leszcz

Contents

Co i Jak (dla “deweloperów”) **2**
- Co (“środowisko uruchomieniowe”) . . . 2
- Jak (“uruchamianie środowiska”) . . . 2
- bitboardy (Gerd n.d.) . . . 2
- cytaty . . . 3

Architektura **4**

Sieć **5**
- p2p . . . 5
- przez sieć internet . . . 5

protokół **5**
- typy pakietów . . . 5
- opisy pakietów . . . 5
- Przykłady . . . 9

Odniesienia **11**

Co i Jak (dla “deweloperów”)

Co (“środowisko uruchomieniowe”)

- win64 – skompilowałem te biblioteki dla *ciebie* czytelniku 💜
  - raylib[1]
  - libffi[2]
- unix
  - raylib z taga 5.5 skompilowany z wsparciem dla PNG i `RAYLIB_LIBTYPE=LIBTYPE_SHARED` skopiowane do `$(PWD)/raylib5.5.so`
  - libffi typu jakiekolwiek

Jak (“uruchamianie środowiska”)

hmm, preferowalnie emacs z slime a potem `M-x slime RET`

```
CL-USER> (push (pathname (uiop:getcwd)) ql:*local-project-directories*)
[...]
CL-USER> (ql:quickload :leszcz)
[...]
CL-USER> (in-package :leszcz)
#<PACKAGE "LESZCZ">
LESZCZ> (sb-thread:make-thread #'main)
[...]
```

Jeśli repl nie jest czymś co Ci się podoba, można wykonać `make run` by automatycznie i łatwo uruchomić kod.

:3

bitboardy (Gerd n.d.)

```
(setf (game-fb ...) (game->fast-board ...))
```

Generacja ruchów niektórych figur opiera się na bitplanszach `fast-board`, `fast-board-1` w pliku `fast.lisp`.

[1] https://pub.krzysckh.org/msc25/raylib5.5.dll
[2] https://pub.krzysckh.org/msc25/libffi-8.dll

cytaty

> *jezu chryste p **(6 6)** to piece(PAWN[BLACK])@**point(5 7)***
>
> — pre-possible-moves-for/upgrade, leszcz.lisp

> *point-checked-p disagree on point (4, 0) w/ fen r3k2r/p1pp1qbn/bn2p1p1/3PN3/1p2P3/2N4p/PPPBBPPP/R3K1R1 b Qkq - 0 1 being checked by WHITE (NIL vs T)*
>
> — point-checked-p, leszcz.lisp

> *"❌ 97818 is expected to be 97862"*
>
> — t/test.lisp

> *"safe piece type of point (7 7) is NIL in contrary to the unsafe one which is ROOK"*
>
> — maybe-castling-moves, leszcz.lisp

> *"King couldn't be found in #S(FAST-BOARD-1 ...)"*
>
> — fb1-king-of, fast.lisp

Architektura

Aplikacja napisana jest w Common Lispie i korzysta z implementacji sbcl[3]. Do obsługi międzyplatformowej grafiki/IO korzysta z bliblioteki raylib[4]. Napisałem ad-hoc nakładkę korzystającą z cffi[5] i libffi[6] która ładuje dll rayliba i binduje funkcje do takich łatwo używalnych z lispa.

Dzięki takiemu podejściu aplikacja jest dość międzyplatformowa[7] i jeśli pominie się budowanie pliku wykonywalnego (albo użyje buildapp[8]) może[9] być uruchamiana innych implementacjach CL.

[3] https://sbcl.org
[4] https://raylib.com
[5] https://cffi.common-lisp.dev/
[6] https://www.chiark.greenend.org.uk/doc/libffi-dev/html/
[7] przynajmniej MS Windows (x64), Debian GNU/Linux, OpenBSD.
[8] https://www.xach.com/lisp/buildapp
[9] sprawdziłem, działa przynajmniej w ECL

Sieć

(to jest pomysł, jeszcze nie napisałem ani linijki kodu :3)

p2p

1v1 po lokalnej sieci LUB sieci internet przy otwartych portach

- Komputer 1 (`master`) uruchamia serwer na jakimśtam porcie gotowy do akceptowania 1 połącznia
- Komputer 2 (`slave`) łączy się wprost z komputerem 1
- dogadują się przez jakiśtam protokół
- …
- profit

przez sieć internet

- Komputer 1 łączy się z jakimśtam (pewnie moim) serwerem po adresie IP w sieci internet
- Komputer 2 łączy się z tym samym serwerem
- dogadują się przez jakiśtam protokół
- dogadują się **z kim** chcą grać (po nazwie użytkownika? po adresie ip? :3)
- serwer (ten jakiśtam [pewnie mój]) przekazuje dane z Komputera 1 do Komputera 2 bez łączenia ich bezpośrednio ze sobą
- …
- profit

protokół

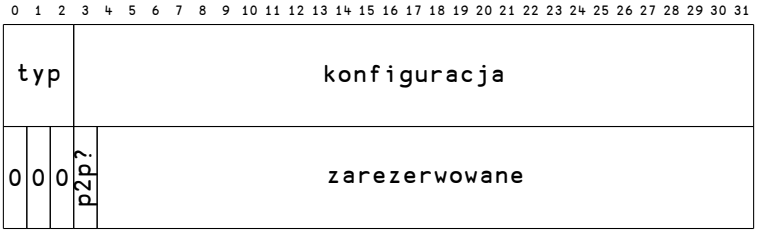
typy pakietów

typ	nazwa	opis
0	hii :3	wysyłany na początku by dostać informacje o typie serwera i możliwościach
1	gdata	informacje o grze przed rozpoczęciem/podczas
2	lgames	prośba/wylistowanie możliwych gier przez serwer
3	pgame	wybierz grę i dołącz
4	ping	ping, czy komputer jest nadal aktywny?
5	move	ruch
6	rdata	dane wysyłane jako kontynuacja poprzedniego pakietu

opisy pakietów

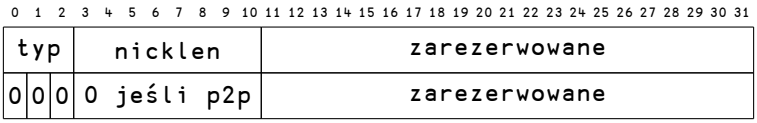
TODO: każda gra musi mieć ID i musi być ich mniej niż 2^29

- *hii :3*
  - serwer wysyła



Rysunek 1: Pakiet hii :3 wysyłany przez serwer z informacją o konfiguracji

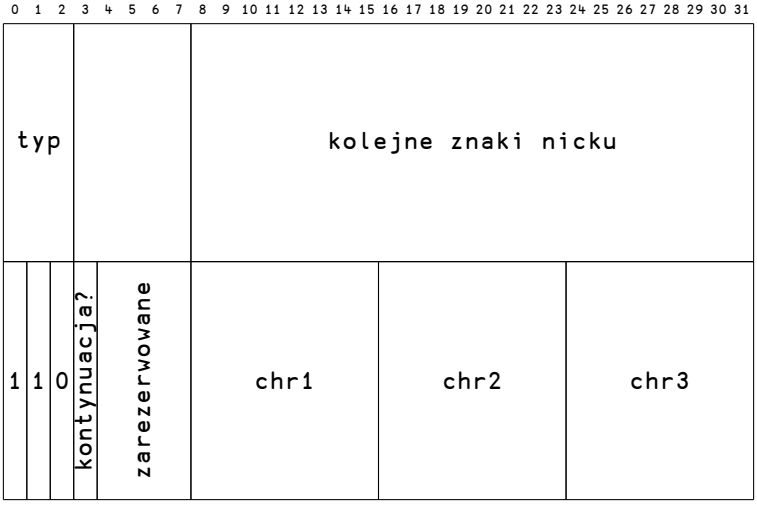
- klient odpowiada



Rysunek 2: Pakiet hii :3 wysyłany przez klient

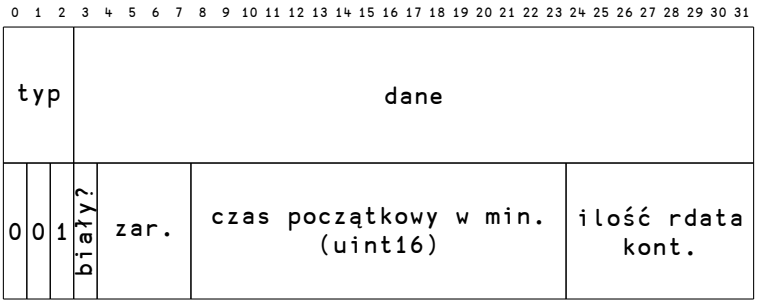
TODO: nicklen w ilości pakietów? łoł czyli 0-3*255

i *nicklen* pakietami kontynuacyjnymi



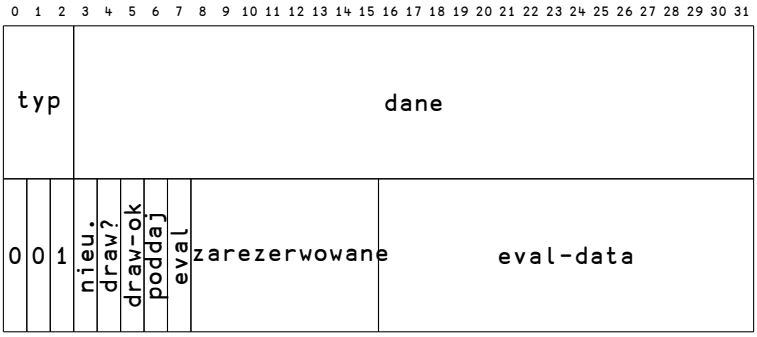
Rysunek 3: Pakiety rdata z danymi o nicku

- *gdata*
  - W etapie początkowym
    * serwer wysyła



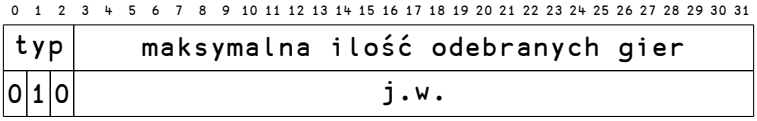
Rysunek 4: Pakiet gdata z danymi o grze

      i *N* pakietów kontynuacyjnych zawierających FEN gry
  - W każdym momencie gry, każdy może wysłać



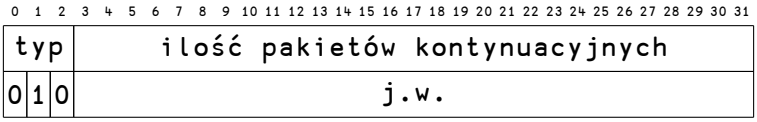
Rysunek 5: Pakiet gdata do wysyłania zaszłości w grze

- *lgames*
  - klient wysyła



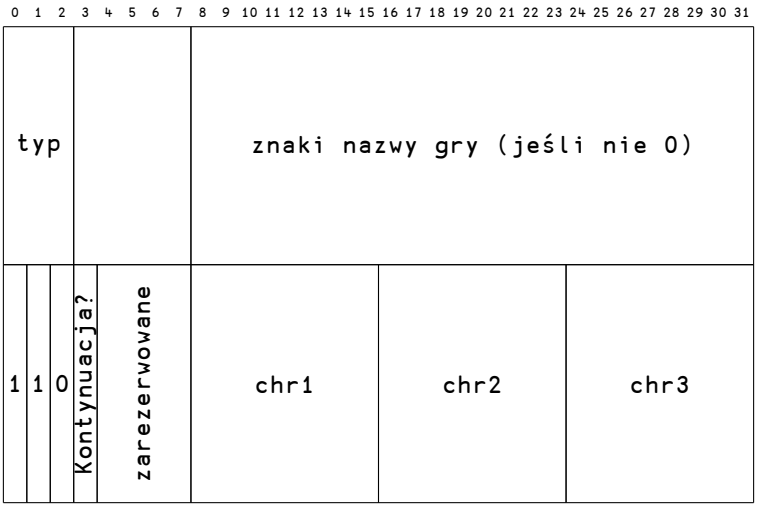
Rysunek 6: Pakiet lgames do proszenia o listę możliwych gier

  - serwer odpowiada



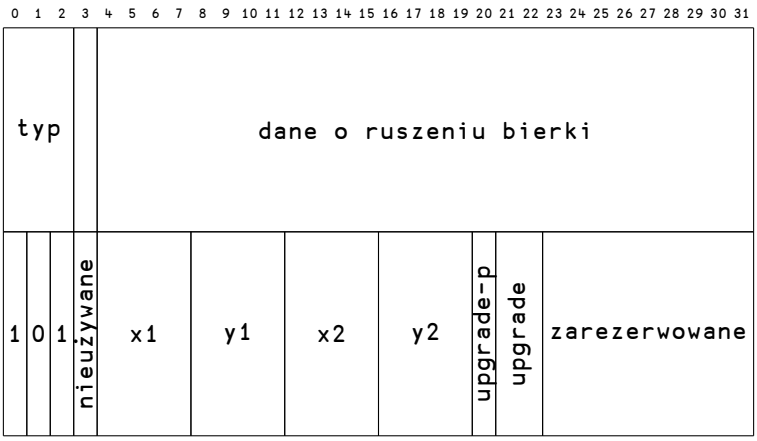
Rysunek 7: Pakiet lgames odpowiadający na prośbę o listę możliwych gier

i *N* pakietami kontynuacyjnymi



Rysunek 8: Pakiety rdata z danymi (nazwami) gier

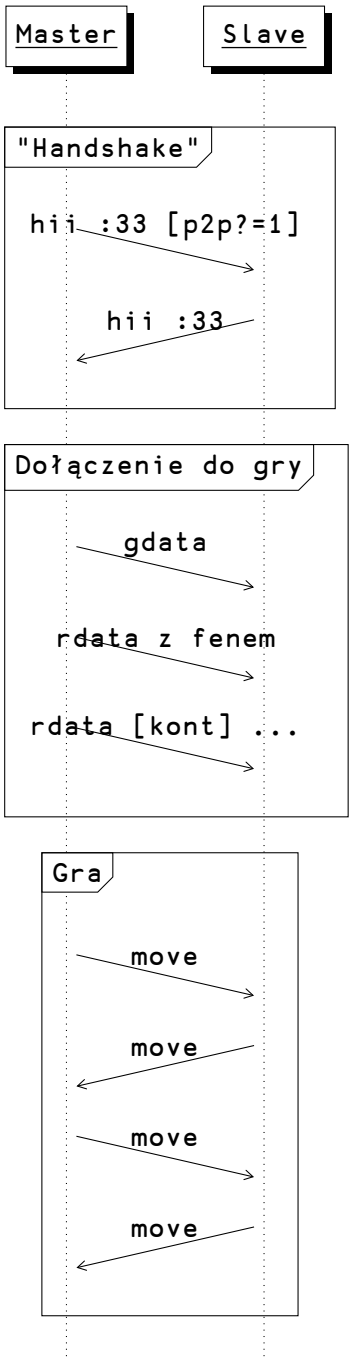
- *move*



Rysunek 9: Pakiet move z danymi o ruchu wykonanym przez przesyłającego

Przykłady

p2p



Rysunek 10: Działanie protokołu w trybie p2p

```
A: 16  0   0   0   ; hii :33 [p2p]
B: 0   0   0   0   ; hii :33
A: 48  0   0   19  ; gdata [packets=19]
   192 114 110 98  ; rdata rnb
   208 113 107 98  ; rdata qkb [kont]
   208 110 114 47  ; rdata nr/ [kont]
   208 112 112 112 ; rdata ppp [kont]
   208 112 112 112 ; rdata ppp [kont]
   208 112 112 47  ; rdata pp/ [kont]
   208 56  47  56  ; rdata 8/8 [kont]
   208 47  56  47  ; rdata /8/ [kont]
   208 56  47  80  ; rdata 8/P [kont]
   208 80  80  80  ; rdata PPP [kont]
   208 80  80  80  ; rdata PPP [kont]
   208 80  47  82  ; rdata P/R [kont]
   208 78  66  81  ; rdata NBQ [kont]
   208 75  66  78  ; rdata KBN [kont]
   208 82  32  119 ; rdata R w [kont]
   208 32  75  81  ; rdata  KQ [kont]
   208 107 113 32  ; rdata kq  [kont]
   208 45  32  48  ; rdata - 0 [kont]
   208 32  49  0   ; rdata  1  [kont]
B: 163 99  64  0   ; move [x1=3, y1=6, x2=3, y2=4] (d4)
A: 163 19  48  0   ; move [x1=3, y1=1, x2=3, y2=3] (d5)
```

Rysunek 11: Transkrypcja działania protokołu w trybie p2p

Odniesienia

Gerd, Isenberg. n.d. “Bitboards (Board Representation).” Accessed February 19, 2025. https://www.chessprogramming.org/Bitboards#General_Bitboard_Techniques.