Max Script for 3D Studio MAX R4 część V

by Adrian Majchrzak

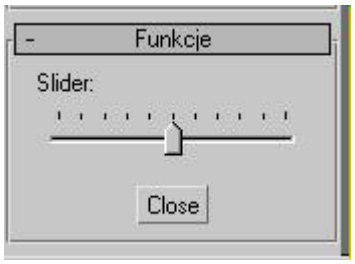
Witam w kolejnej części kursu skryptowania w maxie. Dzisiaj powiemy sobie co nie co szerzej o programowaniu. Warto było by już napisać bardziej złożony skrypt. Niestety bez pomocy funkcji , struktur, tablic i pętli mało możemy zrobić tak więc dzisiaj poznamy kolejny obiekt nazywający się Slider no i może coś jeszcze . Omówimy też w podstawowym stopniu zasadę działania funkcji. Tak więc do dzieła

Slider

Co to jest slider ?. No więc najprościej mówiąc jest to zwykły suwak. Jak jeszcze nie kojarzysz to zaraz zobaczysz na przykładzie. Poniżej zamieszczam kod tworzący taki obiekt. Zasada działania jest taka sama jak w przypadku obiektu spinner więc skoro już tamten poznałeś to tu nie ma czego omawiać.

Konstrukcja obiektu Slider

```
utility funkcje "Funkcje"
(
  Slider sli "Slider:" range:[-10,10,0] enabled:true
)
```



Warto tu powiedzieć jeszcze o tym co mamy w nawiasie kwadratowym oraz opcje boolowskie. Pierwszym parametrem w nawiasie kwadratowym jest minimalny zakres dalej mamy maksymalny zakres oraz ostatni parametr to aktualne położenie suwaka na podziałce pomiędzy –10 a 10 . dalej mamy funkcję enabled. Możemy za jej pomocą decydować czy obiekt jest w tej chwili włączony czy nie. Wartości dla funkcji enabled to true – aktywny i false – wyłączony. Dzięki tej funkcji możemy sterować już w ambitniejszy sposób naszymi obiektami.

Zanim przejdziemy do opisu funkcji przedstawię jeszcze jeden obiekt który każdy z was bardzo dobrze zna. Mowa tu o tym kolorowym kwadraciku do wyboru koloru dla siatki, obiektu itp.

Colorpicker

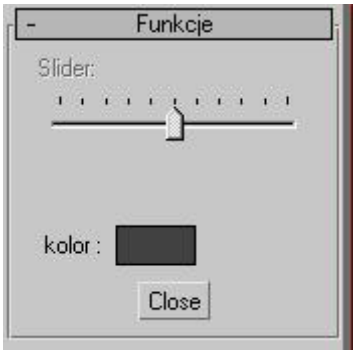
No to wiadomo co tu pokażę więc przejdźmy do razu do kodu tworzącego ten obiekcik.

Kod tworzący obiekt Colorpicker

```
utility funkcje "Funkcje"
(
    colorpicker col " kolor : " color:[255,255,255] enabled:true offset:[15,0]

)
```

Tak wygląda kod tworzący nasz kwadracik do wyboru koloru. Wpisz sobie poniższy kod i zobacz jak on działa. Zasada budowy tego obiektu jest taka jak dla obiektu slider i spinner więc nie ma znowu co omawiać no poza paroma elementami. W tablicy ustawiamy sobie kolor jaki aktualnie ma się wyświetlić. Dalej znana już nam funkcja enabled oraz coś nowego. Mamy to funkcję offset. Jak chcemy to możemy jej nie stosować. W każdym bądź razie służy ona do ustalenia położenia naszego obiektu w ramce naszego skryptu. Pierwsza wartość odpowiada współrzędnej X a druga Y. Poniżej zamieszczam obrazek z tym obiektem.



No dobra, przyszedł czas na omówienie funkcji. Wreszcie coś nowego. Domyślam się że już wiesz mniej więcej do czego są funkcje. Jeżeli nie to już mówię. Czasem zachodzi taka potrzeba aby zbudować fragment kodu zupełnie w innym miejscu. Co więcej, taką funkcję mogą wykorzystywać wszystkie funkcje które uruchamiają nam przyciski itp. Funkcje mogą pobierać i zwracać wartości. Co w tym fajnego ?. Sam się przekonasz. Wyobraź sobie że w kilku miejscach chcesz policzyć np. sinus jakiegoś kąta. Powiedzmy że funkcja która nam będzie to liczyć zajmie nam max 5 linijek kodu. Teraz wyobraź sobie że musisz dokonać takich obliczeń koło 10 i wszędzie wpisywać ten sam fragment kodu tylko że wartości się zmienią. To w sumie zajęło by 50 linijek. Tak więc tworzysz sobie uniwersalną funkcję która będzie odbierać dane przez zmienne, będzie je przeliczać i zwracać w odpowiednio wskazanym miejscu. W ten sposób zamiast pisać za każdym razem 5 linijek kodu będziesz pisał jedną linijkę wywołującą funkcję liczącą wartość dla kątów. I to jest najlepsze. Kod staje się krótszy, szybszy i bardziej przejrzysty. Oszczędzisz sobie i innym późniejsze analizowanie kodu w poszukiwaniu błędów oraz w późniejszym jego zrozumieniu .

Dzisiaj omówię podstawową postać funkcji i najprostszy skrypt który pokaże nam jak się posługiwać taką funkcją. Tak więc zaczynajmy.

Podstawowa postać funkcji

Tradycyjnie na początek zobaczysz sobie kod skryptu. Przeanalizuj sobie go na spokojnie, zobacz jakie są zmiany i jak skończysz to czytaj dalej.

Kod pokazujący tworzenie obiektu Sphere za pomocą funkcji

```
utility funkcje "Funkcje"
(

    fn funkcja =
    (
        Sphere segs:20 smooth:off
    )

    button sph "SPHERE"

    on sph pressed do
    (
        funkcja()
    )

)
```

Widzisz funkcję ?. No właśnie, to ta która rozpoczyna się przez **fn** dalej mamy zmienną funkcji czyli jej nazwę. Za znakiem równości rozpoczyna się cała jej zawartość. Każda funkcja musi mieć swój nawias otwierający i zamykający. Zawartość tej funkcji jest już Ci chyba znana. Po prostu tworzymy kulę na 20 segmentów i wyłączyłem opcję smooth. Teraz zobacz co się zmieniło w funkcji która obsługuje działanie przycisku. Zamiast kodu zawartego w funkcji mamy tylko jej wywołanie. Każdą funkcję wywołujemy przez jej nazwę i dwa nawiasy. Te nawiasy można traktować tak jak coś takiego żeby kompilator wiedział że pomiędzy tymi nawiasami mieści się cała zawartość gdzieś tam stworzonej funkcji. Jak widzisz budowa funkcji takiej podstawowej jest całkiem prosta, zarazem stosowanie jej jest bardzo wygodne. Teraz mi pewnie powiesz że ten przykład jest beznadziejny. Hmm..... może i tak, ponieważ funkcja tylko tworzy kulkę ale teraz zmodyfikuj sobie funkcję obsługującą przycisk na coś takiego

Funkcja kontrolująca położenie kuli

```
utility funkcje "Funkcje"
(

    fn funkcja =
    (
        Sphere segs:20 smooth:off
    )

    button sph "SPHERE"

    on sph pressed do
    (
        a = funkcja()
        a.pos = [0,0,-20]
    )

)
```

A teraz wyobraź sobie że masz tam bardziej rozbudowaną konstrukcję kuli ze wszystkimi możliwymi dla niej parametrami, załóżmy też że masz dużo bardziej rozbudowany kod. Nagle stwierdzasz że gdzieś masz taką funkcję która tworzy tą kulkę ale jej położenie Ci teraz nie odpowiada, szukać się nie chce więc skoro pamiętasz jak się nazywa ta funkcja po prostu modyfikujesz jej parametry z innego miejsca. Możesz dołożyć jej jakiś modyfikator i takie tam. Zresztą kiedyś zobaczysz że się to przydaje.

Jak zauważyłeś przypisałem całą zawartość funkcji przypadkowo wybranej zmiennej w moim przypadku litera „a". Teraz litera a to nasza kula. Więc nic prostszego gdy chcemy dołożyć lub zmienić parametry. Do tego posłuży nam operator kropki. Tu jest taka mała rozbieżność pomiędzy samą budową obiektu a odwoływaniem się do niego przez funkcje.

Tak więc aby dostać się do właściwości naszej kuli poprzez funkcję musimy napisać tak jak poniżej

```
a = funkcja()
a.pos = [0,0,-20]
```

a.pos daje nam dostęp do pozycji naszej kuli. Teraz żeby nadać jej odpowiednie parametry musimy po przez znak równości podać całą zawartość tablicy i to wszystko. No i to by było na tyle w dzisiejszej lekcji. Pomyśl sam nad tym gdzie można by zastosować funkcję w twoim skrypcie. Spróbuj napisać jakiś prosty skrypt który będzie ją wykorzystywał. Przy następnej okazji jak będzie potrzeba to omówię bardziej rozszerzoną postać funkcji która będzie odbierać dane od użytkownika. Stanie się wtedy bardziej przydatna. Postaram się też powiedzieć parę słów o tablicach i strukturach w następnych częściach. Tak więc ćwicz bo masz coraz więcej rzeczy do zapamiętania.

Autor :	**Adrian Majchrzak**
Przeznaczenie :	**www.max3d.pl**