Studium podyplomowe

- Programowanie w OpenGL

Charakterystyka (I)

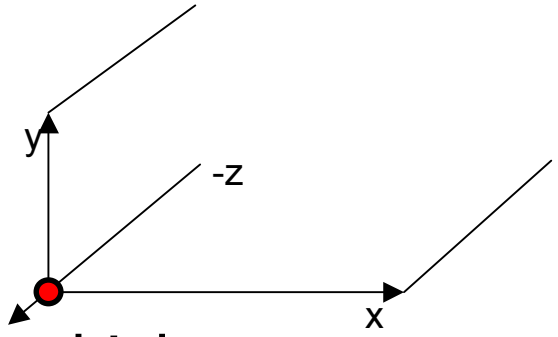
- OpenGL - (**Open G**raphics **L**ibrary)
- Graficzna biblioteka 2D/3D
- Liczne porty biblioteki, w tym także akcelerowane sprzętowo
- Format otwarty - możliwość definiowania dowolnych rozszerzeń, również akcelerowanych przez producentów sprzętu
- Dodatki funkcjonalne, umożliwiające tworzenie interfejsów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

Charakterystyka (II)

- Konwencja nazewnictwa:
  - 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
  - Przedrostki stałych przedefiniowanych: GL_
  - Przedrostki obiektów OpenGL: GL, GLU, GLUT itp..

Podstawy matematyczne - układ współrzędnych

- x - „w prawo”
- y - „w górę”
- z - „w głąb”



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

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

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

Transformacje Macierzy (II)

- Funkcje przekształcające bieżącą macierz:
  - 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)

Transformacje Macierzy (I)

- Translacja - pomnożenie aktywnej macierzy przekształcenia przez:

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Skalowanie - pomnożenie aktywnej macierzy przekształcenia przez:

$$\begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotacja względem osi: x y z

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) & 0 \\ 0 & \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformacje Macierzy (II)

- Operacja mnożenia dwóch macierzy :

$$c_{ij} = \sum_{k=1}^{m} a_{ik} \cdot b_{kj}$$

- Transformacje 3D w OpenGL polegają na mnożeniu odpowiednich macierzy o wymiarach 4x4.
- Załadowanie macierzy z tablicy:
  - glLoadMatrixf (const GLfloat *tablica);
- Mnożenie bieżącej macierzy przez tablicę liczb 4x4:
  - glMultMatrixf (const GLfloat *tablica);

Transformacje Macierzy (III)

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

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

Transformacje Macierzy (IV)

- W OpenGL istnieją trzy niezależne macierze przekształcenia, trzy stosy macierzy, mające zastosowanie do:
  - Projekcji (rzutowania) ekranu : GL_PROJECTION
  - Modelu w widoku: GL_MODELVIEW
  - Teksturowania: GL_TEXTURE
- Jedna z tych macierzy jest atywna. Wszystkie operacje 3D są wówczas wykonywane na niej i na jej stosie.
- Zmiana macierzy aktywnej:
  - glMatrixMode(GLenum macierz); - gdzie macierz to GL_PROJECTION, GL_MODELVIEW lub GL_TEXTURE
  - Pobranie informacji o aktywnej macierzy: glGet(GL_MATRIX_MODE);

Prezentacja

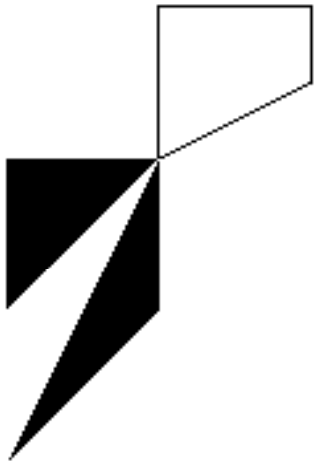
- Szablon - transformacje macierzy

Rysowanie figur (I)

- Funkcje glBegin(GLenum tryb) i glEnd(void)
- Tworzą blok rysujący, w który umieszczamy wywołania (między innymi):
  - funkcji znakujących wierzchołki (glVertex*()
  - funkcji ustalających kolor (glColor*())
  - funkcje wywołujące listę operacji(glCallList())
  - funkcje modyfikujące krawędzie rysowania (glEdgeFlag*())
  - funkcje ustalające koordynaty tekstur figury (glTexCoord*())
  - funkcje definiujące materiał przypisany figurze (glMaterial*())

Rysowanie figur (II)

- Przykład:
  - glBegin(GL_LINE_LOOP);
  - glVertex2f (0,0);
  - glVertex2f (0,1);
  - glVertex2f (1,1);
  - glVertex2f (1,0.5);
  - glVertex2f (0,0);
  - glEnd();
  - glBegin(GL_TRIANGLES);
  - glVertex2f (0,0);
  - glVertex2f (0,-1);
  - glVertex2f (-1,-2);
  - glVertex2f (0,0);
  - glVertex2f (-1,0);
  - glVertex2f (-1,-1);
  - glEnd();



Rysowanie figur (III)

- Parametr glBegin(GLenum tryb) ustala globalne dla bloku znaczenie znajdujących się w nim wywołań def. punkty:
  - GL_POINTS - rysowanie pojedynczych punktów,
  - GL_LINES - odcinki (para punktów to jeden odcinek),
  - GL_LINE_STRIP - linia łamana otwarta,
  - GL_LINE_LOOP - linia łamana zamknięta,
  - GL_TRIANGLES - trójkąty (trzy punkty to jeden trójkąt),
  - GL_TRIANGLE_STRIP - trójkąty w połączeniu tworzące „pasek”,
  - GL_TRIANGLE_FAN - trójkąty w połączeniu tworzące „wachlarz”,
  - GL_QUADS - czworokąty (cztery punkty to jeden czworokąt),
  - GL_QUAD_STRIP - czworokąty w połączeniu tworzące „pasek”,
  - GL_POLYGON - wielokąt

Rysowanie figur (IV)

- Funkcje pomocnicze:
  - glPointSize(GLfloat size) - wielkość punktu
  - glLineWidth(GLfloat size) - szerokość linii
  - glLineStripple(GLint zwielokrotnienie, Glushort wzor) - rysowanie linii rastrem bitowym (linia przerywana według wzoru). Aby funkcja działała, należy włączyć rysowanie linii rastrem : glEnable(GL_LINE_STRIPPLE)
  - glShadeModel() - sposób cieniowania (GL_FLAT - płaskie, jednokolorowe, GL_SMOOTH - wygładzone gradientem)
  - glPolygonMode (GLenum strona, tryb) - sposób wypełeniania wielokątów. Strona - z której strony (GL_FRONT_AND_BACK,GL_BACK, GL_FRONT), tryb - jak (GL_LINE, GL_POINT, GL_FILL)

Prezentacja

- Szablon - obiekty 2D

Macierz rzutowania

- W OpenGL występuje kilka macierzy transformujących, wśród nich - macierz rzutowania ekranu, stosowania przy ustawianiu sceny.
- Wybór macierzy (GL_PROJECTION to macierz rzutowania ekranu): glMatrixMode(GL_PROJECTION);
- Po wyborze możliwe jest zastosowania funkcji przetwarzających macierz (wśród nich np. glPerspective, glOrtho, glFrustum) i korygujących tym samym widok do np. perspektywy lub widoku równoległego.

Funkcje korygujące macierz rzutowania (I)

- gluPerspective - kolejne parametry to:
  - kąt rozwarcia widoku
  - proporcja wysokości do szerokości widoku
  - odległość przedniej płaszczyzny odcięcia
  - odległość tylnej płaszczyzny odcięcia
- glOrtho - kolejne parametry to:
  - cztery pierwsze: współrzędne ekranu 2d, który będzie widokiem
  - odległość przedniej płaszczyzny odcięcia
  - odległość tylnej płaszczyzny odcięcia

Funkcje korygujące macierz rzutowania (II)

- gluOrtho2D - kolejne parametry to:
  - cztery pierwsze: współrzędne ekranu 2d, który będzie widokiem
- glFrustum - kąt rozwarcia widoku
  - cztery pierwsze: współrzędne ekranu 2d, który będzie widokiem
  - odległość przedniej płaszczyzny odcięcia i zarazem ekranu 2d - tym samym „powiększenie” widoku
  - odległość tylnej płaszczyzny odcięcia

Prezentacja

- Szablon - ustawianie widoku

Biblioteka funkcji pomocniczych - AUX

- Nagłówek (C++): glaux.h
- Obsługa interfejsu użytkownika (rejestrowanie funkcji obsługi zdarzeń), obsługa okien i wyświetlania
- Wybrane funkcje narzędziowe:
  - auxInitWindow(LPCTSTR);
  - auxCloseWindow(void);
  - auxMainLoop(AUXMAINPROC);
  - auxExposeFunc(AUXEXPOSEPROC);
  - auxReshapeFunc(AUXRESHAPEPROC);
  - auxIdleFunc(AUXIDLEPROC);
  - auxKeyFunc(int, AUXKEYPROC);
  - auxMouseFunc(int, int, AUXMOUSEPROC);

Biblioteka funkcji pomocniczych - AUX

- Funkcje rysujące:
- auxWireSphere(GLdouble); auxSolidSphere(GLdouble);
- auxWireCube(GLdouble); auxSolidCube(GLdouble);
- auxWireBox(GLdouble, GLdouble, GLdouble); auxSolidBox(GLdouble, GLdouble, GLdouble);
- auxWireTorus(GLdouble, GLdouble); auxSolidTorus(GLdouble, GLdouble);
- auxWireCylinder(GLdouble, GLdouble); auxSolidCylinder(GLdouble, GLdouble);
- auxWireIcosahedron(GLdouble); auxSolidIcosahedron(GLdouble);
- auxWireOctahedron(GLdouble); auxSolidOctahedron(GLdouble);
- auxWireTetrahedron(GLdouble); auxSolidTetrahedron(GLdouble);
- auxWireDodecahedron(GLdouble); auxSolidDodecahedron(GLdouble);
- auxWireCone(GLdouble, GLdouble); auxSolidCone(GLdouble, GLdouble);
- auxWireTeapot(GLdouble); auxSolidTeapot(GLdouble);

Prezentacja

- Szablon - AUX

Obiekty 3D biblioteki GLU

- Nagłówek (C++): glu.h
- Alternatywa dla AUX w generowaniu siatek 3D
- Obiekty GLUquadricObj umożliwiają wygenerowanie siatki 3D według zadanych w obiekcie parametrów
- Procedura:
  - GLUquadricObj *obiekt;
  - obiekt = gluNewQuadric(); // tworzenie
  - gluQuadricDrawStyle (obiekt, GLU_LINE); // konfiguracja
  - gluSphere(obiekt, 2,10,10); // rysowanie

Prezentacja

- Szablon - obiekty 3D

Światła (I)

- OpenGL wykonuje obliczenia dla ośmiu niezależnych światłami, oznaczonymi stałymi: GL_LIGHT0 .. GL_LIGHT8. Stałe te podawane są do funkcji sterującymi światłami.
- System świateł wymaga włączenia funkcją:
  - glEnable(GL_LIGHTING);
- Do prawidłowej projekcji świateł należy także włączyć użytkowanie bufora głębokości:
  - glEnable(GL_DEPTH_TEST);

Światła (II)

- Dla każdego ze świateł można definiować właściwości, w podstawowej wersji adresowane stałymi: GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION:
- Jedna z funkcji modyfikujących światła: glLightfv(GLenum id_swiatla, GLenum id_wlasciwosci, GLfloat wektor_danych);
- Przykład:
  - GLfloat swiatloSpecular1[4] = {0.5,1,1,1};
  - glLightfv(GL_LIGHT0,GL_SPECULAR, swiatloSpecular1);

Prezentacja

- Szablon - światła rozproszone

Światła - ciąg dalszy (I)

- Ustawianie pozycji światła następuje poprzez zmianę jej właściwości GL_POSITION np.:
  - GLfloat swiatloPoz[4] = {0.5,1,1,1};
  - glLightfv(GL_LIGHT1,GL_POSITION,swiatloPoz);
- Możliwość modyfikowania modelu oświetlenia sceny, poprzez wywarcie wpływu na sposób obliczania świateł,zdefiniowanie oświetlenia globalnego itp.
- Przykład (oświetlenie globalne)
  - GLfloat ambient[4] = {0.3,0.3,0.3,1};
  - glLightModelfv(GL_LIGHT_MODEL_AMBIENT,ambient);

Światła - ciąg dalszy (II)

- Światła o ograniczonym kącie emisji - są definiowane poprzez zmianę parametrów:
  - GL_SPOT_CUTOFF (kąt rozwarcia)
  - GL_SPOT_EXPONENT (wytłumienie na granicy kąta)
  - GL_SPOT_DIRECTION (wektor kierunku świecenia)
- Przykład:
  - GLfloat swiatloKierunek[4] = {0.5,1,1,1};
  - glLightf(GL_LIGHT1,GL_SPOT_CUTOFF,6.0);
  - glLightf(GL_LIGHT1,GL_SPOT_EXPONENT,50.0);
  - glLightfv(GL_LIGHT1,GL_SPOT_DIRECTION, swiatloKierunek);

Prezentacja

- Szablon - światła spot

Materiały (I)

- Atrybuty materiału obowiązują w momencie rysowania obiektów. Można je zmieniać przed każdym rysowaniem
- Funkcja: glMaterial[if][v](GLenum strona, GLenum atrybut, TYPE wartosc);
- Parametr strona to powierzchnia renderowania materiału: GL_FRONT, GL_BACK, lub GL_FRONT_AND_BACK
- Parametr atrybut to rodzaj ustawianej właściwości. Ważniejsze z nich: GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_SHININESS

Materiały (II)

- GL_DIFFUSE:
  RGB(1,1,1)

RGB(0,1,0)

- GL_AMBIENT:
  RGB(1,1,1)

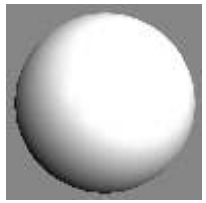
RGB(0,1,0)

Materiały (III)

- GL_SPECULAR:
  RGB(1,1,1)

RGB(0,1,0)

- GL_SHININESS:
  0.05



0.2

1

Prezentacja

- Szablon - materiały, faktura

Materiały (IV)

- Korygowanie materiałów aktualnym kolorem.
- Funkcja:
  glColorMaterial(GLenum strona,
  GLenum wlasnosc)
- Umożliwia wybór trybu koloryzacji materiałów, definiuje to, które właściwości materiału mają być korygowane przez wybór koloru
- Parametr strona - jak przy glMaterial()
- Parametr własność:
  GL_EMISSION, GL_AMBIENT, GL_DIFFUSE,
  GL_SPECULAR, GL_AMBIENT_AND_DIFFUSE

Materiały (IV)

- Wybór trybu koloryzacji materiałów:
  - glEnable (GL_COLOR_MATERIAL)
  - glDisable (GL_COLOR_MATERIAL)
- Przy włączonym trybie wybrane właściwości materiałów można korygować poprzez glColor()
- „Wygładzanie” materiałów:
- Funkcja glShadeModel(GLenum tryb)
- Parametr tryb: GL_SMOOTH, GL_FLAT

Prezentacja

- Szablon - materiały, kolor

Efekty mgły (I)

- Efekt mgły należy włączyć poprzez wywołanie: glEnable(GL_FOG)
- Definiowana na scenie mgła posiada szereg parametrów, definiowanych funkcją glFog[i,f,d][v] w postaci:

  glFogf (GLenum ustawienie, GLfloat wartosc)

  Zależnie od wartości ustawienie drugi parametr może być także wektorem. Możliwe wartości parametru ustawienie: GL_FOG_MODE, GL_FOG_COLOR, GL_FOG_DENSITY, GL_FOG_START, GL_FOG_END

Efekty mgły (II)

- Typ mgły:
  glFogi(GL_FOG_MODE,Glenum typ); gdzie typ to GL_EXP, GL_EXP2, lub GL_LINEAR
- Kolor mgły:
  glFogfv(GL_FOG_COLOR, kolor);
- Gęstość mgły:
  glFogf(GL_FOG_DENSITY, 0.1f);
- Granice mgły (licząc od położenia kamery):
  glFogf(GL_FOG_START, 5.0f);
  glFogf(GL_FOG_END, 400.0f);

Prezentacja

- Szablon - mgła

Tekstury (I)

- Problem ładowania tekstur leży poza OpenGL - konieczny jest moduł ładujący obraz z pliku o danym formacie
- Tekstury są rejestrowane w OpenGL i adresowane przy użyciu uchwytów typu int. Takie podejście pozwala na szybkie zmiany tekstur aktywnych oraz jednokrotne ich ładowanie - od razu do pamięci akceleratora 3D
- Umożliwienie wyboru tekstury (przy ładowaniu i użyciu): glBindTexture(GLenum cel, GLint identyfikator); gdzie cel to GL_TEXTURE_2D lub GL_TEXTURE_1D

Tekstury (II)

- Włączenie trybu teksturowania: glEnable(GLenum cel); (cel to GL_TEXTURE_2D lub GL_TEXTURE_1D)
- Teksturowanie następuje poprzez wytypowanie wierzchołków i koordynat tekstur dla każdego z nich:
  - glBegin(GL_TRIANGLES);
  - glTexCoord2f(0,0);
  - glVertex2f (0,0);
  - glTexCoord2f(0,1);
  - glVertex2f (0,-1);
  - glTexCoord2f(1,1);
  - glVertex2f (-1,-2);
  - glEnd();

Tekstury (III)

- Możliwość modyfikowania macierzy teksturowania - przy użyciu funkcji glRotate, glTranslate itp..:
- Przykład:
  - glMatrixMode(GL_TEXTURE);
  - glScalef(1.3,1.3,1.3);
  - glRotatef(30,0,0,1);
  - glMatrixMode(GL_MODELVIEW);
  - // ciąg dalszy operacji teksturowania

Prezentacja

- Szablon - teksturowanie

Automatyczne teksturowanie (I)

- OpenGL posiada możliwość automatycznego generowania koordynat tekstur dla dowolnej siatki - w przypadku braku koordynat w modelu.
- Technika przydatna tylko w przypadku pokrywania siatki jednolitą teksturą
- Włączenie generowania koordynat tekstur:

```
glEnable(GL_TEXTURE_GEN_S); // dla współrzędnej poziomej tekstury
glEnable(GL_TEXTURE_GEN_T); // dla współrzędnej poziomej tekstury
```

Automatyczne teksturowanie (II)

- Wybór techniki automatycznego generowania koordynat tekstur:
  glTexGeni (GLenum współrzędna
  ,GL_TEXTURE_GEN_MODE, GLenum tryb)
  gdzie dla tekstury 2D:
  - współrzędna to GL_S lub GL_T
  - tryb to GL_OBJECT_LINEAR, GL_EYE_LINEAR, lub GL_SPHERE_MAP

Prezentacja

- Szablon - modele 3DS z automatycznym teksturowaniem