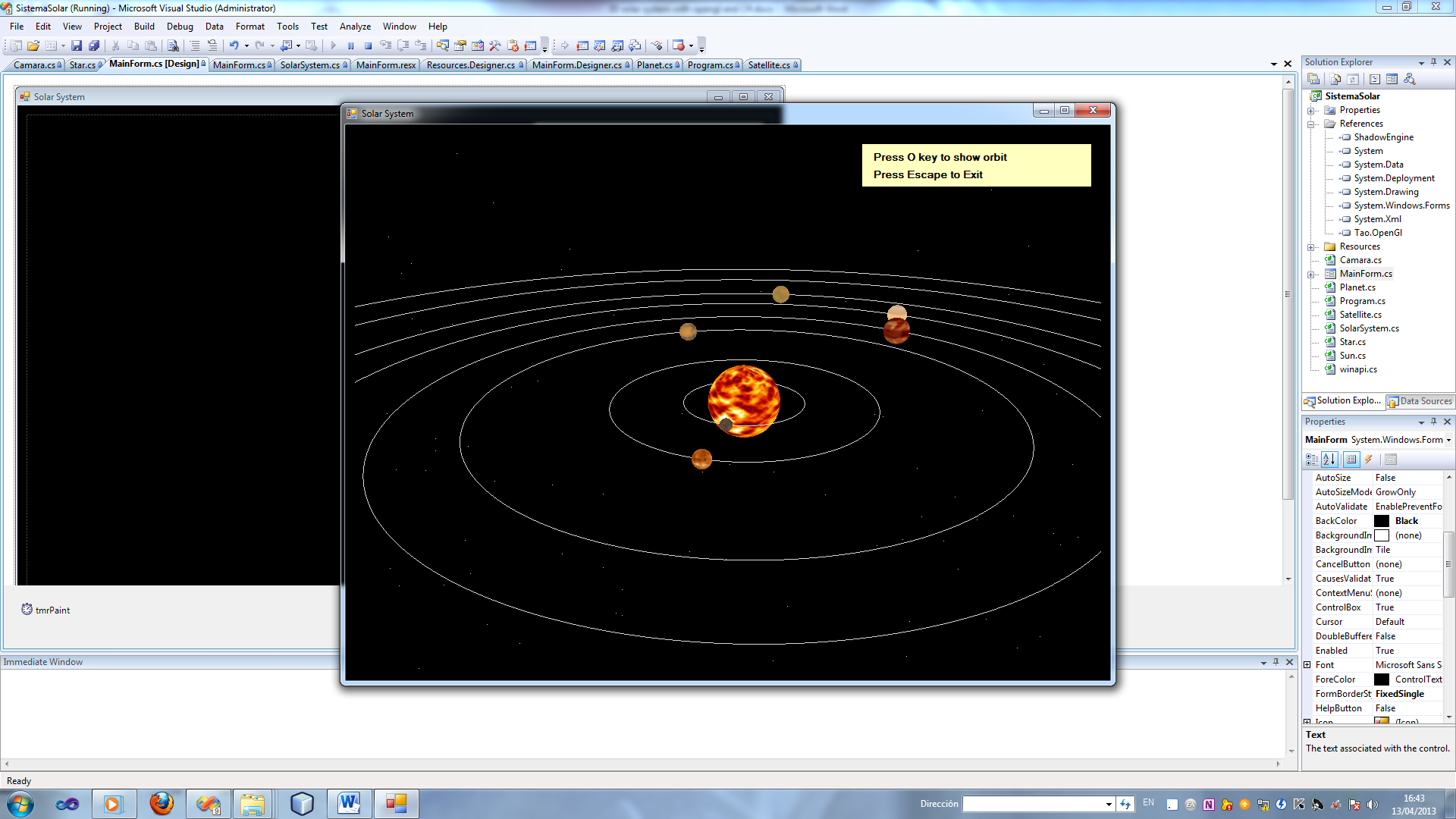
**3D solar system with Opengl and C#**



Hello this is a 3D solar system implementation with OpenGL and C#. I tried to keep it simple because this demo it’s only for educational purposes. It contains the sun, the planets, our moon, the planet’s orbit and some stars. It is programmed in Visual Studio 2008 and I have upgraded it to Visual Studio 2010 without any kind of issue. I used for this demo the TAO namespace which it’s an interop between the opengl dll and the .Net Framework. I also used Shadowengine, a small graphic framework developed by me to get rid you of the tedium of hard coding the loading of textures, the initialization of the graphic context among others. Well let’s dive into the project.

The project contains the following classes which I will explain briefly what they do and how.

|  |  |
| --- | --- |
|  | As you can see in the project, references to ShadowEngine and TAO.OpenGL. I will like to point out that I don’t create a graphic context in a standalone window Like XNA, GLUT, ETC. My graphic context it’s created in a common .NET win form. This is very convenient because you can draw 3D content in any window mixing it with 2D components. Later on you will see that you can draw 3D content in almost any 2D component. The OpenGl initialization function only needs a valid component handler to start drawing 3D. |

**Here is the list of the project classes:**

**Camara.cs**

This is a classic FPS (First Person Shooter) camera. The explanation of how a FPS works goes beyond the scope of this article. They work the following way:

* The mouse it’s centered on the middle of the screen.
* When the user moves the mouse a delta X and Delta Y are calculated from the beginning point.
* Those Delta X and Delta Y are translated into angles and how is how the camera it’s rotated.
* When you wish to move forward or backward the camera will move in the direction that are the angles pointing.
* You may take a look at public void Update(int pressedButton) at the camera class to have a better understanding

**MainForm.cs**

This class name its auto explanatory, it is the main and only form of the project. It contains the call to the texture loading, the 3D context initialization, the drawing of the 3D content, among others. It also handles the user key and mouse input. Because the 3D content requires at least 30 frames per second to be drawn I used a timer and placed all the drawing code inside it.

**Planet.cs**

A planet contains the following variables:

* Position
* Texture
* Orbit(current distance from the sun)
* Current rotation angle
* Current orbit speed

I used OpenGL quadrics to draw the planets sphere. Quadrics are OpenGL predefined shapes to help in small drawing tasks. In each frame the planet moves through its orbit according to its orbit speed. Also there is a bool variable called hasMoon to specify if you want to draw a moon for that planet. I have only our moon but if you like, for example, to draw mars moons Phobos and Deimos you can use that code. Another interesting function that contains the planet class is the one used to draw its orbit. First I generate the points with a sin function and then I connect them using GL\_LINE\_STRIP. Here is the code:

public void DrawOrbit()

{

Gl.glBegin(Gl.GL\_LINE\_STRIP);

for (int i = 0; i < 361; i++)

{

Gl.glVertex3f(p.x \* (float)Math.Sin(i \* Math.PI / 180),

0, p.x \* (float)Math.Cos(i \* Math.PI / 180));

}

Gl.glEnd();

}

**Satellite.cs**

A satellite contains everything that a planet does. The only difference is that it’s rotation point it’s not the sun but the planet that contains it. So anytime it draws it has to receive the position of its containing planet.

**SolarSystem.cs**

This is the class that contains the list of planets, stars and satellites. It only create and draw them.

**Star.cs**

This is the class the draws the stars. The stars are single GL\_POINTS which are generated in random positions. This is the function that generates them:

public void CreateStars(int amount)

{

Random r = new Random();

int count = 0;

while (count != amount)

{

Position p = default(Position);

p.x = (r.Next(110)) \* (float)Math.Pow(-1, r.Next());

p.z = (r.Next(110)) \* (float)Math.Pow(-1, r.Next());

p.y = (r.Next(110)) \* (float)Math.Pow(-1, r.Next());

if (Math.Pow(Math.Pow(p.x, 2) + Math.Pow(p.y, 2) +

Math.Pow(p.z, 2), 1 / 3f) > 15)

{

stars.Add(p);

count++;

}

}

}

**Sun.cs**

The sun class is the most simple it’s like the planet class only it has no orbit. It has only a rotation around its axis.

**Last thoughts**

Well, these are all the classes involving this project I hope it is useful and that will encourage developers to start in 3D programming. Feel yourself free to play with the code and to ask any question you want. If you like this demo you can visit my personal dev blog at <http://vasilydev.blogspot.com>.