WARSZTAT: DANE TEKSTOWE

Możliwość dostępu do danych zawartych w plikach zewnętrznych z poziomu AutoLISP-u ma wielkie znaczenie dla różnego typu programów i aplikacji AutoCAD-a. Trudno sobie wyobrazić jakiekolwiek oprogramowanie do parametryzacji obiektów bez dostępu do danych zewnętrznych. Przesyłanie danych do edytora graficznego AutoCAD-a z programów obliczeniowych tworzących zestawienia itp. często odbywa się poprzez pliki zewnętrzne. Są to z reguły pliki tekstowe. Mimo pewnych ograniczeń (przede wszystkim długi czas dostępu i wielkość plików danych), stosowanie mechanizmów AutoLISP-u dostępu do plików tekstowych nierzadko jest jedyną metodą na tworzenie wydajnych i elastycznych programów. Za stosowaniem plików tekstowych do wymiany danych z AutoCAD-em przemawiają również takie cechy, jak: prostota składni, możliwość edycji bez używania specjalnych narzędzi, wygoda i szybkość wprowadzania zmian oraz możliwość przechowywania danych w jednym miejscu.

AutoLISP posiada narzędzia do wykonywania operacji wejścia i wyjścia. Na początek zajmiemy się operacjami wejścia czyli pobierania danych zewnętrznych. Najogólniej proces odczytywania pliku możemy podzielić na trzy etapy:

→ znalezienie i otwarcie pliku do odczytu;
→ odczyt danych (i ewentualnie przetworzenie);
→ zamknięcie pliku.

Tutaj największy nacisk położymy na operację drugą, czyli odczyt i przetworzenie danych.

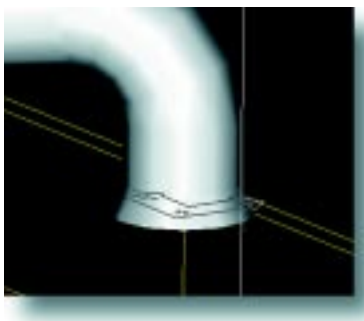
Do odczytania linii (całego wiersza) z pliku służy funkcja o nazwie *read-line*. Funkcja *read-line* posiada pewne ograniczenia. Oto one:

- wynikiem działania funkcji jest łańcuch tekstowy;
- każde wywołanie funkcji zwraca tylko jedną linię;

→ dokończenie ze strony 33

tion–Join, Body Type–Normal, Termination–Path Only, Draft Angle 0. Nasza wylewka jest już prawie gotowa.

Musimy teraz na jej końcu wykonać lekkie rozszerzenie. Wybieramy *New Sketch Plane* i wskazujemy krawędź okręgu na końcu wylewki. Jeśli teraz wybierzemy *Assist→Icon at Origin*, zauważymy, że ikona początku współrzędnych przesunie się ku punktowi środkowemu okręgu. Rysujemy okrąg o współrzędnych środka 0,0 i promieniu 9 i tworzymy z niego *Single Profile*. Teraz naciskamy *Work Features→Work Plane* i w opcjach okna dialogowego wpisujemy *Planar Parallel, Offset* 3; wskazujemy krawędź końca wylewki. Przy potwierdzaniu zwrotu osi strzałka osi Z musi być skierowana ku górze. Rysujemy okrąg: środek 0,0, promień 10 i przekształcamy w kolejny *Profil*. Tworzymy następną płaszczyznę roboczą przesuniętą o kolejne 2 milimetry, aby uzyskać jeszcze jeden *Single Profile* z okręgu o promieniu 11.

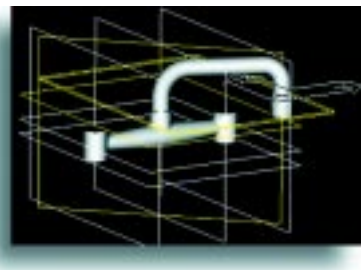


Wykonujemy operację *Loft* przy standardowych ustawieniach, wskazując po kolei każdy z trzech kolejno wykonanych profili. Niebieska strzałka będzie wskazywała kierunek wytłaczania. Efekt na ilustracji.

Pora teraz na wykonanie stożkowatego elementu, tzw. perlatora. Jako nową płaszczyznę szkicową (*New Sketch Plane*) wybieramy ostatnią kolistą krawędź. Musimy pamiętać o prawidłowym zwrocie osi. Rysujemy okrąg w punkcie 0,0 o promieniu 12 i przekształcamy go w *Single Profile*. Rozpoczynamy operację *Sketched Features→Extrude*, wybieramy parametry *Join, Blind, Distance* 20, *Draft Angle* 5, wskazujemy nasz profil i zatwierdzamy kierunek wytłaczania.

Doszliśmy już do końca pierwszego etapu naszej pracy.

W przyszłym odcinku utworzymy następne elementy: pokrętła kurków i nakrętki, połączymy również wszystko w całość. Zobaczymy także, jak tworzy się połączenia gwintowane. Mamy nadzieję, że każdy uczestnik kursu dotrzymuje nam kroku. Czasami wykonanie poszczególnych operacji może być kilkakrotnie powtarzane dla uzyskania pożądanego efektu. Umożliwi to stopniowe zapoznawanie się z programem i nabieranie wprawy. Praca przy modelowaniu zacznie stawać się coraz większą przyjemnością. Rezultatem może być również odnajdywanie własnych rozwiązań, może prostszych i efektywniejszych niż podane. Wdzięczni będziemy za podzielenie się z nami własnymi doświadczeniami. Umożliwi to stworzenie forum użytkowników AMD3 na łamach naszego pisma.



Andrzej Pierowski
pierowski@avia.com.pl

- kolejne linie czytane są poprzez każdorazowe wywołanie funkcji;
- *read-line* nie potrafi cofnąć się do poprzednich linii (wcześniej czytanych).

Biorąc pod uwagę te ograniczenia, bardzo wielką rolę odgrywa tu odpowiednie formatowanie pliku tekstowego. Oczywiście sposobów formatowania może być bardzo wiele. Dwa najbardziej rozpowszechnione formaty plików do przechowywania danych to SDF i CDF.

Format SDF (*Space Delimited Format*). Dane sformatowane są w kolejnych liniach, a jako separatory używana jest jedna lub więcej spacji. Dane tworzą równej szerokości kolumny. Zawartość takiego pliku może wyglądać tak:

```
100.00     120.50     185.00     20
100.50     122.75     190.50     35
itd.
```

W pewnych sytuacjach formatowanie takie, może okazać się niewygodne (dla danych o zmiennych lub nieznanych długościach).

Format CDF (*Comma Delimited File*). Dane sformatowane są w kolejnych liniach, separatorem jest przecinek. Dane mogą mieć różną długość. Plik może wyglądać tak:

```
10.20,12.67,18.98
100.20,122.67,138.98
itd.
```

Innym sposobem formatowania danych jest zastosowanie odpowiedniego klucza, według którego linia jest przeszukiwana. Po odnalezieniu słowa lub znaku kluczowego powinna być czytana jedna lub więcej linii zapisanych w formacie CDF (wyglądają tak pliki AutoCAD-a typu LIN i PAT)

W obu przypadkach zadaniem programu AutoLISP analizującego plik tekstowy jest odczytanie określonej linii pliku i zwrócenie wyniku jako listy danych. Pierwszym etapem realizacji tego zadania jest odczytanie linii pliku (wynikiem jest łańcuch tekstowy). W tym celu zdefiniowałem funkcję o nazwie JK:READ-FILES. Jest ona tak skonstruowana, aby odczytywać dane z pliku na dwa sposoby:

- odczytanie całego pliku i zwrócenie listy łańcuchów tekstowych;
- odczytanie jednej linii określonej przez jej pozycję.

Funkcja JK:READ-FILES jest dostępna na stronie WWW Magazynu 3D (http://www.3d.pl). Aby ją przetestować wykonajmy następujące ćwiczenie. Po załadowaniu pliku LSP i wpisaniu w linii poleceń:

```
(JK:READ-FILES NIL "DANE.DAT") lub (JK:READ-
    FILES 3 "DANE.DAT")
```

program wyświetli okno ostrzegawcze i zwróci NIL, ponieważ nie istnieje plik DANE.DAT. Natomiast wywołanie:

```
(JK:READ-FILES NIL "ACAD.LIN")
```

zwróci listę wszystkich linii w pliku acad.lin.

Wywołanie:

```
(nth 12 (JK:READ-FILES NIL "ACAD.LIN"))
```

zwróci łańcuch:

```
"*CENTER,Center ____ _ ____ _ ____ _ ____ _
    ____ _ ____"
```

Inny sposób wybrania tylko dwunastej linii z tego pliku wygląda następująco:

```
(JK:READ-FILES 12 "ACAD.LIN")
```

Wówczas pobranie trzynastego wiersza pliku z acad.lin za pomocą polecenia:

```
(JK:READ-FILES 13 "ACAD.LIN")
```

zwróci:

```
"A,1.25,-.25,.25,-.25"
```

Oczywiście zamiast pliku ACAD.LIN można wykorzystać każdy inny plik tekstowy.

Funkcja JK:READ-FILES oczywiście spełnia swoje zadanie, kiedy należy odczytać cały plik, lub odczytać linię na znanej pozycji. W przypadku gdy nie wiemy, w którym miejscu pliku znajdują się dane, a znamy identyfikator, według którego linia jest rozpoznawana, wykorzystamy funkcję o nazwie JK:READ-LINES, której działanie sprowadza się do:

- odczytania linii, która występuje jako następna po linii zawierającej poszukiwany klucz;
- odczytania linii, w której występuje poszukiwany klucz.

Funkcja JK:READ-LINES jest dostępna na stronie WWW Magazynu 3D (http://www.3d.pl). Do testowania można ponownie wykorzystać plik acad.lin. Wówczas wywołanie:

```
(JK:READ-LINES T "acad.lin" "*DASHDOT,")
```

zwróci:

```
"*DASHDOT,Dash dot __ . __ . __ . __ . __ .
    __ . __ . __"
```

czyli linię, w której wystąpił klucz. Natomiast:

```
(JK:READ-LINES NIL "acad.lin" "*DASHDOT,")
```

zwraca:

```
"A,.5,-.25,0,-.25"
```

czyli następną linię po wystąpieniu klucza. Warto zaznaczyć, że funkcję tę można wykorzystać również do czytania plików typu INI. Na przykład wywołanie:

```
(JK:READ-LINES T "mtextmap.ini" "isocp.shx=")
```

zwraca:

```
"isocp.shx=ISOCP,0,0,0,2"
```

Pierwsze zadanie, jakim było odczytanie dowolnej linii z pliku tekstowego, zostało zatem zrealizowane. Ponieważ READ-LINE zawsze zwraca dane jako łańcuch tekstowy osobnym problemem jest teraz obróbka takiego łańcucha. Tym zagadnieniem zajmiemy się w następnym numerze.

Jacek Kożuszek
jako@wr.onet.pl