Studium podyplomowe

- Programowanie grafiki z użyciem OpenGL

Charakterystyka OpenGL

- OpenGL - (**Open G**raphics **L**ibrary),
- Graficzna biblioteka 2D/3D,
- Liczne warianty biblioteki, także akcelerowane sprzętowo,
- Format OpenGL jest otwarty - istnieje możliwość definiowania dowolnych rozszerzeń, również akcelerowanych przez producentów sprzętu,
- Dodatki funkcjonalne, umożliwiające tworzenie interfejsów użytkownika i dostarczające narzędzi:
  - GLUT
  - GLU
  - GLAUX

Podstawy C

- Blok komentarza:
  /* *tu jest komentarz*
  */
- Komentarz w jednej linii:
  po znakach // *tu jest komentarz*
- Bloki kodu wykonywanego sekwencyjnie (linia po linii):
  { *tu jest kod*
  }
- Kod to ciąg instrukcji. Każda instrukcja zakończona jest znakiem średnika: „;”
- W jednej linii może zostać zapisanych wiele instrukcji - oddzielanych średnikami.

C - deklarowanie zmiennych (I)

- Podstawowe typy zmiennych i ich deklaracje:
  - int wielkosc = 7;
    - typ całkowity o nazwie *wielkosc* i wartości początkowej 7
  - float odleglosc = 30.5;
    - typ rzeczywisty o nazwie *odleglosc* i wartości początkowej 30.5
  - double wPrawo = 1;
    - typ rzeczywisty podwójnej precyzji o nazwie wPrawo i wartości początkowej 1.

C - deklarowanie zmiennych (II)

- Podstawowe typy i ich deklaracje:
  - char tekst[40] = "To jest testowy tekst...";
    - typ łańcuchowy (tablica znaków) o nazwie tekst i wartości początkowej „*To jest testowy tekst...*" i maksymalnej długości 40-1
  - char znak = 'X';
    - typ znakowy (pojedynczy znak) o nazwie *znak* i wartości początkowej 'X'
  - bool dodatkowaKula = false;
    - typ boolean ("prawda lub fałsz") o nazwie *dodatkowaKula* i wartości false (fałsz)

Prezentacja

- Szablon - Deklarowanie zmiennych

C - wywoływanie funkcji (I)

- Funkcję wywołujemy w celu zlecenia wykonania różnych czynności w niej zapisanych,
- Każda funkcja posiada swoją nazwę, przez którą jest identyfikowana,
- Funkcja może także (opcjonalnie) otrzymywać zestaw parametrów (poprzez wartości których można wpłynąć na przebieg jej wykonania). Wartości parametrów są podawane w momencie wywołania funkcji,
- Funkcja może (opcjonalnie) zwracać wartość.

C - wywoływanie funkcji (II)

- Aby wywołać funkcję należy zapisać w linii kodu jej nazwę, oraz w występujących po tej nazwie nawiasach „(”, „)” należy podać wartości parametrów. Gdy parametrów nie ma, nawiasy pozostają puste „()”.
- Przykład:

```
funkcja1 ();
funkcja2 (10,15);              // tu dwa parametry
        // liczbowe o wartościach kolejno 10 i 15
```

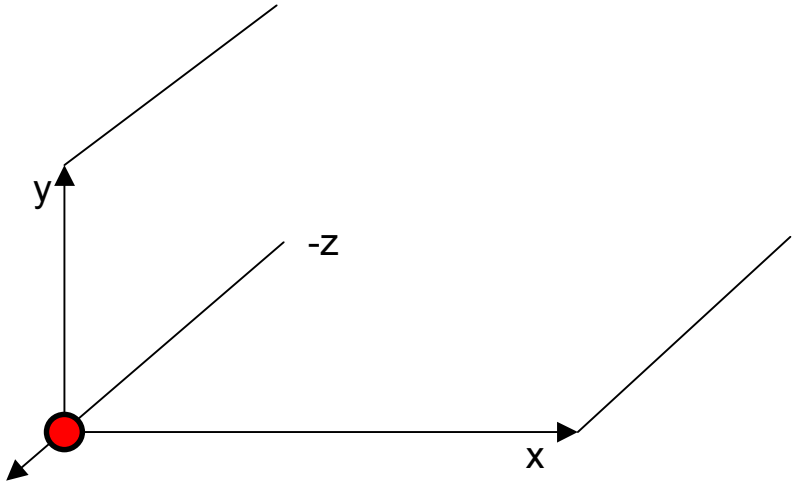
- Funkcje można tworzyć samodzielnie lub korzystać z funkcji gotowych (bibliotecznych). OpenGL jest przykładem biblioteki (zbioru) funkcji, które można wywoływać.

Funkcje w OpenGL (II)

- Konwencja nazewnictwa funkcji w OpenGL:
  - Przedrostki funkcji: gl, glu, glut - zależnie od pakietu, do którego funkcja przynależy
  - Przyrostki funkcji:
    - Pierwszy: 2,3,4 - zależnie od liczby argumentów, jaki dana wersja funkcji pobiera,
    - Drugi: d, f, i, b, s - zależnie od typu argumentów, jaki dana wersja funkcji pobiera (GLdouble, GLfloat, GLint, GLbyte, GLshort itp)
    - Trzeci (opcjonalny): v - gdy wersja funkcji pobiera wskaźnik na wektor zamiast luźnych parametrów
  - Przykład funkcji (definiuje kolor): glColor4f, glColor3d, glColor3iv

Podstawy matematyczne OpenGL (I)

- Układ współrzędnych:



  - x - „w prawo”
  - y - „w górę”
  - z - „w głąb”, ale w kierunku do kamery

Podstawy matematyczne OpenGL (II)

- Przekształcenie obiektu w przestrzeni 3D to mnożenie jego współrzędnych przez tzw. **macierz przekształcenia**:

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1]\begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix}$$

$$x' = (M_{11} \times x) + (M_{21} \times y) + (M_{31} \times z) + (M_{41} \times 1)$$
$$y' = (M_{12} \times x) + (M_{22} \times y) + (M_{32} \times z) + (M_{42} \times 1)$$
$$z' = (M_{13} \times x) + (M_{23} \times y) + (M_{33} \times z) + (M_{43} \times 1)$$

- Modyfikacje macierzy „nakładają się na siebie”, tworząc efekt końcowy będący wynikiem wielu przekształceń.

Przekształcenia 3D

- Funkcje przekształcające:
  - Rotacja:
    - glRotatef (GLfloat kąt, GLfloat x, GLfloat y, GLfloat z);
    - glRotated (GLdouble kąt, GLdouble x, GLdouble y, GLdouble z)
  - Przesunięcie:
    - glTranslatef (GLfloat x, GLfloat y, GLfloat z);
    - glTranslated (GLdouble x, GLdouble y, GLdouble z)
  - Skalowanie:
    - glScalef (GLfloat x, GLfloat y, GLfloat z);
    - glScaled (GLdouble x, GLdouble y, GLdouble z)

Zachowanie i odtworzenie pozycji

- Stosy macierzy - umożliwiają zapamiętanie pozycji i późniejsze jej przywołanie:
  - glPushMatrix - kładzie bieżącą macierz na stosie
  - glPopMatrix - pobiera macierz ze stosu
- „Kasowanie” macierzy do postaci wyjściowej (nic nie przekształcającej):
  - glLoadIdentity();

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Prezentacja

- Szablon - Wywoływanie funkcji

Operacje arytmetyczne (I)

- Operatory arytmetyczne C można pogrupować w zależności od ilości argumentów pobieranych. Większość to operatory dwuargumentowe, potem jednoargumentowe.
- Podstawowe operatory C:

CZYNNOŚĆ	SYMBOL	PRZYKŁAD
Dodawanie:	+	a = c + b
Zwiększenie o wartość podaną:	+=	a += b
Odejmowanie:	-	a = c - b
Zmniejszenie o wartość podaną:	-=	a -= b

Operacje arytmetyczne (II)

- Podstawowe operatory dwuargumentowe C:
  - Mnożenie: `*` `a = c * b`
  - Mnożenie przez wartość podaną: `*=` `a *= b`
  - Dzielenie: `/` `a = c / b`
  - Dzielenie przez wartość podaną: `/=` `a /= b`
  - Reszta z dzielenia (modulo): `%` `a = c % b`
  - Reszta z dziel. przez w podaną: `%=` `a %= b`

Prezentacja

- Szablon - Operacje arytmetyczne

Operacje arytmetyczne (III)

- Specyficzne między innymi dla języka C są operatory jednoargumentowe zwiększenia o 1 (inkrementacji) i zmniejszenia o 1 (dekrementacji). Mogą one występować zarówno przed jak i po argumencie, dając skutek w postaci korekty wartości przed użyciem w innym wyrażeniu, lub już po nim:

CZYNNOŚĆ	SYMBOL	PRZYKŁAD
Zwiększenie o 1 „po”:	++	a++
Zmniejszenie o 1 „po”:	--	b--
Zwiększenie o 1 „przed” :	++	++a
Zmniejszenie o 1 „przed” :	--	--b

Prezentacja

- Szablon - Operacje arytmetyczne 2

Rzutowanie typów

- Przed wykorzystaniem danej zmiennej w programie możliwa jest korekta typu danych przez nią przechowywanych. Dzięki temu możliwe jest np. podanie do funkcji argumentu typu float za zmiennej, mimo iż dysponujemy jedynie zamienną typu int.
- Część konwersji wykonywana jest automatycznie (niejawnie). Do wykonania wymuszonej (jawnej) konwersji stosuje się konstrukcję:
  (*typ*)starytyp , gdzie *typ* to żądany typ wynikowy.
- Przykład (konwersja zmiennej *zm* typu *float* na *int*):
  a = (int) zm;

Prezentacja

- Szablon - Rzutowanie typów danych

Wyrażenia logiczne i instrukcja „if”

- Instrukcja „if” umożliwia wykonanie fragmentu kodu warunkowo - w zależności od wartości logicznej wyrażenia, będącego warunkiem.
- Składnia instrukcji warunkowej „if” - podstawowe warianty:

```
if (warunek) instrukcja_tak;

if (warunek) instrukcja_tak; else instrukcja_nie;

if (warunek) {
        instrukcja1; instrukcja2; instrukcja3; //...
}
```

Operatory porównania

- Operacje porównania:

OPERACJA	SYMBOL	PZYKŁAD
Czy mniejsze	<	a < b
Czy większe	>	a > b
Czy równe	==	a == b
Czy nierówne	!=	a != b
Czy mniejsze lub równe	<=	a <= b
Czy większe lub równe	>=	a >= b

Prezentacja

- Szablon - Wyrażenia logiczne i instrukcja „if”

Złożone wyrażenia logiczne

- Wyrażenia złożone konstruowane są w wyników różnych operacji porównania
- Operatory porównania:

OPERATOR	SYMBOL	PRZYKŁAD
Koniunkcja („i”)	&&	a > 1 && a < 5
Alternatywa („lub”)	||	a < 1 || a > 5
Zaprzeczenie („nie”)	!	! a

Prezentacja

- Szablon - Złożone wyrażenia logiczne

Iteracje - pętla „for”

- Zastosowanie pętli „for” jest najprostszym sposobem na wykonanie pewnych ustalonych operacji wielokrotnie - kontrolując liczbę wykonań, także pomiędzy poszczególnymi wykonaniami.
- Przykład:

```
for (iterator = 0; iterator <10; iterator ++){
      instrukcje;
}
```

oznacza wykonywanie instrukcji dla wartości *iterator* od 0 do 9 (mniejszej od 10) i zwiększanej o 1 za każdym wykonaniem.

Prezentacja

- Szablon - Pętla „for”

Pętla „while” i „do..while”

- Alternatywą dla pętli „for” jest „while”. Ten wariant nie zapewnia automatycznego korygowania iteratora i inicjalizowania go początkową wartością. Trzeba to zrobić we własnym zakresie.
- W przypadku „do..while” instrukcje zostaną na pewno wykonane przynajmniej raz
- Przykłady:

```
while (a < 10 ) { instrukcje;}
do { instrukcje;} while (a < 10 );
```

przy czym *instrukcje* muszą wpływać na wartość a.

Prezentacja

- Szablon - Pętla „while”