**Laboratorium grafiki komputerowej i animacji**

**Ćwiczenie IV - Biblioteka OpenGL -**

**transformacje przestrzenne obiektów**

Przygotowanie do ćwiczenia:

1. Zapoznać się z transformacjami przestrzennymi (obrót, przesunięcie, skalowanie),
2. Zapoznać się z opisem jednorodnym transformacji przestrzennych,
3. Zapoznać się z zestawem komend OpenGL umożliwiającym dokonywanie transformacji przestrzennych obiektów na scenie wraz z podstawowymi zasadami posługiwania się tymi komendami.

Przebieg ćwiczenia:

1. Założenia:
   1. Celem prac na zajęciach laboratoryjnych jest wykonanie ruchomego modelu siatki manipulatora Puma.
   2. Wynikiem prac na dzisiejszych zajęciach ma być program zbliżony w działaniu do programu „**puma\_siatka.exe**” dostarczonego do materiałów laboratoryjnych.
   3. Realizacja ćwiczenia polega na uzupełnieniu kodu programu „**gl\_template**” modyfikowanego na ostatnich zajęciach.
   4. W realizacji prac wzorować się należy na rozwiązaniach przyjętych w programie „**robot1**” również dołączonym do materiałów laboratoryjnych.
2. Przebieg ćwiczenia:
   1. Załadować do programu VS projekt Gl\_Template modyfikowany na poprzednich zajęciach i zawierający opracowane siatki: sześcianu, walca oraz ramienia robota.
   2. Utworzyć funkcję o prototypie:

**void robot(double d1, double d2, double d3);**

Model robota będzie posiadał 3 stopnie swobody: obród wokół podstawy, obrót pierwszego ramienia i obrót drugiego ramienia, stąd do funkcji zostaną przekazane 3 parametry określające jego bieżącą konfigurację. Funkcja będzie korzystała z funkcji rysujących walec i ramię robota.

* 1. Powołać do życia 3 zmienne globalne typu double do przechowywania konfiguracji robota:   
     **double rot1, rot2, rot3;**
  2. Wywołać funkcję **robot** wewnątrz funkcji **RenderScene()** z parametrami wywołania rot1, rot2, rot3:   
     **robot(rot1, rot2, rot3);**
  3. Wewnątrz funkcji robot() przetransformować bazowy układ współrzędnych do miejsca, w którym będzie rysowana podstawa robota. Zmianę położenia układu współrzędnych pokazano na rysunku 1   
     

Rys. 1 Oczekiwana zmiana orientacji i położenia układu współrzędnych.

Układ bazowy obrócono o -90 stopni wokół osi x, a następnie przesunięto o wektor [0,0,-50].

W bibliotece OpenGL można dokonać takiej transformacji na macierzy modelowania-transformacji wywołując komendy:

**glRotated(-90,1,0,0);**

**glTranslated(0,0,-50);**

* 1. W przetransformowanym układzie współrzędnych wyrysować podstawę robota:  
     **walec(30,5);**

Wynik rysowania pokazano na rysunku 2.



Rys. 2. Wyrysowanie podstawy robota.

* 1. Przesunąć układ współrzędnych o wektor [0,0,5] (5-wysokość walca modelującego podstawę robota). Wyrysować model walca obrazujący 1 część kolumny robota:

**glTranslated(0,0,5);**

**walec(10,40);**

Rezultat wykonania nowych komend OpenGL pokazano na rysunku 3.



Rys. 3. Wyrysowanie 1 części kolumny robota.

* 1. Przesunąć układ współrzędnych o wektor [0,0,40] (40-wysokość walca modelującego podstawę robota), a następnie obrócić układ współrzędnych o wartość parametru **d1** wokół osi. Należy pamiętać, że wartość parametru **d1** zależy od wartości zmiennej globalnej **rot1**.   
     Następnie wyrysować kolejny walec stanowiący drugą część kolumny robota:  
     **glTranslated(0,0,40);**

**glRotated(d1,0,0,1);**

**walec(10,40);**

Rezultat dotychczasowego ciągu komend pokazano na rysunku 4.   


Rys. 4. Wyrysowanie 2 części kolumny robota.

* 1. Obsługę komunikatu **WM\_KEYDOWN** w funkcji **WndProc()** uzupełnić (przed wywołaniem funkcji **InvalidateRect(hWnd,NULL,FALSE);**) o następujący kod:

**if(wParam == '1')**

**rot1 -= 5.0f;**

**if(wParam == '2')**

**rot1 += 5.0f;**

Od tej chwili przyciśnięcie klawisza ‘1’ powoduje zmniejszenie rot1, z kolei przyciśnięcie klawisza ‘2’ - zwiększenie wartości zmiennej rot1, a następnie wymuszenie wyrysowania sceny. W konsekwencji druga część kolumny robota obraca się pod wpływem przyciskania klawiszy ‘1’ i ‘2’.

* 1. Uzupełnić obsługę komunikatu **WM\_KEYDOWN** o możliwość modyfikacji stanu zmiennej **rot2** po przyciśnięciu klawiszy ‘3’, ‘4’ oraz modyfikacji stanu zmiennej **rot3** po przyciśnięciu klawiszy ‘5’, ‘6’.
  2. Przesunąć układ współrzędnych o wektor [0,0,40], obrócić układ współrzędnych o kąt 90 stopni wokół osi y i przesunąć układ współrzędnych o wektor [0,0,-20]:

**glTranslated(0,0,40);**

**glRotated(90,0,1,0);**

**glTranslated(0,0,-20);**

* 1. Wyrysować kolejny walec w nowej pozycji lokalnego układu współrzędnych:  
     **walec(10,40);**

Rezultat dotychczasowego skryptu pokazano na rysunku 5.



Rys. 5. Uzupełnienie kolumny robota o element umożliwiający przyłączenie ramienia.

* 1. Przesunąć układ współrzędnych o wektor [0,0,40], obrócić układ współrzędnych o (+90º+d2) wokół osi z, wyrysować ramię robota:

**glTranslated(0,0,+40);**

**glRotated(90+d2,0,0,1);**

**ramie(15,10,5,30);**

Dotychczasowy rezultat wykonania skryptu pokazano na rys. 6.



Rys. 6. Uzupełnienie modelu robota o pierwsze ramię.

* 1. Przesunąć układ współrzędnych o wektor [30,0,-5], obrócić układ współrzędnych o kąt d3 wokół osi z, wyrysować ramię robota:   
     **glTranslated(30,0,-5);**

**glRotated(d3,0,0,1);**

**ramie(15,10,5,30);**

Dotychczasowy rezultat wykonania skryptu pokazano na rys. 7.



Rys. 7. Kompletna siatka robota.

* 1. Skrypt rysujący model robota rozpocząć od polecenia glPushMatrix() i zakończyć poleceniem glPopMatrix():   
     **glPushMatrix();**

**// skrypt rysujący robota…**

**glPopMatrix();**

Takie zastosowanie funkcji glPushMatrix() i glPopMatrix() powoduje, że wszystkie transformacje przestrzenne zastosowane w skrypcie tworzącym model robota nie wpływają na rysowanie innych elementów sceny.

* 1. Obsługę komunikatu **WM\_CREATE** w funkcji **WndProc()** uzupełnić o wywołanie funkcji:

**SetTimer(hWnd,101,200,NULL);**

Spowoduje to zainstalowanie w programie budzika, który będzie powiązany z oknem programu, będzie miał identyfikator 101 i będzie wysyłał specjalny komunikat **WM\_TIMER** do procedury okna co 200 [ms].

* 1. Obsługę komunikatu **WM\_DESTROY** w funkcji **WndProc()** uzupełnić o wywołanie funkcji:

**KillTimer(hWnd,101);**

Funkcja oddaje systemowi operacyjnemu budzik tuż przed zakończeniem działania programu.

* 1. Powołać globalną całkowitoliczbową zmienną **licznik.**
  2. Wprowadzić do funkcji **WndProc()** mechanizm obsługi nowego komunikatu **WM\_TIMER:**  
     **case WM\_TIMER:**

**if(wParam==101)**

**{**

**licznik++;**

**if(licznik<15)**

**rot2+=15.0;**

**if(licznik>15 && licznik < 30)**

**rot2-=15.0;**

**if(licznik>30)**

**{licznik=0;}**

**InvalidateRect(hWnd,NULL,FALSE);**

**}**

**break;**

Tak zmodyfikowany program będzie stosował budzik do automatycznego generowania kolejnych klatek animacji.

* 1. Zaproponować realizację funkcji dwa\_roboty(), która będzie modelowała gniazdo robotów składające się z 2 egzemplarzy robota opracowanego wcześniej.

Uwaga: Należy umiejętnie posłużyć się wywołaniem komend glPushMatrix() i glPopMatrix() w celu odizolowania transformacji przestrzennych.

* 1. Opracować własny scenariusz poruszania się robotów w gnieździe.