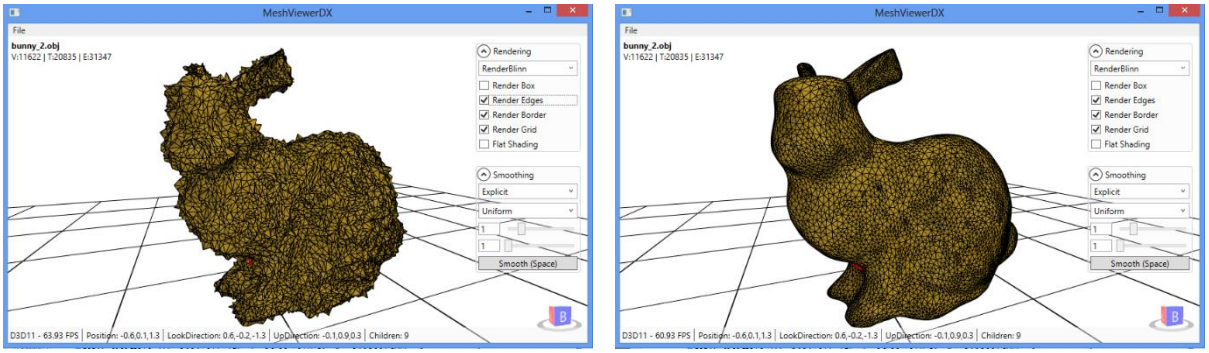
# ASSIGNMENT # 1:
# MESH LAPLACIAN AND MESH SMOOTHING




## 1. ENVIRONMENT

The provided framework is implemented in C# 4.0 and uses HelixToolkit-SharpDX library for rendering with DirectX 11 API.

The application is self-contained. You should be able to build it with Visual Studio 2012/2013. If you don't have the 2012/2013 version, please download it from MSDNAA. You also need an internet connection, since on the first build it will download additional libraries from NuGet (http://www.nuget.org/). In some cases you will have to activate automatic NuGet updates in Visual Studio 2012:

TOOLS->OPTIONS->Package Manager->Check: Allow NuGet to download missing packages during build.

You also might need to update NuGet to its newest version:

TOOLS->Extensions and Updates->Updates->Visual Studio Gallery->NuGet Package Manager: Update

Furthermore, the application uses WPF and the MVVM pattern. While you do not need this knowledge necessarily, you might look it up:

http://msdn.microsoft.com/en-us//library/ms754130.aspx
http://msdn.microsoft.com/en-us/magazine/dd419663.aspx
http://www.codeproject.com/Articles/36545/WPF-MVVM-Model-View-View-Model-Simplified
http://en.wikipedia.org/wiki/Model_View_ViewModel
http://msdn.microsoft.com/en-us/library/gg405484(v=pandp.40).aspx

In the case you get missing DirectX-DLL errors on execute, please install the DirectX End-User Runtimes (June 2010) from: http://www.microsoft.com/en-us/download/details.aspx?id=8109

## 2. COMMUNICATION & HONOR CODE

The communication runs over the TUWEL forum. **Christian Hafner** is the teaching assistant (Tutor). In case you have questions or problems regarding the assignment, please:

- read the forum and check if your question has already been posted, and
- post your question in case it has not been handled yet.

### HONOR CODE: A SET OF ETHICAL PRINCIPLES GOVERNING THIS COURSE

- **It is okay** to share information and knowledge with your colleagues, but
- **it is not okay to share the code**,
- **it is not okay to post or give out your code** to others (also in the future!),
- **it is not okay to use code** from others (also from the past) for this assignment!

## 3. DEADLINE & SUBMISSION

### THE SUBMISSION DEADLINE IS ON **19.11.2014, 23:55.**

By this date you should:

- implement as many tasks as possible,
- write a **2-3 pages documentation**, that includes **which tasks you implemented**, and a brief discussion and conclusions,
- zip your solution folder,
  please delete ./bin/, ./obj/ directories and all packeges in ./packages/ folder
  **Do not delete** repositories.config in the ./packages/ folder!
- submit your ZIP on TUWEL before 23:55.

### NAMING CONVENTION

- The documentation file should be named (0000000 indicates your student-number)
  **0000000-lastname-firstname-A1.pdf**
- Rename the solution file to
  **0000000-lastname-firstname-A1.sln**
- The whole archive should be named:
  **0000000-lastname-firstname-A1.zip**

### LATE POLICY

- You will receive a **-20% penalty** of final grading for **every day you are late**.
- **Max** late submission is **4 days** (-80%, you still qualify to pass the course).
- If your program **does not compile**, or **does not run** you will receive a penalty **(-10%)** and you will have to fix the solution (if we can fix the solution there will be only 5% penalty).

## 4. TASKS

1. Implement **uniform Laplacian** matrix (5 points)
2. Implement basic **explicit Euler (Taubin)** smoothing with uniform Laplacian (see Taubin 2000) (7)
3. Implement **cotangents Laplacian** matrix (see Meyer et al. 2002) (13)
4. Implement **implicit Euler** smoothing (see Desbrun et al. 1999) (10)
5. ~~Implement **least-squares optimization** smoothing (see Nealen et al. 2006)~~ SOLVED!
6. Implement **triangle quality optimization** (see Nealen et al. 2006) (10)
7. Implement **edge-preserving smoothing** in the least-squares framework (see Nealen et al. 2006) (10)
8. Compute and visualize **mean curvature** and **Gaussian curvature** (10)
9. Compare and discuss results of **two chosen methods** (5)
10. Compare and discuss results of **uniform and cotangent Laplacians** on one method (5)

Search the code for `// TODO_A1` in order to find potential tasks.

## DOCUMENTATION & CODE COMMENTS

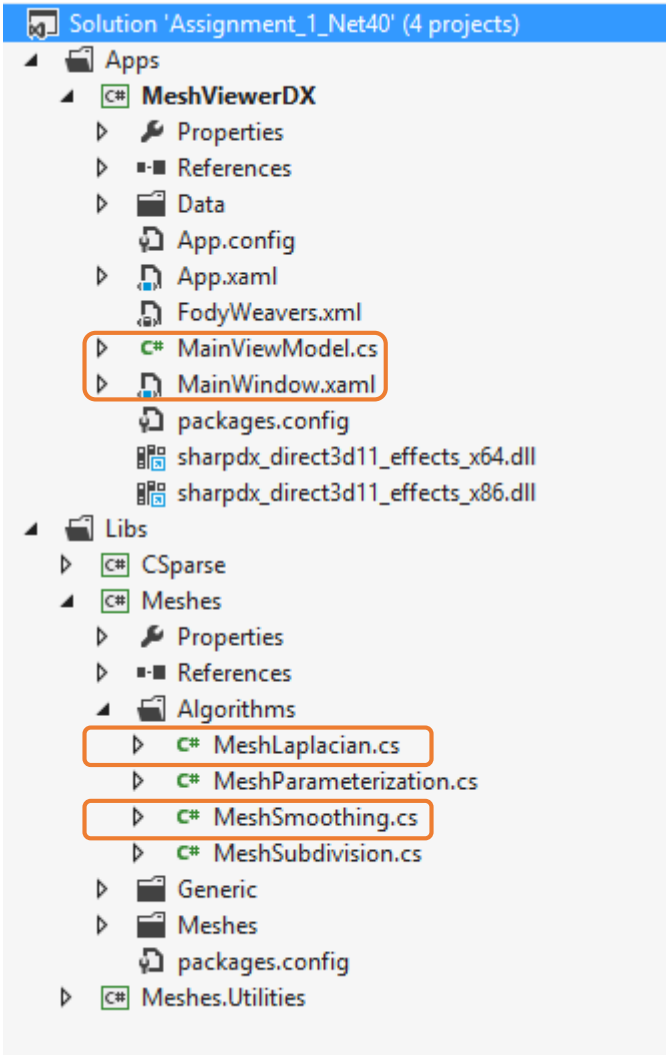
Please include a **2-3 pages PDF** report describing **which tasks you implemented**, as well as a brief **discussion** and your **conclusions**. Also add as **many comments in code** as possible---it will help us judging your work. The **first section** of the documentation should look like this:

- Task 1: Done
- Task 2: Partially Done,
- Task 3: not approached, etc.

## GRADING

To pass this course you will need in each assignment **more than 0 points** (so do not submit empty solutions). Depending on the delivered quality and quantity, you might **earn up to 150%**. If you get more than 50 points, you can compensate lower percentages in other assignments!

- 01-19 points: 5
- 20-29 points: 4
- 30-39 points: 3
- 40-49 points: 2
- 50-75 points: 1
- More than 50 points: points accumulation up to 75 points




## 5. HINTS

### IMPORTANT FILES

Files you should know:

You might need to add something to the GUI, so check out the MVVM pattern and the files in the **MeshViewerDX** project

The main task is to work with the mesh. For this purpose, check out the file MeshLaplacians.cs and MeshSmoothing.cs.

### LAPLACIAN MATRICES:

For tasks with the least-squares optimization framework, you will need augmented matrices. Since the sparse library (CSparse.NET) does not support matrix concatenation, it is recommended to implement two functions for each Laplacian, where one generates n x n, and the other 2*n x n matrices:

```
        /// <summary>
        /// Creates a square uniform n x n laplacian matrix
        /// eye*[ I ] - lambda*[ L ];
        /// Note that depending on lambda you can use this function to create both:
        /// [I - λL]
        /// [I + λL]
        /// If the normalized flag is true, each row of L is normalized to sum to 1
        /// </summary>
        public static SparseMatrix CreateUniformLaplacian(TriangleMesh mesh, double lambda = 0.0, double eye = 0.0, bool normalized =
false)



        /// <summary>
        /// Creates an extended uniform n x n laplacian matrix extended by a n x n weight-matrix:
        ///       [ L ]
```

```
/// lambda*[ I ]
/// If the normalized flag is true, each row of L is normalized to sum to 1
/// </summary>
public static SparseMatrix CreateExtendedUniformLaplacian(TriangleMesh mesh, double lambda = 1.0, bool normalized = true)
```

This applies of course also to the Cotan-Laplacian. You are also free to create further functions and helper procedures, this is up to you!

## HOW TO USE THE SOLVER

This code shows how to use the CSparse.NET solver:

```
/// <summary>
/// Performs global least-squares mesh smoothing with the extended matrix
/// [ L  ] [x'] =      [ 0 ]
/// [ Wp ]       [ Wp ][ x ]
/// </summary>
private void GlobalLeastSquaresSmoothing(TriangleMesh mesh)
{
    /// output vertex positions (after solving/smoothing)
    double[] xx, xy, xz;
    xx = xy = xz = null;

    /// get current world positions
    double[] px, py, pz;
    GetEuclideanCoordinates(mesh, out px, out py, out pz);

    /// get n = number of vertices
    int n = mesh.Vertices.Count;

    /// zero vector
    var zero = new double[n];

    /// extended px vector  (LS-meshes, sorkine 2004)
    /// it is vertically stacked block vector:
    /// [ 0 ]
    /// [ p ]
    /// 0 is the zero vector
    double[] apx = zero.Concat(px.Select(x => x * this.lambda)).ToArray();
    double[] apy = zero.Concat(py.Select(x => x * this.lambda)).ToArray();
    double[] apz = zero.Concat(pz.Select(x => x * this.lambda)).ToArray();

    if (!this.hasMatrix)
    {
        /// create Laplacian
        /// the matrix is created as extended column block matrix:
        /// [ L ]
        /// [ W ]
        /// W is the weight-matrix
        switch (this.SelectedLaplacian)
        {
            default:
            case LaplacianType.Uniform:
                this.L = MeshLaplacian.CreateExtendedUniformLaplacian(mesh, this.Lambda, this.Normalize);
```

```
                    break;
                case LaplacianType.Cotan:
                    this.L = MeshLaplacian.CreateExtendedCotanLaplacian(mesh, this.Lambda, this.Normalize);
                    break;
            }

            /// factorize L
            this.qr = QR.Create(order, L);
            this.hasMatrix = true;
        }

        /// Copy b to x
        xx = apx.Select(x => x).ToArray();
        xy = apy.Select(x => x).ToArray();
        xz = apz.Select(x => x).ToArray();

        /// solve Ax = b
        qr.Solve(xx);
        qr.Solve(xy);
        qr.Solve(xz);

        /// update mesh
        UpdateMesh(mesh, xx, xy, xz);
    }
```

GOOD LUCK!

## 6. REFERENCES

1. DirectX 11 Reference:
   http://msdn.microsoft.com/en-us/library/windows/desktop/ff476080(v=vs.85).aspx
2. DirectX 11 SDK:
   http://www.microsoft.com/en-us/download/details.aspx?id=6812
3. DirectX 11Effect framework:
   http://msdn.microsoft.com/en-us/library/windows/desktop/ff476136(v=vs.85).aspx
4. WPF:
   http://msdn.microsoft.com/en-us//library/ms754130.aspx
5. MVVM Introduction:
   http://msdn.microsoft.com/en-us/library/gg405484(v=pandp.40).aspx
   http://msdn.microsoft.com/en-us/magazine/dd419663.aspx
6. SharpDX:
   http://sharpdx.org/
7. DirectX End-User Runtimes (June 2010)
   http://www.microsoft.com/en-us/download/details.aspx?id=8109