

```

using namespace System::Collections::Generic; //List
using namespace System::Drawing::Drawing2D; //Matrix

//ablak betölt
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e) {
    Text = "Grafika";
    tabPage1->Text = "2D";
    tabPage2->Text = "3D";
    // Fgv. ábrázolás
    checkBox1->Text = "Arányos";
    checkBox1->Checked = true;
    comboBox1->Items->Add("Sin(x)*Ln(x)");
    comboBox1->Items->Add("Sin(x)*Cos(2x)");
    comboBox1->Items->Add("Sin(x)*Sin(2x)");
    comboBox1->SelectedIndex = 0;

    // 3D
    Random ^ randi = gcnew Random();
    comboBox1->SelectedIndex = randi->Next(3);
    comboBox2->Items->Add("Izometrikus");
    comboBox2->Items->Add("Cavalier");
    comboBox2->Items->Add("Centrális");
    comboBox2->SelectedIndex = randi->Next(3);
}

////////// Fgv. ábrázolás //////////

// kiválasztható függvények
private: double fgv(double x)
{
    switch (comboBox1->SelectedIndex)
    {
        case 1: return Math::Sin(x) * Math::Cos(2 * x); break;
        case 2: return Math::Sin(x) * Math::Sin(2 * x); break;
        default:
            if (x <= 0) return 0; //csalás a Log miatt
            return Math::Sin(x) * Math::Log(x);
            break;
    }
}

// Paint esemény használata a folyamatos újra rajzolás miatt
private: System::Void tabPage1_Paint(System::Object^ sender,
System::Windows::Forms::PaintEventArgs^ e) {

    //függvényábrázolás határai
    float xmin = 0;
    float xmax = 2 * Math::PI;
    float ymin = -2;
    float ymax = 2;

    //képernyő transzformációja
    Matrix ^ m = gcnew Matrix();
    m->Translate(tabPage1->Width / (xmax - xmin) * -1 * xmin // x tengely eltolása
        , tabPage1->Height / (ymax - ymin) * ymax); // y tengely eltolása
    float m_x = tabPage1->Width / (xmax - xmin); // x arány
    float m_y = tabPage1->Height / (ymax - ymin); // y arány
    float arany = m_x < m_y ? m_x : m_y; // arányos megjelenítéshez,
    mindig a kisebb kell
    if (checkBox1->Checked) // arányos megjelenítés
        m->Scale(arany, -arany);
    else m->Scale(m_x, -m_y); // nyújtott megjelenítés
    e->Graphics->Transform = m; // tarnszformáció végrehajtása

    e->Graphics->Clear(this->BackColor); // háttér törlése
}

```

```

Pen ^ p = gcnew Pen(Color::Red, 2 / arany); //0.005f // toll arányos vonalvastagsággal

//függvény kirajzoltatása
float dx = 0.1;
for (float x = xmin; x < xmax - dx; x += 0.1)
    e->Graphics->DrawLine(p, x, fgv(x), x+dx, fgv(x+dx)); // végpont
    behelyettesítéssel

//tengelyosztások és tengely vonalak
p->Color = Color::Gray;
p->Width = 1 / arany; //0.001f
for (float i = 0; i < xmax; i += Math::PI / 2) // x osztás PI/2
    e->Graphics->DrawLine(p, PointF(i, 0.1f), PointF(i, -0.1f));
// y osztás
e->Graphics->DrawLine(p, PointF(0.1f, 1), PointF(-0.1f, 1));
e->Graphics->DrawLine(p, PointF(0.1f, -1), PointF(-0.1f, -1));
// pontvonal készítés GT3 :)
array<float> ^ dp = { 0.2f, 0.05f, 0.05f, 0.05f };
p->DashPattern = dp;
e->Graphics->DrawLine(p, PointF(xmin, 0), PointF(xmax, 0));
e->Graphics->DrawLine(p, PointF(0, ymin), PointF(0, ymax));
}

// Arányos/Nyújtott megjelenítés
private: System::Void checkBox1_CheckedChanged(System::Object^ sender, System::EventArgs^
e) {
    tabPage1->Refresh(); //Paint esemény meghívása
}

// Fgv. kiválasztása
private: System::Void comboBox1_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e) {
    tabPage1->Refresh(); //Paint esemény meghívása
}

//////////////////////////////////// 3D //////////////////////////////////////

// 2D-s pont
ref class p2D
{
public:
    float x, y;
    p2D(float x, float y) :x(x), y(y) {};
};

// 3D-s pont
ref class p3D
{
public:
    float x, y, z;
    p3D(float x, float y, float z) :x(x), y(y), z(z) {};
};

// Vetítés interfészrelése
interface class Vetites
{
public: p2D^ vetit(p3D^ be);
};

// Axonometikus vetítést megvalósító osztály
ref class Axonometrikus : Vetites
{
public:
    float qx, qy, qz, alfa, beta, gamma;
    Axonometrikus(float qx, float qy, float qz, float alfa, float beta, float gamma)

```

```

    {
        this->qx = qx;
        this->qy = qy;
        this->qz = qz;
        this->alfa = alfa;
        this->beta = beta;
        this->gamma = gamma;
    }
    virtual p2D^ vetit(p3D^ be)
    {
        return gcnew p2D(-be->x*Math::Cos(deg2rad(alfa)) +
            be->y*Math::Cos(deg2rad(beta)) +
            -be->z*Math::Sin(deg2rad(gamma)),
            -be->x*Math::Sin(deg2rad(alfa)) +
            -be->y*Math::Sin(deg2rad(beta)) +
            be->z*Math::Cos(deg2rad(gamma))
        );
    }
};

// Centrális vetítést megvalósító osztály
ref class Centralis : Vetites
{
public:
    float d, zmin = 2;
    Centralis(float d) :d(d), zmin(2) {};
    Centralis(float d, float zmin) :d(d), zmin(zmin) { if (zmin < 2) zmin = 2; };
    virtual p2D^ vetit(p3D^ be)
    {
        return gcnew p2D(be->x*d / (be->z + zmin), // z eltolása, mert 0 távolságból
            be->y*d / (be->z + zmin)); // végtelen nagy lenne a kép
    }
};

// fok -> rad átváltó
static float deg2rad(float fok)
{
    return fok*Math::PI / 180;
}

// Képernyőre helyezés
ref class WindowViewPort
{
public:
    int x, y, n;
    WindowViewPort(int x, int y, int n) :x(x), y(y), n(n) {};
    PointF wvport(p2D^ vp)
    {
        return PointF(vp->x*n + x, y - vp->y*n);
    }
};

// 3D-s pont levetítése és megjelenítése a képernyőn
void Draw3DLine(Graphics ^gr, Pen ^toll, p3D^ p1_3d, p3D^ p2_3d, Vetites^ v, WindowViewPort
^ w)
{
    p2D^ p1_2d = v->vetit(p1_3d);
    p2D^ p2_2d = v->vetit(p2_3d);
    PointF p1 = w->wvport(p1_2d);
    PointF p2 = w->wvport(p2_2d);
    gr->DrawLine(toll, p1, p2);
}

```

```

// Paint esemény használata a folyamatos újra rajzolás miatt
private: System::Void tabPage2_Paint(System::Object^ sender,
System::Windows::Forms::PaintEventArgs^ e) {

    // Képernyő közepén legyen az origó, 100x nagyítással
    WindowViewport ^ wv = gcnew WindowViewport(tabPage2->Width / 2, tabPage2->Height / 2,
    100);

    // Vetítési módok:
    Axonometrikus ^ izo = gcnew Axonometrikus(1, 1, 1, 30, 30, 0); // izometrikus
    Axonometrikus ^ cavalier = gcnew Axonometrikus(0.5, 1, 1, 45, 0, 0); // cavalier
    Centralis ^cent = gcnew Centralis(2); // centrális
    távolság 2

    // Kiválasztható vetítési módok
    List<Vetites^>^ vetites = gcnew List<Vetites^>();
    vetites->Add(izo);
    vetites->Add(cavalier);
    vetites->Add(cent);

    Graphics ^gr = tabPage2->CreateGraphics(); //grafika átadása
    gr->Clear(this->BackColor); //törlés háttérszínnel

    int a = 1; //oldalhossz
    int mode = comboBox2->SelectedIndex; //kiválasztott mód
    p3D^ p1, ^p2; //egyenessel összekötésre kerülő pontok

    //x tengely
    Pen ^toll = gcnew Pen(Color::Red);
    toll->DashStyle = DashStyle::Dash;
    p1 = gcnew p3D(0, 0, 0); p2 = gcnew p3D(2 * a, 0, 0); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    //y tengely
    toll->Color = Color::Blue;
    p1 = gcnew p3D(0, 0, 0); p2 = gcnew p3D(0, 2 * a, 0); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    //z tengely
    toll->Color = Color::Green;
    p1 = gcnew p3D(0, 0, 0); p2 = gcnew p3D(0, 0, 2 * a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);

    //tengely feliratok
    System::Drawing::Font^ font = gcnew System::Drawing::Font("Arial", 10);
    gr->DrawString("X", font, gcnew SolidBrush(Color::Red),
    wv->wvport(vetites[mode]->vetit(gcnew p3D(2 * a, 0, 0))));
    gr->DrawString("Y", font, gcnew SolidBrush(Color::Blue),
    wv->wvport(vetites[mode]->vetit(gcnew p3D(0, 2 * a, 0))));
    gr->DrawString("Z", font, gcnew SolidBrush(Color::Green),
    wv->wvport(vetites[mode]->vetit(gcnew p3D(0, 0, 2 * a))));

    //téglatest
    toll->Color = Color::Black;
    toll->Width = 2;
    toll->DashStyle = DashStyle::Solid;
    //alsó lap
    p1 = gcnew p3D(a, a, 0); p2 = gcnew p3D(-a, a, 0); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    p1 = gcnew p3D(-a, a, 0); p2 = gcnew p3D(-a, -a, 0); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    p1 = gcnew p3D(-a, -a, 0); p2 = gcnew p3D(a, -a, 0); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    p1 = gcnew p3D(a, -a, 0); p2 = gcnew p3D(a, a, 0); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    //felső lap
    p1 = gcnew p3D(a, a, a); p2 = gcnew p3D(-a, a, a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);

```

```

    p1 = gcnew p3D(-a, a, a); p2 = gcnew p3D(-a, -a, a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    p1 = gcnew p3D(-a, -a, a); p2 = gcnew p3D(a, -a, a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    p1 = gcnew p3D(a, -a, a); p2 = gcnew p3D(a, a, a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    //oldalsó vonalak
    p1 = gcnew p3D(a, a, 0); p2 = gcnew p3D(a, a, a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    p1 = gcnew p3D(-a, a, 0); p2 = gcnew p3D(-a, a, a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    p1 = gcnew p3D(-a, -a, 0); p2 = gcnew p3D(-a, -a, a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    p1 = gcnew p3D(a, -a, 0); p2 = gcnew p3D(a, -a, a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);

    //gúla
    p1 = gcnew p3D(a, a, 0); p2 = gcnew p3D(0, 0, 2 * a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    p1 = gcnew p3D(-a, a, 0); p2 = gcnew p3D(0, 0, 2 * a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    p1 = gcnew p3D(-a, -a, 0); p2 = gcnew p3D(0, 0, 2 * a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
    p1 = gcnew p3D(a, -a, 0); p2 = gcnew p3D(0, 0, 2 * a); Draw3DLine(gr, toll, p1, p2,
    vetites[mode], wv);
}

// Vetítési mód kiválasztása
private: System::Void comboBox2_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e) {
    tabPage2->Refresh();
}

```

