

Grafică pe calculator (MLR5060)

Elemente de grafică 3_D

1. *OpenGL - Noțiuni introductice*
2. Realizarea unei aplicații simple
 - Introducere *Tao.OpenGl* folosind *SimpleOpenGLControl*.

Ce este *SimpleOpenGLControl* ?

SimpleOpenGLControl este un control simplu din OpenGL bazat pe *Windows Forms*.

Scopul său este de a obtine o executie rapida a aplicatiei OpenGL cu *Windows Forms*.

Acesta permite scrierea rapida a unei aplicatii *Windows Forms-based OpenGL* într-un mod simplu.

SimpleOpenGLControl este recomandat pentru incepatorii in OpenGL sau pentru cei care doresc să scrie rapid o *aplicatie* bazată pe OpenGL-*Windows Forms*.

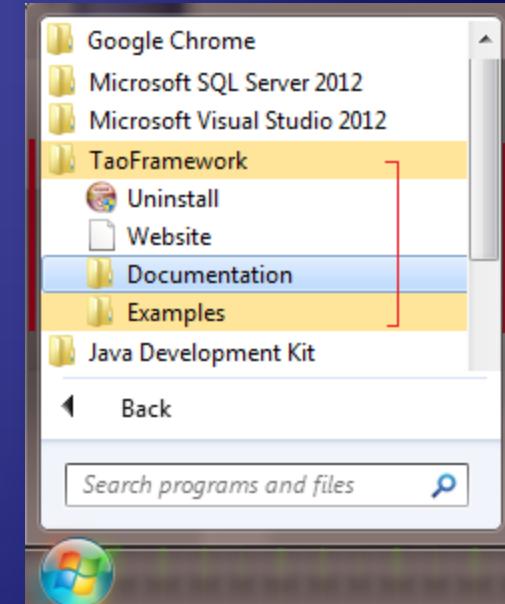
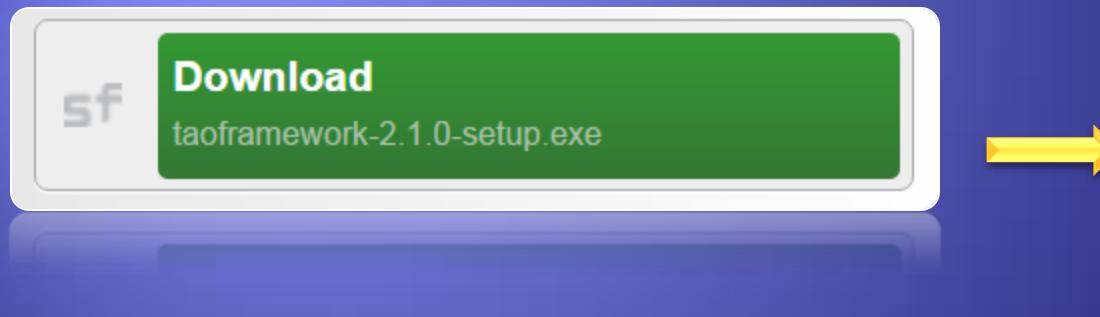
Dacă *SimpleOpenGLControl* nu se potrivește, de obicei este relativ ușor de a utiliza alta metoda pentru funcționalitatea dorita, oferită de *SimpleOpenGLControl*.

Soluție (proiect) cu SimpleOpenGLControl

a) Se instalează *Tao Framework* : http://sourceforge.net/projects/taoframework/?source=typ_redirect

- [Home](#)/[Browse](#)/[Development](#)/[Frameworks](#)/The Tao Framework

http://sourceforge.net/projects/taoframework/files/latest/download?source=typ_redirect



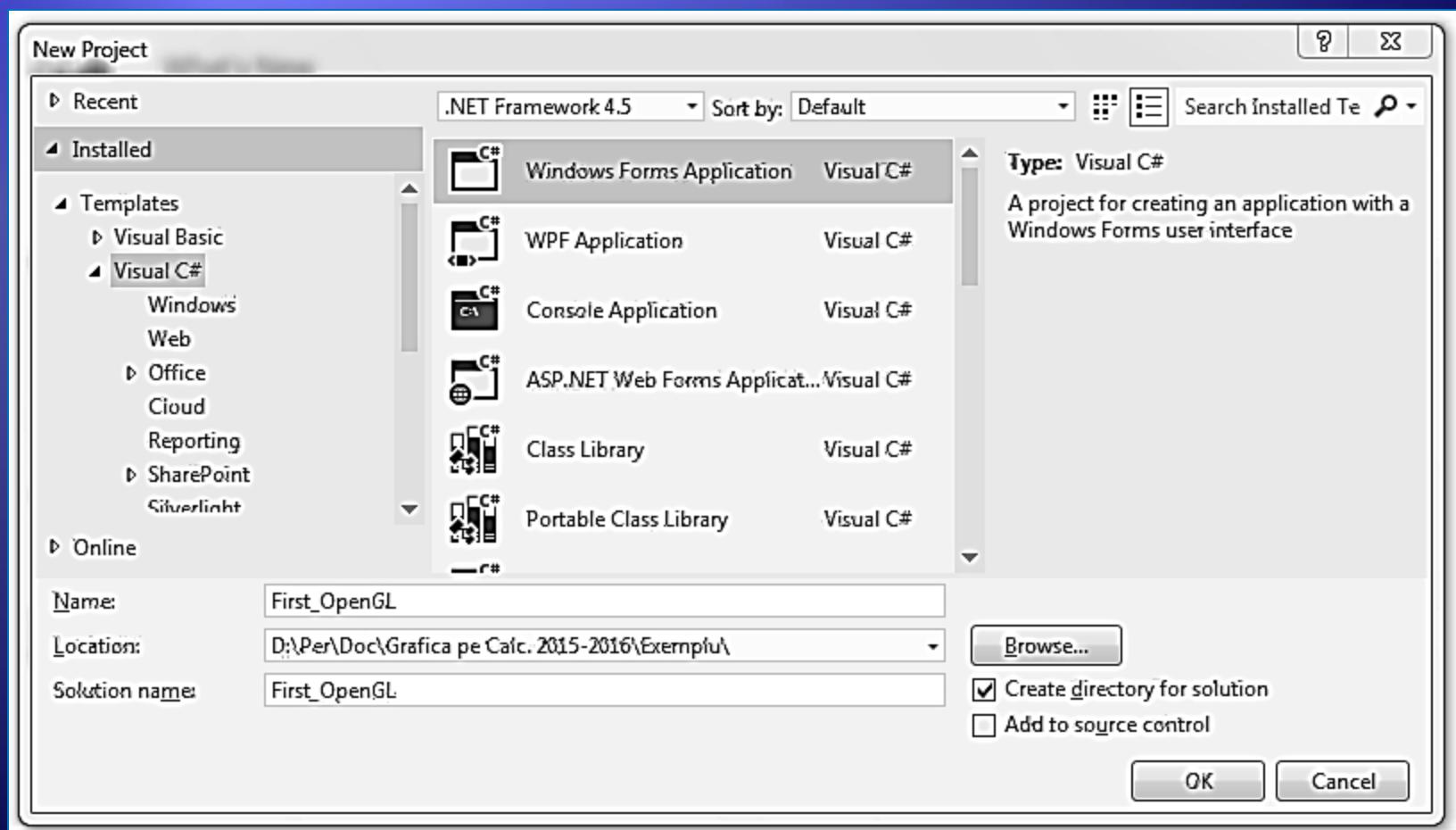
The Tao Framework for .NET is a collection of bindings to facilitate cross-platform game-related development utilizing the .NET platform.

The Open Toolkit is an advanced, cross-platform, C# OpenGL, OpenAL and OpenCL wrapper for Mono/.Net. It is especially suitable to RAD development and can be used in games, GUIs (WinForms, WPF, GTK#) and scientific applications.

... Soluție (proiect) cu *SimpleOpenGLControl*

b) Se va crea un nou proiect (*new Windows application project*):

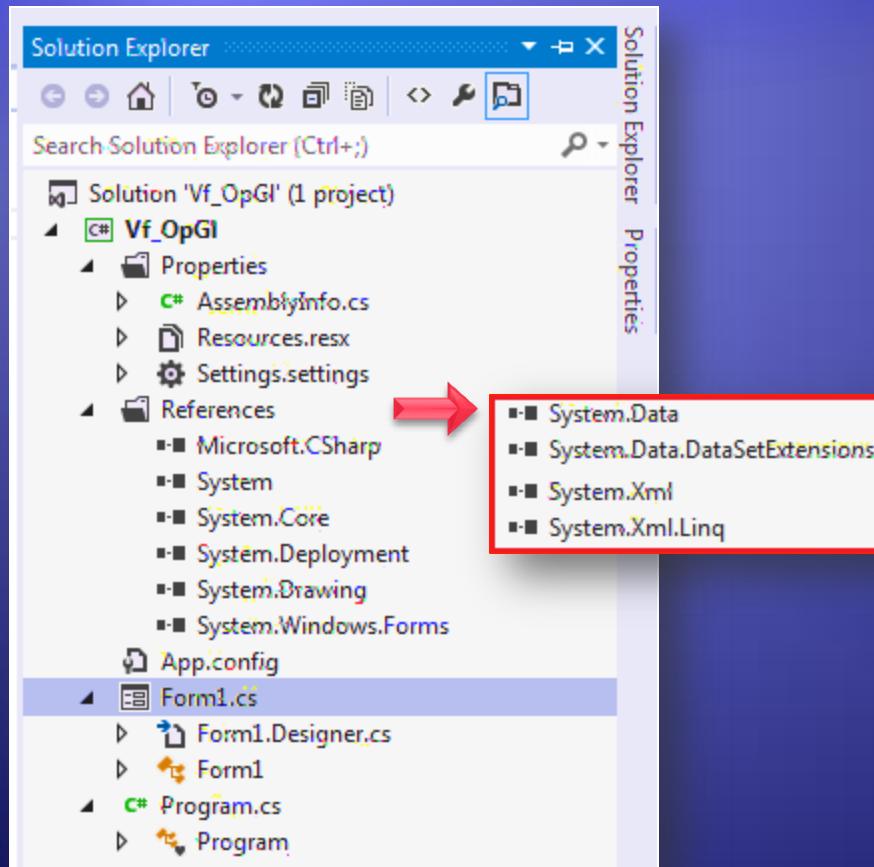
First_OpenGL



... Soluție (proiect) cu *SimpleOpenGLControl*

c) Se sterg din:

- *References*: *System.Data* și *System.XML*,
- *Form1.cs*: *using System.Data;*



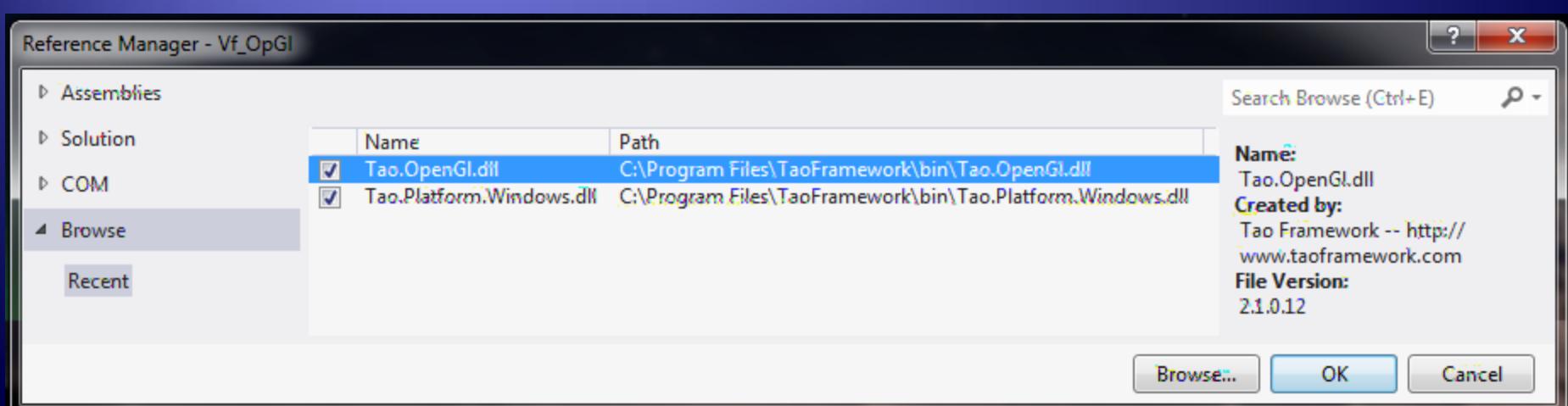
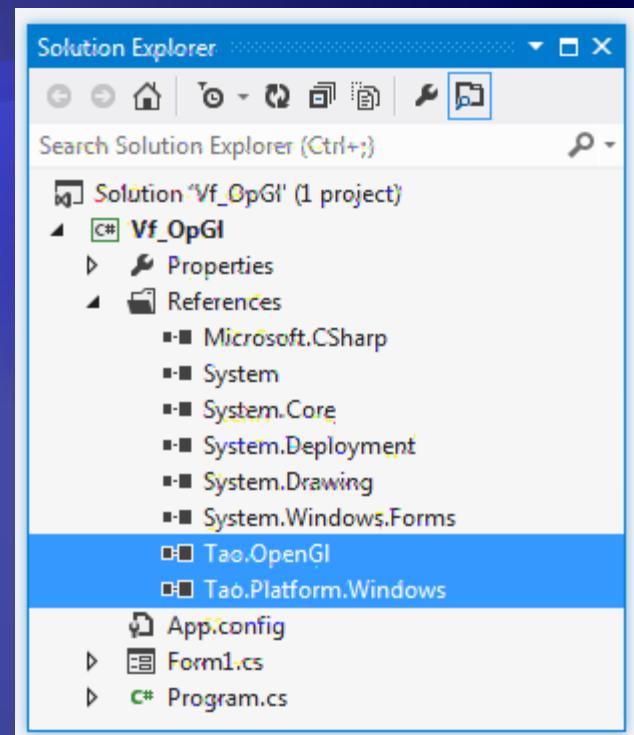
using System;
using System.Collections.Generic;
using System.ComponentModel;
// using System.Data; Sters!
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```
namespace Vf_OpGl
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

d) Se adaugă la *References*

(*Add Reference ...*) : 

- *Tao.OpenGl.dll*,
- *Tao.Platform.Windows.dll*.



e) Se adaugă în *form1.cs* using:

- *Tao.OpenGl.dll* ,
- *Tao.Platform.Windows.dll* .

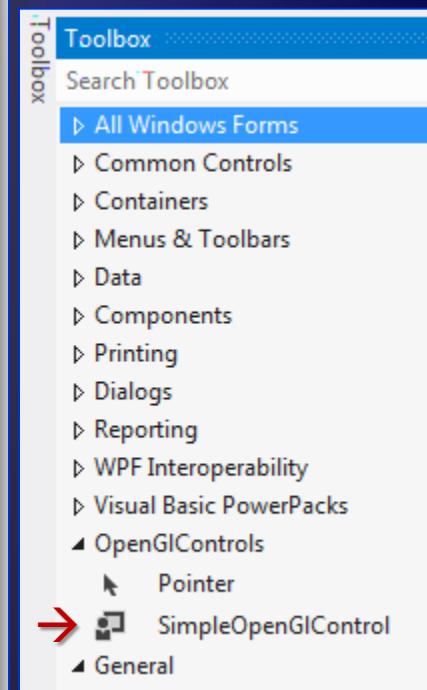
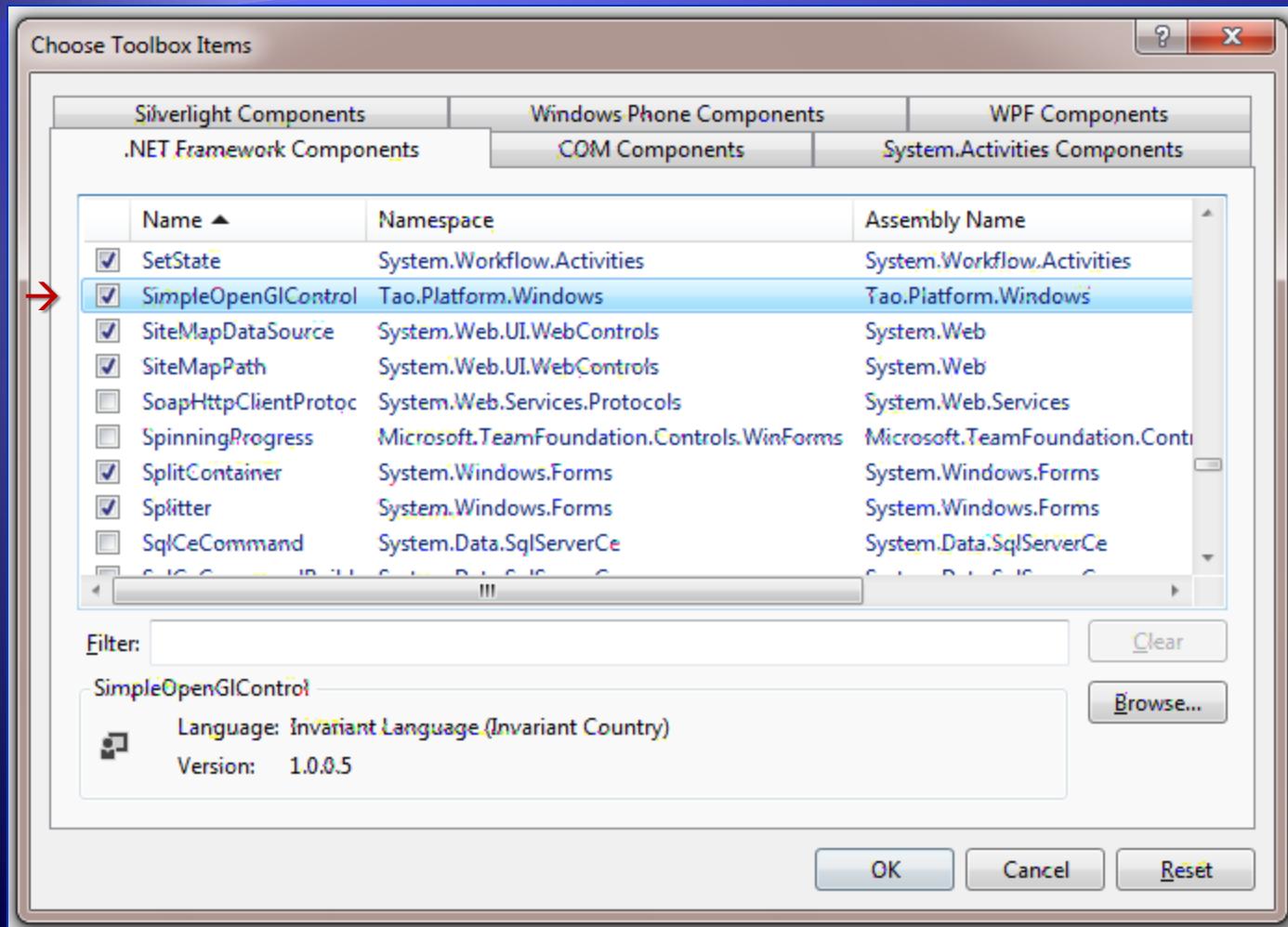


```
using System;
using System.Collections.Generic;
using System.ComponentModel;
// using System.Data; Sters!
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Tao.OpenGl;
using Tao.Platform.Windows;
namespace Vf_OpGl
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

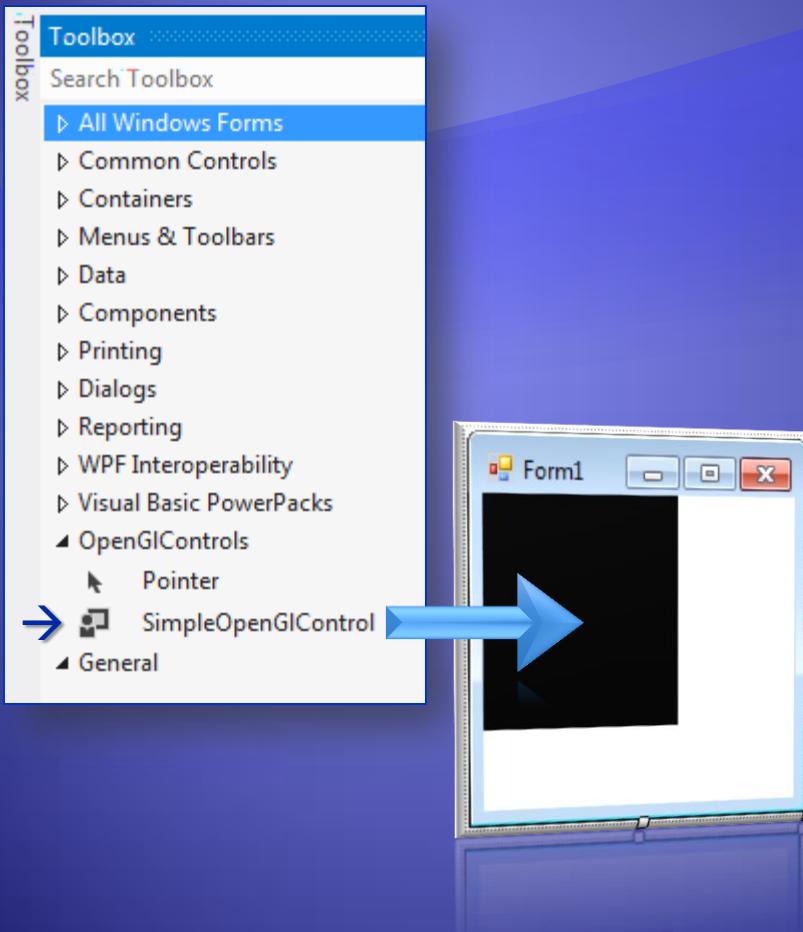
f) Se adaugă în *SimpleOpenGLControl* (*Choose Items ...*) :

➤ [Browse ...](#)

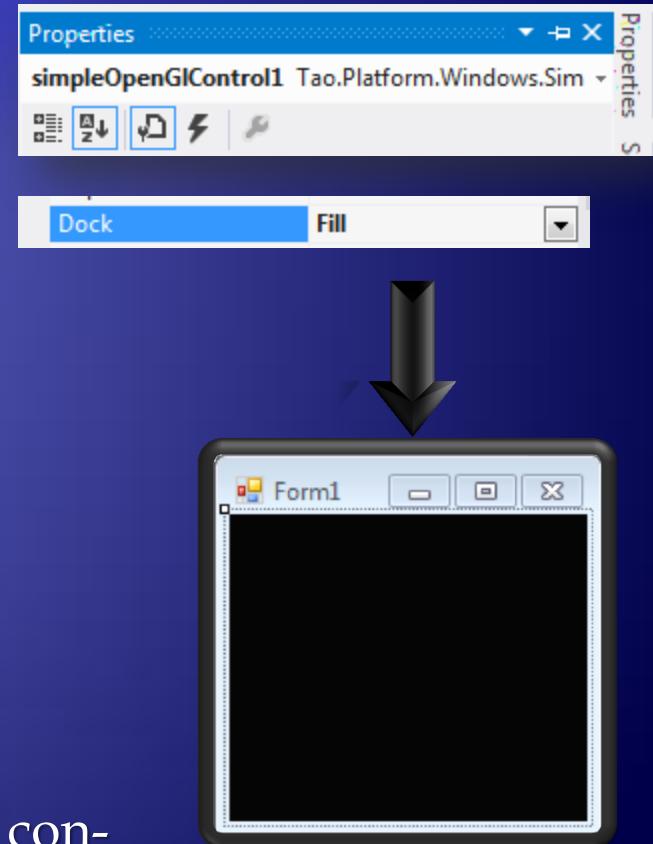
- *Tao.Platform.Windows.dll*



g) Se depune *SimpleOpenGLControl* pe formă:



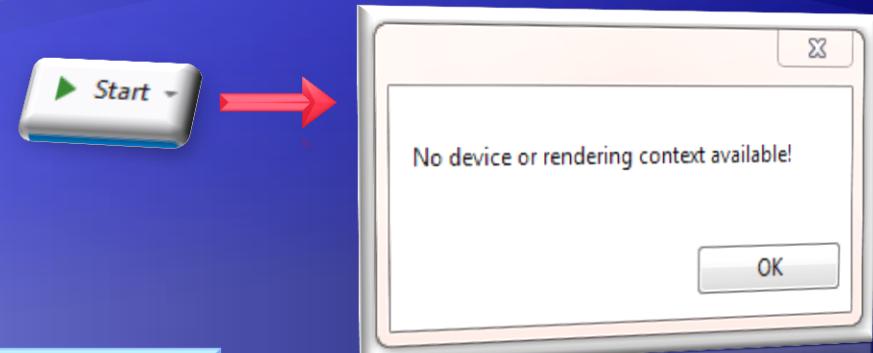
h) *Properties / Dock : Fill*



i) Putem schimba unele proprietăți [*OpenGL*] ale controlului *SimpleOpenGLControl* care controlează diferite valori de *creație*, sau putem pastra valorile implicate.

- j) Dacă rulați aplicația primiți o eroare care să ateste că:
Nici un dispozitiv sau context de randare nu este disponibil !

Acest lucru se datorează faptului că nu am inițializat de fapt contextul *OpenGL*.



```
public Form1()
{
    InitializeComponent();
    simpleOpenGLControl1.InitializeContexts();
}
```

```
protected override void Dispose(bool disposing)
{
    if (disposing && (components != null))
    {
        simpleOpenGLControl1.DestroyContexts();
        components.Dispose();
    }
    base.Dispose(disposing);
}
```

- k) Vom face acest lucru prin adăugarea unui apel la metoda *InitializeContexts* după ce *SimpleOpenGLControl* este instantiat.
- l) vom apela *DestroyContexts* în metoda *Dispose*.

m) Fereastra (neagră) obținută are un *context OpenGL*, o *Fereastră OpenGL Forms app în funcțiune*, deci putem adăuga unele inițializări *OpenGL*:

```
public Form1()
{
    InitializeComponent();
    simpleOpenGLControl1.InitializeContexts();
    Gl.glClearColor(0, 0, 0, 0);
    Gl.glMatrixMode(Gl.GL_PROJECTION);
    Gl.glLoadIdentity();
    Gl.glOrtho(0, 1, 0, 1, -1, 1);
}
```

Inițializări *OpenGL*:

- *glClearColor* - culoare de stergere a ecranului (*Red, Green, Blue, Alpha*), cu valori reale (*float*) in intervalul [0,1].
- *glMatrixMode* precizeaza matricea utilizata de pe stiva de proiecție. Exista stivă *Modelview, de proiecție* (mapează spațiul 3D pe fereastra 2D), *de texturi* și *de culoare*. Modul initial pentru *OpenGL* este modul *Modelview*.

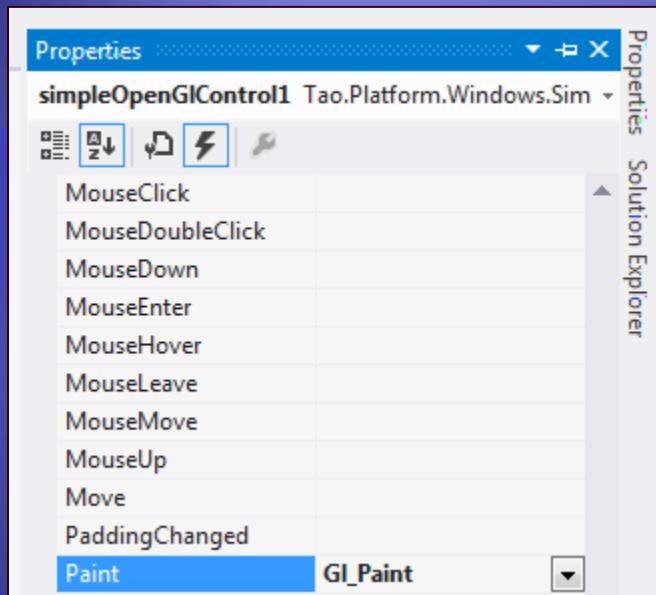
... Inițializări *OpenGL*:

- *glLoadIdentity* – încarcă matricea *unitate* pe stiva de *proiecție*
- *glOrtho* – înmulțeste matricea *curentă* cu matricea *ortogonală*.

```
public Form1() // ...
{
    InitializeComponent();
    simpleOpenGLControl1.InitializeContexts();
    Gl.glClearColor(0, 0, 0, 0);
    Gl.glMatrixMode(Gl.GL_PROJECTION);
    Gl.glLoadIdentity();
    Gl.glOrtho(0, 1, 0, 1, -1, 1);
}
```

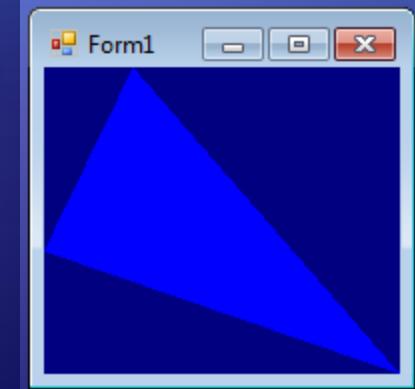
- Am setat valoarea **0** pentru *planul de tăiere stânga* și **1** pentru *planul de tăiere dreapta*.
- Am setat valoarea **0** pentru *planul de tăiere jos* și **1** pentru *planul de tăiere sus*.
- Am setat valoarea **-1** pentru *planul de tăiere spate* și **1** pentru *planul de tăiere față*.

n) Stergem fereastra și desenăm un triunghi, adăugând un eveniment *Paint* la componenta *SimpleOpenGLControl*:



```
private void Gl_Paint(object sender, PaintEventArgs e)
{
    Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);
    Gl.glColor3f(0, 0, 1); // Blue
    Gl.glBegin(Gl.GL_TRIANGLES);
    Gl.glVertex3f(0.25f, 1, 0);
    Gl.glVertex3f(0, 0.4f, 0);
    Gl.glVertex3f(1, 0, 0);
    Gl.glEnd();
    Gl.glFlush();
}
```

```
public Form1()
{
    InitializeComponent();
    simpleOpenGLControl1.InitializeContexts();
    Gl.glClearColor(0, 0, 0.5f, 0); // ~ Blue
    Gl.glMatrixMode(Gl.GL_PROJECTION);
    Gl.glLoadIdentity();
    Gl.glOrtho(0, 1, 0, 1, -1, 1);
}
```



Corp reprezentat prin muchii

Vom arăta funcțiile și elementele folosite pentru a desena un poliedru. Segmentele vor fi desenate cu *GL_LINE_LOOP* precizand capetele acestuia cu *glVertex3d*.

a) Inițializarea spațiului de lucru și a variabilelor care vor controla rotația.

- *simpleOpenGLControl1.InitializeContexts()* inițializează zona de lucru.
- *Gl.glViewport()* ~ Seteaza Viewport,
- *Gl.glMatrixMode()* ~ Precizeaza matricea curenta,
- *Gl.GL_PROJECTION* ~ Defineste proiectia,
- *Gl.GL_MODELVIEW* ~ Defineste matricea de transformari,
- *Glu.gluPerspective* ~ Seteaza proiectia.

public Form1():

```
public Form1()
{
    InitializeComponent();

    int height = simpleOpenGLControl1.Height;
    int width = simpleOpenGLControl1.Width;
    simpleOpenGLControl1.InitializeContexts();
    Gl.glViewport(0, 0, width, height);
    Gl.glMatrixMode(Gl.GL_PROJECTION);
    Gl.glLoadIdentity();
    Glu.gluPerspective(45.0f, (double)width / (double)height, 0.01f, 5000.0f);
}
```

```
double xrot, yrot, zrot = 0;
```

```
    }
```

```
    Ciștigării să se sprijine (xrot) \ (yrot) \ (zrot) \ (width) \ (height) \ (depth) \ (near) \ (far);
```

b) Desenam prin evenimentul *Paint*:

```
private void Gl_Paint(object sender, PaintEventArgs e)
{
    Gl.glClear(Gl.GL_COLOR_BUFFER_BIT | Gl.GL_DEPTH_BUFFER_BIT); //clear buffers to preset values
    Gl.glMatrixMode(Gl.GL_MODELVIEW);
    Gl.glLoadIdentity();           // load the identity matrix
    Gl.glTranslated(0, 0, -4);    //moves our figure (x,y,z)

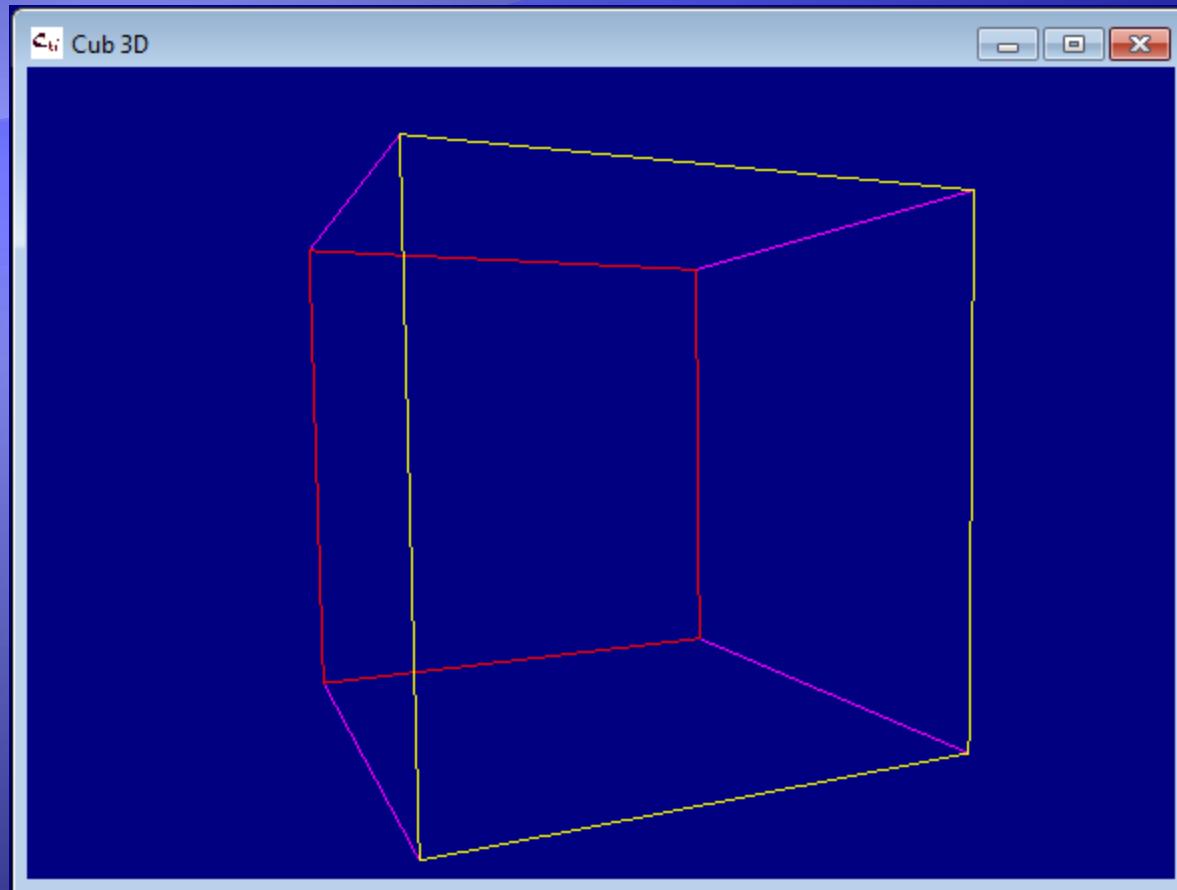
    Gl.glRotated(xrot += 3.25, 1, 0, 0); //rotate on x
    Gl.glRotated(yrot += 23.23, 0, 1, 0); //rotate on y
    Gl.glRotated(zrot += 0.92, 0, 0, 1); //rotate on z

    Gl.glBegin(Gl.GL_LINE_LOOP); //Drawing GL_LINE_LOOP
    Gl glColor4d(255, 0, 255, 100); //Magenta _ Jos
    Gl glVertex3d(-1, -1, -1);
    Gl glVertex3d(1, -1, -1);
    Gl glVertex3d(1, -1, 1);
    Gl glVertex3d(-1, -1, 1);
    Gl glEnd();
    ...
    ...
    Gl glBegin(Gl.GL_LINE_LOOP);
    Gl glColor4d(255, 0, 255, 100); // Red _ Spate
    Gl glVertex3d(1, 1, -1);
    Gl glVertex3d(-1, 1, -1);
    Gl glVertex3d(-1, -1, -1);
    Gl glVertex3d(-1, 1, -1);
    Gl glEnd();
}
```

```
...
Gl glBegin(Gl.GL_LINE_LOOP);
Gl glColor4d(255, 255, 0, 100); // Yellow ~ Fata
Gl glVertex3d(-1, 1, 1);
Gl glVertex3d(-1, -1, 1);
Gl glVertex3d(1, -1, 1);
Gl glVertex3d(1, 1, 1);
Gl glEnd();
```

```
...
Gl glBegin(Gl.GL_LINE_LOOP);
Gl glColor4d(255, 0, 255, 100); // Magenta _ Sus
Gl glVertex3d(-1, 1, -1);
Gl glVertex3d(-1, 1, 1);
Gl glVertex3d(1, 1, 1);
Gl glVertex3d(1, 1, -1);
Gl glEnd();
```

c) Rezultatul obtinut:



Referințe

1. *A gentle introduction to Tao.OpenGl using SimpleOpenGLControl,*
 - http://members.hellug.gr/nkour/Tao.OpenGL_Builder/SimpleIntro_Borland.html
2. *UZIEL GC, Basic Drawings with OpenGL using C#,*
 - <http://www.c-sharpcorner.com/uploadfile/517b0b/basic-drawings-with-opengl-using-C-Sharp/>
3. *Dezvoltarea aplicațiilor OpenGL pe platforma .NET folosind suita TAO,*
 - http://profs.info.uaic.ro/~alaiba/mw/index.php?title=Dezvoltarea_aplica%C5%A3iilor_OpenGL_pe_platforma_.NET_folosind_suita_TAO
4. *Cursuri L_T,*
 - http://www.cs.ubbcluj.ro/~per/Grafica/L_T/L_T.htm
5. *Cursuri, Laboratoare G_A,*
 - <http://www.cs.ubbcluj.ro/~anca/grafica/> , <http://www.cs.ubbcluj.ro/~anca/graphics/>

Temă

Realizarea unei aplicatii simple care sa permita:

- ✓ *vizualizarea unui obiect 3D,*
- ✓ *rotirea si*
- ✓ *miscarea sa.*

Success!