Max Script for 3D Studio MAX R4 część IV

by Adrian Majchrzak

Witam wszystkich w czwartej już części skryptowania w maxie. Wreszcie dzisiaj zaczniemy budować skrypt który tak jak plugin umożliwi nam utworzenie obiektu na razie standardowego oraz będziemy mogli na bieżąco kontrolować zachowywanie się obiektu w scenie. Ten art. jest uzupełnieniem wcześniejszej lekcji więc jeśli nie czytałeś to dobrze by było gdybyś to zrobił teraz. Zanim jednak zaczniemy bawić się obiektami napiszę parę słów jeszcze na temat kontrolek i ich formatowaniu.

Menu typu Rollout.

W każdym większym pluginie mamy do wyboru masę opcji które są pogrupowane w ramki a te ramki jeszcze są poumieszczane w odpowiednich sekcjach czyli roletach które możemy zwijać i rozwijać. Być może i Ty stworzysz kiedyś bardzo zaawansowany skrypt i będziesz chciał też pogrupować swoje opcje. Poniżej zamieszczam taki kod który tworzy zwiniętą i rozwiniętą roletę z przyciskami.

Skrypt pokazujący zasadę budowania rolet

```
utility skrypt "Rolety"
(
	rollout bout "O skrypcie"
	(
	button bout "About" width:65 height:20
	on bout pressed do
	messagebox "Moj pierwszy skrypt z roletami”
	title:"O moim skrypcie
	)

		rollout buduj "Obiekty"
		(
		button dodaj "dodaj" width:60 height: 30
			on dodaj pressed do
				messagebox "he he" \
				title:"Cos tam"
		 )

	on skrypt open do
	(
	    addRollout bout
	    addRollout buduj
	)

	on skrypt close do
	(
	    removeRollout bout
	    removeRollout buduj
	)
)
```

No i tak wygląda nasz skrypt z roletami. Teraz wypadało by coś napisać o tym jak to działa. Więc zaczynam.

Pierwszych dwóch linijek kodu nie muszę opisywać to już wiecie co one robią. Dalej mamy coś takiego

Tworzymy roletę

```
rollout bout "O skrypcie"
       (
       button bout "About" width:65 height:20
       on bout pressed do
       messagebox "Moj pierwszy skrypt z roletami”
       title:"O moim skrypcie
       )
```

Najpierw mamy słówko rollout które informuje kompilator że tworzymy sobie taką roletę. Dalej mamy zmienną (wcześniej mówiłem nazwę własną) no i dalej w cudzysłowie mamy nazwę rolety. Dalej pomiędzy nawiasami otwierającym i zamykającym mamy opis przycisku „About” znamy jego konstrukcję podstawową. Pamiętacie jak mówiłem że są dodatkowe różne opcje dla kontrolek ?. No właśnie. Za pomocą opcji width i height możemy ustalić sobie wysokość i długość naszego przycisku.

Teraz coś o funkcji na której oparte jest działanie naszego przycisku. Podstawową postacią tej funkcji jest

on <zmienna> pressed do
(

)

Uwaga	Jeżeli w rolecie jest tylko jeden przycisk możesz nie stosować nawiasów. Jeśli masz więcej przycisków będziesz musiał stosować nawiasy do oddzielania między sobą poszczególnych funkcji żeby sobie nie przeszkadzały i nie było problemów przy kompilacji.

Tak więc aby uruchomić nasz przycisk musimy zbudować funkcję która nam go uruchomi. W miejsce gdzie powinna być zmienna w funkcji wstawiamy tą jak to mówiłem nazwę własną naszego przycisku. Teraz mając tak zbudowaną funkcję możemy rozszerzyć ją o różne opcje. Np. dodajmy komunikat.

Dodawanie komunikatów

Wszyscy którzy programują w C++ czy basicu znają polecenie messagebox. Dla tych którzy nie wiedzą o co chodzi to jest to taka funkcja która tworzy nam okienko reagujące na jakieś zdarzenie. Np. jeśli w windowsie wystąpi błąd to pokazuje się nam okienko informujące nas o wystąpieniu błędu. Takich komunikatów jest masa, każdy programista może za ich pomocą informować użytkownika o tym że program wykonał błąd lub zapytać czy na pewno chce coś zrobić. Funkcję messagebox pewnie jeszcze omówię ale na razie same podstawy więc jej składnią która będzie nam potrzebna jest

messagebox ”tresc skryptu w cudzysłowie” title: ”nazwa okienka”

Taką wiadomość zobaczysz gdy uruchomisz skrypt i wciśniesz przycisk About. W drugiej rolecie jest to samo więc nie ma co omawiać.

Dalej bardzo ważne są dwie klasy które mają za zadanie kontrolować nasze rolety przy zamykaniu i otwieraniu. Ich postać jest następująca

Klasy kontrolujące zachowanie się rolet

```
on skrypt open do
(
    addRollout bout
    addRollout buduj
)

on skrypt close do
(
    removeRollout bout
    removeRollout buduj
)
```

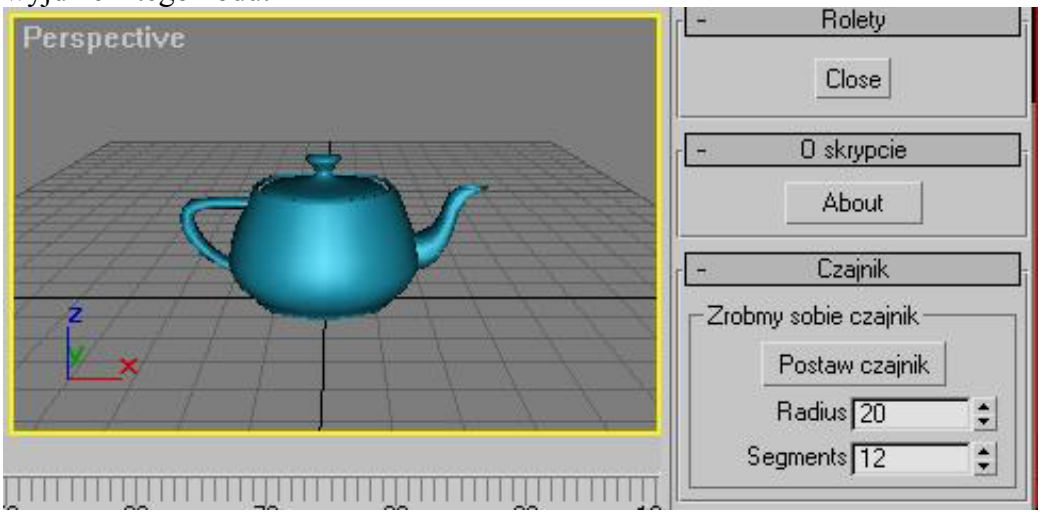
No i co my tu mamy ?. Po pierwsze mamy dwie funkcje które nadzorują działanie rolet w czasie zamykania i otwierania skryptu. Zwróć uwagę że przy tworzeniu tej funkcji odwołuję się do zmiennej skryptu (chodzi o zmienną którą tworzę w pierwszej linii całego skryptu) a nie rolety. To daje dostęp do wszystkich rolet w całym skrypcie. Dalej mamy opcję open do. Czyli po prostu utworzyliśmy funkcję która nadzoruje otwieranie się rolet w momencie otwarcia skryptu.. Dalej po między nawiasami mamy klasę **addRollout**. Czyli dodaj roletę. Dalej wpisujemy nazwę zmiennej określającą daną roletę. I tak wygląda działanie tej funkcji.

Funkcja która nadzoruje zamykanie się rolet to ta poniżej. Podajemy że zamykamy skrypt a w nawiasach za pomocą klasy **removeRollout** czyli usuń rolety czyścimy pamięć ze zbędnych śmieci. Działa to na zasadzie konstruktora i destruktora.

Dobra teraz to na co czekaliście czyli wreszcie zobaczysz jak napisać skrypt który będzie budował nam czajnik, kontrolował ilość jego segmentów oraz zmieniał jego rozmiar. Fajnie co ?. Mnie też się podoba. Zaczynamy.

Budujemy skrypt tworzący i kontrolujący obiekty

Jak zwykle na początek załączam kod całego skryptu a potem opis. Przejrzyj dokładnie co jest nowego i czytaj dalej. Poniżej przedstawiam również rysunek tego co nam wyjdzie z tego kodu.



Podoba się ?. To do roboty. Najpierw pokażę cały kod skryptu ze wszystkim co dziś robiliśmy i dodatkowe funkcje które odpalą nam czajnik.

Kompletny skrypt budujący czajnik i wyświetlający komunikaty

```
utility skrypt "Rolety"
(

local pot

	rollout bout "O skrypcie"
	(
	button bout "About" width:65 height:20
	on bout pressed do
	messagebox "skrypt z roletami" \
	title:"O moim skrypcie"
	)

		rollout buduj "Czajnik"
		(
		group "Zrobmy go wreszcie"
		(
		 button tea "Teapot"
		 spinner rad "Radius" range:[10,50,20] type:#integer
		 spinner seg "Segments" range:[4,32,12] type:#integer
		)
				on tea pressed do
					(
					pot=teapot radius:rad.value
					pot.name="TestPot"
					pot.segs=seg.value
					)

				on rad changed value do
					pot.radius=value

				on seg changed value do
					pot.segs=seg.value
		)


	on skrypt open do
	(
	    addRollout bout
	    addRollout buduj
	)

	on skrypt close do
	(
	    removeRollout bout
	    removeRollout buduj
	)
)
```

Cały nowy kod zaznaczyłem żeby łatwiej było zobaczyć co nowego doszło. Na samym początku zrobimy sobie zmienną lokalną czyli taką żeby w całym skrypcie było wiadomo że akurat nazwa pot to jest taka nazwa która tyczy się naszego czajnika. Deklarujemy zmienne lokalne przez słówko **local** potem przypisujemy mu odpowiednią zmienną i koniec. Teraz jedziemy dalej. Widzimy tu konstrukcję rolety dla naszego dzbanka, dalej mamy konstrukcję przycisku tworzącego nasz czajnik. Niżej dwie funkcje które już wcześniej poznaliśmy czyli spinner. Posłużą one nam do ustawienia średnicy i ilości segmentów dla czajnika. Tak jak mówiłem każdy obiekt ma swoje parametry. Chodzi tu o zmienne sterujące naszym obiektem. Nasz obiekt spinner został określony zmienną Integer.

Co nam daje określenie typu obiektu ?. Ano np. coś takiego że czasem możemy używać parametrów zmiennoprzecinkowych. Do takich parametrów zastosujemy zmienną typu float, dla liczb całkowitych do 65535 zastosujemy zmienną Integer . Uniemożliwia ona całkowicie działanie na liczbach zmiennoprzecinkowych. Tak więc nasz czajnik nie może mieć 1,5 segmentu dlatego zastosujemy zmienną Integer. Jak stosować typy zmiennych dla obiektów przedstawia poniższy przykład

```
spinner rad "Radius" range:[10,50,20] type:#integer
```

za nawiasem kwadratowym podajemy że definiujemy obiekt według jakiegoś typu więc piszemy **type:#integer**

Teraz jak to się dzieje że pokazuje się nam czajnik. To pokazuje nam poniższy kod

```
on tea pressed do
	(
	pot=teapot radius:rad.value
	pot.name="TestPot"
	pot.segs=seg.value
	)
```

To jest jak się domyślacie funkcja obsługująca przycisk stawiający nam czajnik. Do zmiennej lokalnej pot przypisujemy prawdziwą nazwę dzbanka czyli teapot. Dzięki temu mamy dostęp do wszystkich funkcji naszego obiektu. Dalej mamy odwołanie się do naszego spinnera który nam ustawi średnicę czajnika. Piszemy więc **radius:rad.value** Dla prawdziwej wartości radius przypisujemy zmienną rad z opcji spinner. Po kropce mamy opcję value. Działa to tak że przez value przesyłane są dane ze spinnera do zmiennej rad która określa nam obiekt spinner a znowu ta zmienna rad przesyła już przetworzone informacje do właściwości radius naszego czajnika. Niżej mamy przykład odwołania się do nazwy czajnika. Jeśli postawimy czajnik to nazwa pokaże się TestPot a nie TeaPot01 jak normalnie. Ostatnia opcja pokazuje nam jak możemy przesyłać wartość ilości segmentów do właściwości czajnika segs.

Poniżej w tabeli znajdują się funkcje które reagują na zdarzenie wywoływane przez funkcję Value. Jak to działa.?. A no tak. Jeśl wartość Value dla rad się zmienia to prześlij dane z Value do zmiennej właściwości radius obiektu pot. Analogicznie działa druga funkcja która nadzoruje przesył danych z wartości Value do właściwości segs obiektu pot.

Funkcje kontrolujące przesyłanie danych przez opcję Value
on rad changed value do pot.radius=value on seg changed value do pot.segs=seg.value

No i to by było na tyle. Mam nadzieję że ta lekcja nie była zbytnio trudna i wszystko było w miarę jasne. Teraz zacznie się coraz trudniej. No ale o tym następnym razem.

Autor :	**Adrian Majchrzak**
Wyłącznie do użytku dla :	**www.max3d.pl**