METU **Department of Computer Engineering** **Spring 2025**

CENG 589—Digital Geometry Processing

ASSIGNMENT I March 17, 2025
Instructor: Y. Sahillioğlu 2 Weeks

Distances and Sampling/Patching on Meshes (start early!)

3D Graphics Programming Setup {*0 points*}
You can use any 3D SDK in this and all other assignments. For OpenGL, OpenMesh, OSG, libigl, and VTK setup, see http://youtu.be/D9B_cxUB_LU, http://www.openmesh.org, https://youtu.be/1l5PAVCj2iY, http://libigl.github.io/libigl, and https://youtu.be/IgvbhyDh8r0, respectively. I prefer an SDK called Coin3D Open Inventor which is built on top of OpenGL. It comes with a nice viewer that provides a trackball navigation system as well as different rendering modes[1] so that you do not have to implement these features using OpenGL's primitive GLUT or something. Here is how you can set it up on Windows:

1) Download Coin3D, an independent implementation of API: ceng.metu.edu.tr/∼ys/Coin3D.zip

2) Unzip it to, say, C:\Coin3D

3) Visual Studio → New Project → Visual C++ → Win32 Console Application → Application Settings → Empty Project

4) Place your code file, e.g., ∼ys/HelloWorld.cpp, under Source Files

5) Right click on your Project Name and then hit Properties → C/C++ → General → Additional Include Directories: write C:\Coin3D\include;

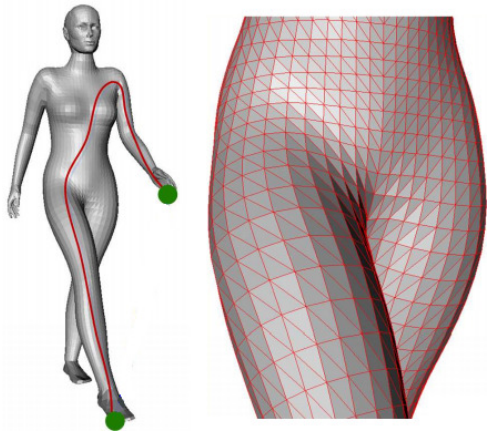
6) C/C++ → Preprocessor: add COIN_DLL;SOWIN_DLL;

7) Linker → General → Additional Library Directories: write C:\Coin3D\lib;

8) Linker → Input → Additional Dependencies: add coin2d.lib;sowin1d.lib;

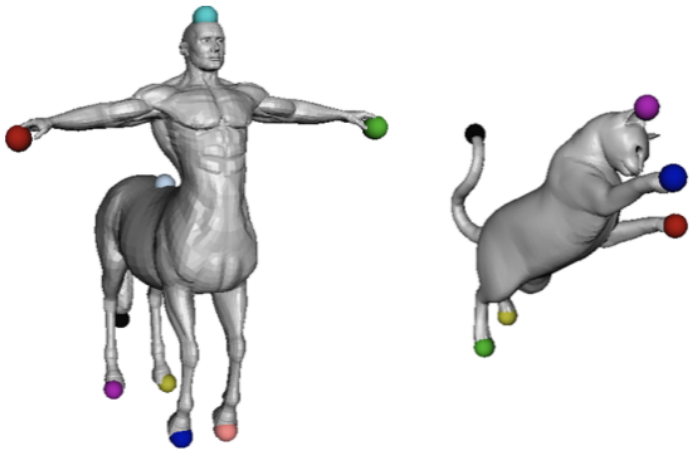
9) Paste Coin3D\coin2d.dll and sowin1d.dll files into your-work-folder\Debug folder (google other missing files reported when you try to execute the program, if any, and put them in this folder too, e.g., MSVCR71D.dll)

Geodesics on Meshes - Dijkstra{*30 points*} Geodesic distance between two mesh vertices is the length of the shortest path along the surface that connects the two, as shown below. Since surface is discretized as an undirected graph, all you have to do is implement the Dijkstra's Shortest Path algorithm discussed in the class. Print the $N \times N$ geodesic distance matrix M to file for the N-vertex input mesh. Visualize the path between two query points, e.g., by redrawing the edges on the path with thicker red lines as below. Report timings with array, min heap, and optionally Fibonacci heap (disable fprints).

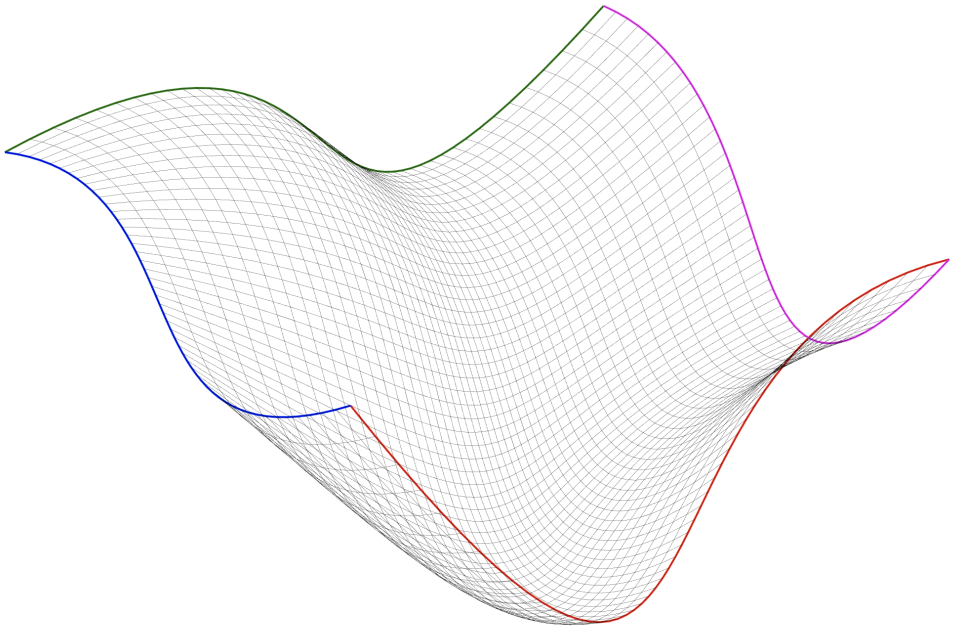


[1] See my short demo at http://youtu.be/lK7aoc1AO8w

Sampling{*20 points*} Implement Farthest Point Sampling (FPS) as discussed in class (see lecture notes).



Patching{*50 points*} Compute the patch at the intersection of 4 paths/geodesics like the one shown below. Note that your result will not be that smooth since you will be using linear interpolation for the filling. Nevertheless, you will get an interesting shape abstraction. To proceed, you need to process each quadruple of paths/geodesics that run between your FPS samples. Discard the quadruple if it does not produce 4 intersection points.



Submission This assignment constitutes 20% of your final grade. Use the meshes provided in ~ys/geomesh.zip for parts 1 and 2, and https://segeval.cs.princeton.edu for part 3. Good luck. Send to ys@ceng.metu.edu.tr your code, executable, and resulting files as well as screenshots. In a text file, mention the encountered problems and interesting observations.