

Deal - rozbudowany generator rozdań

Piotr Beling

24 września 2010

1 Czym jest Deal?

Deal to darmowy program o otwartych źródłach¹ autorstwa Thomasa Andrewsa² służący do generowania rozkładów kart spełniających wybrane przez użytkownika kryteria.

Program nie posiada przyjaznego, graficznego interfejsu użytkownika jednak jego ogromne możliwości w definiowaniu precyzyjnych kryteriów jakie muszą spełniać generowane rozdania sprawiają iż jest naprawdę godny uwagi.

Deal wraz z dokumentacją można pobrać z jego strony domowej: <http://bridge.thomasoandrews.com/deal/>.

2 Jak uruchomić Deal w Windows?

1. pobrać Deal dla Windows ze strony domowej i rozpakować go (w dalszej części będę zakładał, że program rozpakowano do folderu `c:\deal`)
2. uruchomić wiersz polecenia (można to uczynić wpisując w menu Uruchom komendę `cmd` albo odnajdując **Wiersz polecenia** w **Akcesoriach** znajdujących się w menu Start)
3. wydać (napisać i zaakceptować enterem) polecenie `cd c:\deal` by przejść w Wierszu polecenia do katalogu w którym znajduje się Deal
4. uruchomić program wydając odpowiednie polecenie, np. `deal` (więcej przykładów w dalszej części artykułu)

3 Podstawy obsługi

Deal pracuje w trybie tekstowym, do tego nie jest interaktywny. Po uruchomieniu po prostu wypisze na standardowym wyjściu (domyślnie na ekranie) żądane rozdania i zakończy działanie (nigdy nie zada użytkownikowi żadnych dodatkowych pytań). Wszelkie ustawienia przekazujemy do program w parametrach wywołania.

Proste przykłady wywołań:

- `deal` (wywołanie bez parametrów) wydrukuje na ekranie 10 rozdań

¹Napisany w C i TCL program objęty jest licencją GPL której szczegółowa treść znajduje się pod adresem <http://www.gnu.org/copyleft/gpl.html>

²adres e-mail Thomasa: deal@thomasoandrews.com

- `deal 25` wypisze na ekranie 25 rozdań
- `deal 25 > plik.txt` tym razem wydruk 25 rozkładów znajdzie się w pliku o nazwie `plik.txt` (znak `>` oznacza³ przekierowanie standardowego wyjścia programu do pliku), ewentualnie nadpisując jego wcześniejszą zawartość
- `deal 30 >> plik.txt` kolejne 30 rozdań zostanie dopisane do końca pliku `plik.txt`
- `deal -i format/practice 8 > plik.txt` zapisze 8 rozdań w pliku `plik.txt`, dodatkowo (dzięki `-i format/practice` wskazującemu format wyjściowy `practice`⁴) zostaną wygenerowane pliki `out.north`, `out.east`, `out.south`, `out.west` zawierające ręce poszczególnych graczy
- `deal -S "- KQJT62 T9876 84"` wypisze na ekranie 10 rozdań w których zawodnik S będzie posiadał wskazane karty (`-` oznacza renons - w tym przypadku w pikach, honory oznaczone są anglojęzycznymi skrótami: Q oznacza damę, J waleta, T dziesiątkę); możliwe jest też narzucenie kart zawodnika N (`-N "..."`), E (`-E "..."`) oraz W (`-W "..."`)
- `deal -i kryteria.tcl 50` wypisze 50 rozdań spełniających kryteria z pliku `kryteria.tcl` będącym skryptem napisanym w języku programowania TCL.
- `deal -v -i kryteria.tcl -i format/practice 100 > rozdania.txt` 100 rozdań spełniających kryteria opisane przez `kryteria.tcl` trafi do pliku `rozdania.txt`; dzięki zastosowaniu formatu `practice` zostaną wygenerowane pliki (`out.north`, `out.east`, itd.) z kartami poszczególnych graczy; dodatkowo opcja `-v` spowoduje wypisanie na ekran⁵ dodatkowych informacji

4 Skrypty TCL (Tool Command Language)

4.1 Wstęp

Wywołanie `deal -i skrypt.tcl` powoduje uwzględnienie skryptu TCL⁶ z pliku `skrypt.tcl` podczas generowania i wypisywania rozdań. Sam skrypt może wpływać zarówno na rozkłady jak i na format⁷ ich wydruku. W tym rozdziale zajmę się jedynie tym pierwszym przypadkiem.

4.2 Składnia TCL

Tcl jest językiem komend. Program w tym języku składa się z kolejnych wierszy komend. Każdy wiersz składa się z kolejnych słów, pierwsze z nich to komenda do

³w większości powłok, w tym w windowsowej (cmd)

⁴poszczególne formaty są zdefiniowane w plikach znajdujących się w folderze `format`

⁵mówiąc ściślej to na standardowe wyjście błędów

⁶więcej informacji o wymyślonym przez Johna Ousterhouta języku TCL można znaleźć w Wikipedi, pod adresem <http://pl.wikipedia.org/wiki/Tcl>

⁷przykładem są skrypty z katalogu `format`

wykonania, reszta to jej argumenty. Jedynymi elementami składniowymi Tcl-a są następujące znaki: \$ [] { } () ” ; ::.

Podstawowe komendy Tcl-a opisane są w pkt. 4.5

4.3 Przykład „otwarcie z pikami”

Załóżmy że chcemy wygenerować rozdania w których gracz N będzie miał co najmniej 5 pików i nie mniej niż 12PC. Stwórzmy w tym celu plik `piki.tcl` o następującej zawartości:

```
main {
  if {[spades north]>=5 && [hcp north]>=12} { accept }
  reject
}
```

Teraz `deal -i piki.tcl` wypisze 10 rozdań spełniających nasze wymagania.

Jak to właściwie zadziała? Deal wygeneruje losowe rozdania do czasu gdy skrypt nie zaakceptuje zażądaną ich ilość (u nas 10). Każdy rozkład poddawany jest działaniu skryptu, który może go albo zaakceptować (`accept` - wtedy rozkład jest wypisywany) albo odrzucić (`reject` - wtedy rozkład jest porzuca-ny).

Sam program składa się z komendy `main`, tj. zawiera się w bloku pomiędzy `main { a }` na końcu i jest wykonywany linia po linii od góry do dołu.

W pierwszej linii występuje komenda warunkowa `if` z dwoma argumentami, tj. `if { warunek } { instrukcje }`, która oznacza, iż instrukcje zostaną wykonane wtedy i tylko wtedy gdy warunek zostanie spełniony. W naszym przypadku jedyną instrukcją jest `accept` który oznacza akceptację rozważanego rozdania (powoduje natychmiastowe zakończenie działania skryptu dla rozważanego rozdania i wypisanie samego rozdania). Nasz warunek jest zaś koniunkcją (`&&`, czyt. „i”) dwóch wyrażeń (wartości logicznych⁸). Pierwsze `[spades north]>=5` sprawdza czy gracz N posiada co najmniej 5 pików, zaś drugie `[hcp north]>=12` czy ma on 12 lub więcej PC. Składnia `[funkcja argumenty]` oznacza wartość komendy `funkcja` dla podanych argumentów. Wartością (standardowej w Deal) komendy `spades` jest ilość pików, zaś wartością `hcp` jest ilość punktów honorowych. Obie przyjmują 1 argument określający rękę. Operator `>=` (czyt. większe lub równe) porównuje dwie liczby⁹.

Komenda `reject` w drugiej linii odrzuca rozdanie i (podobnie jak `accept`) powoduje natychmiastowe przerwanie skryptu. Tak naprawdę skrypt działałby identycznie bez tej instrukcji. Byłoby tak dlatego, że każde rozważane rozdanie zostaje domyślnie odrzucone jeśli nie zostanie jawnie zaakceptowane komendą `accept` w trakcie wykonywania skryptu.

4.4 Przykład „N otwiera BA, S ma 5 kierów”

Gracz N ma 15-18PC w składzie równym, zaś S co najmniej 5 kierów. Oto odpowiedni filtr:

⁸w TCL liczba 0 oraz napisy „false” i „no” reprezentują fałsz, zaś pozostałe wartości liczbowe, oraz napisy „true” oraz „yes” reprezentują prawdę

⁹jego wartością jest 1 jeśli wartość po lewej jego stronie jest większa lub równa od tej po prawej lub 0 w pozostałych przypadkach

```

main {
  # odrzuć rozdania w których N nie ma składu równego:
  if {[balanced north]} { reject }

  # ustaw zmienną hn na ilość PC w ręce N:
  set hn [hcp north]
  # odrzuć rozdania gdzie N ma złą ilość PC:
  if {$hn>18 || $hn<15} { reject }

  # jeśli skrypt wykona się do tego miejsca
  # to znaczy że N ma zakładaną rękę

  # akceptujemy rozdania w których S ma 5 kierów:
  if {[hearts south]>=5} { accept }

  # reszta jest domyślnie odrzucana
}

```

Działania poszczególnych fragmentów programu wyjaśnione są w komentarzach. Znak `#` rozpoczyna komentarz i wszystkie znaki stojące za nim, aż do końca linii są przez interpreter TCL ignorowane (nie mają wpływu na działanie skryptu). Warto zwrócić uwagę na użytą komendę `set` (objaśnioną w pkt. 4.5). Operator `!` („nie”) jest logicznym zaprzeczeniem, zaś `||` („lub”) to alternatywa. Komenda `balanced` której argumentem jest ręka, zwraca 1 (prawdę) tylko jeśli ręka jest zrównoważona oraz 0 (fałsz) jeśli nie jest.

4.5 Standardowe komendy

Standardowe komendy TCL:

- **set** nazwa_zmiennej wartość - ustawia wartość zmiennej o nazwie nazwa_zmiennej na podaną (dalej odwołujemy się do niej pisząc \$nazwa_zmiennej)
- **proc** nazwa argumenty ciało - definiuje komendę. Argumenty powinny być podane w formie listy. Komenda posiada wartość zwracaną, którą zwraca się za pomocą komendy `return`. Wartością zwracaną jest zawsze tekst (być może różnie interpretowany po stronie odbierającej), domyślnie pusty. Przykład znajduje się w pkt. 4.6.

Spis wszystkich zdefiniowanych w Deal komend można znaleźć na stronie <http://bridge.thomasoandrews.com/deal/commands.html>. Oto wybrane z nich:

- `north`, `east`, `south`, `west` - wartościami tych komend są ręce odpowiednio: N, E S i W. Opcjonalnym argumentem jest kolor, np. `north spades` oznacza piki gracza N.
- `balanced`, `semibalanced` - przyjmują nazwę ręki jako argument i zwracają prawdę tylko jeśli podana ręka jest zrównoważona: `balanced` dla układów 4333, 4432, lub 5332 (z 5-ką w młodym), zaś `semibalanced` dla wszystkich układów bez renonsu, singletona, starszego 6-kartu i młodszego 7-kartu.

- hcp - pierwszym argumentem jest nazwa ręki, zaś drugim (opcjonalnym) nazwa koloru. Wartością jest ilość miltonów w podanej ręce lub kolorze.

4.6 Pisanie i używanie własnych komend

Poniższa komenda zwraca 1 lub 0, przy czym 1 tylko gdy podana jako jej argument ręka ma otwarcie 1 trefl w systemie Wspólny Język:

```

proc polishclub {hand} {
  set hcp [hcp $hand]
  # dowolna ręka od 19 PC jest ok
  if {$hcp>=19} { return 1 }

  #jeżeli jest skład równy lub 4441k
  if {[balanced $hand] || [specialdiamond $hand]} {
    if {$hcp>=12 && $hcp<=14} { return 1 } # 12-14 PC
    if {$hcp==11 && [controls $hand]>=4} {
      return 1      # 11PC i 4 kontrole
    }
    if {$hcp>=18} { return 1 } # 18+PC
    return 0 # zła siła, zwracamy 0
  }
  if {[myclubs $hand] && $hcp>=15} {
    return 1 # 15+PC i 5 trefli
  }
  return 0
}

```

Komenda polishclub zdefiniowana jest za pomocą standardowej w TCL komendy **proc** (por. pkt. 4.5). Deal umożliwia także inne metody definiowania komend. Oto definicje komend myclubs i specialdiamond (są one używane w treści komendy polishclub więc muszą znaleźć się one w treści programu przed jej definicją):

```

shapecond myclubs { # 5 trefli i brak innej 5ki
  $c>=5 && $d<=4 && $h<=4 && $s<=4
}

shapecond specialdiamond { # 4441k
  $s==4 && $h==4 && $d==1 && $c==4
}

```

Jak widać komenda shapecond umożliwia łatwe definiowanie komend sprawdzających czy skład ręki jest odpowiedni. Jej argumentem jest warunek w którym można używać zmiennych z wartościami równymi długością poszczególnych kolorów, tj. \$s dla pików, \$h dla kierów, \$d dla kar oraz \$c dla trefli.

Definicję naszych komend możemy umieścić zarówno w pliku z właściwym skrypcem (przed komendą main) jak i w osobnym pliku. W tym drugim przypadku musimy wskazać owy plik przed ich użyciem. Oto przykład (N i E mają wspólnojęzykowe otwarcie 1t):

```

source nazwa_pliku_z_definicjami_naszych_komend

```

```
# od teraz możemy używać naszych komend
main {
    if {[polishclub north] && [polishclub east]} {
        accept
    }
}
```