Grafika 3D.

Zadanie

Napisać program do dwuwymiarowej wizualizacji obiektów trójwymiarowych wyświetlanych metodą krawędziową. Definicje obiektów należy wczytywać z plików z rozszerzeniem *.geo. Każdy taki plik zawiera zestaw wektorów definiujących obiekt w przestrzeni 3D. Z każdym wektorem związany jest jego kolor. Obiekt 3D powinien zostać wyświetlony w rzucie perspektywicznym. Suwaki umieszczone na panelu kontrolnym powinny pozwalać na przesuwanie obiektu w przestrzeni, jego obracanie wokół trzech osi głównych obiektu (a nie wokół osi układu współrzędnych) oraz skalowanie wzdłuż każdej z osi układu. Proszę przyjąć lewoskrętny układ współrzędnych i zadbać, aby obroty wokół każdej osi odbywały się w kierunku kąta dodatniego. Odcinki przebijające rzutnię należy obciąć w punkcie przebicia. Najlepiej do tego celu zastosować algorytm Cohena-Sutherlanda 1 ograniczony tylko do jednej ściany (rzutni).

Cel

Zapoznanie się z podstawowymi transformacjami 3D, składaniem przekształceń w przestrzeni trójwymiarowej oraz rzutem perspektywicznym.

Środki

Biblioteka wxWidgets.

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 wartośd 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: $\begin{pmatrix} 2.3 \\ 1.2 \\ 3.3 \\ 1.0 \end{pmatrix}$ 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 oraz funkcję:

void GUIMyFrame1::m_button_load_geometry_click(wxCommandEvent& event)

która w reakcji na wciśnięcie kontrolki typu **wxButton** otwiera okno dialogowe wyboru pliku, a następnie z wybranego pliku z rozszerzeniem *.geo wczytuje całą strukturę obiektu 3D. Struktura ta, po wczytaniu, przechowywana jest w wektorze **data** w postaci struktury **Segment** zawierającej początek (**begin**), koniec (**end**) oraz kolor (**color**) wszystkich wektorów które należy wyrysować.

Kod do uzupełnienia

W zadaniu należy jedynie uzupełnić metodę **void GUIMyFrame1::Repaint()** znajdującą się w piku **GUIMyFrame1.cpp**. Całość ryzujemy na obiekcie **wxPanel**. Suwaki sterujące mają nazwy przedstawione poniżej:

WxSB_TranslationX – przesunięcie wzdłuż osi X

WxSB_TranslationY – przesunięcie wzdłuż osi Y

WxSB_TranslationZ – przesunięcie wzdłuż osi Z

WxSB_RotateX – obrót wokół osi X

WxSB_RotateY – obrót wokół osi Y

WxSB_RotateZ – obrót wokół osi Z

WxSB_ScaleX – skalowanie wzdłuż osi X

WxSB_ScaleY – skalowanie wzdłuż osi Y

WxSB_ScaleZ – skalowanie wzdłuż osi Z

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).
- Zasadę rzutu perspektywicznego z jednym punktem zbieżności.