Two games working with Caia explained
Lamistra & Rolit

In this distribution of Caia you can find software to organize CodeCup-like games and competitions. The software is called Caia and is released to play games between different players for the CodeCup 2006 game 'Turn Right'. It includes four compiled players.

But for those who want to get more out of Caia, we have written this document and included two example games in the package to illustrate how it works and what you can do with it.

For the latest information and updates, please see: www.codecup.nl

The CodeCup 2005 game Lamistra

We have written and tested the software of Caia when the Lamistra competitions were going on in the winter of 2004. For a full description of the game and the protocol we refer you to the website www.codecup.nl. You will find the game under 'Past Contests'.

You can find in the source folders of Lamistra two client players which competed in the competition as 'unofficial'. The Pascal-player *(d11)* is from Willem van der Vegt and the C-player (*versie6VBLCS*) is from Marcel Vlastuin. They are as they are.

More interesting from an educational point of view are the manager and the referee. They are explained now in general. For a more detailed description of how to communicate with the caiaio: see the documentation of Caia.

The general structure of the manager for two players

```
PRINT("I cpu_speed 2800 850")
REM This line suggests that the CPU speed of your pc is 850 MHz
FOR A GAME BETWEEN tom AND jerry WITH REFEREE bob DO
  BEGIN
     PRINT("I lock")
     READ(ANSWER) UNTIL (ANSWER == "lock_ok")
     PRINT("I number_players 2")
     PRINT("I player 1 tom 30000 ../playerlogs/tom.log")
     PRINT("I start 1 100")
     READ(ANSWER)
     REM ANSWER = "no_firsterror" OR "firsterror MESSAGE_FROM_TOM"
     PRINT("I player 2 jerry 30000 ../playerlogs/jerry.log")
     PRINT("I start 2 100")
     READ(ANSWER)
     REM ANSWER = "no_firsterror" OR "firsterror MESSAGE_FROM_JERRY"
     PRINT("I unlock")
     PRINT(I referee bob ../refereelogs/bob.log)
     READ(REPORT)
     REM REPORT = "report MESSAGE_FROM_BOB"
     PRINT("I kill 1")
     PRINT("I kill 2")
     PRINT("I kill_referee")
  END
PRINT("I stop_caiaio")
```

Note that all printed information must be flushed in Pascal and in C to get the commands immediately to the caiaio. Without flushing you will get frustrating time delays.

The general structure of the referee for two players

```
PRINT("1 Start")
FOR EACH TWO MOVES OF BOTH PLAYERS DO
  BEGIN
     PRINT("I lock")
     READ(ANSWER) UNTIL (ANSWER == "lock_ok")
     PRINT("I listen 1")
     READ(MOVE1)
     PRINT("I unlock")
     IF (MOVE1 == CORRECT) THEN PROCESS MOVE1
     ELSE GOTO MISTAKE_PLAYER1
     PRINT("2 MOVE1")
     PRINT("I lock")
     READ(ANSWER) UNTIL (ANSWER == "lock_ok")
     PRINT("I listen 2")
     READ(MOVE2)
     PRINT("I unlock")
     IF (MOVE2 == CORRECT) THEN PROCESS MOVE2
     ELSE GOTO MISTAKE_PLAYER2
     IF (MOVE2 == LASTMOVE) GOTO END_GAME
     PRINT("1 MOVE2")
  END

END_GAME:
PRINT("I request_time 1")
READ(TIME1)
PRINT("I request_time 2")
READ(TIME2)
PRINT("I report MESSAGE_THE_MANAGER_WHO_HAS_WON")
REM The message should include details like scores and playing time
STOP PROGRAM REFEREE

MISTAKE_PLAYER1:
PRINT("2 X")
REM An 'X' is sent to a client player to make it stop
PRINT("I report PLAYER_2_WINS")
REM Extra information about playing time etc. should be included as well
STOP PROGRAM REFEREE

MISTAKE_PLAYER2:
PRINT("1 X")
PRINT("I report PLAYER_1_WINS")
REM Extra information about playing time etc. should be included as well
STOP PROGRAM REFEREE
```

The source code of manager.cc of Lamistra is provided with comments. The referee is not because the protocol for the input and output is too complicated to make it really useful to do. You can find the interesting parts in *referee.cc* in the main function on line 1218.

Play the example games with Lamistra

Execute from the command line in the bin folder:

```
./cacaio -d
```

Now two games between *versie6VBLCS* and *d11* are played. For less information you can leave out the –d option. In the log file *../playerlogs/versie6VBLCS.log* you can see what the player did sent to its stderr. The player *d11* did not print anything.

Rolit: a game for four players

This game demonstrates how Caia can be used for organizing games between more than two players.

The starting position of the four players in Rolit is:

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	1	2	30	31	32
33	34	35	4	3	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

The aim of the game is to get as many of your own pieces on the board. You can do this by adding a new piece somewhere to the structure. You must enclose one or more pieces of your opponents; this can be done either vertically, horizontally or diagonally. The enclosed pieces may be replaced by those of the enclosing player. From the starting position player 1 may choose out of the spots 30, 46 or 44. After placing on spot 30, the board now looks like this:

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	1	1	1	31	32
33	34	35	4	3	38	39	40
41	42	43	***44***	45	***46***	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Player 2 now, cannot enclose something and therefore he may place a new piece somewhere on the board. He can only make a choice out of: 19, 20, 21, 22, 23, 31, 39, 38, 46, 45, 44, 43, 35 and 27. Suppose he chooses spot 19:

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	2	***20***	***21***	***22***	***23***	24
25	26	***27***	1	1	1	***31***	32
33	34	***35***	4	3	***38***	***39***	40
41	42	***43***	***44***	***45***	***46***	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Now, player 3 has four options: 10, 21, 23 and 35. If now he chooses spot 10, he will get:

1	2	3	4	5	6	7	8
9	3	11	12	13	14	15	16
17	18	3	20	***21***	22	***23***	24
25	26	27	3	1	1	31	32
33	34	***35***	4	3	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

The game goes on like this. The player with the most pieces at the end of the game wins.

The protocol for the Rolit players

In the players folder of Rolit you can find four different players. Player *rolit_niv_5* is a better player than *rolit_niv_4* etc. After having all four players are started by the manager, the referee sends “1” to player 1, “2” to player 2, etc. Now player 1 must replay with his first move. The referee forwards this move to the other three players. Now is a move expected from player 2, etcetera. After having received or given the last move, a player must stop himself. The referee must stop himself after sending a report to the manager the final result.

If one of the players is making an invalid move, the referee sends the other three players a zero “0” to inform them they must stop and returns a report to the manager with a negative score for the player with the invalid move.

The structure of the manager for Rolit

The manager is structured in the same way as in Lamistra; now for four players:

```
PRINT(“I cpu_speed 2800 850”)
REM This line suggests that the CPU speed of your pc is 850 MHz
FOR A GAME BETWEEN p1, p2, p3 AND p4 WITH REFEREE bob DO
  BEGIN
     PRINT(“I lock”)
     READ(ANSWER) UNTIL (ANSWER == ”lock_ok”)
     PRINT(“I number_players 4”)
     PRINT(“I player 1 p1 30000 ../playerlogs/p1.log”)
     PRINT(“I start 1 100”)
     READ(ANSWER)
     REM ANSWER = “no_firsterror” OR “firsterror MESSAGE_FROM_P1”
     PRINT(“I player 1 p2 30000 ../playerlogs/p2.log”)
     PRINT(“I start 2 100”)
     READ(ANSWER)
     REM ANSWER = “no_firsterror” OR “firsterror MESSAGE_FROM_P2”
     PRINT(“I player 1 p3 30000 ../playerlogs/p3.log”)
     PRINT(“I start 3 100”)
     READ(ANSWER)
     REM ANSWER = “no_firsterror” OR “firsterror MESSAGE_FROM_P3”
     PRINT(“I player 1 p4 30000 ../playerlogs/p4.log”)
     PRINT(“I start 4 100”)
     READ(ANSWER)
     REM ANSWER = “no_firsterror” OR “firsterror MESSAGE_FROM_P4”
     PRINT(“I unlock”)
     PRINT(I referee bob ../refereelogs/bob.log)
     READ(REPORT)
     REM REPORT = “report MESSAGE_FROM_BOB”
     PRINT(“I kill 1”)
     PRINT(“I kill 2”)
     PRINT(“I kill 3”)
     PRINT(“I kill 4”)
     PRINT(“I kill_referee”)
  END
PRINT(“I stop_caiaio”)
```

The structure of the referee for Rolit

The referee of Rolit is structured in a slightly different way as in Lamistra:

```
PRINT(“1 Start”)
PRINT(“2 Start”)
PRINT(“3 Start”)
PRINT(“4 Start”)
FOR EACH FOUR MOVES OF ALL PLAYERS DO
  BEGIN
     PRINT(“I lock”)
     READ(ANSWER) UNTIL (ANSWER == ”lock_ok”)
     PRINT(“I listen 1”)
     READ(MOVE1)
     PRINT(“I unlock”)
     IF (MOVE1 == CORRECT) THEN PROCESS MOVE1
     ELSE GOTO MISTAKE_PLAYER1
     PRINT(“2 MOVE1”)
     PRINT(“3 MOVE1”)
     PRINT(“4 MOVE1”)


     PRINT(“I lock”)
     READ(ANSWER) UNTIL (ANSWER == ”lock_ok”)
     PRINT(“I listen 2”)
     READ(MOVE2)
     PRINT(“I unlock”)
     IF (MOVE2 == CORRECT) THEN PROCESS MOVE2
     ELSE GOTO MISTAKE_PLAYER2
     PRINT(“1 MOVE2”)
     PRINT(“3 MOVE2”)
     PRINT(“4 MOVE2”)

     PRINT(“I unlock”)
     READ(ANSWER) UNTIL (ANSWER == ”lock_ok”)
     PRINT(“I listen 3”)
     READ(MOVE3)
     PRINT(“I unlock”)
     IF (MOVE3 == CORRECT) THEN PROCESS MOVE3
     ELSE GOTO MISTAKE_PLAYER3
     PRINT(“1 MOVE3”)
     PRINT(“2 MOVE3”)
     PRINT(“4 MOVE3”)

     PRINT(“I unlock”)
     READ(ANSWER) UNTIL (ANSWER == ”lock_ok”)
     PRINT(“I listen 4”)
     READ(MOVE4)
     PRINT(“I unlock”)
     IF (MOVE4 == CORRECT) THEN PROCESS MOVE4
     ELSE GOTO MISTAKE_PLAYER4
     PRINT(“1 MOVE4”)
     PRINT(“2 MOVE4”)
     PRINT(“3 MOVE4”)
  END

END_GAME:
PRINT(“I request_time 1”)
READ(TIME1)
PRINT(“I request_time 2”)
READ(TIME2)
PRINT(“I request_time 3”)
READ(TIME3)
PRINT(“I request_time 4”)
READ(TIME4)
PRINT(“I report MESSAGE_THE_MANAGER_WHO_HAS_WON”)
REM The message should include details like scores and playing time
STOP PROGRAM REFEREE

MISTAKE_PLAYER1:
PRINT(“2 0”)
PRINT(“3 0”)
PRINT(“4 0”)
REM An ‘0’ is sent to a client player to make it stop
PRINT(“I report PLAYER_1_LOOSES”)
REM Extra information about playing time etc. should be included as well
STOP PROGRAM REFEREE

MISTAKE_PLAYER2:
PRINT(“1 0”)
PRINT(“3 0”)
PRINT(“4 0”)
PRINT(“I report PLAYER_2_LOOSES”)
REM Extra information about playing time etc. should be included as well
STOP PROGRAM REFEREE
MISTAKE_PLAYER3:
PRINT(“1 0”)
PRINT(“2 0”)
PRINT(“4 0”)
```

```
REM An ‘0’ is sent to a client player to make it stop
PRINT(“I report PLAYER_3_LOOSES”)
REM Extra information about playing time etc. should be included as well
STOP PROGRAM REFEREE

MISTAKE_PLAYER4:
PRINT(“1 0”)
PRINT(“2 0”)
PRINT(“3 0”)
PRINT(“I report PLAYER_4_LOOSES”)
REM Extra information about playing time etc. should be included as well
STOP PROGRAM REFEREE
```

The source code of *manager.cc* and *referee.cc* of Rolit is provided with extra comments.

Play the example game with Rolit

Execute from the command line in the bin folder:

```
./cacaio -d
```

Now one game between *rolit_niv_4*, *rolit_niv3*, *rolit_niv_5* and *rolit_niv_2* is played. For less information you can leave out the –d option. The board of the game after each move is printed to the screen. That is because the referee prints it to his own stderr.

Only *rolit_niv_4* and *rolit_niv_5* do print information to their stderr. They only print their own ID. This can be verified in the player logs.

The referee sends in his report to the manager the number of pieces of player 1 to player 4 at the end of the game. The last four numbers are the playing times of the four players in milliseconds. The player that looks ahead most far wins.