Лекція 5

ОСНОВИ ПРОГРАМУВАННЯ
ГРАФІКИ В QT

Лекція 5. Основи програмування графіки в Qt

План

1. Реалізація двовимірної графіки із застосуванням класу QPainter.
2. Основи графічного стандарту OpenGL.
3. Рисування тривимірної графіки із застосуванням OpenGL та Qt.

1. Реалізація двовимірної графіки із застосуванням класу QPainter

Бібліотека Qt підтримує можливість зображення дво- та тривимірної графіки. Для безпосереднього зображення на віджеті використовується клас **QPainter**.

Рисування двовимірних графічних сцен простіше за все може реалізувати у стандартному методі **paintEvent()** класу **QWidget**, який автоматично викликається при необхідності відображення змісту віджета. Наприклад:

```
// ...
class MainWindow : public QMainWindow
{
    Q_OBJECT

protected:
    void paintEvent(QPaintEvent *event);

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
// ...
```

```
// ...
void MainWindow::paintEvent(QPaintEvent*)
{
        QPainter painter(this);

        // Реалізація зображення сцени
        // ...
}
// ...
```

1. Реалізація двовимірної графіки із застосуванням класу QPainter

Бібліотека Qt реалізує візуалізацію двовимірних графічних сцен із застосуванням трьох стандартних інструментів: **пера** (pen), **кісті** (brush) та **шрифту** (font).

Перо (інкапсулюється класом **QPen**) застосовується для зображення всіх ліній, як окремих, так і границь замкнутих фігур. Атрибутами пера є колір, товщина, стиль (суцільна чи пунктирна) і т. п.

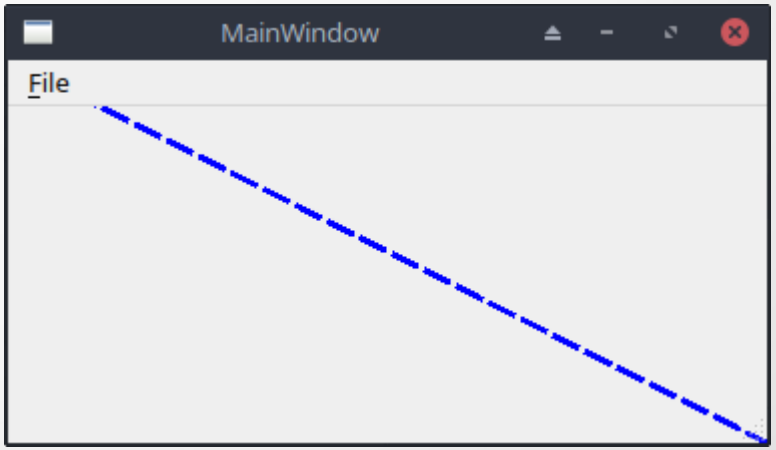
Кість (**QBrush**) використовується для зафарбовування замкнутих геометричних фігур. Як і перо, кість також має певний колір, стиль, прозорість тощо.

Шрифт (**QFont**) використовується для виведення текстів. Шрифти мають певний розмір, стиль відображення, форму і т.п.

1. Реалізація двовимірної графіки із застосуванням класу QPainter

Наприклад, наступний код реалізує виведення пунктирної діагональної лінії синього кольору товщиною 3.

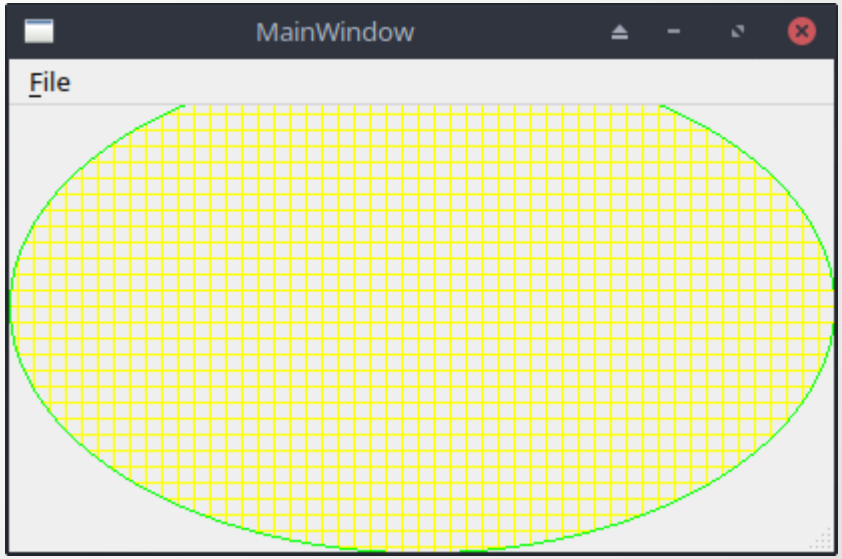
```
void MainWindow::paintEvent(QPaintEvent *)
{
    QPainter painter(this);
    QPen pen;

    pen.setColor(Qt::blue);
    pen.setWidth(3);
    pen.setStyle(Qt::DashLine);
    painter.setPen(pen);
    painter.drawLine(0, 0, width(), height());
}
```



1. Реалізація двовимірної графіки із застосуванням класу QPainter

Виведення еліпсу, зафарбованого жовтим штрихом, і обмеженого лінією зеленого кольору, можна реалізувати наступним чином.

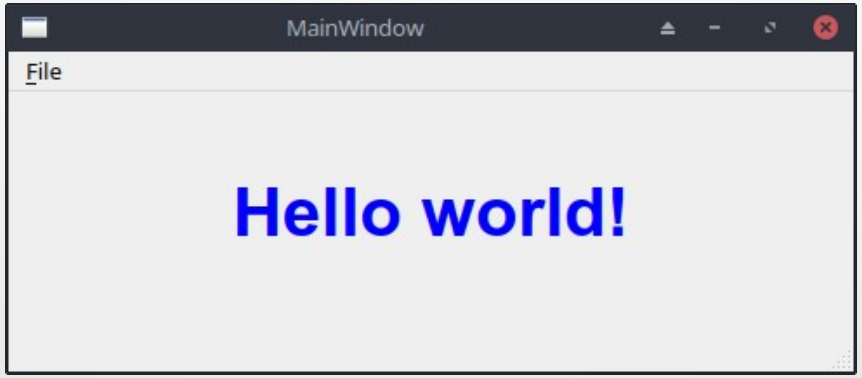
```
void MainWindow::paintEvent(QPaintEvent *)
{
    QPainter painter(this);
    QPen pen(Qt::green);
    QBrush brush(Qt::yellow);

    brush.setStyle(Qt::CrossPattern);
    painter.setPen(pen);
    painter.setBrush(brush);
    painter.drawEllipse(0, 0, width(), height());
}
```



1. Реалізація двовимірної графіки із застосуванням класу QPainter

Вивести надпис заданого шрифту, розміру та кольору в центрі віджету можна, наприклад, так.

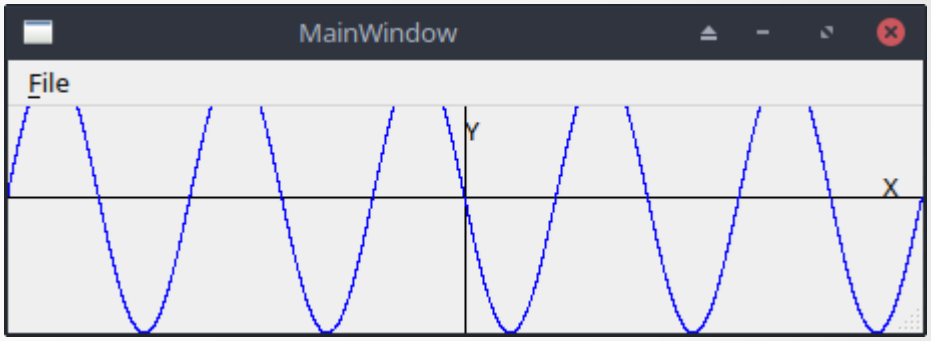
```
void MainWindow::paintEvent(QPaintEvent *)
{
    QPainter painter(this);
    QFont font("Arial", 30);

    font.setBold(true);
    painter.setPen(Qt::blue);
    painter.setFont(font);
    painter.drawText(rect(), Qt::AlignCenter, "Hello world!");
}
```



1. Реалізація двовимірної графіки із застосуванням класу QPainter

Нарисувати графік синусу на відрізку [-5π; 5π] можна таким чином.

```
void MainWindow::paintEvent(QPaintEvent *)
{
    QPainter painter(this);
    int h = height(), w = width(), cx, cy;
    double y;
    painter.drawLine(w / 2, 0, w / 2, h);
    painter.drawLine(0, h / 2, w, h / 2);
    painter.drawText(w - 20, h / 2, "X");
    painter.drawText(w / 2, 40, "Y");
    painter.setPen(Qt::blue);
    for (double x = -5 * M_PI; x <= 5 * M_PI; x += (10.0 * M_PI * 0.0001))
    {
        y = sin(x);
        cx = (x * w) / (10.0 * M_PI) + w / 2;
        cy = (y * h) / 2.0 + h / 2;
        painter.drawPoint(cx, cy);
    }
}
```



2. Основи графічного стандарту OpenGL

OpenGL (Open Graphics Library) – це промисловий крос-платформовий графічний стандарт на розробку графічного програмного забезпечення. OpenGL не залежить від конкретної мови програмування та апаратної платформи.

Фактично OpenGL є **прикладним програмним інтерфейсом** (API) для написання програм, що використовують дво- та тривимірну комп'ютерну графіку.

Стандарт OpenGL було розроблено у 1992 році. Він підтримується більшістю компаній, що випускають апаратне та програмне забезпечення (Microsoft, IBM, Intel, AMD, Nvidia, HP, ...).

2. Основи графічного стандарту OpenGL

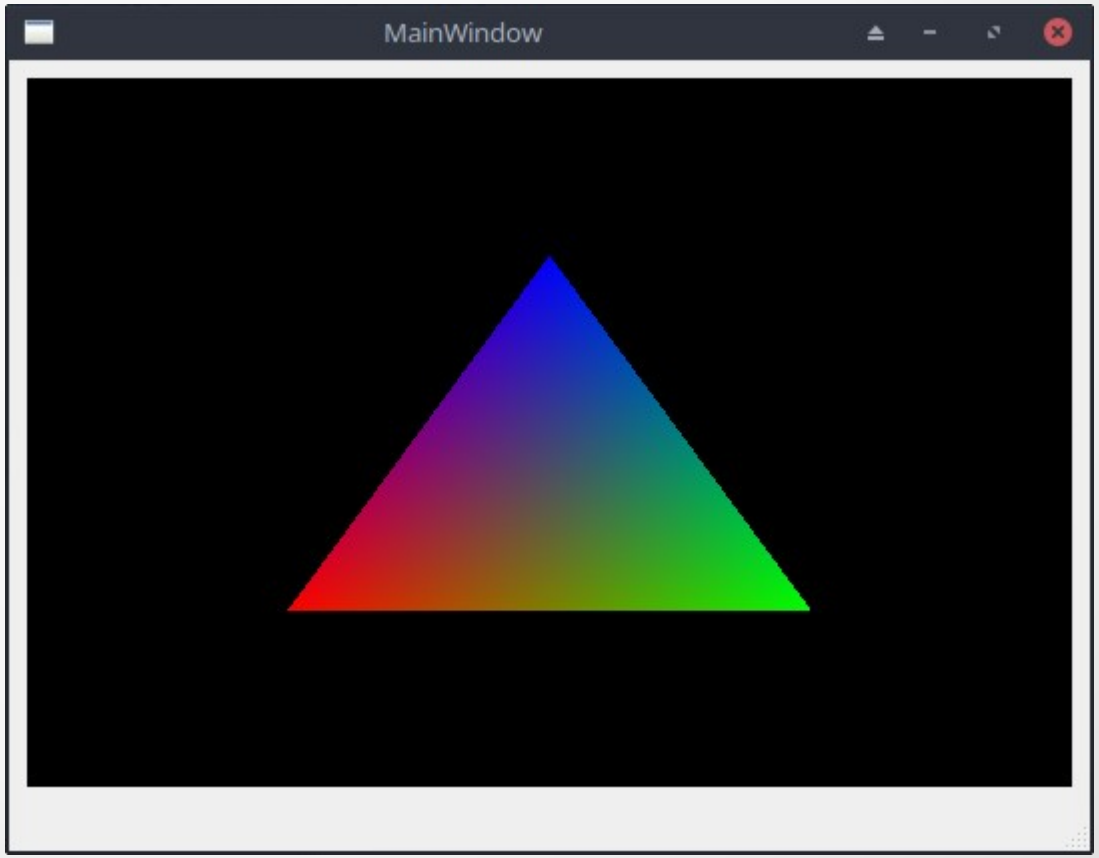
В OpenGL наявні наступні можливості:

- **графічні примітиви** (точки, лінії, полігони);
- **B-сплайни** (рисування кривих за опорними точками);
- **перетворення координат** (масштабування, обертання, зсув);
- **робота з кольором**;
- **видалення невидимих ліній і поверхонь**;
- **подвійна буферизація** (кадр зображення спочатку готується в пам’яті, а потім виводиться на екран, що запобігаю мерехтінню сцени);
- **робота з текстурами** (підвищення реалістичності графічних сцен за рахунок накладення растрових зображень на дво- та тривимірні фігури);
- **згладжування** (приховування ступінчастості, що зазвичай притаманна растровим зображенням);
- **освітлення** (дозволяє задавати джерела світла, їх розташування, інтенсивність і т .п., що значно підвищує реалістичність зображення);
- **атмосферні ефекти** (туман, дим тощо);
- **прозорість** об’єктів, що зображуються.

2. Основи графічного стандарту OpenGL

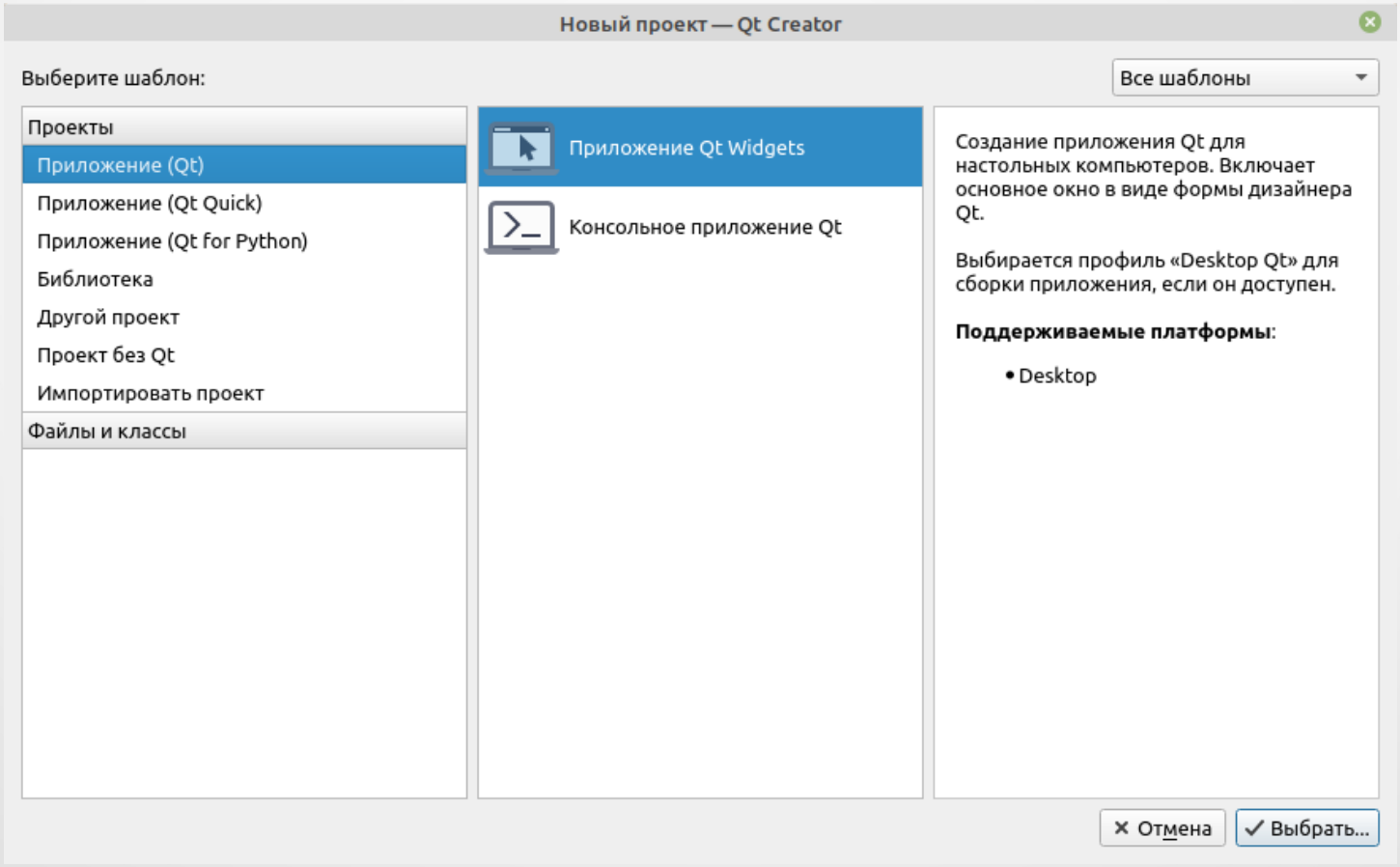
Наприклад, зображення різнокольорового трикутника із застосуванням OpenGL реалізується наступним чином:

```
glBegin(GL_TRIANGLES);
glColor3f(1.0, 0.0, 0.0);
glVertex3f(-0.5, -0.5, 0);
glColor3f(0.0, 1.0, 0.0);
glVertex3f( 0.5, -0.5, 0);
glColor3f(0.0, 0.0, 1.0);
glVertex3f( 0.0,  0.5, 0);
glEnd();
```



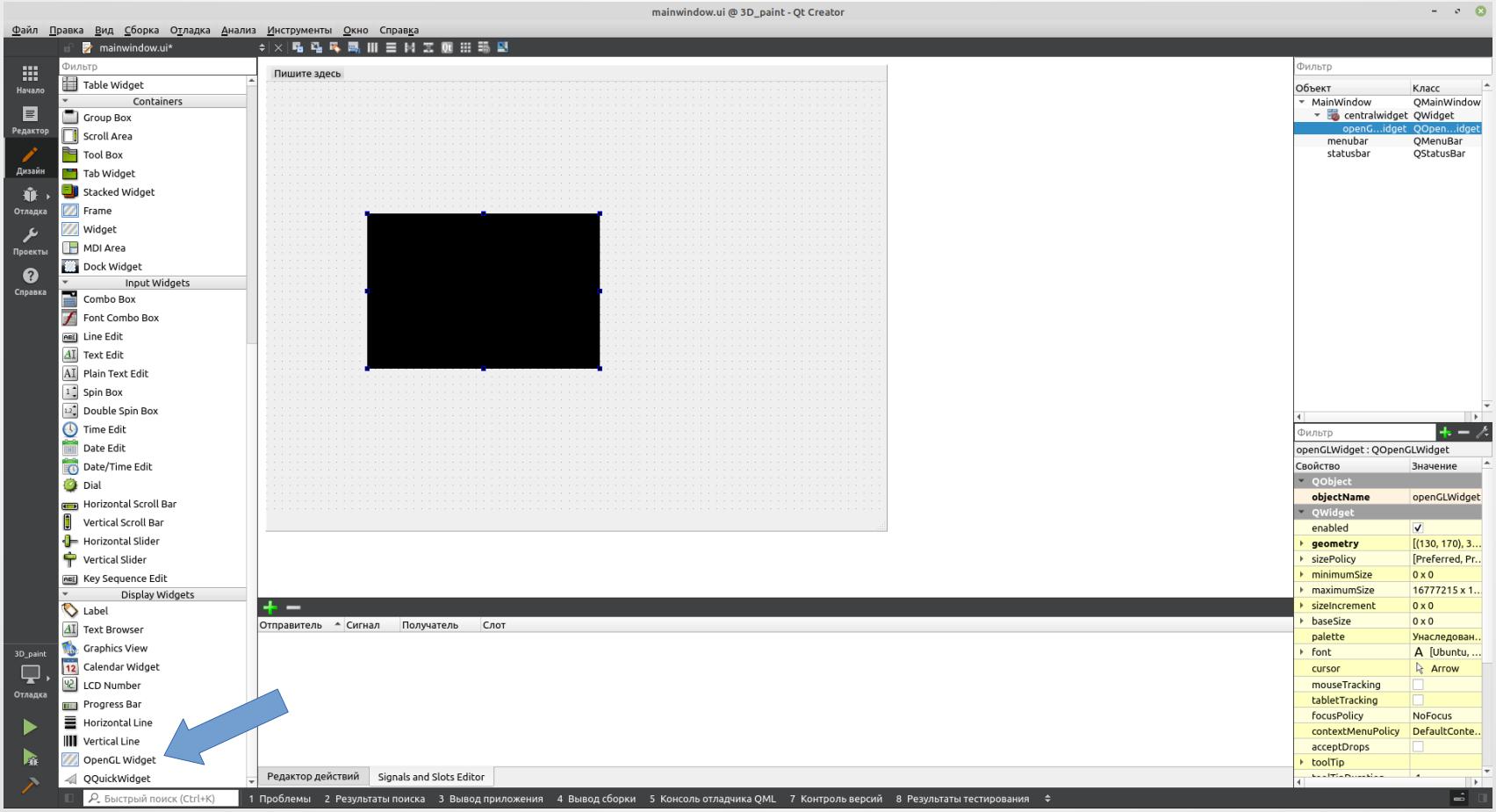
3. Рисування тривимірної графіки із застосуванням OpenGL та Qt

Для створення в Qt Creator програми, що реалізує роботу з тривимірною графікою із застосуванням OpenGL, в першу чергу, створимо проєкт “Приложение Qt Widgets”.



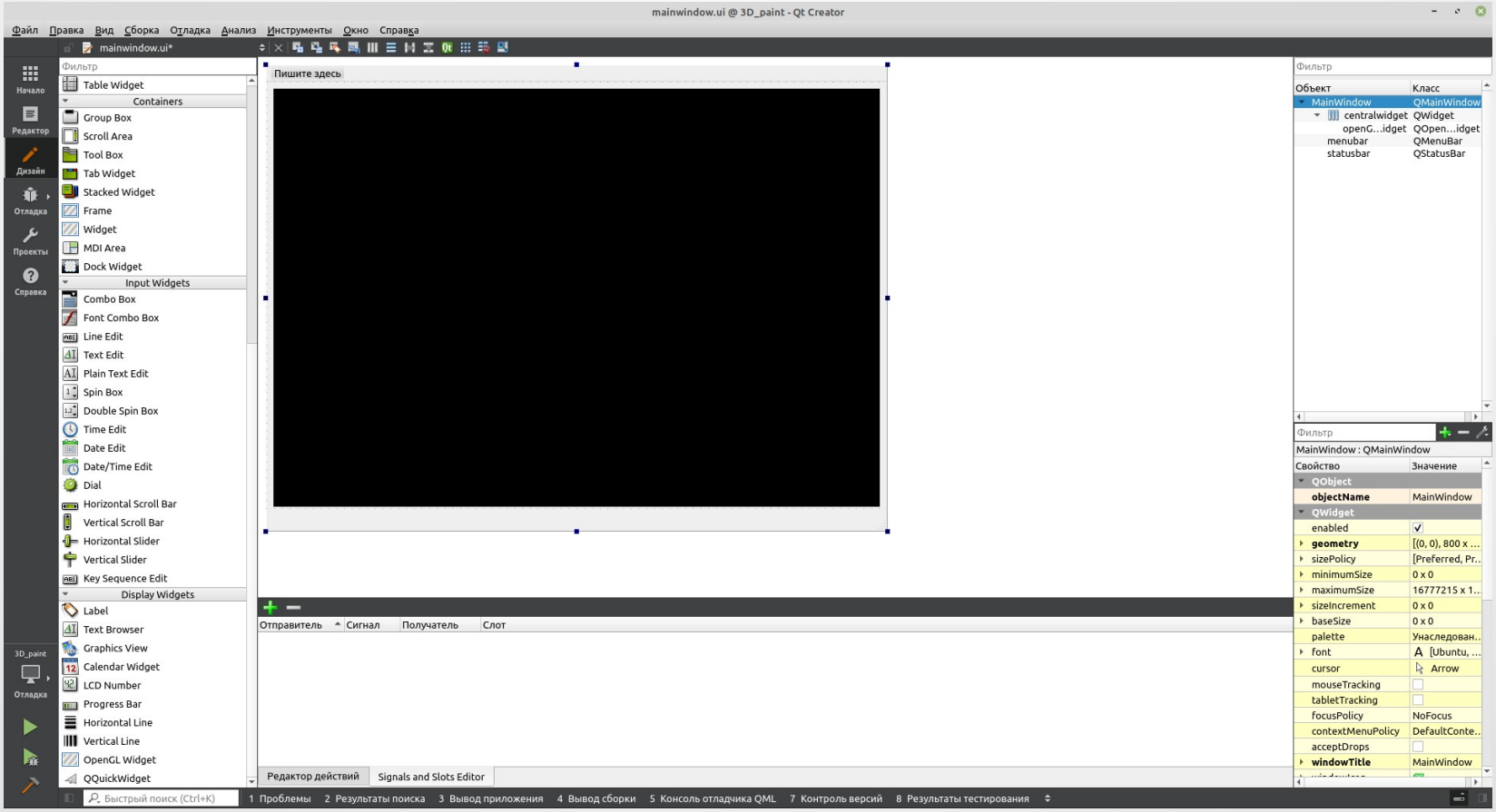
3. Рисування тривимірної графіки із застосуванням OpenGL та Qt

Після чого в Qt Designer відкриваємо форму головного вікна і додаємо до неї віджет **OpenGLWsdget** (клас QOpenGLWsdget)



3. Рисування тривимірної графіки із застосуванням OpenGL та Qt

Для **компонування автозаповненням** цього віджету на формі слід клікнути правою кнопкою миші на формі MainWindows і у контекстному меню, що з’явиться, вибрати команду “Компоновка | Скомпоновать по горизонтали”.



3. Рисування тривимірної графіки із застосуванням OpenGL та Qt

Після чого для завершення налаштування проєкту у файл налаштувань (*.pro) слід додати підключення відповідних модулів:

QT += core gui **opengl openglwidgets**

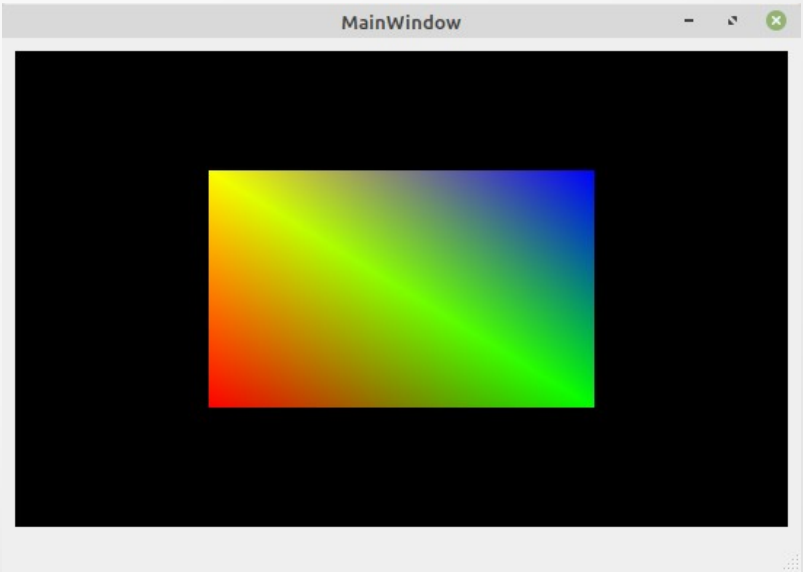
...

Компіляція та запуск програми повинні дати наступний результат:

3. Рисування тривимірної графіки із застосуванням OpenGL та Qt

Для зображення чотирикутника за допомогою OpenGL реалізуємо метод **paintEvent()** головного вікна програми (об’єкт класу QMainWindow) наступним чином:

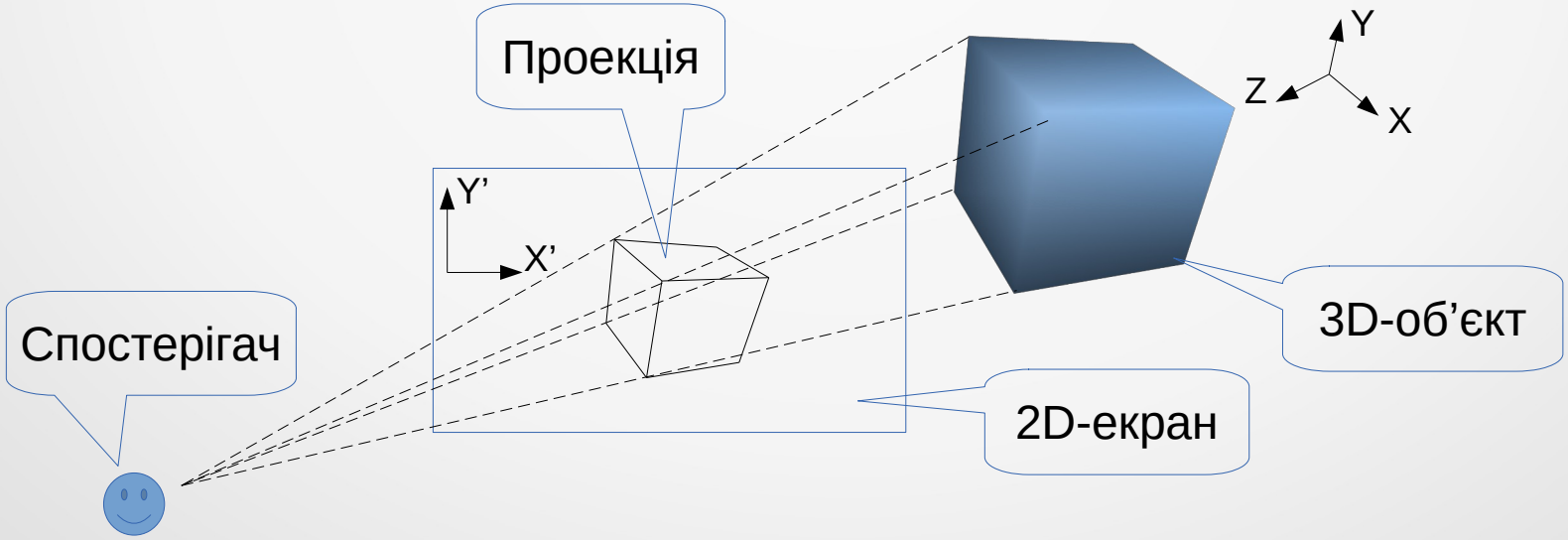
```
void MainWindow::paintEvent(QPaintEvent*)
{
    // Очищення сцени
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Рисування різнокольорового чотирикутника
    glBegin(GL_QUADS);
        glColor3f(1.0, 0.0, 0.0);
        glVertex3f(-0.5, -0.5, 0);
        glColor3f(0.0, 1.0, 0.0);
        glVertex3f( 0.5, -0.5, 0);
        glColor3f(0.0, 0.0, 1.0);
        glVertex3f( 0.5, 0.5, 0);
        glColor3f(1.0, 1.0, 0.0);
        glVertex3f( -0.5, 0.5, 0);
    glEnd();
}
```



3. Рисування тривимірної графіки із застосуванням OpenGL та Qt

В загальному випадку для зображення тривимірної сцени необхідно реалізувати виконання трьох стандартних дій:

1) **початкове налаштування сцени** (ініціалізація OpenGL, задання базових параметрів зображення, джерел світла і таке інше, які рідко змінюються, або взагалі не змінюються);
2) **налаштування камери вікна перегляду** (позиції спостерігача, відстані до об’єкту, що спостерігається, параметрів трансформації координат тощо);
3) **безпосереднє рисування** сцени та виведення її на екран.



3. Рисування тривимірної графіки із застосуванням OpenGL та Qt

В найбільш простому випадку всі ці етапи можна об’єднати і реалізувати у функції paintEvent() головного вікна програми, наприклад, так:

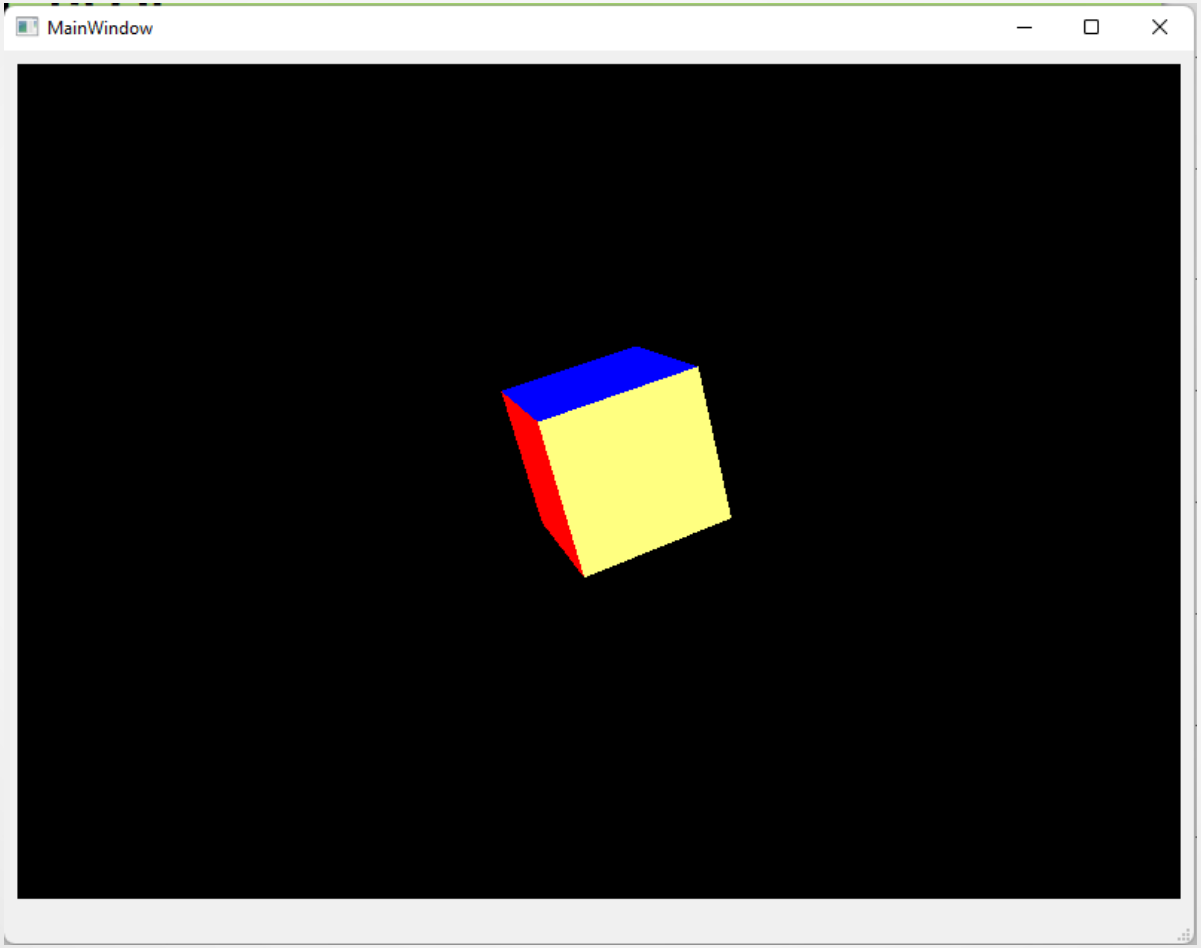
```
void MainWindow::paintEvent(QPaintEvent*)
{
    double radius = 5.0, fnear = 0.01 * radius, ffar = 10 * radius, angle = 60,
                h = tan(angle * M_PI / 360.0) * fnear,
                a = (height() == 0) ? 1.0 : double(width()) / double(height()),  w = a * h;


    glEnable(GL_DEPTH_TEST); // Включення режиму тестування глибини

    glMatrixMode(GL_PROJECTION); // Налаштування камери сцени
    glLoadIdentity();
    glFrustum(-w, w, -h, h, fnear, ffar);
    glTranslatef(0, 0, -radius);
    glViewport(0, 0, width(), height());
    glMatrixMode(GL_MODELVIEW);

    // Очищення сцени
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // Поворот сцени на 30 градусів навколо осей X,Y та Z
    glRotatef(30, 1, 1, 1);
```

3. Рисування тривимірної графіки із застосуванням OpenGL та Qt

```
// Рисування різнокольорового кубика шляхом зображення його граней
glBegin(GL_QUADS);
    glColor3f(1.0, 0.0, 0.0);
    glNormal3f(0, 0, -1);
    glVertex3f(-0.5, -0.5, -0.5);
    glVertex3f( 0.5, -0.5, -0.5);
    glVertex3f( 0.5, 0.5, -0.5);
    glVertex3f( -0.5, 0.5, -0.5);

    glColor3f(0.0, 1.0, 0.0);
    glNormal3f(0, 0, 1);
    glVertex3f(-0.5, -0.5, 0.5);
    glVertex3f( 0.5, -0.5, 0.5);
    glVertex3f( 0.5, 0.5, 0.5);
    glVertex3f( -0.5, 0.5, 0.5);
    // ...

glEnd();
```

3. Рисування тривимірної графіки із застосуванням OpenGL та Qt

Компіляція та запуск програми дадуть наступний результат:



Примітка. Кожного разу при перерисовані вікна програми кубик буде обертатися на 30 градусів навколо всіх трьох осей.