**Laboratorium grafiki komputerowej i animacji**

**Ćwiczenie VI - Biblioteka OpenGL -**

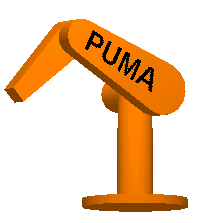
**teksturowanie**

Przygotowanie do ćwiczenia:

1. Zapoznać się z zasadami teksturowania obiektów w OpenGL.
2. Zapoznać się z zestawem komend OpenGL umożliwiającym definiowanie kwadryk (biblioteka GLU).

Przebieg ćwiczenia:

1. Założenia:
   1. Celem prac na zajęciach laboratoryjnych jest uzupełnienie trójwymiarowego modelu robota Puma o teksturę z nazwą robota (rysunek 1.1)
   2. Wynikiem prac na dzisiejszych zajęciach ma być program zbliżony w działaniu do programu „**puma\_tekstury.exe”** dołączonego do materiałów laboratoryjnych.
   3. Realizacja ćwiczenia polega na uzupełnieniu kodu programu „**gl\_Template**” modyfikowanego na ostatnich zajęciach.



Rys 1.1 Teksturowany model manipulatora Puma

1. Przebieg ćwiczenia:
   1. Program glTemplate zawiera już pewne składniki pokazujące, jak można dokonywać teksturowania wielokątów:
      * Wprowadzono zmienne globalne:

BITMAPINFOHEADER bitmapInfoHeader; // nagłówek obrazu

unsigned char\* bitmapData; // dane tekstury

unsigned int texture[2]; // obiekt tekstury

* + - Wprowadzono definicję, zastosowaną później w sprawdzeniu, czy otwarto plik typu bitmapa:

#define BITMAP\_ID 0x4D42

* + - Wprowadzono funkcję odczytującą plik typu \*.bmp i zwracającą wskazanie na tablicę pikseli dostoswaną do konwersji na teksturę OpenGL (szczegóły we wprowadzeniu):

unsigned char \*LoadBitmapFile(char \*filename, BITMAPINFOHEADER \*bitmapInfoHeader;)

* + - Zmodyfikowano obsługę komunikatu WM\_CREATE o zestaw wywołań funkcji, które tworzą dwie tekstury OpenGL (szczegóły we wprowadzeniu).
    - Wprowadzono funkcje:

void skrzynka(void);

void walec01(void);

void kula(void);

* 1. Proszę aktywować w funkcji **RenderScene()** funkcje: skrzynka(), walec01(), kula(). W wymienionych funkcjach pokazano kolejno: sposób rozpinania tekstury na wielokącie, zastosowanie kwadryk, pokrywanie kwadryk teksturami.
  2. **Proszę rozpiąć na jednym z ramion robota teksturę z napisem „PUMA” dostarczoną w materiałach do ćwiczenia (plik ”napis.bmp”) w podobny sposób.**

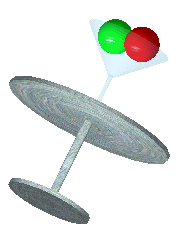
**UWAGA: Aby napis był odpowiednio rozłożony na trapezie, trzeba odpowiednio dobrać współrzędne tekstury.**

* 1. Uwaga: Istnieje możliwość przedefiniowania w programie sposobu odwzorowywania przestrzeni z rzutu prostokątnego na rzut perspektywiczny. W tym celu należy „zablokować” w funkcji ChangeSize() wykonywanie funkcji glOrtho(), a „odblokować” działanie funkcji gluPerspective(). Wszystkie transformacje przestrzenne wprowadzone do sceny w funkcji RenderScene() należy także poprzedzić wywołaniem funkcji:   
     **glTranslated(0,0,-200);.**

Spowoduje to „cofnięcie” kamery od centrum sceny.

* 1. Zadnie dodatkowe:

Należy skonstruować prostą scenę z wykorzystaniem kwadryk, podobną do sceny w programie „**lody.exe**” (rysunek 1.2)



Rys 1.2 Scena wykorzystująca kwadryki OpenGL

Uwagi do postawionego problemu:

Efekt przezroczystości osiąga się postępując według następujących zasad:

* + - „Przezroczyste” elementy sceny muszą być rysowane na końcu
    - Funkcję SetupRC() należy uzupełnić o wywołanie komendy:  
      glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);
    - Elementy „przezroczyste” umieszcza się na scenie w następujący sposób (wartość A\_val decyduje o przezroczystości obiektu):  
       glEnable(GL\_BLEND);

glColor4f(R\_val, G\_val, B\_val, A\_val);

gluCylinder(obj,

0.0,

30.0,

0.0,

20.0,

3.0

);

glDisable(GL\_BLEND);

* + - Dla zapewnienia odpowiedniego mieszania kolorów elementów teksturowanych z elementami przezroczystymi przed wywołaniem obiektu zateksturowanego należy wywołać komendę:  
      glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_BLEND);