Laboratorium grafiki komputerowej i animacji

Ćwiczenie IV - Biblioteka OpenGL - transformacje przestrzenne obiektów

Przygotowanie do ćwiczenia:

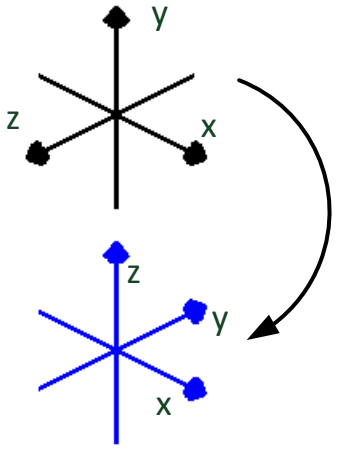
1. Zapoznać się z transformacjami przestrzennymi (obrót, przesunięcie, skalowanie),
2. Zapoznać się z opisem jednorodnym transformacji przestrzennych,
3. Zapoznać się z zestawem komend OpenGL umożliwiającym dokonywanie transformacji przestrzennych obiektów na scenie wraz z podstawowymi zasadami posługiwania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

Przebieg ćwiczenia:

1. Założenia:
    a. Celem prac na zajęciach laboratoryjnych jest wykonanie ruchomego modelu siatki manipulatora Puma.
    b. Wynikiem prac na dzisiejszych zajęciach ma być program zbliżony w działaniu do programu „**puma_siatka.exe**" dostarczonego do materiałów laboratoryjnych.
    c. Realizacja ćwiczenia polega na uzupełnieniu kodu programu „**gl_template**" modyfikowanego na ostatnich zajęciach.
    d. W realizacji prac wzorować się należy na rozwiązaniach przyjętych w programie „**robot1**" również dołączonym do materiałów laboratoryjnych.

2. Przebieg ćwiczenia:
    a. Załadować do programu VS projekt Gl_Template modyfikowany na poprzednich zajęciach i zawierający opracowane siatki: sześcianu, walca oraz ramienia robota.
    b. Ustalić następujący tryb rysowania wielokątów: przednie i tylne ściany rysowane jako siatka (należy w funkcji RenderScene() uaktywnić wywołanie funkcji **glPolygonMode(GL_FRONT_AND_BACK,GL_LINE);**)
    c. 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.
    d. Powołać do życia 3 zmienne globalne typu double do przechowywania konfiguracji robota:
    double rot1, rot2, rot3;
    e. Wywołać funkcję **robot** wewnątrz funkcji **RenderScene()** z parametrami wywołania rot1, rot2, rot3:
    robot(rot1, rot2, rot3);
    f. Wewnątrz funkcji robot() przetransformować bazowy układ współrzędnych do miejsca, w którym będzie rysowana podstawa robota. Zmianę położenia układu

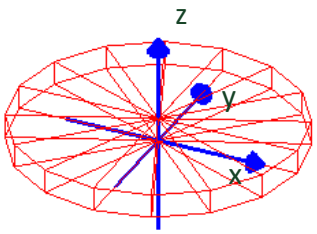
współrzędnych pokazano na rysunku 1



Rys. 1 Oczekiwana zmiana orientacji i położenia układu współrzędnych.

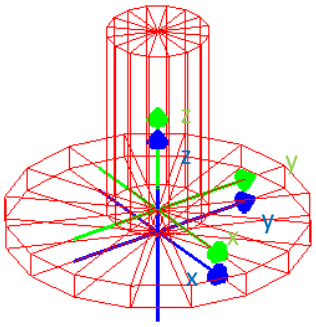
Układ bazowy obrócono o -90 stopni wokół osi x, a następnie przesunięto o wektor [0,0,-50].
W bibliotece OpenGL można dokonać takiej transformacji na macierzy modelowania-transformacji wywołując komendy:
glRotated(-90,1,0,0);
glTranslated(0,0,-50);

g. W przetransformowanym układzie współrzędnych wyrysować podstawę robota:
walec(30,5);
Wynik rysowania pokazano na rysunku 2.



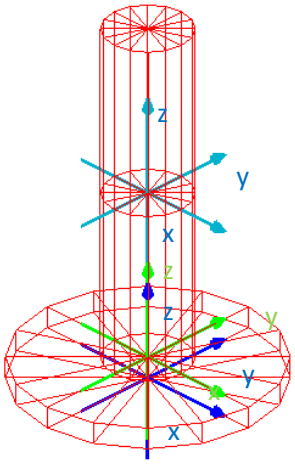
Rys. 2. Wyrysowanie podstawy robota.

h. 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.

i. 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.

j. Obsługę komunikatu **WM_KEYDOWN** w funkcji **WndProc()** uzupełnić (przed wywołaniem funkcji **InvalidateRect(hWnd,NULL,FALSE);**) o następujący kod:

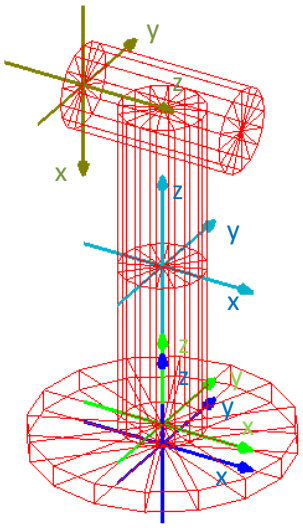
```
if(wParam == '1')
    rot1 -= 5.0f;

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’.

k. 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’.

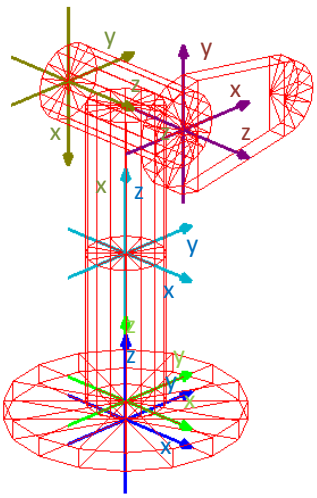
l. Przesunąć układ współrzędnych o wektor [0,0,40], obrócić układ współrzędnych o kąt 90 stopni wokół osi y i przesunąć układ współrzędnych o wektor [0,0,-20]:
glTranslated(0,0,40);
glRotated(90,0,1,0);
glTranslated(0,0,-20);

m. Wyrysować kolejny walec w nowej pozycji lokalnego układu współrzędnych:
walec(10,40);
Rezultat dotychczasowego skryptu pokazano na rysunku 5.



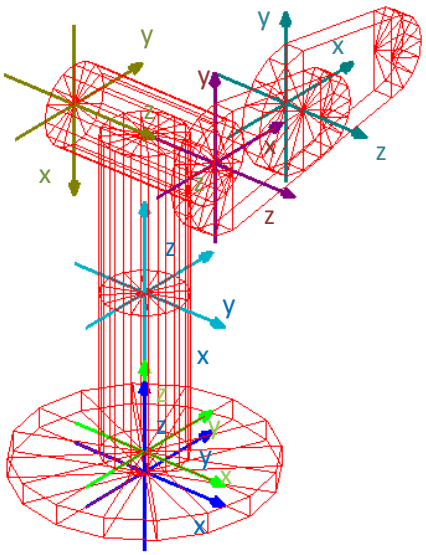
Rys. 5. Uzupełnienie kolumny robota o element umożliwiający przyłączenie ramienia.

n. 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.

o. 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.

p. 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.

q. 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].

r. Obsługę komunikatu **WM_DESTROY** w funkcji **WndProc()** uzupełnić o wywołanie funkcji:
KillTimer(hWnd,101);

Funkcja oddaje systemowi operacyjnemu budzik tuż przed zakończeniem działania programu.

s. Powołać globalną całkowitoliczbową zmienną **licznik.**

t. 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.

u. 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.

v. Opracować własny scenariusz poruszania się robotów w gnieździe.