Using Chess Programming in Computer Education

Dmitri A Gusev
Purdue University College of Technology
4444 Kelly Street
Columbus, IN 47203
812-348-2029
dgusev@purdue.edu

Abstract

Chess programming is an old and venerable branch of game development. In the modern computing environment, it encompasses the development of chess engines, chess user interfaces (UIs), chess tournament broadcast tools, chess databases, endgame tablebases (EGTBs) and opening books. In this paper, the author shares his multi-year experience of utilizing most of these components of chess programming in teaching undergraduate courses in computer science (CS) and computer and information technology (CIT). The courses at different levels were taught at Edinboro University of Pennsylvania and, most recently, at Purdue Polytechnic Columbus, formerly called Purdue College of Technology Columbus. They involved teaching programming in C, C++ and Java, as well as system administration. We will discuss the courses' structure and learning outcomes, the choices of tools and equipment, how the courses progressed, individual assignments and team projects, the lessons learned, the specifics of getting the students engaged, and the author's plans for the future work aimed at improvement of programming and computer networking skills of Purdue Polytechnic students via their involvement in different aspects of chess programming.

Introduction

Chess is a widely popular and well-respected turn-based centuries-old game played on an 8x8 board with white and black pieces (pawns, knights, bishops, rooks, queens, and kings) that move according to the rules that can be quickly learned even by young kids. The art of the game is not easy to master, though, so many software products have been developed to implement chess as a video game on numerous platforms and assist the beginner, amateur, and professional chess players alike. Some of these software products are free and open source, letting the developers learn by examining the existing solutions. Chess programming involves the development of the following products.

1. Chess engines capable of playing the game and analyzing its positions,
2. Chess user interfaces (UIs) that facilitate communication between the user and the engine(s) and/or databases,
3. Chess tournament broadcast tools that allow viewers to follow games online,
4. Chess databases, such as Shane's Chess Information Database (Scid),
5. Endgame tablebases (EGTBs), and

6. Opening books for chess engines to use early in the game and users to explore and learn.

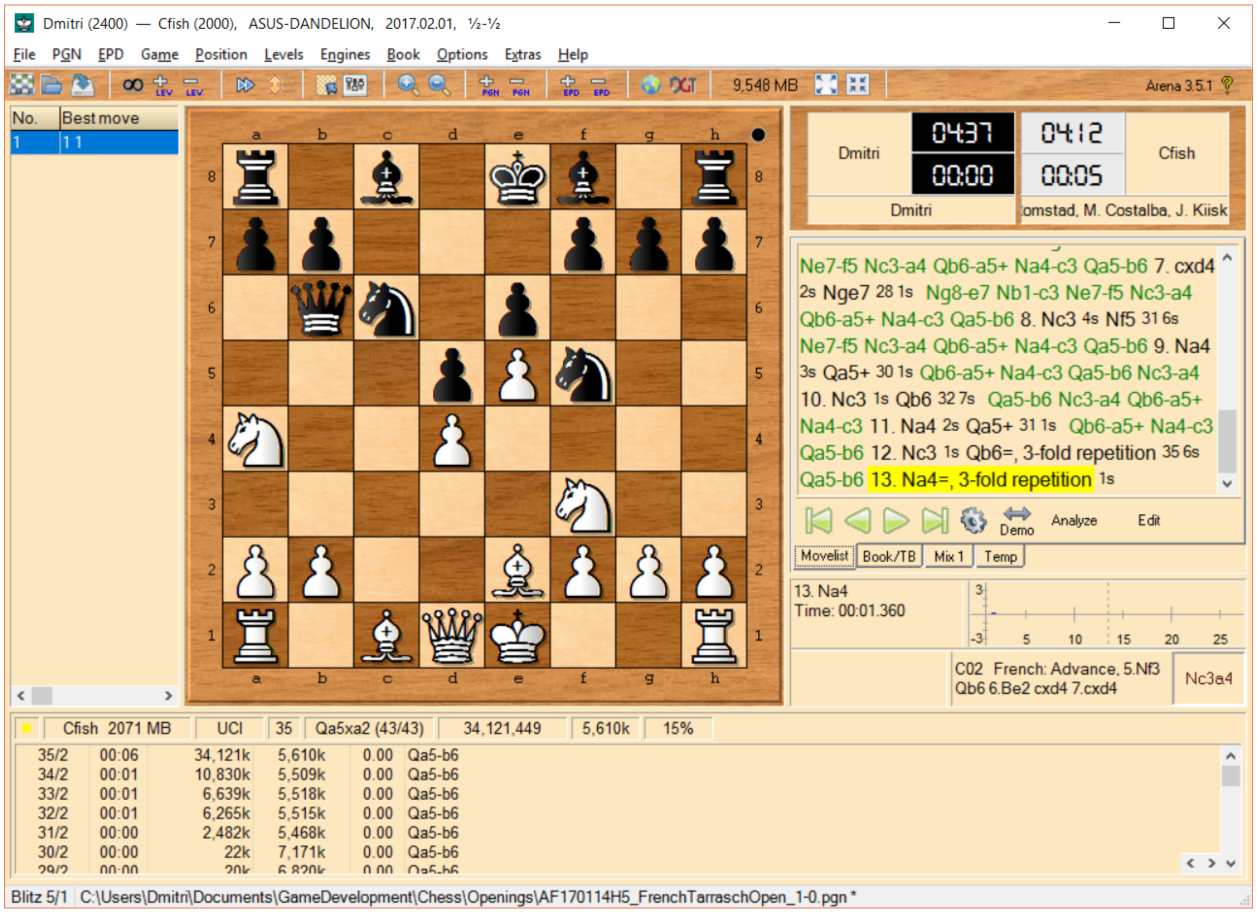
In this paper, the author shares his multi-year experience of utilizing most of the aforementioned components of chess programming in teaching undergraduate courses in computer science (CS) and computer and information technology (CIT). The courses were taught at Edinboro University of Pennsylvania and, most recently, at Purdue Polytechnic Columbus, formerly called Purdue College of Technology Columbus. Purdue Polytechnic Institute (PPI) of Purdue University has a statewide system that offers programs throughout the state of Indiana. One of the main goals of the system is to make technology programs available throughout the state of Indiana. One of such program majors supported in Columbus, Indiana, is computer and information technology (CIT). At each statewide site, Purdue Polytechnic collaborates with a local college that provides non-major classes such as English, Mathematics, etc. The partner of Purdue Polytechnic Columbus is Indiana University Purdue University Columbus (IUPUC). Purdue Polytechnic has its own admissions, programs and recruitment activities in the community.

In the next sections of the paper, we will discuss the courses' structure and learning outcomes, the choices of tools and equipment, how the courses progressed, individual assignments and team projects, the lessons learned, and the specifics of getting the students engaged. The courses involved teaching programming in C, C++ and Java, as well as system administration. They will be grouped into sections according to their level, from 100 to 400. Finally, the author will outline his plans for the future work aimed at improvement of programming and computer networking skills of Purdue Polytechnic students through involving them in different aspects of chess programming.

Chess in a 100-Level Programming Course

CNIT 105 Introduction to C Programming was a 3-credit-hour course offered in Spring 2017 to non-CIT majors – the undergraduate students of robotics engineering technology (RET) and mechanical engineering technology (MET) at Purdue Polytechnic Columbus. Fifteen students enrolled. The students were required to use Code::Blocks 16.01 with the MinGW-w64 v. 6.2.0 compiler to produce accurately functioning 64-bit C applications. The first learning objective of the course being to be able to compile and run a program using the C language, chess software came in handy for the first of the two lab exams, each of which cost 100 points, or ~15% of the final grade.

To show the power of the tools that the students were mastering, the objective of Lab Exam 1 was chosen for them to demonstrate ability to use a modern IDE to compile, debug, run, and configure a multi-file C program. That program was Cfish (2017), the fruit of translation of Stockfish, the strongest free, open-source chess engine in the world, from C++ into C by Ronald de Man to speed up the compile. The students configured Cfish under Arena (2017), a free chess GUI. For 5 points of extra credit, the students were asked to set the Arena timer to 5 minutes and play an optional quick game of chess against Cfish in the lab. The students who did not know how to play chess were allowed to ask the instructor for help and/or follow hints from Cfish. Twelve students received the extra credit, even though only a couple of them had requested the instructor's assistance to make valid moves. This lab exam did not involve programming; it was given during Week 4 of the semester. Figure 1 shows the instructor's screenshot of Arena running Cfish that was used as part of the lab exam assignment.


Dmitri (2400) — Cfish (2000), ASUS-DANDELION, 2017.02.01, ½-½
File PGN EPD Game Position Levels Engines Book Options Extras Help
9,548 MB
Arena 3.5.1
No. Best move
1 1 1
Dmitri
04:37
00:00
04:12
00:05
Cfish
omstad, M. Costalba, J. Kiisk
Analyze
Edit
Demo
Movelist
Book/TB
Mix 1
Temp

```
Ne7-f5 Nc3-a4 Qb6-a5+ Na4-c3 Qa5-b6 7. cxd4
2s Nge7 28 1s  Ng8-e7 Nb1-c3 Ne7-f5 Nc3-a4
Qb6-a5+ Na4-c3 Qa5-b6 8. Nc3 4s Nf5 31 6s
Ne7-f5 Nc3-a4 Qb6-a5+ Na4-c3 Qa5-b6 9. Na4
3s Qa5+ 30 1s Qb6-a5+ Na4-c3 Qa5-b6 Nc3-a4
10. Nc3 1s Qb6 32 7s  Qa5-b6 Nc3-a4 Qb6-a5+
Na4-c3 11. Na4 2s Qa5+ 31 1s  Qb6-a5+ Na4-c3
Qa5-b6 12. Nc3 1s Qb6=, 3-fold repetition 35 6s
Qa5-b6 13. Na4=, 3-fold repetition 1s
```

```
13. Na4
Time: 00:01.360
```

```
C02  French: Advance, 5.Nf3
Qb6 6.Be2 cxd4 7.cxd4
Nc3a4
```

```
Cfish  2071 MB   UCI   35   Qa5xa2 (43/43)   34,121,449   5,610k   15%

35/2   00:06   34,121k   5,610k   0.00  Qa5-b6
34/2   00:01   10,830k   5,509k   0.00  Qa5-b6
33/2   00:01    6,639k   5,518k   0.00  Qa5-b6
32/2   00:01    6,265k   5,515k   0.00  Qa5-b6
31/2   00:00    2,482k   5,468k   0.00  Qa5-b6
30/2   00:00       22k   7,171k   0.00  Qa5-b6
29/2   00:00       20k   6,820k   0.00  Qa5-b6
```

```
Blitz 5/1  C:\Users\Dmitri\Documents\GameDevelopment\Chess\Openings\AF170114H5_FrenchTarraschOpen_1-0.pgn *
```

Figure 1. Cfish playing under Arena: A screenshot from the lab exam assignment.

Chess in a 200-Level Programming Course

The author's first use of chess programming in the academia took place in Spring 2012, when he taught CSCI 230 Principles of Programming II, an undergraduate course at Edinboro University of Pennsylvania. This was the second course in a three-course sequence that taught computer science majors programming in C++. Sixteen students enrolled. Some of those students were members of the Edinboro Chess Club, for which the instructor served as the faculty advisor. In addition to two midterm exams and a final exam, the students were given 12 lab assignments that determined 15% of the final grade, 6 homework assignments (18%), and 4 programs to write (30% of the final grade).

The primary course objective was for the students to be able to "employ the principles of software engineering to design high-quality, structured programs," so I made a decision to base the programming assignments on the code from two famous free, open-source chess engines: Stockfish (2012) by Tord Romstad, Marco Costalba, and Joona Kiiski and Kaissa 1.00 by G.M. Adelson-Velsky, V.L. Arlazarov, and M.V. Donskoy. The code archive of the latter program was received by email from Mr. Anthon Dubetz of DISCo (the now defunct Donskoy Interactive Software Company), along with the written permission to use it.

The first programming assignment directed students to develop a C++ program that would compute all chess proto-moves and output them as a C++ statement that declared and initialized a multi-dimensional array. The students were informed that the resulting statement would be included in the next program, so it had to obey the C++ syntax for multi-dimensional arrays.

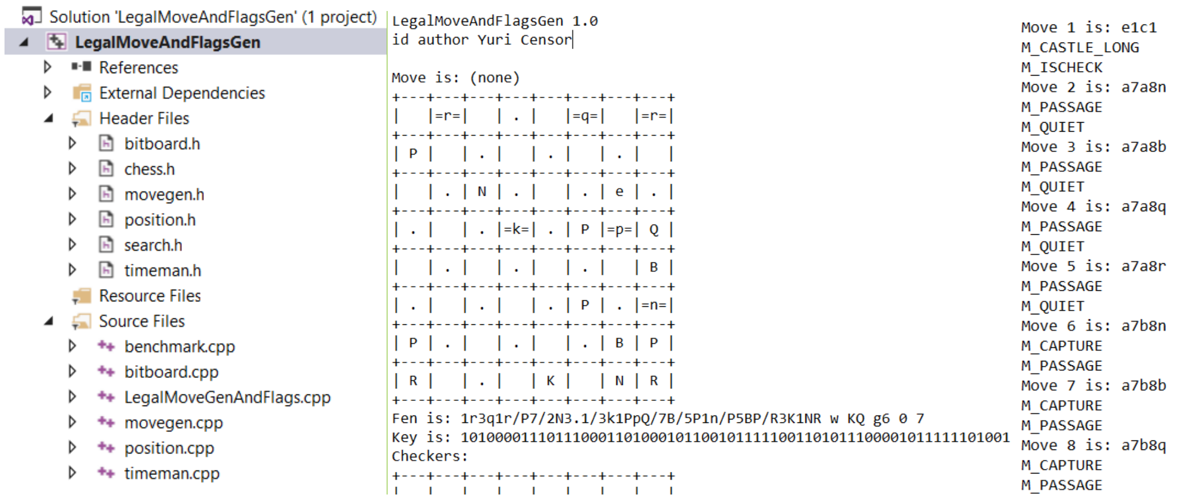
For the second programming assignment, the students were provided a starter project derived from Stockfish 2.2.2 and Kaissa 1.00. Their task was to complete LegalMoveGen, a C++ program that would use the pre-computed multi-dimensional array of proto-moves to produce an array-based list of all legal chess moves for a user-specified position and output the list using the provided **move_to_uci()** function.

The third programming assignment was to complete LegalMoveAndFlagsGen, a C++ program that would:

1. Set all move flags correctly for each legal chess move generated and
2. Output the detailed list of legal moves using the provided **move_to_uci()** function and a new function **print_flags()** that would report how the move flags were set for each legal move.

Finally, the fourth programming assignment asked the students to modify LegalMoveAndFlagsGen in order to complete OneMoveWonder, a UCI chess engine that would make random legal moves when installed in the Arena interface. This last assignment proved too difficult for the students to complete.

Figure 2 shows the multi-file structure and part of the output of LegalMoveAndFlagsGen for a test position provided in the popular FEN format.



```
Solution 'LegalMoveAndFlagsGen' (1 project)
  LegalMoveAndFlagsGen
    References
    External Dependencies
    Header Files
      bitboard.h
      chess.h
      movegen.h
      position.h
      search.h
      timeman.h
    Resource Files
    Source Files
      benchmark.cpp
      bitboard.cpp
      LegalMoveGenAndFlags.cpp
      movegen.cpp
      position.cpp
      timeman.cpp
```

```
LegalMoveAndFlagsGen 1.0
id author Yuri Censor

Move is: (none)
+---+---+---+---+---+---+---+---+
|   |=r=|   | . |   |=q=|   |=r=|
+---+---+---+---+---+---+---+---+
| P |   | . |   | . |   | . |   |
+---+---+---+---+---+---+---+---+
|   | . | N | . |   | . | e | . |
+---+---+---+---+---+---+---+---+
| . |   | . |=k=| . | P |=p=| Q |
+---+---+---+---+---+---+---+---+
|   | . |   | . |   | . |   | B |
+---+---+---+---+---+---+---+---+
| . |   | . |   | . | P | . |=n=|
+---+---+---+---+---+---+---+---+
| P | . |   | . |   | . | B | P |
+---+---+---+---+---+---+---+---+
| R |   | . |   | K |   | N | R |
+---+---+---+---+---+---+---+---+
Fen is: 1r3q1r/P7/2N3.1/3k1PpQ/7B/5P1n/P5BP/R3K1NR w KQ g6 0 7
Key is: 1010000111011100011010001011001011111001101011100001011111101001
Checkers:
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
```

```
Move 1 is: e1c1
M_CASTLE_LONG
M_ISCHECK
Move 2 is: a7a8n
M_PASSAGE
M_QUIET
Move 3 is: a7a8b
M_PASSAGE
M_QUIET
Move 4 is: a7a8q
M_PASSAGE
M_QUIET
Move 5 is: a7a8r
M_PASSAGE
M_QUIET
Move 6 is: a7b8n
M_CAPTURE
M_PASSAGE
Move 7 is: a7b8b
M_CAPTURE
M_PASSAGE
Move 8 is: a7b8q
M_CAPTURE
M_PASSAGE
```

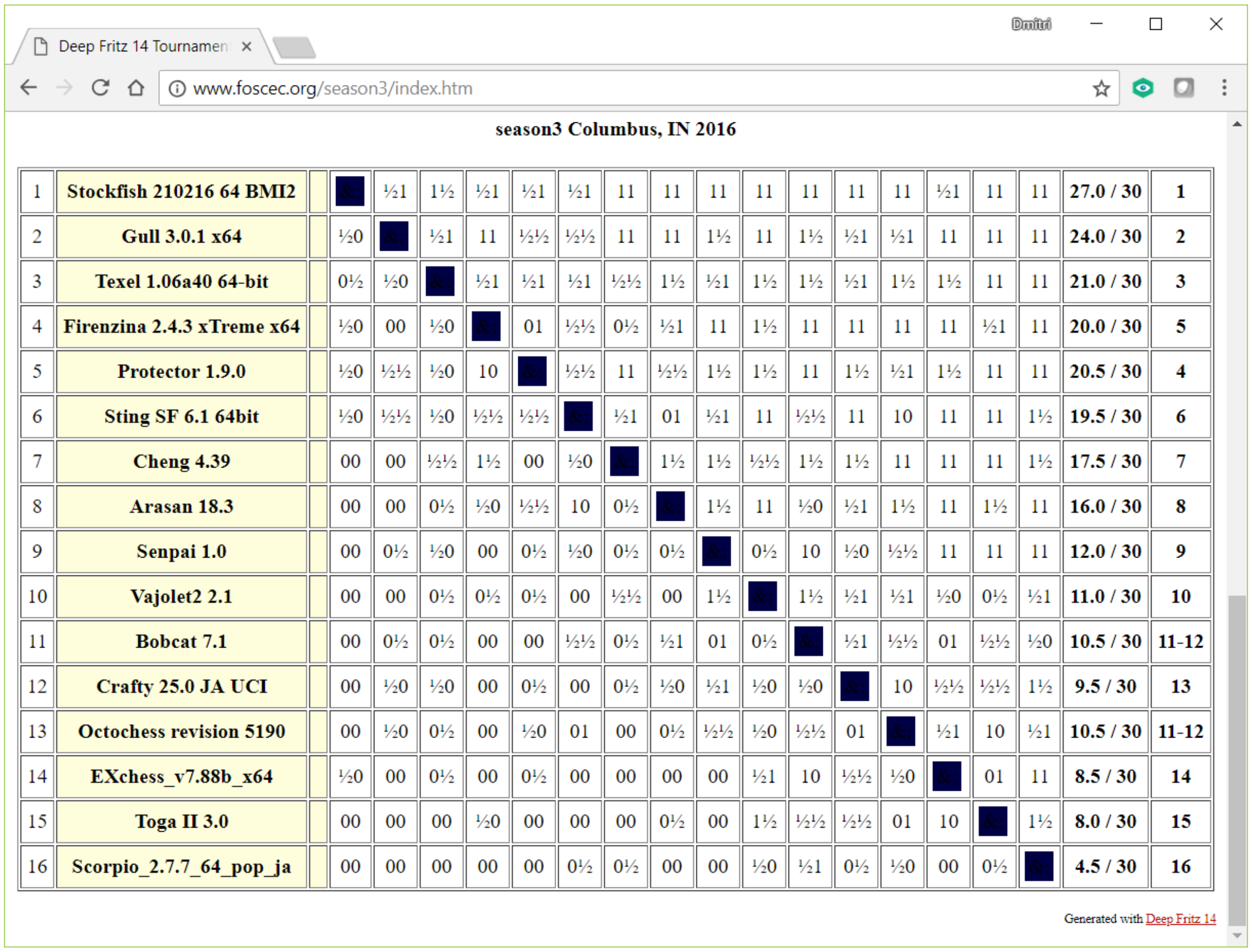
Figure 2. Structure and partial output of LegalMoveAndFlagsGen (Program 3).

Chess in a 200-Level Networking Course

When system administrators have to deal with free, open-source software, one of the common tasks that they are likely to face in their line of work is to compile an executable from the source code and install or otherwise configure the program. Another important thing for the future system administrators to learn is how computers communicate with remote servers via the ports dedicated to FTP and

SFTP connections. CNIT 242 System Administration is a required course in computer networking for CIT undergraduate students at Purdue Polytechnic Columbus. It is taught every spring semester using the resources of Kuber Maharjan Networking Lab housed in the Advanced Manufacturing Center of Excellence. Most recently, in Spring 2018, 12 students initially enrolled in the course, and one of them subsequently withdrew from it. In the spring semesters of 2014, 2015, 2016, 2017 and 2018, the instructor assigned a team project involving a broadcast of a double round-robin tournament called Purdue free open-source chess engine contest (FOSCEC) on the Web. Each time, an effort was made to compile as many of the engines from sources as possible. "Factory" compiles provided by the engine authors were used for the engines that did not compile from the sources, or if the in-house compiles did not run correctly. The tournament games and results are available online at (FOSCEC 2014).

The first three seasons of FOSCEC were played using Deep Fritz 14 (2014), a commercial GUI acquired for the course by the Columbus site. Figure 3 shows the results of the last of those three seasons. The engines are not listed in the table according to their ranking in the tournament, because the table had to be edited manually to account for several engine crashes that Deep Fritz 14 did not automatically count as losses.



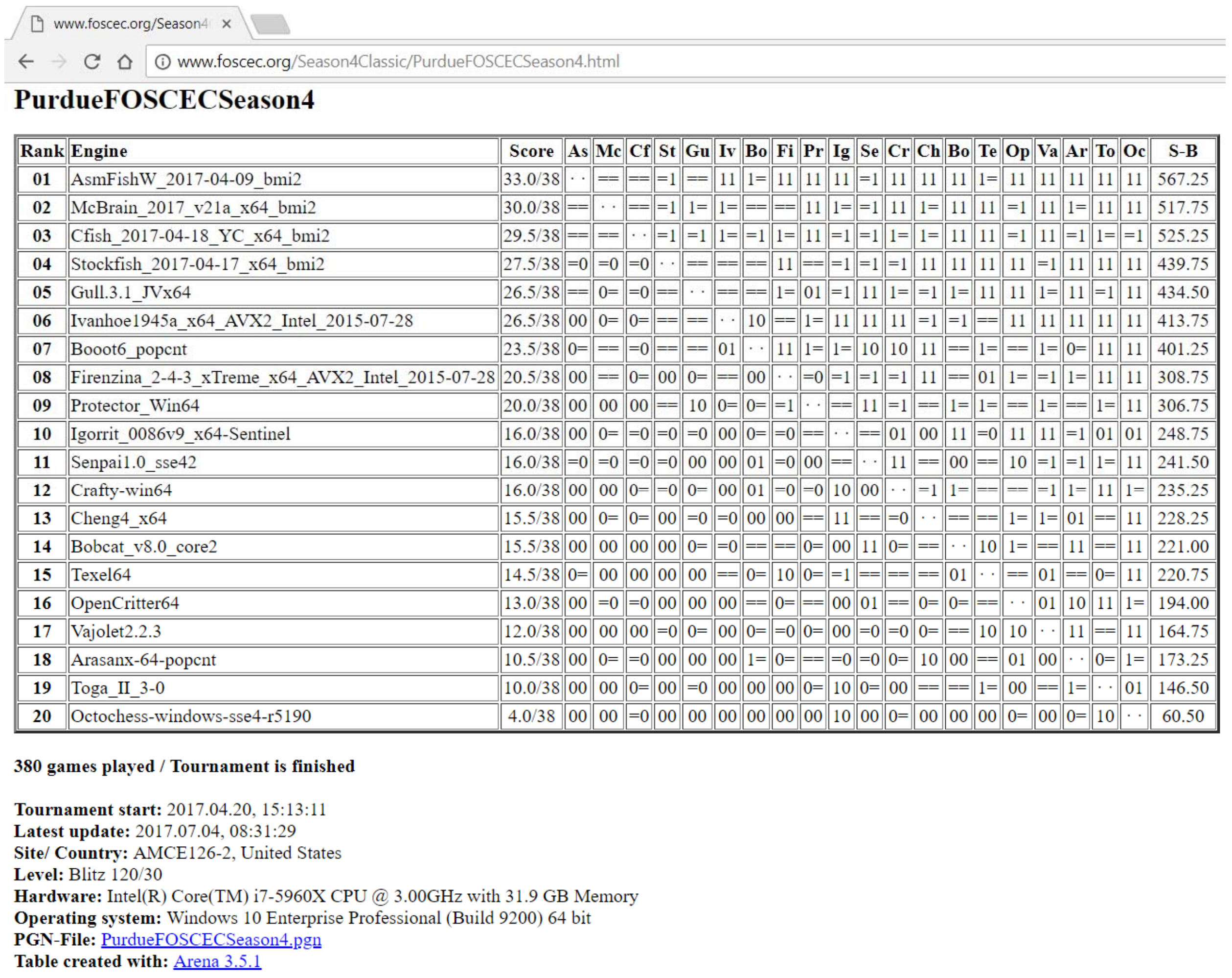
season3 Columbus, IN 2016

1	**Stockfish 210216 64 BMI2**		■	½1	1½	½1	½1	½1	11	11	11	11	11	11	11	½1	11	11	**27.0 / 30**	**1**
2	**Gull 3.0.1 x64**		½0	■	½1	11	½½	½½	11	11	1½	11	1½	½1	½1	11	11	11	**24.0 / 30**	**2**
3	**Texel 1.06a40 64-bit**		0½	½0	■	½1	½1	½1	½½	1½	½1	1½	1½	½1	1½	1½	11	11	**21.0 / 30**	**3**
4	**Firenzina 2.4.3 xTreme x64**		½0	00	½0	■	01	½½	0½	½1	11	1½	11	11	11	11	½1	11	**20.0 / 30**	**5**
5	**Protector 1.9.0**		½0	½½	½0	10	■	½½	11	½½	1½	1½	11	1½	½1	1½	11	11	**20.5 / 30**	**4**
6	**Sting SF 6.1 64bit**		½0	½½	½0	½½	½½	■	½1	01	½1	11	½½	11	10	11	11	1½	**19.5 / 30**	**6**
7	**Cheng 4.39**		00	00	½½	1½	00	½0	■	1½	1½	½½	1½	1½	11	11	11	1½	**17.5 / 30**	**7**
8	**Arasan 18.3**		00	00	0½	½0	½½	10	0½	■	1½	11	½0	½1	1½	11	1½	11	**16.0 / 30**	**8**
9	**Senpai 1.0**		00	0½	½0	00	0½	½0	0½	0½	■	0½	10	½0	½½	11	11	11	**12.0 / 30**	**9**
10	**Vajolet2 2.1**		00	00	0½	0½	0½	00	½½	00	1½	■	1½	½1	½1	½0	0½	½1	**11.0 / 30**	**10**
11	**Bobcat 7.1**		00	0½	0½	00	00	½½	0½	½1	01	0½	■	½1	½½	01	½½	½0	**10.5 / 30**	**11-12**
12	**Crafty 25.0 JA UCI**		00	½0	½0	00	0½	00	0½	½0	½1	½0	½0	■	10	½½	½½	1½	**9.5 / 30**	**13**
13	**Octochess revision 5190**		00	½0	0½	00	½0	01	00	0½	½½	½0	½½	01	■	½1	10	½1	**10.5 / 30**	**11-12**
14	**EXchess_v7.88b_x64**		½0	00	0½	00	0½	00	00	00	00	½1	10	½½	½0	■	01	11	**8.5 / 30**	**14**
15	**Toga II 3.0**		00	00	00	½0	00	00	00	0½	00	1½	½½	½½	01	10	■	1½	**8.0 / 30**	**15**
16	**Scorpio_2.7.7_64_pop_ja**		00	00	00	00	00	0½	0½	00	00	½0	½1	0½	½0	00	0½	■	**4.5 / 30**	**16**

Generated with Deep Fritz 14

Figure 3. Screenshot of the table of the results of Purdue FOSCEC Season 3 (Spring 2016).

In an attempt to reduce crashes and improve broadcast quality, Season 4 was played using Arena 3.5.1, a free chess GUI routinely used to test the engine compiles, in a combination with another freebie, Tom's Live Chess Viewer (TLCV 2017). Figure 4 features the results of Purdue FOSCEC Season 4. Unfortunately, due to the quirks in the Arena configuration, three engines — Stockfish, Texel and Arasan — were misconfigured to use only one thread instead of the usual 16 on an 8-core Windows workstation with hyper-threading on. In another deficiency of the Season 4 setup, TLCV does not have the ability to broadcast to mobile devices, such as smartphones or tablets.



www.foscec.org/Season4Classic/PurdueFOSCECSeason4.html

PurdueFOSCECSeason4

Rank	Engine	Score	As	Mc	Cf	St	Gu	Iv	Bo	Fi	Pr	Ig	Se	Cr	Ch	Bo	Te	Op	Va	Ar	To	Oc	S-B
01	AsmFishW_2017-04-09_bmi2	33.0/38	· ·	==	==	=1	==	11	1=	11	11	11	=1	11	11	11	1=	11	11	11	11	11	567.25
02	McBrain_2017_v21a_x64_bmi2	30.0/38	==	· ·	==	=1	1=	1=	==	==	11	1=	=1	11	1=	11	11	=1	11	1=	11	11	517.75
03	Cfish_2017-04-18_YC_x64_bmi2	29.5/38	==	==	· ·	=1	=1	1=	=1	1=	11	=1	=1	1=	1=	11	11	=1	11	=1	1=	=1	525.25
04	Stockfish_2017-04-17_x64_bmi2	27.5/38	=0	=0	=0	· ·	==	==	==	11	==	=1	=1	=1	11	11	11	11	=1	11	11	11	439.75
05	Gull.3.1_JVx64	26.5/38	==	0=	=0	==	· ·	==	==	1=	01	=1	11	1=	=1	1=	11	11	1=	11	=1	11	434.50
06	Ivanhoe1945a_x64_AVX2_Intel_2015-07-28	26.5/38	00	0=	0=	==	==	· ·	10	==	1=	11	11	11	=1	=1	==	11	11	11	11	11	413.75
07	Booot6_popcnt	23.5/38	0=	==	=0	==	==	01	· ·	11	1=	1=	10	10	11	==	1=	==	1=	0=	11	11	401.25
08	Firenzina_2-4-3_xTreme_x64_AVX2_Intel_2015-07-28	20.5/38	00	==	0=	00	0=	==	00	· ·	=0	=1	=1	=1	11	==	01	1=	=1	1=	11	11	308.75
09	Protector_Win64	20.0/38	00	00	00	==	10	0=	0=	=1	· ·	==	11	=1	==	1=	1=	==	1=	==	1=	11	306.75
10	Igorrit_0086v9_x64-Sentinel	16.0/38	00	0=	=0	=0	=0	00	0=	=0	==	· ·	==	01	00	11	=0	11	11	=1	01	01	248.75
11	Senpai1.0_sse42	16.0/38	=0	=0	=0	=0	00	00	01	=0	00	==	· ·	11	==	00	==	10	=1	=1	1=	11	241.50
12	Crafty-win64	16.0/38	00	00	0=	=0	0=	00	01	=0	=0	10	00	· ·	=1	1=	==	==	=1	1=	11	1=	235.25
13	Cheng4_x64	15.5/38	00	0=	0=	00	=0	=0	00	00	==	11	==	=0	· ·	==	==	1=	1=	01	==	11	228.25
14	Bobcat_v8.0_core2	15.5/38	00	00	00	00	0=	=0	==	==	0=	00	11	0=	==	· ·	10	1=	==	11	==	11	221.00
15	Texel64	14.5/38	0=	00	00	00	00	==	0=	10	0=	=1	==	==	==	01	· ·	==	01	==	0=	11	220.75
16	OpenCritter64	13.0/38	00	=0	=0	00	00	00	==	0=	==	00	01	==	0=	0=	==	· ·	01	10	11	1=	194.00
17	Vajolet2.2.3	12.0/38	00	00	00	=0	0=	00	0=	=0	0=	00	=0	=0	0=	==	10	10	· ·	11	==	11	164.75
18	Arasanx-64-popcnt	10.5/38	00	0=	=0	00	00	00	1=	0=	==	=0	=0	0=	10	00	==	01	00	· ·	0=	1=	173.25
19	Toga_II_3-0	10.0/38	00	00	0=	00	=0	00	00	00	0=	10	0=	00	==	==	1=	00	==	1=	· ·	01	146.50
20	Octochess-windows-sse4-r5190	4.0/38	00	00	=0	00	00	00	00	00	00	10	00	0=	00	00	00	0=	00	0=	10	· ·	60.50

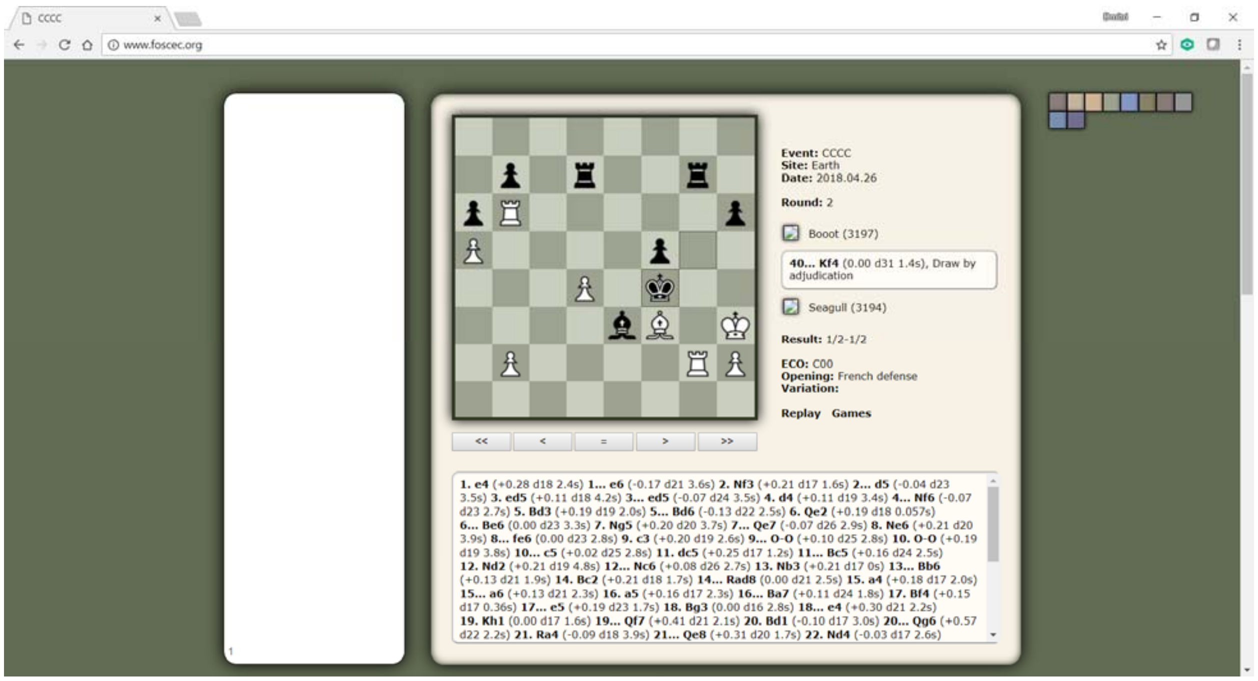
380 games played / Tournament is finished

Tournament start: 2017.04.20, 15:13:11
Latest update: 2017.07.04, 08:31:29
Site/ Country: AMCE126-2, United States
Level: Blitz 120/30
Hardware: Intel(R) Core(TM) i7-5960X CPU @ 3.00GHz with 31.9 GB Memory
Operating system: Windows 10 Enterprise Professional (Build 9200) 64 bit
PGN-File: PurdueFOSCECSeason4.pgn
Table created with: Arena 3.5.1

Figure 4. Screenshot of the table of the results of Purdue FOSCEC Season 4.

To alleviate the problems experienced before, the instructor decided to use Norman Schmidt's free, open-source CCCC (2018) to broadcast Purdue FOSCEC Season 5. By this point in the Spring 2018 semester, a test tournament of two engines was run at a short time control and broadcast to the FOS-CEC website. Figure 5 displays a screenshot that features the second game of the completed test tournament. We are planning to use CCCC to broadcast a double round-robin tournament of 20 free, open-source chess engines at the classic time control of 120 minutes + 30 seconds added per move for the whole game. This time control was common in early seasons of TCEC (2018), the tournament often regarded as the Unofficial World Computer Chess Championship. (Unlike FOSCEC, TCEC allows participation of commercial engines, private engines, and free engines that are not open-source, alongside the free, open-source ones.) For each engine, the number of threads should be set to 16. The hash size is set to 4096M for all engines that can handle it (Booot supports up to 2048M), and the ponder option is set to OFF.

The eleven students were divided into three teams. A team of four students was responsible for compiling as many engines from sources as possible and testing the compiles under the Arena GUI. Sixteen engines were compiled and tested successfully, and working "factory" compiles were located for the other four. Two teams of three students each were responsible for the front end (an 8-core Windows workstation) and the back end (an Apache web server running on a virtual Ubuntu server), respectively.



Event: CCCC
Site: Earth
Date: 2018.04.26

Round: 2

Booot (3197)

40... Kf4 (0.00 d31 1.4s), Draw by adjudication

Seagull (3194)

Result: 1/2-1/2

ECO: C00
Opening: French defense
Variation:

1. e4 (+0.28 d18 2.4s) **1... e6** (-0.17 d21 3.6s) **2. Nf3** (+0.21 d17 1.6s) **2... d5** (-0.04 d23 3.5s) **3. ed5** (+0.11 d18 4.2s) **3... ed5** (-0.07 d24 3.5s) **4. d4** (+0.11 d19 3.4s) **4... Nf6** (-0.07 d23 2.7s) **5. Bd3** (+0.19 d19 2.0s) **5... Bd6** (-0.13 d22 2.5s) **6. Qe2** (+0.19 d18 0.057s) **6... Be6** (0.00 d23 3.3s) **7. Ng5** (+0.20 d20 3.7s) **7... Qe7** (-0.07 d26 2.9s) **8. Ne6** (+0.21 d20 3.9s) **8... fe6** (0.00 d23 2.8s) **9. c3** (+0.20 d19 2.6s) **9... O-O** (+0.10 d25 2.8s) **10. O-O** (+0.19 d19 3.8s) **10... c5** (+0.02 d25 2.8s) **11. dc5** (+0.25 d17 1.2s) **11... Bc5** (+0.16 d24 2.5s) **12. Nd2** (+0.21 d19 4.8s) **12... Nc6** (+0.08 d26 2.7s) **13. Nb3** (+0.21 d17 0s) **13... Bb6** (+0.13 d21 1.9s) **14. Bc2** (+0.21 d18 1.7s) **14... Rad8** (0.00 d21 2.5s) **15. a4** (+0.18 d17 2.0s) **15... a6** (+0.13 d21 2.3s) **16. a5** (+0.16 d17 2.3s) **16... Ba7** (+0.11 d24 1.8s) **17. Bf4** (+0.15 d17 0.36s) **17... e5** (+0.19 d23 1.7s) **18. Bg3** (0.00 d16 2.8s) **18... e4** (+0.30 d21 2.2s) **19. Kh1** (0.00 d17 1.6s) **19... Qf7** (+0.41 d21 2.1s) **20. Bd1** (-0.10 d17 3.0s) **20... Qg6** (+0.57 d22 2.2s) **21. Ra4** (-0.09 d18 3.9s) **21... Qe8** (+0.31 d20 1.7s) **22. Nd4** (-0.03 d17 2.6s)

Figure 5. Screenshot of the CCCC-based Purdue FOSCEC Season 5 website.

Chess in a 300-Level Programming Course

CNIT 325 Object-Oriented Application Development is an undergraduate CIT course at Purdue Polytechnic Columbus that focuses on using Java in the development of modern business applications. Topics include object-oriented design, encapsulation, object interfaces, inheritance, aggregation, abstract classes, polymorphism, data structures, and exception handling.

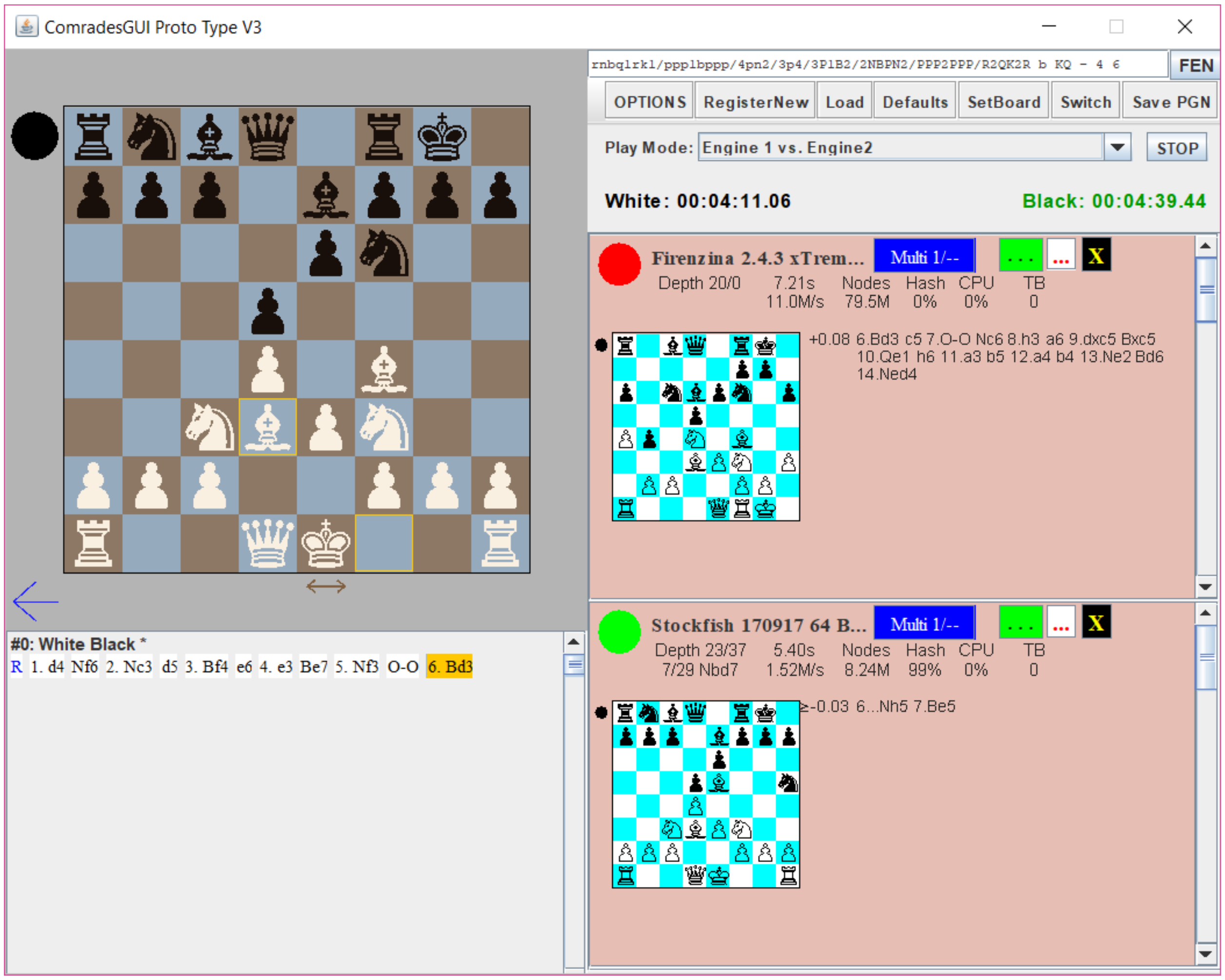
In the Fall 2017 semester, a team of four students successfully completed a project to modify a public domain ComradesGUI chess interface and accomplish the following.

1. Add two timers to the GUI. Once the game is started from a given position, the timer of the side to move will be ticking. Once a move is made, the first timer will be paused and the other side's timer will be activated.

2. Provide means to the user to set the timers initially.

3. Add a Play button that would start a game, provided that at least one chess engine is loaded. The following modes of play are supported: Human vs. Engine1, Engine1 vs. Engine1, Engine 1 vs. Engine2.
4. Once the game begins, GUI will issue UCI commands to the loaded engines to tell them to think and make their moves.
5. Implement game end detection.
6. Make sure that the games can be saved in the PGN format.

The resulting new version was released at the Firenzina website (2017). Figure 6 features a screenshot of two chess engines playing a game under ComradesGUI Proto Type V3. Prior to this release, ComradesGUI worked in the infinite analysis mode only.



```
rnbqlrkl/ppplbppp/4pn2/3p4/3P1B2/2NBPN2/PPP2PPP/R2QK2R b KQ - 4 6
```

Play Mode: Engine 1 vs. Engine2

White: 00:04:11.06   Black: 00:04:39.44

```
Firenzina 2.4.3 xTrem...
Depth 20/0     7.21s     Nodes   Hash   CPU    TB
               11.0M/s   79.5M   0%     0%     0

+0.08 6.Bd3 c5 7.O-O Nc6 8.h3 a6 9.dxc5 Bxc5
      10.Qe1 h6 11.a3 b5 12.a4 b4 13.Ne2 Bd6
      14.Ned4
```

```
Stockfish 170917 64 B...
Depth 23/37    5.40s     Nodes   Hash   CPU    TB
7/29 Nbd7      1.52M/s   8.24M   99%    0%     0

≥-0.03 6...Nh5 7.Be5
```

#0: White Black *

R 1. d4 Nf6 2. Nc3 d5 3. Bf4 e6 4. e3 Be7 5. Nf3 O-O 6. Bd3

Figure 6. Screenshot of two chess engines playing a game under ComradesGUI Proto Type V3.

Chess in a 400-Level Programming Course

As we reported previously (Gusev & Swanson, 2017), a free open-source chess engine called Firenzina was ported for Android by an undergraduate CIT student who took CNIT 425 Software Development for Mobile Computers II in Spring 2014. Figure 7 features a screenshot of Firenzina playing a game of chess against Stockfish on an Android smartphone under the auspices of Chess for All, a free app.

In the most recent Rapidroid rating by Chesstroid (2018), Firenzina 2.4.3 xTreme for Android was ranked in the Top 10 as of January of 2018, as shown in Figure 8. Firenzina 2.4.1 xTreme for Android was released as #4 according to the very first Rapidroid rating released September 14, 2014, as demonstrated in Figure 9.

Our main challenge in the 400-level courses remains low enrollment. This makes it difficult to give a team project to a team of a sufficient size to be up to the challenge.

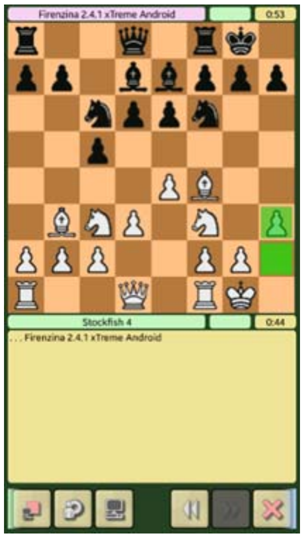
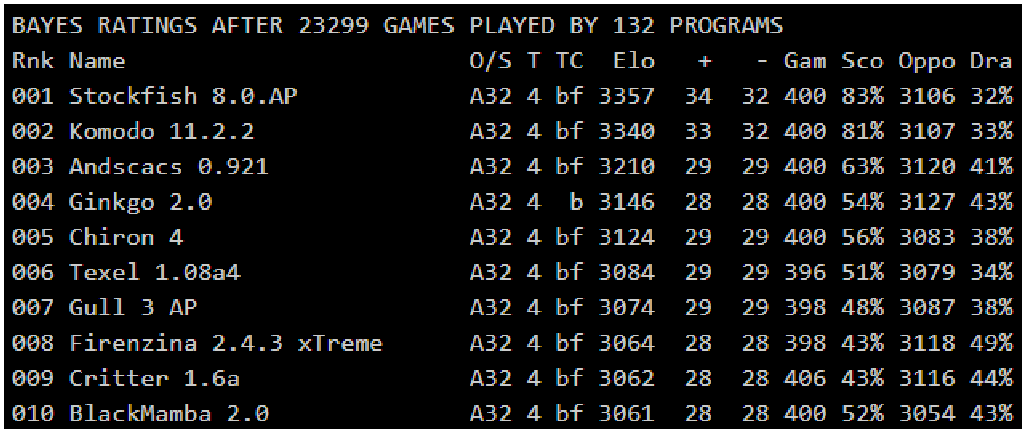
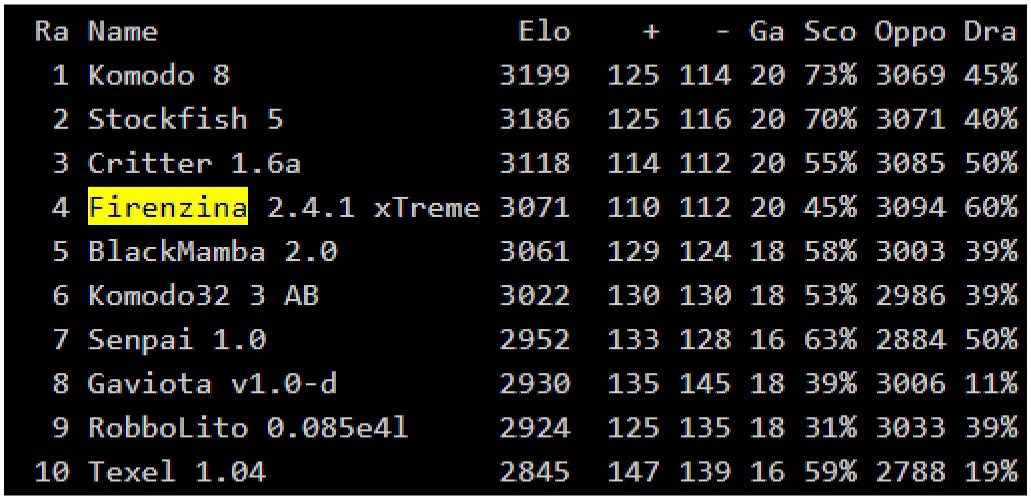

Firenzina 2.4.1 xTreme Android
0:53
Stockfish 4
0:44
. . . Firenzina 2.4.1 xTreme Android

Figure 7. A screenshot from the Chess for All app showing gameplay by Firenzina, a chess engine ported to Android for a Spring 2014 CNIT 425 project. Source: Abshire and Gusev (2015).



```
BAYES RATINGS AFTER 23299 GAMES PLAYED BY 132 PROGRAMS
Rnk Name                        O/S T TC  Elo   +   - Gam Sco Oppo Dra
001 Stockfish 8.0.AP            A32 4 bf 3357  34  32 400 83% 3106 32%
002 Komodo 11.2.2               A32 4 bf 3340  33  32 400 81% 3107 33%
003 Andscacs 0.921              A32 4 bf 3210  29  29 400 63% 3120 41%
004 Ginkgo 2.0                  A32 4  b 3146  28  28 400 54% 3127 43%
005 Chiron 4                    A32 4 bf 3124  29  29 400 56% 3083 38%
006 Texel 1.08a4                A32 4 bf 3084  29  29 396 51% 3079 34%
007 Gull 3 AP                   A32 4 bf 3074  29  29 398 48% 3087 38%
008 Firenzina 2.4.3 xTreme      A32 4 bf 3064  28  28 398 43% 3118 49%
009 Critter 1.6a                A32 4 bf 3062  28  28 406 43% 3116 44%
010 BlackMamba 2.0              A32 4 bf 3061  28  28 400 52% 3054 43%
```

Figure 8. The Top 10 of the January 2018 Rapidroid rating by Chesstroid.



```
 Ra Name                      Elo    +    - Ga Sco Oppo Dra
  1 Komodo 8                 3199  125  114 20 73% 3069 45%
  2 Stockfish 5              3186  125  116 20 70% 3071 40%
  3 Critter 1.6a             3118  114  112 20 55% 3085 50%
  4 Firenzina 2.4.1 xTreme   3071  110  112 20 45% 3094 60%
  5 BlackMamba 2.0           3061  129  124 18 58% 3003 39%
  6 Komodo32 3 AB            3022  130  130 18 53% 2986 39%
  7 Senpai 1.0               2952  133  128 16 63% 2884 50%
  8 Gaviota v1.0-d           2930  135  145 18 39% 3006 11%
  9 RobboLito 0.085e4l       2924  125  135 18 31% 3033 39%
 10 Texel 1.04               2845  147  139 16 59% 2788 19%
```

Figure 9. The Top 10 of the September 2014 Rapidroid rating by Chesstroid.

Conclusions and Future Plans

The author’s long experience shows that chess programming can be used successfully in at all levels of undergraduate computer education, if students are sufficiently motivated.

The instructor intends to continue to use chess programming in the CIT courses to improve programming and computer networking skills of Purdue Polytechnic students. Among possible future programming projects, a re-write of the historically important Kaissa 1.00 chess engine to comply with the modern UCI chess protocol (2018) and a modification of ComradesGUI to use the newer JavaFX GUI library instead of Swing can be named. The research of implementations of chess tournament broadcasts is expected to continue as well.

References

Abshire, C., & Gusev, D. A. (2015). Firenzina: Porting a Chess Engine to Android. In: Mobile and Wireless Technology 2015: Proceedings of the 2nd International Conference on Mobile and Wireless Technology (ICMWT2015) (pp. 163-171). Berlin, Germany. Springer.

Arena website. (2017) Retrieved May 1, 2018 from http://playwitharena.com/.

CCCC website. (2018) Retrieved May 1, 2018 from https://github.com/FireFather/cccc.

Cfish website. (2017) Retrieved May 1, 2018 from https://github.com/syzygy1/Cfish.

Chesstroid website (2018) Retrieved May 1, 2018 from http://chesstroid.blogspot.com/.

Deep Fritz 14 website. (2014) Retrieved May 1, 2018 from https://shop.chessbase.com/en/products/deep_fritz_14_english.

Firenzina website. (2017) http://www.firenzina.org/. ComradesGUI Proto Type V3. Retrieved May 1, 2018 from http://www.firenzina.org/ComradesGUIProtoTypeV3.zip.

FOSCEC website. (2014) Retrieved May 1, 2018 from http://www.foscec.org/.

Gusev, D. A., & Swanson, D. A. (2017). Transitioning to the Polytechnic: The Game Development Aspect. *The 50th ASCUE Annual Conference,* June 11-15, 2017 (pp. 35-47). Myrtle Beach, SC: The Association Supporting Computer Users in Education (ASCUE).

Stockfish website. (2012) Retrieved May 1, 2018 from https://stockfishchess.org/.

TCEC website. (2018) Retrieved May 1, 2018 from http://tcec.chessdom.com/.

TLCV website. (2017) Retrieved May 1, 2018 from http://home.pacific.net.au/~tommyinoz/tlcv.html.

UCI chess protocol website. (2018) Retrieved May 1, 2018 from http://www.shredderchess.com/chess-info/features/uci-universal-chess-interface.html.