

... we don't sell anything.

However if you really need "more" than you get here, especially built-in support for some pokersites, a winning bot for the lowest microstakes (NL 2 — yes, that are blinds of \$0.01/\$0.02) and professional hand-holding, then you might want to consider buying the original Shanky-bot at www.bonusbots.com. To our best knowledge they offer an advertisement-deal of \$30 or \$50 or something for every customer who gets referred. However: we do neither want to nor need to make any money advertising such funny things like "winning poker-bots", but we don't want to waste that money either. So if you really buy their bot, please register to their forum, send greetings to everybody and ask Egor to send that money to

Doctors without Borders.

333 7th Avenue, 2nd Floor New York, NY 10001-5004

Phone: 212-679-6800 Fax: 212-679-7016

https://www.doctorswithoutborders.org/donate/

Thank you very much for your help and cooperation!

Document generated by <u>eLyXer 1.2.5 (2013-03-10)</u> on 2015-10-07T21:50:29.196000