Grafika 3D.

Zadanie

Napisać program do dwuwymiarowej wizualizacji ścian bocznych graniastosłupa o podstawie dwunastokąta foremnego. Graniastosłup powinien zostać wyświetlony w rzucie ortogonalnym. Suwaki umieszczone na panelu kontrolnym powinny pozwalać na obracanie bryły wokół trzech osi głównych obiektu (a nie wokół osi układu współrzędnych). Proszę przyjąć lewoskrętny układ współrzędnych i zadbać, aby obroty wokół każdej osi odbywały się w kierunku kąta dodatniego. Kontrolką umieszczoną po prawej stronie obszaru wyświetlania można kontrolować sposób wyświetlania bryły. Do wyboru mamy obiekt szkieletowy oraz obiekt bez dolnej i górnej podstawy z wypełnionymi ścianami. W tym drugim przypadku ściany od strony zewnętrznej są pokolorowane naprzemiennie na czerwono i niebiesko. Od strony wewnętrznej wszystkie ściany są szare. Wyświetlając ściany musimy zadbać, aby krawędzie niewidoczne nie były wyświetlane.

Cel

Zapoznanie się z podstawowymi transformacjami 3D, składaniem przekształceń w przestrzeni trójwymiarowej oraz algorytmów usuwania linii zasłoniętych.

Środki

Środowisko wxDev-C++.

Opis istniejącego kodu

Plik *vecmat.h* zawiera dwie klasy: **`Vector4`** i **`Matrix4`**. Klasy te reprezentują odpowiednio wektor o czterech składowych oraz macierz o rozmiarach 4x4.

Klasa **`Vector4`** zawiera metodę pozwalającą ustawić współrzędne wektora **`Set(x,y,z)`** oraz pobranie jego współrzędnych **`GetX()`**, **`GetY()`** oraz **`GetZ()`**. Konstruktor klasy ustawia automatycznie wartość czwartej składowej na 1.0.

Klasa **`Matrix4`** posiada przeciążony operator „**`*`**” (mnożenie) dla operacji **`Matrix4*Matrix4`** oraz **`Matrix4* Vector4`**. Konstruktor klasy automatycznie ustawia wartość elementu (4,4) macierzy na 1.0. Do konkretnych składowych macierzy odwołujemy się przez pole data, które jest publiczne.

Użycie tych klas jest bardzo proste. Na przykład pomnożenie wektora $\begin{bmatrix} 2.3 \\ 1.2 \\ 3.3 \\ 1.0 \end{bmatrix}$ przez macierz $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ wygląda następująco:

```
Vector4 y,x;
Matrix4 m;
```

```
x.Set(2.3,1.2,3.3);
m.data[0][0]=1.0;
m.data[1][1]=1.0;
m.data[2][2]=1.0;
y=M*x;
```

Przygotowany kod zawiera już wszystkie niezbędne kontrolki.

Ponieważ wyświetlana bryła jest bardzo prosta, przedstawioną na wykładzie koncepcję list zamieniono na tablice.

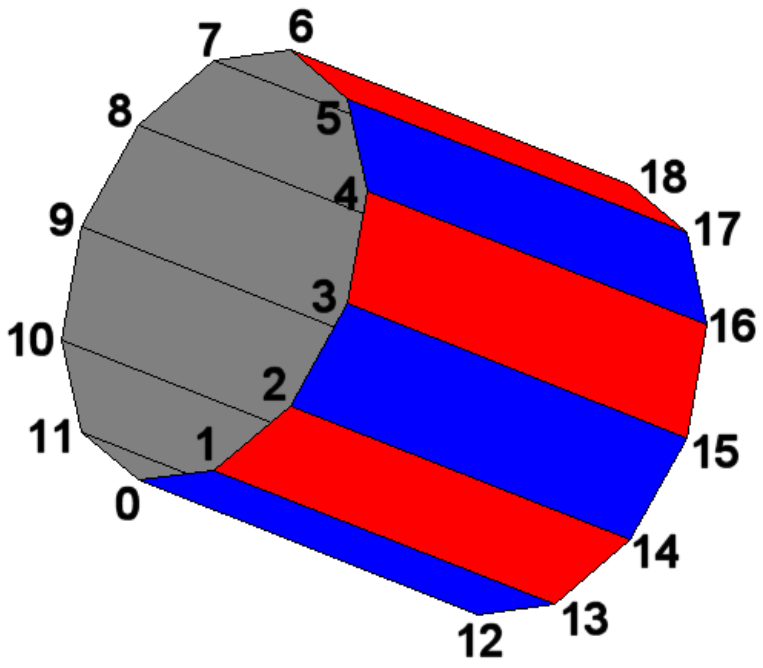
W kodzie programu przygotowano następujące tablice:

`double P_x[24],P_y[24],P_z[24];` - każda z powyższych tablic zawiera jedną współrzędną i-tego wierzchołka

`int E[36][2];` - pierwszy indeks tablicy numeruje krawędzie, drugi numer wierzchołka będącego początkiem i numer wierzchołka będącego końcem krawędzi

`int S[12][4];` - pierwszy indeks tablicy numeruje ścianę, drugi cztery krawędzie tworzące daną ścianę

Pierwsze trzy tablice zostały już wypełnione. Pozostałe dwie należy wypełnić samodzielnie w miejscu oznaczonym w programie. Numerację wierzchołków przedstawia poniższy rysunek:



Kod do uzupełnienia

W zadaniu należy uzupełnić metodę **`void Lab05Frm::Repaint()`** oraz fragment konstruktora klasy **`Lab_05Frm`** odpowiedzialny za wypełnianie tablic krawędzi i ścian.

Suwaki sterujące mają nazwy przedstawione poniżej:

`WxSB_RotateX` – obrót wokół osi X

`WxSB_RotateY` – obrót wokół osi Y

`WxSB_RotateZ` – obrót wokół osi Z

Kontrolka zmiany sposobu wyświetlania nosi nazwę:

`WxCB_HideLines`

Jak się przygotować przed zajęciami

W zasadzie powinny wystarczyć wiadomości z wykładu. W szczególności proszę sobie powtórzyć:

- Transformacje obiektów 3D w postaci macierzowej.
- Składanie przekształceń (proszę zwrócić uwagę na kolejność).
- Algorytm malarski oraz algorytm wektora normalnego.