

Коси квадрат

Задатак

Нацртати квадрат са задатим координатама доње леве тачке, дужином странице и углом који основица заклапа са x осом. Одштампати колика је дужина пројекције нацртаног квадрата на x осу. Задатак урадити:

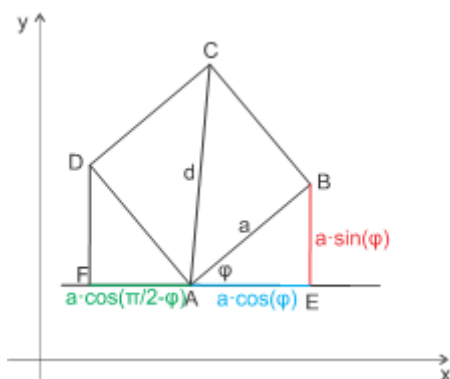
1. Коришћењем тригонометријских функција.
2. Коришћењем вектора. Креирати класу `Vector2D` која ће представљати дводимензионални вектор. Класа треба да садржи методе за одређивање интензитета вектора, нормализацију вектора, штампање вектора, за скаларни и векторски производ, и за пројекцију вектора на дати вектор. Класа треба да садржи и предефинисане операторе за сабирање, одузимање и множење. Дозвољено је додавање и других потребних метода.
3. Коришћењем `GLM` библиотеке.
4. Написати и функцију која црта полигон за задати низ тачака, а поред тога штампа и површину тог полигона.
5. Искористити методу за цртање квадрата под неким углом, и направити анимацију у којој се квадрат окреће око осе која пролази кроз доњу леву тачку квадрата и нормална је на раван xOy .
6. Модификовати методу за цртање косог квадрата тако да се уместо доње леве тачке задаје центар квадрата, а потом направити анимацију у којој се квадрат окреће око осе која пролази кроз центар квадрата и нормална је на раван xOy .

Решење

Нека је задата дужина странице квадрата a , координате доње леве тачке квадрата (x_a, y_a) и угао који треба да заклапа доња страна квадрата са позитивним делом x осе.

1. Нека је тачка E подножје нормале из тачке B на праву која пролази кроз тачку A и која је паралелна x оси (Слика 1). Тада је дужина странице $AE = a \cdot \cos(\varphi)$, а дужина странице $BE = a \cdot \sin(\varphi)$. Како су координате тачке $A(x_a, y_a)$, једноставно се рачунају координате тачке B :

$$x_b = x_a + a \cdot \cos(\varphi) \text{ и } y_b = y_a + a \cdot \sin(\varphi).$$



Слика 1: Коси квадрат

Угао $\angle DAE$ једнак је $\frac{\pi}{2} + \varphi$ па се на сличан начин добијају координате тачке D :

$$x_d = x_a + a \cdot \cos\left(\frac{\pi}{2} + \varphi\right) \text{ и } y_d = y_a + a \cdot \sin\left(\frac{\pi}{2} + \varphi\right).$$

Угао $\angle CAE$ једнак је $\frac{\pi}{4} + \varphi$. Дужина странице AC једнака је дијагонали квадрата па се координате тачке C могу добити са:

$$x_c = x_a + d \cdot \cos\left(\frac{\pi}{4} + \varphi\right) \text{ и } y_c = y_a + d \cdot \sin\left(\frac{\pi}{4} + \varphi\right).$$

Тиме су добијене координате свих тачака квадрата па се исти може нацртати.

Дуж FE представља део x осе на коме би се простирао пројекција квадрата на x осу. Дужина дужи FE једнака је

$$FE = a \cdot \cos\left(\frac{\pi}{2} - \varphi\right) + a \cdot \cos(\varphi) = a \cdot \sin(\varphi) + a \cdot \cos(\varphi) = a \cdot (\sin(\varphi) + \cos(\varphi)).$$

Пример 1. Коси квадрат – коришћењем тригонометријских функција.

```
void drawSquareTrig(double a_length, double fi, double xa, double ya)
{
    double xb, yb, xc, yc, xd, yd;

    double senka = a_length * cos(fi) + a_length * sin(fi);
    cout << "Senka ce biti duzine " << senka << "\n";

    xb = xa + a_length * cos(fi);
    yb = ya + a_length * sin(fi);

    xd = xa + a_length * cos(M_PI/2.0 + fi);
    yd = ya + a_length * sin(M_PI/2.0 + fi);

    double d = sqrt(a_length*a_length + a_length*a_length);

    xc = xa + d * cos(M_PI/4.0 + fi);
    yc = ya + d * sin(M_PI/4.0 + fi);

    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex2f (xa, ya);
        glVertex2f (xb, yb);
        glVertex2f (xc, yc);
        glVertex2f (xd, yd);
    glEnd();
}
```

2. Са слике 1 се види да је јединични вектор у правцу вектора $\overrightarrow{AB}$ једнак

$$\overrightarrow{ABnorm} = (\cos(\varphi), \sin(\varphi)),$$

а јединични вектор у правцу вектора $\overrightarrow{AD}$ једнак

$$\overrightarrow{ADnorm} = \left(\cos\left(\frac{\pi}{\varphi} + \varphi\right), \sin\left(\frac{\pi}{\varphi} + \varphi\right) \right).$$

Тада је:

$$\overrightarrow{AB} = a \cdot \overrightarrow{ABnorm},$$

$$\overrightarrow{AD} = a \cdot \overrightarrow{ADnorm}.$$

Даље важи: $\vec{B} = \vec{A} + \overrightarrow{AB}$ и $\vec{D} = \vec{A} + \overrightarrow{AD}$, где су $\vec{A}, \vec{B}$ и $\vec{D}$ радијус вектори тачака A, B и D .

Тачка C се може одредити на следећи начин: $\vec{C} = \vec{A} + \overrightarrow{AB} + \overrightarrow{AD}$.

Дужина дужи FE рачуна се као збир интезитета вектора $\overrightarrow{AF}$ и $\overrightarrow{AE}$. Вектор $\overrightarrow{AE}$ се добија као пројекција вектора $\overrightarrow{AB}$ на x осу, а вектор $\overrightarrow{AF}$ као пројекција вектора $\overrightarrow{AD}$ такође на x осу.

Пример 2. Коси квадрат – Коришћењем вектора.

У примеру 2 коришћена је метода која је дефинисана у класи *Vector2D*, а која враћа вектор пројекције посматраног вектора на задати вектор:

`Vector2D ProjectionVector(Vector2D &V).`

```
void drawSquare(double a_length, double fi, Vector2D &A)
{
    Vector2D ABnorm(cos(fi), sin(fi));
    Vector2D ADnorm(cos(M_PI/2.0 + fi), sin(M_PI/2.0 + fi));

    Vector2D AB = ABnorm * a_length;
    Vector2D AD = ADnorm * a_length;
    Vector2D AC = AB + AD;

    Vector2D xAxis(1.0, 0.0);

    double senka = AB.ProjectionVector(xAxis).Intensity() +
AD.ProjectionVector(xAxis).Intensity();
    cout << "Vektorski: Senka ce biti duzine " << senka << "\n";

    Vector2D B = A + AB;
    Vector2D C = A + AC;
    Vector2D D = A + AD;

    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex2f (A.m_x, A.m_y);
        glVertex2f (B.m_x, B.m_y);
        glVertex2f (C.m_x, C.m_y);
        glVertex2f (D.m_x, D.m_y);
    glEnd();
}
```

3. Уместо класе *Vector2D* коју је потребно написати, може се користити библиотека *GLM* (*OpenGL Mathematics*) која садржи ефикасне и тестиране имплементације 2D вектора и других класа које се често користе у раду са OpenGL-ом. За одређивање пројекције вектора на задати вектор коришћена је функција:

```
detail::tvec2<T> glm::gtx::projection::proj( detail::tvec2<T> const &x,
                                           detail::tvec2<T> const &Normal
                                           )
```

Пример 3. Цртање полигона и израчунавање његове површине.

```
using namespace glm;

void drawSquare(float a_length, float fi, vec2 A )
{
    vec2 ABnorm(cos(fi), sin(fi));
    vec2 ADnorm(cos(M_PI/2.0 + fi), sin(M_PI/2.0 + fi));
    vec2 AB = ABnorm * a_length;
    vec2 AD = ADnorm * a_length;
    vec2 AC = AB + AD;
    vec2 xAxis(1.0,0.0);

    float senka = length(proj(AB,xAxis)) + length(proj(AD,xAxis));
    printf("%f\n",senka);

    vec2 B = A + AB;
    vec2 C = A + AC;
    vec2 D = A + AD;

    glColor3f (1.0, 1.0, 1.0);

    glBegin(GL_POLYGON);
        glVertex2f (A.x, A.y);
        glVertex2f (B.x, B.y);
        glVertex2f (C.x, C.y);
        glVertex2f (D.x, D.y);
    glEnd();
}
```

4. Да би се израчунала површина полигона површ полигона се може поделити на троуглове, а потом израчунати површина полигона као сума површина тих троуглова. Површина троугла се може израчунати Хероновим обрасцем $P = \sqrt{s(s-a)(s-b)(s-c)}$, где је s полуобим троугла, а a , b и c дужине страница троугла, као и преко интезитета векторског производа два вектора.

Пример 3. Цртање полигона и израчунавање његове површине.

```
double sqr(double x) { return x * x; }

void drawPolygon(vector<Vector2D>&points, int n)
{
    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
    for(int i = 0; i < n; i++)
        glVertex2d(points[i].X(), points[i].Y());
    glEnd();

    // Povrsina
    double p = 0.0;
    for(int i = 1; i < n-1; i++)
    {
        Vector2D v1(points[i] - points[0]);
        Vector2D v2(points[i+1] - points[0]);
        p += (0.5*fabs(v1.CrossProductIntensity(v2)));
    }

    cout << "Polygon area = " << p << endl;

    p = 0.0;
    for(int i = 1; i < n-1; i++)
    {
        double a = sqrt(sqr(points[i].m_x - points[0].m_x) +
sqr(points[i].m_y - points[0].m_y));
        double b = sqrt(sqr(points[i+1].X() - points[i].m_x) +
sqr(points[i+1].m_y - points[i].m_y));
        double c = sqrt(sqr(points[i+1].m_x- points[0].m_x) +
sqr(points[i+1].m_y - points[0].m_y));

        double s = 0.5*(a+b+c);
        p += sqrt(s*(s-a)*(s-b)*(s-c));
    }

    cout << "Polygon area (HERONOV) = " << p << endl;
}
```

Поред представљеног алгоритма, површину многоугла је могуће ефикасно израчунавати помоћу формуле пертле, која се још назива и Гаусова површинска формула и која гласи:

$$P = \frac{1}{2} \left| \sum_{i=1}^n x_i (y_{i+1} - y_{i-1}) \right|$$

5. Анимација

При креирању анимације потребно је користити:

```
void glutTimerFunc(unsigned int msecs, void (*func)(int value), value);
```

Брзину анимације подесити на 60 fps. Дефинишемо методу `timer` у којој увећавамо вредност угла φ .

```
void timer(int v)
{
    fi += deltaFi;

    if (fi > 360)
        fi = 0.0;

    glutTimerFunc(1000 / FPC, timer, v);
    glutPostRedisplay();
}
```