Algorytmy geometryczne

Maciej Hojda

Uwaga: Słowa „dany”, „zadany”, „podany”, „wybrany” itd. w kontekście parametrów (zmiennych) oznacza parametr zadany przez użytkownika (a nie na stałe, przez programistę), a implementacja wykorzystująca taki parametr powinna obsługiwać jego różne wartości.

1 Zadanie nr 0 – pyopengl

Uwaga: Można korzystać z innej biblioteki graficznej. W takim wypadku zadanie nr 0 należy pominąć.

Zainstaluj biblioteki OpenGL w oprogramowaniu PyCharm, na Linuksie (np. na uprzednio przygotowanej maszynie wirtualnej).

Uwaga: znaków ”>>” nie należy wpisywać

1. Zainstaluj pakiety (w konsoli Linuksa)

```
>> sudo apt install python-opengl python3-dev freeglut3-dev
```

2. Zainstaluj pakiety (w konsoli PyCharm-a)

```
>> import pip
>> pip.main(['install', 'numpy']);
>> pip.main(['install', 'pyopengl']);
```

3. Uruchom i zrozum program (w razie potrzeby, zapoznaj się z dokumentacją `http://pyopengl.sourceforge.net/documentation/index.html`)

– treść programu –

```
from OpenGL.GL import *;
from OpenGL.GLU import *;
from OpenGL.GLUT import *;

def rysuj():
    # czyszczenie ekranu
    glClear(GL_COLOR_BUFFER_BIT);

    # rysowanie punktów
    glPointSize(50.0);
    glBegin(GL_POINTS);
    glColor3f(1.0, 0.0, 0.0);
    glVertex2f(0.0, 0.0);
    glColor3f(0.0, 1.0, 0.0);
    glVertex2f(50.0, 50.0);
    glColor3f(0.0, 0.0, 1.0);
    glVertex2f(50.0, -50.0);
```

```
    glEnd();

    # rysowanie prostych
    glLineWidth(10);
    glColor3f(0.0, 0.0, 0.0);
    glBegin(GL_LINES)
    glVertex2f(0.0, 0.0);
    glVertex2f(50.0, 50.0);
    glVertex2f(50.0, 50.0);
    glVertex2f(50.0, -50.0);
    glVertex2f(50.0, -50.0);
    glVertex2f(0.0, 0.0);
    glEnd();

    # wyświetlanie
    glFlush();

# inicjalizacja OpenGL i okna
glutInit();
glutInitWindowSize(600, 400);
glutInitWindowPosition(0, 0);
glutCreateWindow(b"Zadanie 1");
# parametry OpenGL -- single-buffer i rgb
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
# funkcje rysujące
glutDisplayFunc(rysuj);
glutIdleFunc(rysuj);
# kolor tła
glClearColor(1.0, 1.0, 1.0, 1.0);
# projekcja ortograficzna
gluOrtho2D(-100.0, 100.0, -100.0, 100.0);
# główna pętla (zapętla funkcję "rysuj")
glutMainLoop();
```

– koniec treści programu –

2 Zadanie nr 1 – odcinki

Przygotuj aplikację, która pobiera od użytkownika listę wektorów (zdefiniowanych przez punkt początkowy i końcowy) w $\mathbb{R}^2$.

- Wyświetl pobrane wektory. Graficznie odróżnij początek od końca.
- Znajdź i wyświetl współrzędne punktów przecinania się par wektorów. Punkty przecięcia też zaznacz graficznie.
- Dla zadanego punktu (wyświetl go) wskaż wszystkie wektory, dla których jest on z ich prawej strony (prawej, patrząc od początku wektora, na koniec wektora).

3 Zadanie nr 2 – figury

- Wyświetl trójkąt o zadanych wierzchołkach.

- Dokonaj obrotu tego trójkąta o podany kąt α, względem podanego punktu (niekoniecznie środka układu współrzędnych). Wykonaj animację realizującą obrót w pętli (co iterację obrót o kolejne α). Wykorzystaj macierz rotacji dookoła środka układu współrzędnych (w $\mathbb{R}^2$), czyli $\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$.
- Dla zadanego punktu zbadaj, czy jest on w środku obracanego trójkąta. Zawieranie się zasygnalizuj graficznie (np. zmianą rozmiaru/koloru punktu). Wykorzystaj warunek zawierania wynikający z rozwiązania poniżej podanego (tym razem nie „przez użytkownika") układu równań i nierówności (np. korzystając z reguły Cramera). Załóż, że trójkąt jest niezdegenerowany (nie jest odcinkiem, ani punktem).

  Układ

$$\begin{aligned} (p_x - a_x) &= \alpha(b_x - a_x) + \beta(c_x - a_x), \\ (p_y - a_y) &= \alpha(b_y - a_y) + \beta(c_y - a_y), \end{aligned}$$
$$\alpha \geq 0, \beta \geq 0, \alpha + \beta \leq 1,$$

  gdzie (p_x, p_y) to badany punkt, a $(a_x, a_y), (b_x, b_y), (c_x, c_y)$ to wierzchołki trójkąta.

4 Zadanie nr 3 – pokrycie

Dla zadanej figury (listy punktów połączonych odcinkami) znajdź jej najmniejszą otoczkę wypukłą (najmniejszą figurę wypukłą, która zawiera w sobie oryginalną figurę). Wyświetl figurę i jej pokrycie. Na podstawie rezultatu wnioskuj, czy oryginalna figura była wypukła (konkluzję też wyświetl).

5 Zadanie nr 4* – figury w 3d

Zweryfikuj, czy dwa trójkąty zadane w $\mathbb{R}^3$ przecinają się ze sobą. Dla uproszczenia załóż, że trójkąty leżą na różnych płaszczyznach i nie są zdegenerowane (nie są odcinkami lub punktami). Wyświetl trójkąty i zaznacz punkty (odcinki) przecięcia.