Uzupełnienie o tablicach wierzchołków oraz obsługa klawiatury i myszy w bibliotece GLUT

1. *Dane w tablicach z przeplotem*

   Z oczywistego powodu bedziemy się na razie posługiwali nadal tylko danymi dotyczącymi geometrii wierzchołków oraz ich kolorów. Tym razem jednak stworzymy jedną tablicę zawierającą obie te grupy danych. Podobnie jak w przypadku zwykłych tablic użyjemy przykładu rysującego kwadrat przy pomocy funkcji `kwadrat()` (zob. pliki `opengl4.pdf`). Zdefiniujmy tablicę zawierajacą paczki danych typu: `trzy współrzędne wierzchołka , trzy współrzędne koloru wierzchołka` dla kwadratu. Oznacza to potrzebę wpisania do tablicy co najmniej 24 liczb.

```
float kolory_wierzcholki[] = {1.0, 0.0, 0.0, -0.5, -0.5, 0.0,
                               0.0, 1.0, 0.0,  0.5, -0.5, 0.0,
                               0.0, 0.0, 1.0,  0.5,  0.5, 0.0,
                               1.0, 1.0, 0.0, -0.5,  0.5, 0.0,
                              };
```

   Funkcja odczytująca takie tablice nazywa się

```
glInterleavedArrays(format, offset, wskaznik_do_tablicy),
```

   gdzie `offset` i `wskaznik_do_tablicy` mają takie samo znaczenie jak w przypadku funkcji `glVertexPointer()` (zob. pliki `opengl4.pdf`), natomiast `format` przybiera jedną z następujących form: `GL_V2F, GL_V3F, GL_C4UB_V2F, GL_C4UB_V3F, GL_C3F_V3F`. Stałe te oznaczają format paczek danych: np. `GL_C3F_V3F` oznacza paczkę dwóch danych typu `float`: trójelementowego koloru i trójelementowego wierzchołka. Ponadto funkcja `glInterleavedArrays()` wykonuje za jednym zamachem dwie pierwsze fazy używania tablic wierzchołków w zwykłym przypadku (fazy (a), (b) opisane w pliku `opengl4.pdf`, punkt 2.), tzn. że nie trzeba używać żadnej funkcji typu `glEnable()`. Przykład pokazujący użycie tablicy z przeplotem znajduje się w pliku `ogl51.cpp`. Używa on konwencji z funkcją `glDrawElement()`, ale oczywiście dwa pozostałe sposoby odnoszenia się do tablic wierzchołków opisane w pliku `opengl4.pdf` są również mo możliwe do zastosownia. Przetestowanie tego zostawia się czytelnikowi.

2. *Obsługa klawiatury w bibliotece* `GLUT`.

   Za obsługę klawiatury odpowiedzialne są funkcje których nazwy są parametrem jednej z dwóch funkcji biblioteki `GLUT`, a mianowicie: `glutKeyboardFunc()` lub `glutSpecialFunc()`. Rozpatrzmy je po kolei, zakładając dla ustalenia uwagi, że zdarzenia związane z klawiaturą opisane są w funcji `klawiatura()`.

(a) Funkcja `glutKeyboardFunc(klawiatura)` odpowiada za obsługę znaków ASCII na klawiaturze. Parametr `klawiatura` musi być nazwą funkcji o następującym nagłówku:

```
void klawiatura(unsigned char kod_klawisza, int x, int y)
```

Parametr `key` powoduje automatyczne przekazanie do wnętrza funkcji kodu ostatnio wciśniętego klawisza ze znakiem ASCII, natomiast parametry `x,y` śledzą położenie kursora myszy w oknie w czasie operacji wciskania klawisza. W przeciętnym przypadku nie ma potrzeby jednoczesnego śledzenia myszy i klawiatury, więc typowa postać funkcji `klawiatura()` jest następująca:

```
void klawiatura(unsigned char kod_klawisza, int x, int y)
{
  switch (kod_klawisza) {

  case 'a':
           //COS ZROB
           glutPostRedisplay();
           break;

  case 'b':
           //ZROB COS INNEGO
           glutPostRedisplay();
           break;

  case '\033':
            //ZROB COS JESZCZ INNEGO
            glutPostRedisplay();
           break;
  }
}
```

Przykład użycia takiej funkcji znajduje się w pliku `ogl52.cpp`.

(b) Funkcja `glutSpecialFunc(klawiatura)` odpowiada za obsługę klawiszy specjalnych klawiatury. Ich kody są reprezentowane przez następujące stałe:

```
GLUT_KEY_F1
GLUT_KEY_F2
GLUT_KEY_F3
GLUT_KEY_F4
GLUT_KEY_F5
GLUT_KEY_F6
```

```
GLUT_KEY_F7
GLUT_KEY_F8
GLUT_KEY_F9
GLUT_KEY_F10
GLUT_KEY_F11
GLUT_KEY_F12
GLUT_KEY_LEFT
GLUT_KEY_UP
GLUT_KEY_RIGHT
GLUT_KEY_DOWN
GLUT_KEY_PAGE_UP
GLUT_KEY_PAGE_DOWN
GLUT_KEY_HOME
GLUT_KEY_END
GLUT_KEY_INSERT
```

Parametr `klawiatura` musi być nazwą funkcji o następującym nagłówku:

```
void klawiatura(int kod_klawisza, int x, int y)
```

Znaczenie parametrów funkcji `klawiatura` jest identyczne jak w przypadku funkcji `glutKeyboardFunc()`. W jej kodzie różnica wystąpi tylko przy opisie kodów klawiszy: zamiast

```
case 'znak':
```

wystąpi tutaj

```
case kod_klawisza :
```

gdzie `kod_klawisza` jest jedną z wyżej wymienionych stałych. Przykład znajduje się w pliku `ogl53.cpp`.

3. *Obsługa myszy w bibliotece* `GLUT`.

   Za obsługę myszy odpowiedzialne są funkcje których nazwy są parametrem jednej z trzech funkcji biblioteki `GLUT`, a mianowicie: `glutMouseFunc()`, `glutMotionFunc()` lub `glutPassiveMotionFunc()`. Rozpatrzmy je po kolei, zakładając dla ustalenia uwagi, że zdarzenia związane z myszą opisane są w funcji `mysz()`.

   (a) Funkcja `glutMouseFunc(mysz)` odpowiada za obsługę zdarzeń wciśnięcia i zwolnienia danego przycisku myszy. Parametr `mysz` musi być funkcją o następującym nagłówku:

   ```
   void mysz(int przycisk, int stan, int x, int y),
   ```

gdzie `przycisk` przyjmuje wartość jednej ze stałych: `GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON` (lewy, środkowy, prawy), `stan` oznacza czy przycisk został wciśnięty - stała `GLUT_DOWN`, czy też zwolniony - stała `GLUT_UP`. Parametry `x,y` przekazują automatycznie aktualne położenie kursora myszy w oknie. Typowa postać funkcji `mysz()` nie oprogramowuje parametrów `x,y` i jest następująca:

```
void mysz(int przycisk, int stan, int x, int y)
{
  if (stan == GLUT_DOWN) {
    switch (przycisk) {
      case GLUT_LEFT_BUTTON:
                            //COS ZROB
                            glutPostRedisplay();
                            break;

      case GLUT_MIDDLE_BUTTON:
                            //ZROB COS INNEGO
                            glutPostRedisplay();
                            break;
      case GLUT_RIGHT_BUTTON:
                            //ZROB COS JESZCZE INNEGO
                            glutPostRedisplay();
                            break;
    }
  }
  else if (stan == GLUT_UP){
    switch (przycisk) {
      case GLUT_LEFT_BUTTON:
                            //COS ZROB_1
                            glutPostRedisplay();
                            break;

      case GLUT_MIDDLE_BUTTON:
                            //ZROB COS INNEGO_1
                            glutPostRedisplay();
                            break;
      case GLUT_RIGHT_BUTTON:
                            //ZROB COS JESZCZE INNEGO_1
                            glutPostRedisplay();
                            break;
    }
  }
}
```

Przykład znajduje się w pliku `ogl54.cpp`.

(b) Funkcja `glutMotionFunc(mysz)` odpowiada za obsługę zdarzeń ruchu kursora myszy przy wciśniętym danym przycisku myszy. Parametr `mysz` musi być funkcją o następującym nagłówku:

```
void mysz(int x, int y),
```

gdzie parametry `x,y` przekazują automatycznie aktualne położenie kursora myszy w oknie.

(c) Funkcja `glutPassiveMotionFunc(mysz)` odpowiada za obsługę zdarzeń ruchu kursora myszy bez wciśnięcia jakiegokolwiek przycisku myszy. Nagłówek funkcji `mysz()` jest taki sam jak w przypadku funkcji `glutMotionFunc()`. Przykład jednoczesnego użycia wszystkich trzech funkcji oprogramowujących mysz znajduje się w pliku `ogl55.cpp`.

4. Wyłącznie danego rodzaju obsługi klawiatury lub myszy zachodzi przy przy wywołaniu którejkolwiek z pięciu omówionych powyzej funkcji z parametrem `NULL`, np. `glutKeyboardFunc(NULL)`.

5. *Zadanie*

(a) Zdefiniować funkcje `czworościan()` oraz `prostopadłościan()` konstruującą te bryły przy pomocy tablic z przeplotem zawierających dane o geometrii wierzchołków i ich kolorach. Przetestować wszystkie warianty odwoływania się do tablic. Obie bryły mają jeden wierzchołek w środku układu i trzy krawędzie rozpięte wzdłuż dodatnich osi układu współrzędnych.

(b) Zdefiniować funkcje rysujące wielokątowe przybliżenie powierzchni *walca, stożka i kuli*. Dokładność takiego przybliżenia powinna być określona w parametrach funkcji (np. w przypadku walca może to być liczba całkowita określająca ilość prostokątów przybliżających powierzchnię boczną walca).

(c) Oprogramować przy pomocy klawiatury lub myszy ruch obserwatora w dowolnej wykonanej scenie.