

Implementacja i analiza wydajnościowa generatora silników gier logicznych



DYPLOMANT:

Paweł Troka

OPIEKUN:

prof. dr hab. inż. Krzysztof Giaro

KONSULTANT:

mgr inż. Tomasz Goluch



Cel pracy



- „Oficjalny” cel pracy:
- Celem pracy jest zbadanie wydajności generatorów silników gier logicznych.



Zadania do wykonania



- 1. Przegląd i analiza dostępnych implementacji silników popularnych gier (Go, szachy, warcaby, itp.)
- 2. Implementacja generatora silników wybranej gry, parametryzowanego takimi wartościami jak: rozmiary planszy, liczba pól aktywnych planszy, zasady gry, liczba graczy itp.
- 3. Analiza wydajnościowo, poprawnościowa zaimplementowanego generatora silników gier



Dlaczego taki temat?



- Był wolny ☺
- Zupełnie nowy obszar naukowy, generalnie dotąd mi znany słabo
- Sporo cudzych kodów źródłowych do obejrzenia, zrozumienia i porównania
- Coś innego niż robię na „co dzień”



„Narzucone” wymagania



- Wybrana gra powinna być rozgrywana na olimpiadzie ICGA
- Osobno GUI i osobno program grający
- Raczej C/C++ ze wstawkami Assemblerowymi, jeżeli implementacja ma rywalizować z innymi
- Dostępność kodów źródłowych porównywanych silników





The International Computer Games Association (ICGA) was founded as the International Computer Chess Association (ICCA) in 1977 by computer chess programmers to organise championship events for computer programs and to facilitate the sharing of technical knowledge via the ICCA Journal.

Renamed the 'ICGA' in 2002, the association now more broadly fosters the computer games and game artificial intelligence community by organizing the Computer Olympiad, the World Computer Chess Championship, and the International Conference on Computers and Games. The ICGA also publishes a quarterly journal, the ICGA Journal, and maintains relationships with Computer Science, Commercial, and Game organizations throughout the world. The ICGA is led by David Levy.



ICGA 2016



The 22nd World Computer-Chess Championship will take place from June 27 – July 3, 2016 in Leiden, the Netherlands, in the Snellius building of the Leiden University, Niels Bohrweg 1, 2333 CA Leiden, the Netherlands. We are grateful to LIACS and the Boerhaave Museum that they have offered to sponsor and organize five events in cooperation with the ICGA, viz. the 22nd WCCC, the 6th WCSC, the 2016 World Computer Chess Blitz Championship, the 19th Computer Olympiad, and the 9th Computer and Games Conference (CG2016). Here we recall that the Maastricht Triennial Meeting in 2002, i.e., the first ICGA meeting (instead of ICCA), decided that the WCCC should be held annually without distinguishing any type of machines. The observation was clear: all kinds of differences between microcomputers, personal computers, “normal” computers, and supercomputers were in some sense obsolete and the classification thus was considered artificial. So was the division into the classes of single processors and multiprocessors.



ICGA 2016



Snellius Building



ICGA 2016



Another division considered obsolete in the competition since 2002 is that between amateur, semi-professional, and professional. In 2015 we have abandoned also the differences in payments and made the subscription for participants equal for all types of human programmers and their programs (for all games). The only prerequisite is originality. See rule 2 of the 22nd World Computer-Chess Championship which holds for all games and all types of competitions in Leiden 2016. Following the survey conducted amongst chess programmers after the WCCC tournament in Pamplona, the ICGA announced the inauguration of a new tournament for 2010 and subsequent years - The World Chess Software Championship (WCSC).



ICGA 2016



Below we provide the rules for the 22nd World Computer-Chess Championship, for the 2016 World Chess Software Championship, for the 2016 World Computer Chess Blitz Championship, and for the 19th Computer Olympiad. It was agreed at the Maastricht meeting that from the 13th World Computer-Chess Championship onwards the Shannon Trophy will be awarded annually (if it were possible to organize the tournaments). The original trophy is kept at ICGA headquarters in Leiden, and each year a replica of the trophy is presented on a permanent basis to the World Computer-Chess Champion team. A trophy will be given each year to be retained by the winner of the World Chess Software Championship. The years 2012 and 2014 were in all aspects an exception since the ICGA was unable to find a sponsor for the events. For the WCCC and the WCSC, we have split the rules into two sections of general rules and two sections of tournament rules.



ICGA 2016



Below we provide the rules for the 22nd World Computer-Chess Championship, for the 2016 World Chess Software Championship, for the 2016 World Computer Chess Blitz Championship, and for the 19th Computer Olympiad. It was agreed at the Maastricht meeting that from the 13th World Computer-Chess Championship onwards the Shannon Trophy will be awarded annually (if it were possible to organize the tournaments). The original trophy is kept at ICGA headquarters in Leiden, and each year a replica of the trophy is presented on a permanent basis to the World Computer-Chess Champion team. A trophy will be given each year to be retained by the winner of the World Chess Software Championship. The years 2012 and 2014 were in all aspects an exception since the ICGA was unable to find a sponsor for the events. For the WCCC and the WCSC, we have split the rules into two sections of general rules and two sections of tournament rules.



THE 19th COMPUTER OLYMPIAD



- The Leiden Centre of Data Science (LCDS) is enabling the organisation of the Computer Olympiad, i.e., the 19th Computer Olympiad (CO) (June 27 - July 3). Location: the Snellius Building, Niels Bohrweg 1, 2333 CA Leiden, the Netherlands.
- The Computer Olympiad is a multi-games event in which all of the participants are computer programs. The purpose is to find the strongest programs at each of the games, partly as an academic exercise and partly because the competitions are fun. In Table 1 we mention 37 different games for which a program can be submitted to the Olympiad.
- We are willing to host more games, such as Ataxx, Dvonn, Light up, Mediocrity, Onyx, Tamsk, TwixT, and Zèrtz, but we do not know of the existence of adequately playing programs. We are awaiting suggestions and proposals of programmers before we include them in the official list.
- For each game, a tournament will take place provided that at least two programs enter the tournament for that particular game.
- Gold, Silver and Bronze medals will be awarded to the leading programs in each tournament.



RULES AND GENERAL RULES



Each program must be the original work of the entering developers, possibly with the inclusion of game playing code and/or data from other sources for which the entering developers have a legal right of use. Developers whose code is derived from or includes (1) game-playing code; and/or (2) data written by others, must name (a) all the other developers of whom they are aware; and (b) the source of such code and/or data, in their tournament registration details.



RULES AND GENERAL RULES



Programs which are discovered to be undeclared derivatives of others may be designated invalid by the Tournament Director if he is convinced, after seeking advice if he feels that to be necessary, that the closeness of derivation is of such a level as to constitute unfair competition. A listing and an executable version of all game-related code and data running on the system must be available on demand to the Tournament Director prior to the start of and during the tournament. The Tournament Director has the right to submit the executable version of a program for testing for similarity with other known programs, and/or to submit the listing to an expert or experts of his choosing for examination, also to determine similarity. Under all circumstances the Tournament Director will take all reasonable steps to ensure that any such listing and/or executable are treated as being strictly confidential.



RULES AND GENERAL RULES



The entering developers must keep a copy of the source code of their entry until at least one year following the date of conclusion of the tournament, in order to be able to respond accurately to any questions about the source code that might be raised after the event by the Tournament Director.

The entry fee for the Olympiad tournaments is as follows: € 50 for the first game and € 25 for each subsequent game (exclusive ICGA membership).

A participant is expected to be an ICGA member (€ 40). Deadline early registration: May 15, 2016. Any entry received after May 15, 2016 will be subject to a penalty fee, doubling the above fee. Website: www.icga.org.



The ICGA 2016 Events

10-3-2016

Preliminary Schedule (June 27 - July 3, 2016)

event	Mon 27/6	Tue 28/06	Wed 29/06	Thu 30/06	Fri 1/07	Sat 2/07	Sun 3/07	
9th CG2016 conference			Snellius, room 174 8.30 - 12.30 hours	Snellius, room 174 8.30 - 12.30 hours	Snellius, room 174 8.30 - 12.30 hours			
22th WCCC, room 302	1000 Opening and players meeting, 12-16 round 1, 17-21 round 2	9-13 round 3, 14-18 round 4	9-13 round 5, 14-18 round 6	9-13 round 7, 14-18 round 8	9-13 round 9, 14-18 Playoffs			
6th WCSC, room 302						9-11 round 1, 12-14 round 2, 15-17 round 3, 18-20 round 4	9-11 round 5, 12-14 round 6, 15-17 round 7	
Speed tournament, room 302		19-22 hours						
19th Computer Olympiad, room 302 + 304	Olympiad after the opening, players meeting	Olympiad all day 10-21h	Olympiad after the conference at 13.00h	Olympiad after the conference 13.00h	Olympiad after the tri-ennial meeting at 14.00 h	Olympiad Draughts 10.00 h - 17.00 h	Olympiad Draughts 10.00 h - 17.00 h	
				dinner for all at 1900 h			Closing and medals etc 18.00	

open tot	22 uur	22 uur	21 uur	19 uur	21 uur	21 uur	20
----------	--------	--------	--------	--------	--------	--------	----

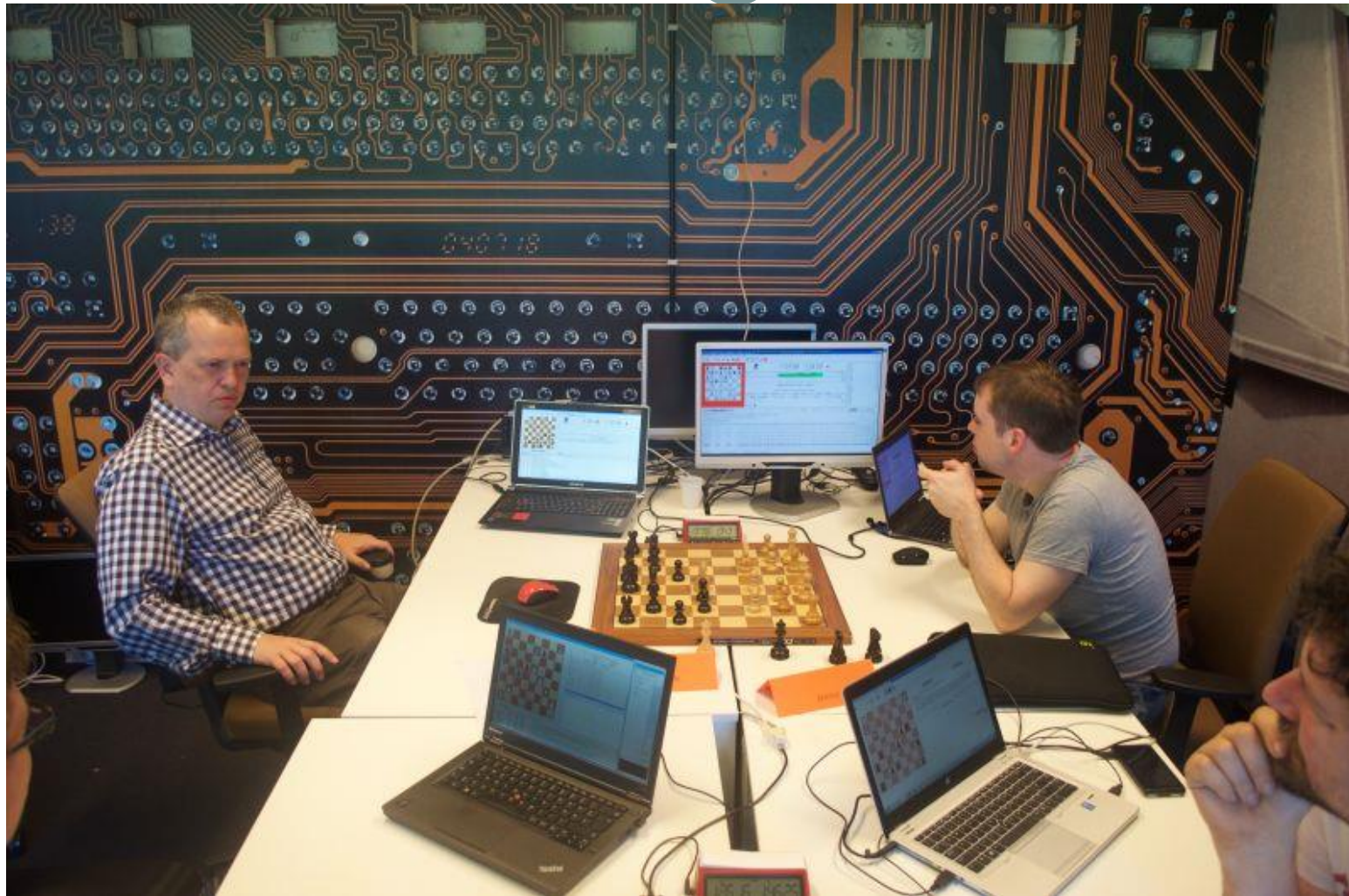
ICGA



ICGA



ICGA



ICGA



ICGA



ICGA



Prize giving ceremony WCCC: Jonny (Johannes Zwanzger) receives the Shannon Trophee.



Johannes Zwanzger



a German mathematician with a Ph.D. from University of Bayreuth, chess programmer and chess player, since 2013 playing for FC Bayern München in the German Chess Bundesliga. His research interests include construction of discrete structures, especially linear codes, the use of heuristics in general, and chess programming. Johannes is author of the chess program Jonny, which competed at all World Computer Chess Championships since Graz 2003, and became World Computer Chess Champion at the WCCC 2015 in Leiden.



Johannes Zwanzger



Honorable sportsmanship intended, Johannes was subject of a difficult and controversial TD-decision during the 11th World Computer Chess Championship, 2003 in Graz. In the last round game, Shredder versus Jonny, Shredder had a winning position, already about to announce mate in X. There would nothing wrong for Johannes to resign on behalf of Jonny. Apparently due to a bug, Shredder allowed a threefold repetition, still high plus score. Johannes, while his engine had a zero score, and the external GUI a "Threefold repetition" Infobox popped up, decided that Info is no draw-claim and continued the game by making the move over the board. TD Jaap van den Herik first miss understood, assuming Johannes would claim a draw rather than asking to continue



Johannes Zwanzger



[Event "11th World Computer Chess Championship"]

[Site "Graz, Austria"]

[Date "2003.11.29"]

[Round "11"]

[White "Shredder"]

[Black "Jonny"]

[Result "1-0"]

1. e4 c5 2. Nf3 e6 3. d4 cxd4 4. Nxd4 Nf6 5. Nc3 d6 6. Bg5 Be7 7. f4 O-O 8. Qf3 e5
9. Nf5 Bxf5 10. exf5 e4 11. Qh3 h6 12. Bh4 Qc7 13. g4 d5 14. g5 hxg5 15. fxg5 Nh5
16. O-O-O Rc8 17. Bg3 Nf4 18. Bxf4 Qxf4+ 19. Kb1 Bxg5 20. f6 Nc6 21. fxg7 Kxg7
22. Be2 d4 23. Rhf1 Qh4 24. Qf5 dxc3 25. Rg1 Kf8 26. Rxg5 Ne7 27. Qf6 Qh7
28. Rd7 Ng6 29. Rxb7 Qg7 30. Qd6+ Kg8 31. Rxf7 Qxf7 32. Rxg6+ Kh7 33. Rg4 Rab8
34. Rh4+ Kg8 35. Rg4+ Kh7 36. Rh4+ Kg8 37. Rg4+ Kh7 38. Bc4 Rxb2+ 39. Ka1 Rxc4
40. Rh4+ Kg8 41. Qd8+ Qf8 42. Rg4+ Kf7 43. Qd7+ Qe7 44. Rf4+ Kg6 45. Qxe7 Rxa2+
46. Kxa2 Ra4+ 47. Kb3 Rb4+ 48. Kxb4 a5+ 49. Kxc3 a4 50. Qf6+ Kh5 51. Rh4# 1-0



Gry rozgrywane w 2015 na ICGA



Abalone	Computational Pool	Go 9×9	Othello
Amazons	Connect 6	Havannah	Phantom Go
Arimaa	Diplomacy	Hex	Poker
Backgammon	Dominoes	Kriegspiel	Quoridor
Bao	Dots and Boxes	Lines of Action	Scrabble
Bridge	10×10 Draughts	Mahjong	Stratego
Chinese Chess	Gipf	MiniShogi	Shogi
Chu Shogi	Go	Nonogram	Shogi 5×5
Clobber	Go 13×13	OCTI	Surakarta



THE 19th COMPUTER OLYMPIAD












Abalone	Computational Pool	Go 9x9	Phantom Go
Amazons	Connect 6	Havannah	Poker
Arimaa	Diplomacy	Hex	Scrabble
Backgammon	Dominoes	Kriegspiel	Stratego
Bao	Dots and Boxes	Lines of Action	Shogi
Bridge	10x10 Draughts	Mahjong	Shogi 5x5
Chinese Chess	Ein Stein Wurfelt Nicht	NoGo	Surakarta
Chinese Dark Chess	Gipf	Nonogram	
Chu Shogi	Go	OCTI	
Clobber	Go 13x13	Othello	

Table 1: Games played at the Computer Olympiad.












21st World Computer Chess Championship

Rank	Program	Country	Hardware	Score	Games	Title
1	<u>Jonny</u>	 <u>DEU</u>	2,400 cores, 2,376 threads, AMD x86-64, 2.8 GHz	7.0	8	World Computer Chess Champion
2	<u>Komodo</u>	 <u>USA</u>	24 cores, 24 threads, Intel i7	6.5	8	Second Place
3	<u>Hiarcs</u>	 <u>GBR</u>	28 threads, Intel Xeon E5-2697, 2.8 GHz	5.0	8	Third Place
4	<u>Protector</u>	 <u>DEU</u>	8 cores, 15 threads, Intel i7 5690x, 4 GHz	5.0	8	
5	<u>Shredder</u>	 <u>DEU</u>	16 cores, 16 threads, Intel i7	4.5	8	
6	<u>Ginkgo</u>	 <u>DEU</u>	6 cores, 6 threads, Intel i5	4.0	8	
7	<u>The Baron</u>	 <u>NLD</u>	16 cores, 16 threads, Intel E5-2687w, 3.1 GHz	3.0	8	
8	<u>Maverick</u>	 <u>USA</u>	1 thread, Intel i7, 2.8 GHz	1.0	8	
9	<u>Fridolin</u>	 <u>DEU</u>	4 cores, 7 threads. Intel i7	0.0	8	

	1	2	3	4	5	6	7	8	9
1 <u>Jonny</u>		1	1	=	=	1	1	1	1
2 <u>Komodo</u>	0		=	1	1	1	1	1	1
3 <u>Hiarcs</u>	0	=		=	1	=	=	1	1
4 <u>Protector</u>	=	0	=		=	=	1	1	1
5 <u>Shredder</u>	=	0	0	=		=	1	1	1
6 <u>Ginkgo</u>	0	0	=	=	=		=	1	1
7 <u>The Baron</u>	0	0	=	0	0	=		1	1
8 <u>Maverick</u>	0	0	0	0	0	0	0		1
9 <u>Fridolin</u>	0	0	0	0	0	0	0	0	



21st World Computer Chess Championship









Rank	Program	Country	Hardware	Score	Games	Title
1	<u>Jonny</u>	 <u>DEU</u>	2,400 cores, 2,376 threads, AMD x86-64, 2.8 GHz	7.0	8	World Computer Chess Champion
2	<u>Komodo</u>	 <u>USA</u>	24 cores, 24 threads, Intel i7	6.5	8	Second Place
3	<u>Hiarcs</u>	 <u>GBR</u>	28 threads, Intel Xeon E5-2697, 2.8 GHz	5.0	8	Third Place
4	<u>Protector</u>	 <u>DEU</u>	8 cores, 15 threads, Intel i7 5690x, 4 GHz	5.0	8	
5	<u>Shredder</u>	 <u>DEU</u>	16 cores, 16 threads, Intel i7	4.5	8	
6	<u>Ginkgo</u>	 <u>DEU</u>	6 cores, 6 threads, Intel i5	4.0	8	
7	<u>The Baron</u>	 <u>NLD</u>	16 cores, 16 threads, Intel E5-2687w, 3.1 GHz	3.0	8	
8	<u>Maverick</u>	 <u>USA</u>	1 thread, Intel i7, 2.8 GHz	1.0	8	
9	<u>Fridolin</u>	 <u>DEU</u>	4 cores, 7 threads. Intel i7	0.0	8	

	1	2	3	4	5	6	7	8	9
1 <u>Jonny</u>		1	1	=	=	1	1	1	1
2 <u>Komodo</u>	0		=	1	1	1	1	1	1
3 <u>Hiarcs</u>	0	=		=	1	=	=	1	1
4 <u>Protector</u>	=	0	=		=	=	1	1	1
5 <u>Shredder</u>	=	0	0	=		=	1	1	1
6 <u>Ginkgo</u>	0	0	=	=	=		=	1	1
7 <u>The Baron</u>	0	0	=	0	0	=		1	1
8 <u>Maverick</u>	0	0	0	0	0	0	0		1
9 <u>Fridolin</u>	0	0	0	0	0	0	0	0	



21st World Computer Chess Championship (software)

The World Computer Software Championship is played on equal hardware (Intel I7 with 4 cores and 8 threads), without internet access, for all participants. Time control: 45 minutes per program per game with 15 seconds per move from move 1.

Rank	Program	Country	Score	Games	Playoff
1	<u>Shredder</u>	 <u>DEU</u>	5.0	7	2.5
2	<u>Ginkgo</u>	 <u>DEU</u>	5.0	7	1.5
3	<u>Protector</u>	 <u>DEU</u>	4.5	7	
3	<u>Komodo</u>	 <u>USA</u>	4.5	7	
5	<u>Hiarcs</u>	 <u>GBR</u>	4.0	7	
6	<u>Jonny</u>	 <u>DEU</u>	3.5	7	
7	<u>Maverick</u>	 <u>USA</u>	1.5	7	
8	<u>Fridolin</u>	 <u>DEU</u>	0.0	7	



Go



Rank	Program	Country	Hardware	Score	Games	Playoff	SOS	SoDOS	Title
1	<u>Zen</u>	<u>JPN</u>	2 x Xeon E5-2690v3, 2 x Xeon E5-2690v2, 2 x Xeon X5680, i7 5960X	7.0	7		21.0	21.00	Gold medal
2	<u>Abakus</u>	<u>DEU</u>	OCuLUS; 8 x 2 x Intel Xeon E5-2670/2.6 GHz, 64 GB (Linux)	5.0	7	1.0	23.0	13.00	Silver medal
3	<u>Nomitan</u>	<u>JPN</u>	7 x 2 x Intel Xeon (6-core each)	5.0	7	0.0	23.0	11.00	Bronze medal
4	<u>CGI</u>	<u>TWN</u>	20 x 3 x AMD Opteron 6174/2.2 GHz (ALPS system by Acer)	4.0	7		24.0	7.00	
5	<u>MC_ark</u>	<u>JPN</u>	2 x Intel Xeon E5-2687W/3.1 GHz, 32 GB (Linux)	3.0	7		25.0	7.00	
6	<u>Jimmy</u>	<u>TWN</u>	2 x Intel Xeon E5-2690, 128 GB	2.0	7		26.0	3.00	
7	<u>Oakfoam</u>	<u>ZAF</u>	Intel Core i7 4790K/4.2 GHz with nVidia GTX970	2.0	7		26.0	2.00	







- SOS: Sum of Opponent Scores
- SoDOS: Sum of Defeated Opponent Scores

		1	2	3	4	5	6	7
1	<u>Zen</u>		1	1	1	1	1	1
2	<u>Abakus</u>	0		1	1	0	1	1
3	<u>Nomitan</u>	0	0		1	1	1	1
4	<u>CGI</u>	0	0	0		1	1	1
5	<u>MC_ark</u>	0	1	0	0		0	1
6	<u>Jimmy</u>	0	0	0	0	1		0
7	<u>Oakfoam</u>	0	0	0	0	0	1	



Go 9x9









Rank	Program	Country	Hardware	Score	Games	Title
1	<u>Zen</u>	 <u>JPN</u>	2 x Xeon E5-2690v3, 2 x Xeon E5-2690v2, 2 x Xeon X5680, i7 5960X	9.5	10	Gold medal
2	<u>Abakus</u>	 <u>DEU</u>	OCuLUS; 8 x 2 x Intel Xeon E5-2670/2.6 GHz, 64 GB (Linux)	7.5	10	Silver medal
3	<u>CGI</u>	 <u>TWN</u>	20 x 3 x AMD Opteron 6174/2.2 GHz (ALPS system by Acer)	7.0	10	Bronze medal
4	<u>Nomitan</u>	 <u>JPN</u>	7 x 2 x Intel Xeon (6-core each)	4.0	10	
5	<u>MC ark</u>	 <u>JPN</u>	2 x Intel Xeon E5-2687W/3.1 GHz, 32 GB (Linux)	2.0	10	
6	<u>Wingo</u>	 <u>TWN</u>		0.0	10	

	1	2	3	4	5	6
1 <u>Zen</u>		2	1.5	2	2	2
2 <u>Abakus</u>	0		1.5	2	2	2
3 <u>CGI</u>	0.5	0.5		2	2	2
4 <u>Nomitan</u>	0	0	0		2	2
5 <u>MC ark</u>	0	0	0	0		2
6 <u>Wingo</u>	0	0	0	0	0	



Go 13x13



Rank	Program	Country	Hardware	Score	Games	Title
1	<u>Zen</u>	 <u>JPN</u>	2 x Xeon E5-2690v3, 2 x Xeon E5-2690v2, 2 x Xeon X5680, i7 5960X	5.0	5	Gold medal
2	<u>Nomitan</u>	 <u>JPN</u>	7 x 2 x Intel Xeon (6-core each)	4.0	5	Silver medal
3	<u>Abakus</u>	 <u>DEU</u>	OCuLUS; 8 x 2 x Intel Xeon E5-2670/2.6 GHz, 64 GB (Linux)	3.0	5	Bronze medal
4	<u>CGI</u>	 <u>TWN</u>	20 x 3 x AMD Opteron 6174/2.2 GHz (ALPS system by Acer)	2.0	5	
5	<u>MC_ark</u>	 <u>JPN</u>	2 x Intel Xeon E5-2687W/3.1 GHz, 32 GB (Linux)	1.0	5	
6	<u>Wingo</u>	 <u>TWN</u>		0.0	5	




	1	2	3	4	5	6
1 <u>Zen</u>		1	1	1	1	1
2 <u>Nomitan</u>	0		1	1	1	1
3 <u>Abakus</u>	0	0		1	1	1
4 <u>CGI</u>	0	0	0		1	1
5 <u>MC_ark</u>	0	0	0	0		1
6 <u>Wingo</u>	0	0	0	0	0	



Lines of Action



Result

Rank	Program	Country	Score	Games	SOS	SoDOS	Title
1	<u>MC-LOA</u>	 <u>NLD</u>	6.0	6	8.0	8.00	Gold medal
2	<u>Deep Nikita</u>	 <u>USA</u>	4.0	6	12.0	0.00	Silver medal
3	<u>Lucky Line</u>	 <u>CHN</u>	0.0	8	40.0	0.00	Bronze medal

- SOS: Sum of Opponent Scores
- SoDOS: Sum of Defeated Opponent Scores

Cross-table

	1	2	3
1 <u>MC-LOA</u>		2	4
2 <u>Deep Nikita</u>	0		4
3 <u>Lucky Line</u>	0	0	



Surakarta



Rank	Program	Country	Title
1	<u>SIA</u>	 <u>NLD</u>	Gold medal
2	<u>Deep Nikita</u>	 <u>USA</u>	Silver medal
3	<u>V&S Surakarta</u>	 <u>CHN</u>	Bronze medal



Surakarta



Rank	Program	Country	Title
1	<u>SIA</u>	 <u>NLD</u>	Gold medal
2	<u>Deep Nikita</u>	 <u>USA</u>	Silver medal
3	<u>V&S Surakarta</u>	 <u>CHN</u>	Bronze medal



Hex



Hex 11×11

Pos	Name	Total	1	2	3
1	Mo-Hex	7	-	3	4
2	Deep-Hex	5	1	-	4
3	Ezo	0	0	0	-

[Top](#)

Hex 13×13

Pos	Name	Total	1	2	3
1	Mo-Hex	6	-	2	4
2	Deep-Hex	6	2	-	4
3	Ezo	0	0	0	-

The play-off was won by Mo-Hex (2-0)

Autorem programu MIMHex jest Polak, Jakub Pawlewicz. Jego algorytm (new virtual connection implementation) został wykorzystany w mistrzowskim programie MoHex



Wybrane technologie



- IDE: Visual Studio 2015 Ultimate
- GUI generatora:
 - Język programowania: C# 6.0
 - .NET 4.6.1
 - WPF
- Silnik:
 - C/C++ ze wstawkami Assemblerowymi
 - Portable
 - Mocno zoptymalizowany kod i build



Wybrane technologie



- Aplikacja testująca wydajność/poprawność:
 - Język programowania: C# 6.0
 - .NET 4.6.1
 - WPF
 - Generowanie dokumentów z wynikami
- Github – publiczne repo, na bieżąco uzupełniane



Część I – przegląd i analiza wyb. Impl.

- Na podstawie dostępnych implementacji uwzględnić ogólny trend programowania silników gier, zastosowane optymalizacje czasowe i pamięciowe (kompresja stanów gry).
- Czy są stosowane optymalizacje sprzętowe (np. kod pisany w assemblerze pod konkretne procesory).



Część I – przegląd i analiza



Stockfish

Strong open source chess engine

[Download Stockfish 7 for Windows](#)

Powerful

Stockfish is one of the strongest chess engines in the world. It is also much stronger than the best human chess grandmasters.

[View CCRL rating list](#)

Open Source

Unlike most chess engines, Stockfish is open source (GPL license). That means you can read the code, modify it, contribute back, and even use it in your own projects.

[Stockfish on GitHub](#)

Run Anywhere

You can use Stockfish on your computer running Windows, OS X, or Linux, or on your iOS or Android device. So you can get world-class chess analysis, wherever you are.

[View all downloads](#)



Część I – przegląd i analiza



Overview

Stockfish is a free UCI chess engine derived from Glaurung 2.1. It is not a complete chess program and requires some UCI-compatible GUI (e.g. XBoard with PolyGlott, eboard, Arena, Sigma Chess, Shredder, Chess Partner or Fritz) in order to be used comfortably.

This version of **Stockfish supports up to 128 cores**. The engine defaults to one search thread, so it is therefore recommended to inspect the value of the *Threads* UCI parameter, and to make sure it equals the number of CPU cores on your computer.

Stockfish has support for **32 or 64-bit CPUs**, the **hardware POPCNT instruction**, big-endian machines such as Power PC, and other platforms.



Część I – przegląd i analiza



POPCNT

These instructions operate on integer rather than SSE registers, and although introduced by AMD with the SSE4a instruction set, they are counted as separate extensions with their own dedicated CPUID bits to indicate support. Intel implements POPCNT beginning with the Nehalem microarchitecture (2008). AMD implements both beginning with the Barcelona microarchitecture. AMD calls this pair of instructions Advanced Bit Manipulation (ABM).

Instruction	Description
-------------	-------------

POPCNT	
--------	--

Population count (count number of bits set to 1). Support is indicated via the CPUID.01H:ECX.POPCNT[Bit 23] flag.[14]



Część I – przegląd i analiza

```
35  /// Determine at compile time the best popcount<> specialization according to
36  /// whether the platform is 32 or 64 bit, the maximum number of non-zero
37  /// bits to count and if the hardware popcnt instruction is available.
38  const BitCountType Full  = HasPopCnt ? CNT_HW_POPCNT : Is64Bit ? CNT_64 : CNT_32;
39  const BitCountType Max15 = HasPopCnt ? CNT_HW_POPCNT : Is64Bit ? CNT_64_MAX15 : CNT_32_MAX15;
40
41
42  /// popcount() counts the number of non-zero bits in a bitboard
43  template<BitCountType> inline int popcount(Bitboard b);
44
45  template<>
46  inline int popcount<CNT_64>(Bitboard b) {
47      b -= (b >> 1) & 0x5555555555555555ULL;
48      b = ((b >> 2) & 0x3333333333333333ULL) + (b & 0x3333333333333333ULL);
49      b = ((b >> 4) + b) & 0x0F0F0F0F0F0F0F0FULL;
50      return (b * 0x0101010101010101ULL) >> 56;
51  }
52
53  template<>
54  inline int popcount<CNT_64_MAX15>(Bitboard b) {
55      b -= (b >> 1) & 0x5555555555555555ULL;
56      b = ((b >> 2) & 0x3333333333333333ULL) + (b & 0x3333333333333333ULL);
57      return (b * 0x1111111111111111ULL) >> 60;
58  }
59
60  template<>
61  inline int popcount<CNT_32>(Bitboard b) {
62      unsigned w = unsigned(b >> 32), v = unsigned(b);
63      v -= (v >> 1) & 0x55555555; // 0-2 in 2 bits
64      w -= (w >> 1) & 0x55555555;
65      v = ((v >> 2) & 0x33333333) + (v & 0x33333333); // 0-4 in 4 bits
66      w = ((w >> 2) & 0x33333333) + (w & 0x33333333);
67      v = ((v >> 4) + v + (w >> 4) + w) & 0x0F0F0F0F;
68      return (v * 0x01010101) >> 24;
69  }
```



Część I – przegląd i analiza



```
140 // Threat[attacking][attacked] contains bonuses according to which piece
141 // type attacks which one.
142 const Score Threat[][PIECE_TYPE_NB] = {
143     { S(0, 0), S( 7, 39), S(24, 49), S(24, 49), S(41,100), S(41,100) }, // Minor
144     { S(0, 0), S(15, 39), S(15, 45), S(15, 45), S(15, 45), S(24, 49) } // Major
145 };
146
147 // ThreatenedByPawn[PieceType] contains a penalty according to which piece
148 // type is attacked by an enemy pawn.
149 const Score ThreatenedByPawn[] = {
150     S(0, 0), S(0, 0), S(80, 119), S(80, 119), S(117, 199), S(127, 218)
151 };
152
153 // Assorted bonuses and penalties used by evaluation
154 const Score KingOnPawnOne   = S(0 , 64);
155 const Score KingOnPawnMany  = S(0 ,128);
156 const Score RookOnPawn      = S(10, 28);
157 const Score RookOpenFile    = S(43, 21);
158 const Score RookSemiOpenFile = S(19, 10);
159 const Score BishopPawns     = S( 8, 12);
160 const Score MinorBehindPawn  = S(16,  0);
161 const Score TrappedRook      = S(92,  0);
162 const Score Unstoppable      = S( 0, 20);
163 const Score Hanging          = S(23, 20);
```



Część I – przegląd i analiza



```
template<>
Move MovePicker::next_move<false>() {

    Move move;

    while (true)
    {
        while (cur == end)
            generate_next_stage();

        switch (stage) {

        case MAIN_SEARCH: case EVASION: case QSEARCH_0: case QSEARCH_1: case PROBCUT:
            ++cur;
            return ttMove;

        case CAPTURES_S1:
            move = pick_best(cur++, end)->move;
            if (move != ttMove)
            {
                if (pos.see_sign(move) >= VALUE_ZERO)
                    return move;

                // Losing capture, move it to the tail of the array
                (endBadCaptures--)->move = move;
            }
            break;
        }
```

```
        case KILLERS_S1:
            move = (cur++)->move;
            if (    move != MOVE_NONE
                && move != ttMove
                && pos.pseudo_legal(move)
                && !pos.capture(move))
                return move;
            break;

        case QUIETS_1_S1: case QUIETS_2_S1:
            move = (cur++)->move;
            if (    move != ttMove
                && move != killers[0].move
                && move != killers[1].move
                && move != killers[2].move
                && move != killers[3].move
                && move != killers[4].move
                && move != killers[5].move)
                return move;
            break;
    }
```



Część II – własna implementacja



- Silnik gry powinien charakteryzować się wydajnością czasową a przekazywane do niego oraz generowane przez niego listy możliwych posunięć powinny być zoptymalizowane pamięciowo, ponieważ mogą być przechowywane w węzłach algorytmów z rodziny best-first search.
- Koniecznie osobno GUI (Arena, Winboard, Xboard) i osobno program grający



Część II – własna implementacja (GUI?)

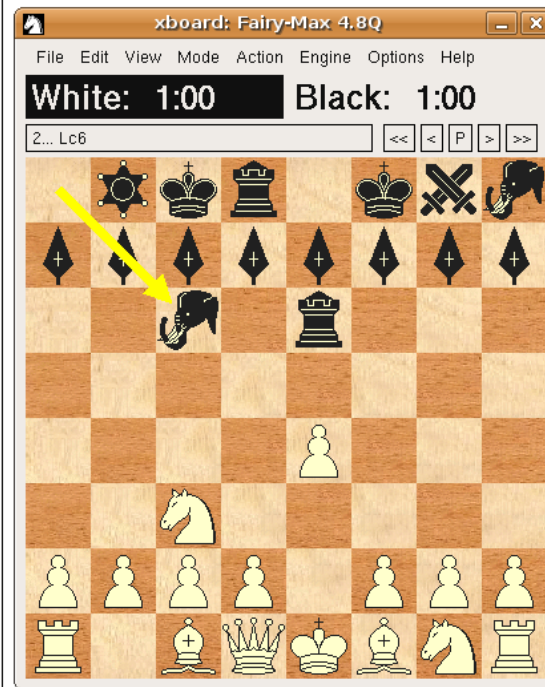
made with the mouse, and loads and saves games in Portable Game Notation (PGN). It serves as a front-end for many different chess services, including:

- Chess engines that will run on your machine and play a game against you or help you analyze, such as GNU Chess, Crafty, or many others.
- Chess servers on the Internet, where you can connect to play chess with people from all over the world, watch other users play, or just hang out and chat.
- Correspondence chess played by electronic mail. The CMail program automates the tasks of parsing email from your opponent, playing his moves out on your board, and mailing your reply move after you've chosen it.

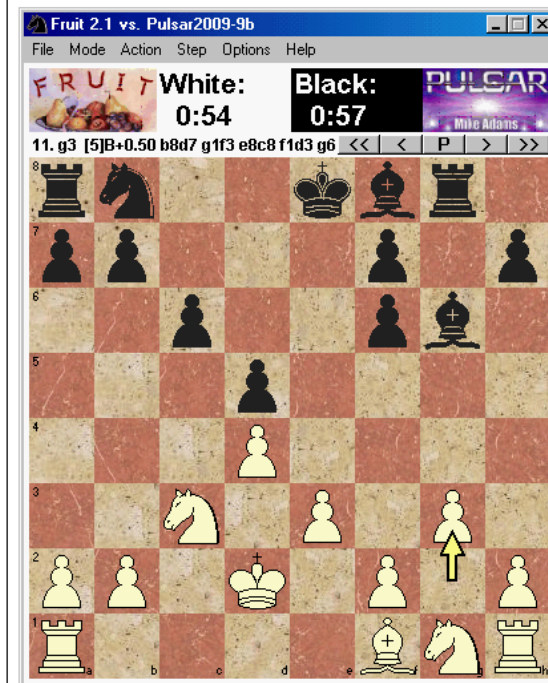
XBoard runs on Unix and Unix-like systems that use the X Window System.

More...

- [WinBoard](#) (MS Windows port)
- [Downloading](#)
- [Documentation](#)
- [On-line User Guide](#)
- [More Screenshots](#)
- [What's New in 4.8.0?](#) (stable)
- [What's New in 4.7.3?](#)
- [What's New in 4.7.2?](#)
- [What's New in 4.7.1?](#)
- [What's New in 4.7.0?](#)
- [What's New in 4.6.2?](#)
- [What's New in 4.6.1?](#) (faulty)
- [What's New in 4.6.0?](#)
- [What's New in 4.5.3?](#)
- [What's New in 4.5.2?](#)
- [What's New in 4.5.1?](#)
- [What's New in 4.5.0?](#)
- [What's New in 4.4.x?](#)
- [How can you help us?](#)
- [Frequently Asked Questions](#)



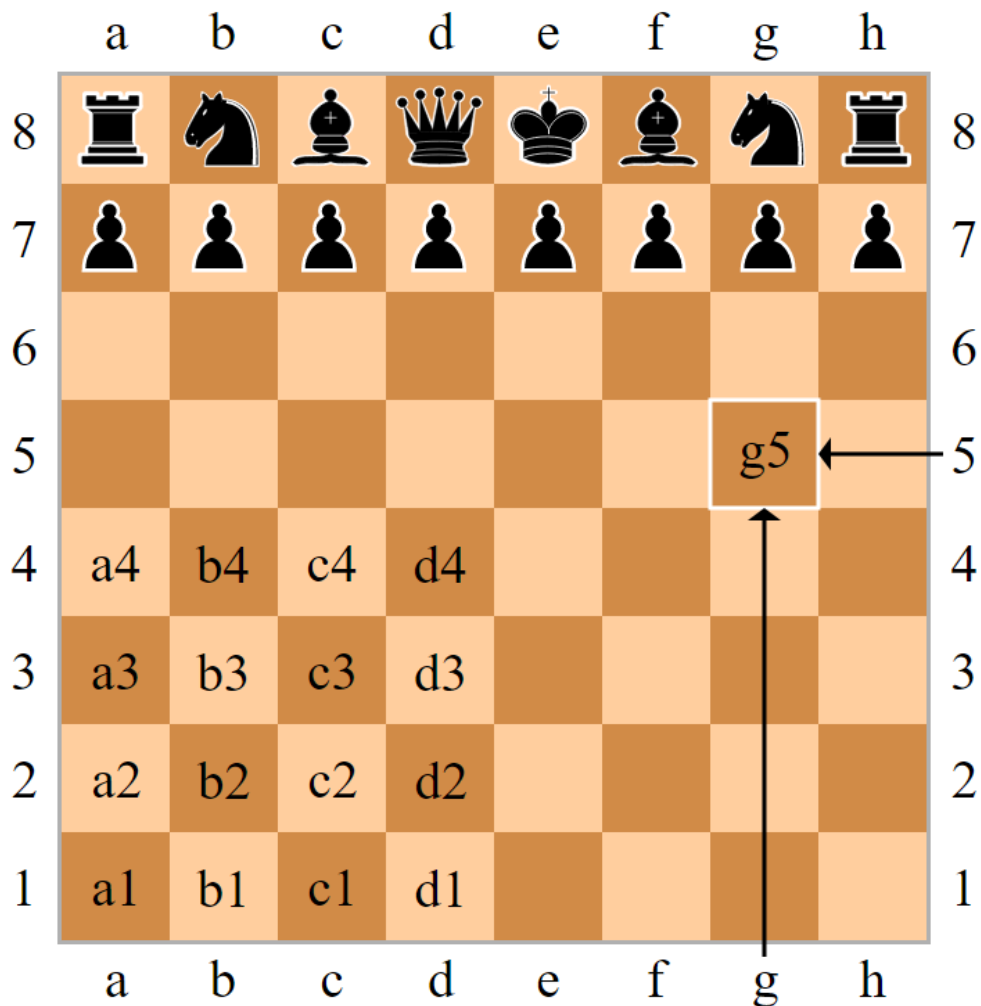
XBoard, playing some Chess variant



WinBoard, configured with marble board



Część II – własna implementacja



Część II – własna implementacja



```
[Event "F/S Return Match"]  
[Site "Belgrade, Serbia Yugoslavia|JUG"]  
[Date "1992.11.04"]  
[Round "29"]  
[White "Fischer, Robert J."]  
[Black "Spasky, Boris V."]  
[Result "1/2-1/2"]
```

```
1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 {This opening is called the Ruy Lopez.}  
4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6 8. c3 O-O 9. h3 Nb8 10. d4 Nbd7  
11. c4 c6 12. cxb5 axb5 13. Nc3 Bb7 14. Bg5 b4 15. Nb1 h6 16. Bh4 c5 17. dxe5  
Nxe4 18. Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6 21. Nc4 Nxc4 22. Bxc4 Nb6  
23. Ne5 Rae8 24. Bxf7+ Rxf7 25. Nxf7 Rxe1+ 26. Qxe1 Kxf7 27. Qe3 Qg5 28. Qxg5  
hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33. f3 Bc8 34. Kf2 Bf5  
35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39. Kd2 Kb5 40. Rd6 Kc5 41. Ra6  
Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```



Stan prac



- Przegląd i analiza dostępnych implementacji silników popularnych gier (Go, szachy, warcaby, itp.)
 - Ogólne rozeznanie /ZROBIONE
 - Wstępna analiza kodów źródłowych /W TRAKCIE
 - Spisanie wniosków, dokończenie analizy /WKRÓTCE
- Implementacja generatora silników wybranej gry
 - 0%
- 3. Analiza wydajnościowo, poprawnościowa zaimplementowanego generatora silników gier
 - 0%

Ocena zagrożeń



- Nowa tematyka (średnie)
- Niewielki postęp prac (wysokie)
- Wiedza o zastosowanych rozwiązaniach może okazać się niewystarczająca (niskie)
- Efektywna implementacja może okazać się znacznie trudniejsza niż się wydaje (średnie)

Wstępny harmonogram prac



- Koniec marca – zakończenie części I pracy – napisana analiza i porównanie silników gier logicznych, rozpoczęcie implementacji
- Koniec kwietnia – wstępna implementacja silnika wybranej gry zakończona, silnik działa ale niekoniecznie jest szybki, napisanie prototypowej wersji części II pracy
- Połowa maja – ukończona i dopracowana aplikacja testująca, szybkie domknięcie (poprawienie) części II pracy
- Połowa czerwca – napisana i ukończona część III pracy, poprawki
- Koniec czerwca – przygotowanie pracy do oddania, złożenie pracy



Podstawowa bibliografia



- 1. Czasopisma ICGA Journal
- 2. Strona olimpiady ICGA <https://icga.leidenuniv.nl>
- 3. Kody źródłowe programów grających
- 4. <http://www.grappa.univ-lille3.fr/icga/programs.php>
- 5. <http://stockfishchess.org>
- 6. <https://github.com/mcostalba/Stockfish>



Ok, jak to wygląda dzisiaj w praktyce

1. Rozdział opisowy – rozpoznanie trendów w silnikach programów grających JEST NIEZALEŻNY od problemu wyboru gry (czyli mógłby być napisany już dawno...)

Ok, jak to wygląda dzisiaj w praktyce

1. Wybór gry jest problematyczny – generalnie:
 1. Im bardziej złożona gra tym więcej wartości które możemy parametryzować - ZALETA
 2. Im więcej parametrów tym więcej pracy, więcej czasu i trudniej – WADA
 3. Gry dobrze znane jak warcaby mają dobre wsparcie ze strony community, dużo materiałów i dobre silniki żeby coś podpatrzyć – ZALETA. Ale trudniej jest też napisać coś szybszego, sprytniejszego, dostępne rozwiązania są bardzo złożone – WADA.
 4. Z drugiej strony wybranie gry mniej znanej jak Quoridor oznaczałoby prawdopodobnie implementację silnika wolnego i potrzeba napisania własnego GUI i tak dalej



Ok, jak to wygląda dzisiaj w praktyce

1. Wstępny wybór: szachy
2. Dlaczego?
 1. Bardzo dobre wsparcie ze strony społeczności
 2. Pełno materiałów, opracowań, artykułów naukowych
 3. Dostępne programy do GUI
 4. Ustandaryzowane protokoły komunikacji silników z GUI
 5. Bardzo dopracowane, bardzo wydajne i szeroko dostępne (otwarte) implementacje programów grających
 6. Dużo wielkości nadających się do parametryzowania
3. Co można parametryzować?...



Ok, jak to wygląda dzisiaj w praktyce

1. Co można parametryzować?

1. -liczbę pól w formacie $n \times m$ (czyli zarówno n jak i m , szachownica nie musi być kwadratowa)
2. -kto ma pierwszy ruch
3. -ile ruchów wykonuje osoba druga
4. -**ilość każdego typu figur** (szczególnie jeśli jakiegoś typu figury się nie pojawią (możliwość optymalizacji pamięci na stan gry))
5. -ilość pionów
6. -ilość bierek
7. -rozstawienie bierek
8. -zasady promocji
9. -może nawet zasady wygranej



Ok, jak to wygląda dzisiaj w praktyce

1. Lekka zmiana koncepcji?

Generalnie ta praca nie miała się skupiać na AI bo to bardzo rozległe zagadnienie.

Chodzi o przegląd dostępnych kodów silników, ich analiza i napisanie równie dobrego, a przy okazji możliwego do modyfikacji. Jakies testy poprawności i wydajności, ew. GUI i to już jest sporo.



Ok, jak to wygląda dzisiaj w praktyce

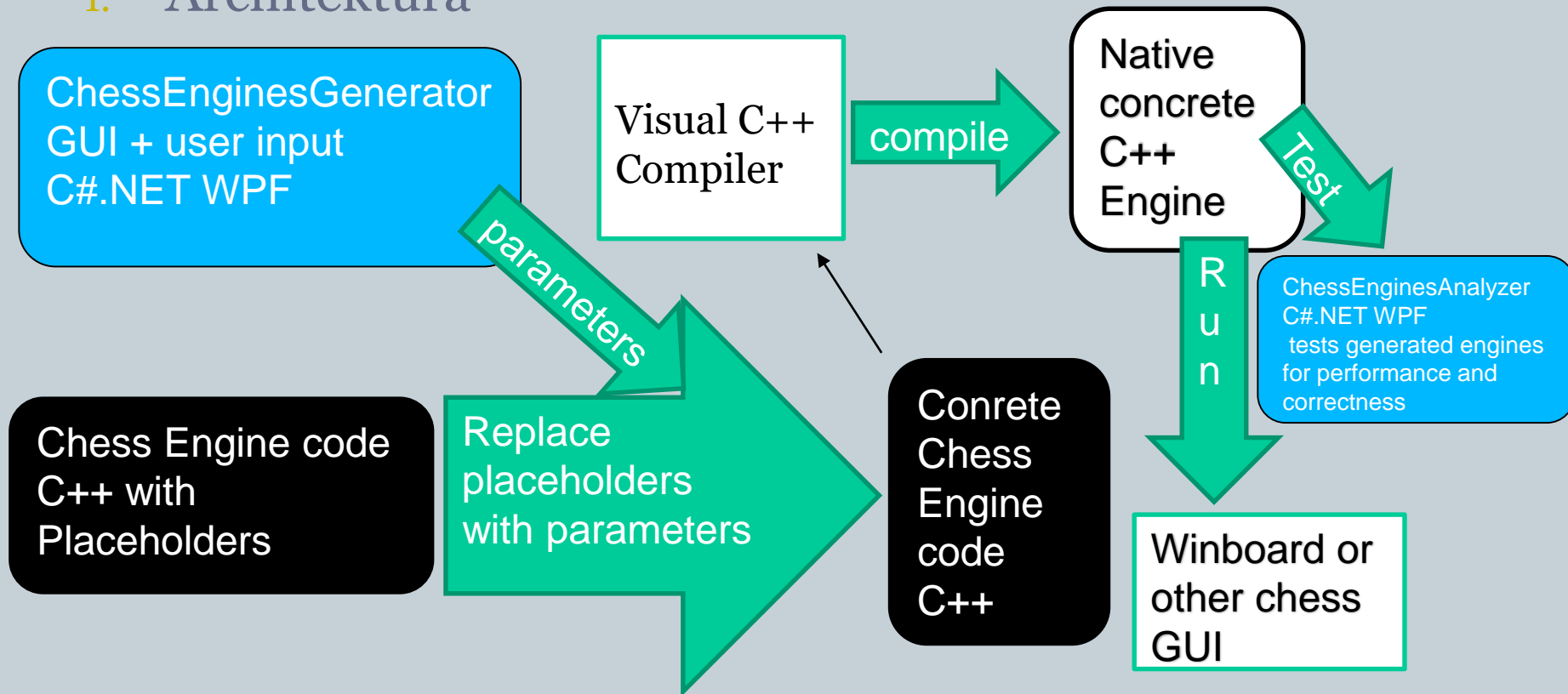
1. Architektura

1. Entry point is ChessEnginesGenerator with GUI
2. Generator creates new native game engine based on parameters and source code in C++ from another project
3. Some GUI app like WinBoard can run generated engine
4. Tester "ChessEnginesAnalyzer" tests generated engines for performance and correctness, also tests Generator app



Ok, jak to wygląda dzisiaj w praktyce

1. Architektura



Ok, jak to wygląda dzisiaj w praktyce

1. Rozdział testowy to tak naprawdę kwestia aplikacji testującej i jej wyjścia.
2. Aplikacja testująca powinna spełniać następujące warunki:
 1. Dobra implementacja protokołu komunikacyjnego rozgrywek szachowych
 2. Wyczerpujący zestaw test case'ów weryfikujących poprawności stanu gry
 3. Pomiar wydajności generatora z możliwie niewielkim błędem
 4. Sensowny output np. w postaci wykresów – tak żeby łatwo i szybko dało się to wstawić do pracy.
 5. Może nawet generowanie raportów w PDF/Doc

