Лабораторна робота № 10
OpenGL та .NET Framework

План

1. Створення програм, що використовують OpenGL.
2. Завдання

1. OpenGL (Open Graphics Library) – промисловий кросплатформний стандарт програмного інтерфейсу для розробки застосунків, що використовують дво- та тривимірну комп'ютерну графіку. В наш час OpenGL підтримується більшістю провідних виробників апаратного та програмного забезпечення (Microsoft, IBM, Intel, Dell, Hewlett-Packard, Apple, …). Крім того OpenGL апаратно підтримується більшістю сучасних відеоадаптерів.

Для розробки програми, що використовують OpenGL, у Microsoft Visual Studio створимо проект Windows Forms (.NET Framework). Далі за допомогою NuGet (див. Лабораторну роботу № 8) встановимо для поточного проекту компоненти «OpenTK» та «OpenTK.GLControl». Після додавання цих компонентів з'явиться можливість програмувати тривимірну графіку з використанням OpenGL. Для цього слід додати до форми новий компонент GLControl (Рис. 1).

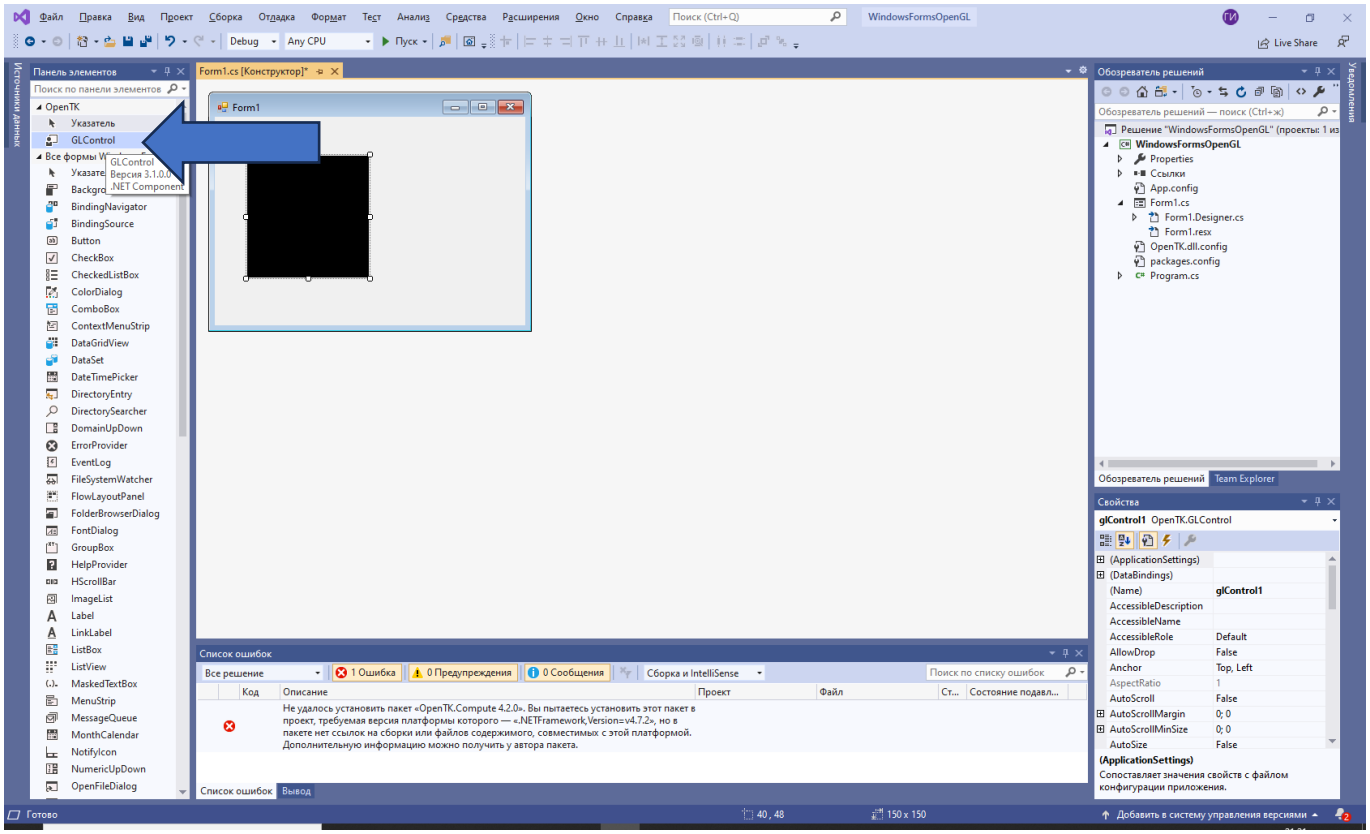

Рис. 1. Додавання GLControl на форму

У вікні «Properties» властивість «Dock» встановимо у значення «Fill» (автоматичне масштабування на всю клієнтську область форми) (Рис. 2).

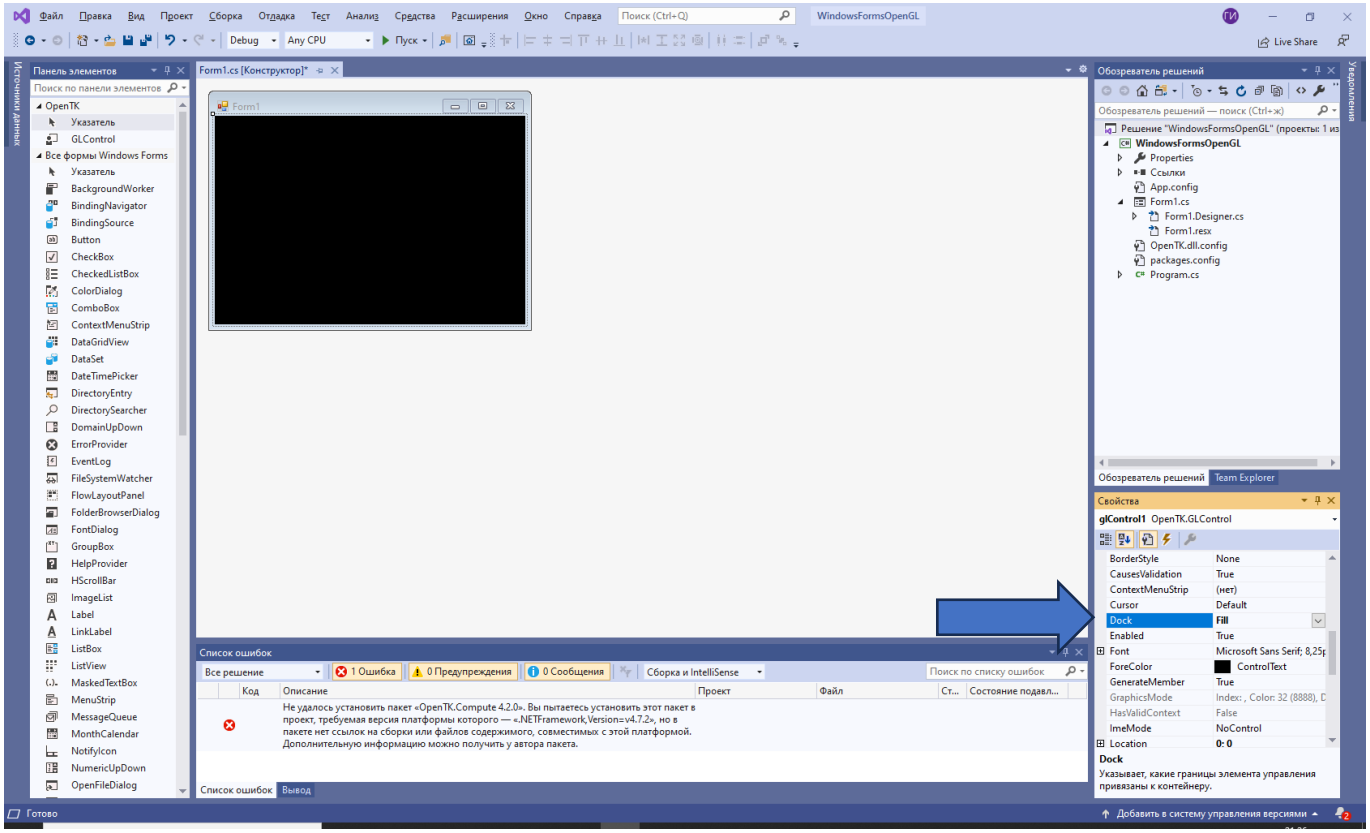

Рис. 2. Налаштування автозаповнення для об'єкта GLControl

Після цього активуємо список подій у вікні «Properties» (натиснувши кнопку з піктограмою у вигляді блискавки) і додамо для об'єкта GLControl обробники подій «Load» (завантаження), «Resize» (зміна розміру) і «Paint» (оновлення зображення) (Рис. 3).

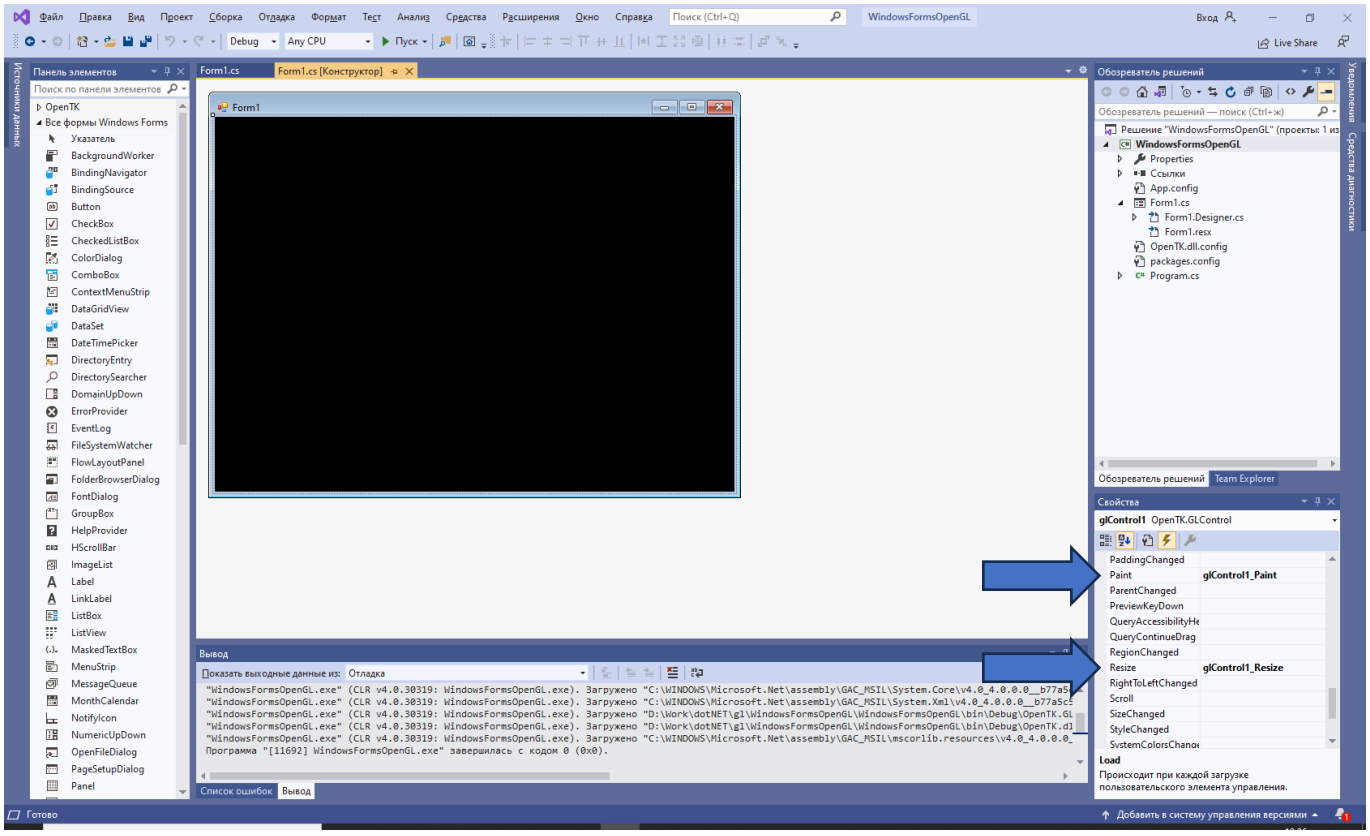

Рис. 3. Додавання обробників подій «Load», «Paint» та «Resize»

Тепер перейдемо у вихідний текст, який описує форму. Для доступу до функцій OpenGL додамо простір імен OpenTK.Graphics.OpenGL:

```
using OpenTK.Graphics.OpenGL;
```

В обробник події «Load» додамо виклик функцій, що здійснюють базові налаштування графіки. У нашому випадку це буде:

```
private void glControl1_Load(object sender, EventArgs e)
{
    // Активізуємо тест глибини (для алгоритму Z-буфера)
    GL.Enable(EnableCap.DepthTest); // glEnable(GL_DEPTH_TEST);
    // Задаємо колір фону
    GL.ClearColor(0.5f, 0.5f, 1, 1);
}
```

У цьому прикладі при виконанні команди OpenTK у коментарях зазначена відповідна їй оригінальна команда OpenGL, що реалізує ту саму функцію.

У обробнику події «Resize» налаштуємо параметри відображення графічної сцени, що залежать від розміру вікна:

```
private void glControl1_Resize(object sender, EventArgs e)
{
    double radius = 2, // Радиус сферы, содержащей всю отображающуюся сцену
           near = 0.01 * radius,
           far = 10 * radius,
           angle = 60,
           h = Math.Tan(angle * Math.PI / 360.0) * near,
           a = (Height == 0) ? 1.0 : (double)Width / (double)Height,
           w = a * h;

      GL.MatrixMode(MatrixMode.Projection); // glMatrixMode(GL_PROJECTION);
      GL.LoadIdentity(); // glLoadIdentity();
      GL.Frustum(-w, w, -h, h, near, far); // glFrustum(-w, w, -h, h, near, far);
      GL.Translate(0, 0, -radius); // glTranslatef(0, 0, -radius);
      GL.Viewport(0, 0, Width, Height);  // glViewport(0, 0, width, height);
      GL.MatrixMode(MatrixMode.Modelview); // glMatrixMode(GL_MODELVIEW);
}
```

Тут слід зазначити, що значення змінної radius в загальному вигляді має обчислюватися виходячи з габаритів сцени, що зображується.

У цьому коді здійснюється налаштування графічної сцени. За допомогою функції Frustum() задається геометрія області, в якій буде знаходитись об'єкт, що відображається, а також параметри візуалізації перспективи.

В обробнику події «Paint» помістимо код, який безпосередньо реалізує побудову графічного образу:

```
private void glControl1_Paint(object sender, PaintEventArgs e)
{
    // Очищення сцени glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```
        GL.Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);

        // Ініціалізація матриці перетворення координат
        GL.MatrixMode(MatrixMode.Projection);
        GL.LoadIdentity();

        // Поворот сцени на 40 градусів відносно x, y та z
        GL.Rotate(40, 1, 1, 1); // glRotate(40, 1, 1, 1);

        // **** Рисування трикутної піраміди ****
        // Включаємо режим відображення трикутників
        GL.Begin(PrimitiveType.Triangles); // glBegin(GL_TRIANGLES);
        // Задаємо поточний колір
        GL.Color3(1.0f, 0, 0); // glColor3f(1, 0, 0);
        // Задаємо координати першої грані
        GL.Vertex3(0, 0, 0); // glVertex3f(0, 0, 0);
        GL.Vertex3(1, 0, 0);
        GL.Vertex3(0, 1, 0);

        // Задаємо колір та координати другої грані
        GL.Color3(0, 1.0f, 0);
        GL.Vertex3(0, 0, 0);
        GL.Vertex3(0, 0, 1);
        GL.Vertex3(0, 1, 0);

        // Третя грань
        GL.Color3(0, 0, 1.0f);
        GL.Vertex3(0, 0, 0);
        GL.Vertex3(1, 0, 0);
        GL.Vertex3(0, 0, 1);

        // Четверта ...
        GL.Color3(1.0f, 1.0f, 0);
        GL.Vertex3(1, 0, 0);
        GL.Vertex3(0, 1, 0);
        GL.Vertex3(0, 0, 1);
        // Кінець рисування трикутників
        GL.End(); // glEnd();

        // Виведення сформованого зображення на форму
        glControl1.SwapBuffers();
}
```

Запуск програми призведе до наступного результату (Рис. 4).

Для обертання піраміди під час руху мишки з натиснутою кнопкою потрібно додати обробники подій «MouseDown», «MouseUp» і «MouseMove» для компонента GLControl.

Нижче наведено повний вихідний текст класу, що реалізує цю можливість.

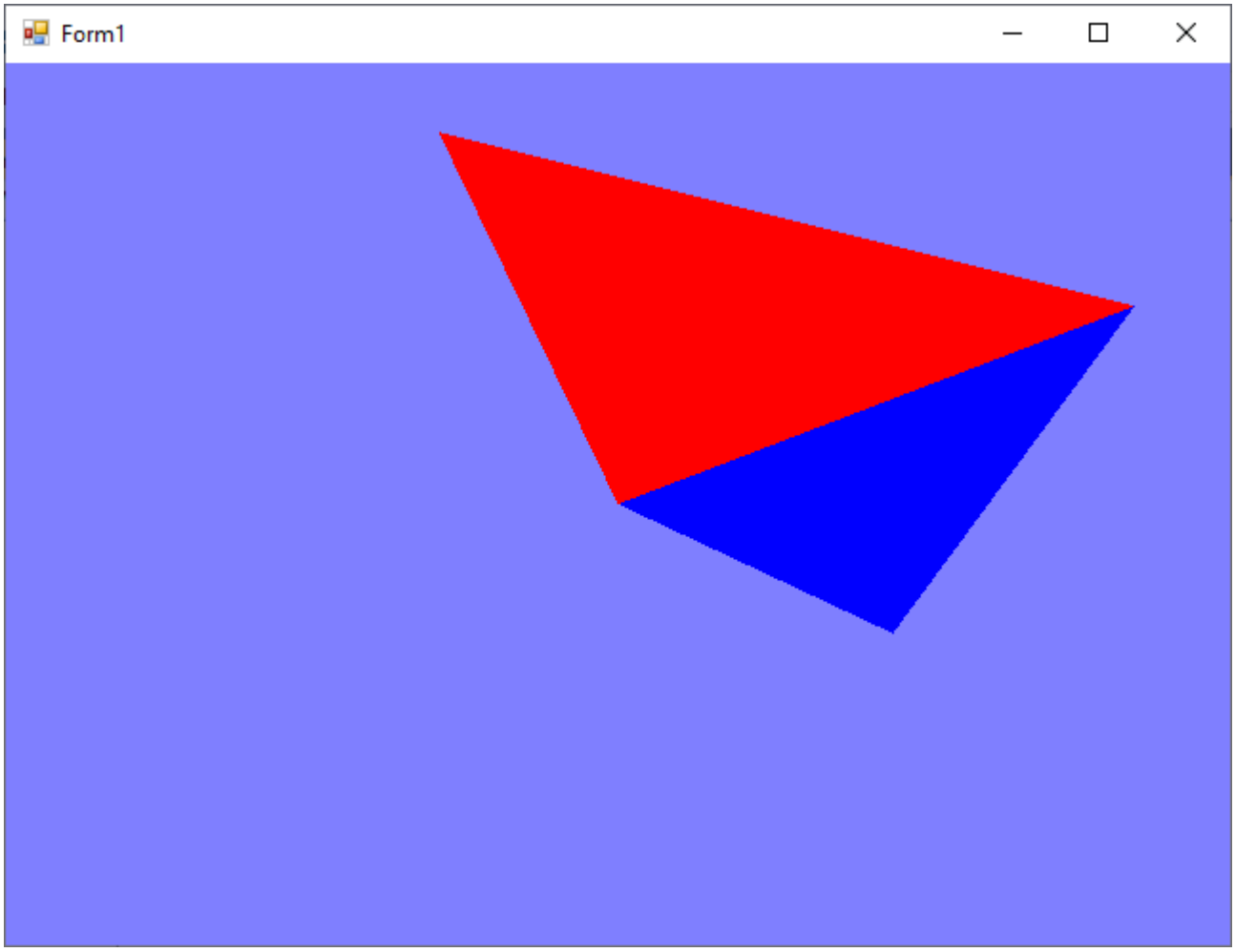

Рис. 4. Результат роботи програми, що зображує трикутну піраміду

```
using System;
using System.Windows.Forms;
using OpenTK.Graphics.OpenGL;

namespace WindowsFormsOpenGL
{
   public partial class Form1 : Form
   {
      double angle = 0;
      bool isMouseDown = false;
      public Form1()
      {
         InitializeComponent();
      }

      private void glControl1_MouseDown(object sender, MouseEventArgs e)
      {
         isMouseDown = true;
      }

      private void glControl1_MouseUp(object sender, MouseEventArgs e)
      {
         isMouseDown = false;
```

```
}

private void glControl1_MouseMove(object sender, MouseEventArgs e)
{
   if (isMouseDown)
   {
      angle += 15;
      glControl1.Refresh();
   }
}

private void glControl1_Paint(object sender, PaintEventArgs e)
{
   GL.Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);

   GL.MatrixMode(MatrixMode.Projection);
   GL.LoadIdentity();

   GL.Translate(-0.25f, -0.25f, -0.25f);

   GL.Rotate(angle, 1, 1, 1); // glRotate(40, 1, 1, 1);

   GL.Begin(PrimitiveType.Triangles);
   GL.Color3(1.0f, 0, 0);
   GL.Vertex3(0, 0, 0);
   GL.Vertex3(1, 0, 0);
   GL.Vertex3(0, 1, 0);

   GL.Color3(0, 1.0f, 0);
   GL.Vertex3(0, 0, 0);
   GL.Vertex3(0, 0, 1);
   GL.Vertex3(0, 1, 0);

   GL.Color3(0, 0, 1.0f);
   GL.Vertex3(0, 0, 0);
   GL.Vertex3(1, 0, 0);
   GL.Vertex3(0, 0, 1);

   GL.Color3(1.0f, 1.0f, 0);
   GL.Vertex3(1, 0, 0);
   GL.Vertex3(0, 1, 0);
   GL.Vertex3(0, 0, 1);
   GL.End();

   glControl1.SwapBuffers();

}

private void glControl1_Load(object sender, EventArgs e)
{
   GL.Enable(EnableCap.DepthTest);
   GL.ClearColor(0.5f, 0.5f, 1, 1);
}
```

```
        private void glControl1_Resize(object sender, EventArgs e)
        {
            double radius = 4,
                   near = 0.01 * radius,
                   far = 10 * radius,
                   angle = 60,
                   h = Math.Tan(angle * Math.PI / 360.0) * near,
                   a = (Height == 0) ? 1.0 : (double)Width / (double)Height,
                   w = a * h;

            GL.MatrixMode(MatrixMode.Projection);
            GL.LoadIdentity(); // glLoadIdentity();
            GL.Frustum(-w, w, -h, h, near, far);
            GL.Translate(0, 0, -radius);
            GL.Viewport(0, 0, Width, Height);
            GL.MatrixMode(MatrixMode.Modelview);
            glControl1.Refresh();

        }
    }
}
```

2. Переробити наведену вище програму таким чином, щоб вона зображала тривимірну цифру:

1 варіант – «2»;
2 варіант – «3»;
3 варіант – «4»;
4 варіант – «5»;
5 варіант – «6».

У заголовку форми виводити ПІБ та номери групи студента, який здає роботу.

ВАЖЛИВО! У разі, якщо при запуску програми не відображається графічна сцена, слід у налаштуваннях компонента GLControl увімкнути режим сумісності (compatibility).