Introduction to GEL

GEometry and **L**inear algebra

…

Graphics Elements Library

- Overview: What does GEL contain?
- In depth:
  - CGLA
  - HMesh
- Todays exercise: Compute the dual of a triangle mesh.

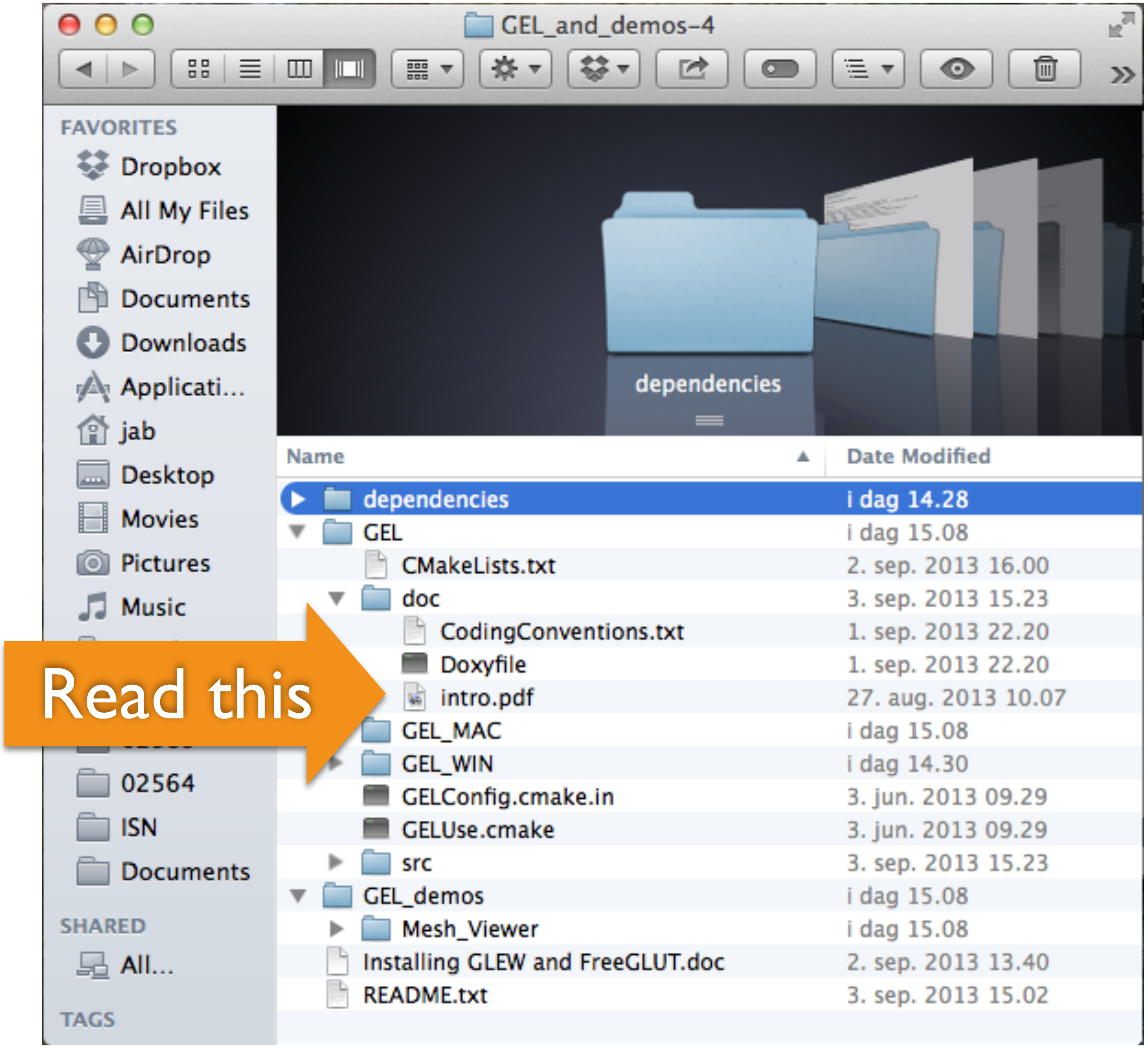
What GEL contains

- C++ libraries:
  - CGLA: Linear algebra for small matrices
  - HMesh: A mesh class
  - Geometry: Many things related to geometry.
  - GLGraphics: Visualization tools
  - Util: What did not fit elsewhere.
- GEL Contains no linear solver, we use Eigen

How to use GEL

- A very brief overview of GEL follows
- Even if something is not mentioned, it might still be there
- Peruse documentation!

GEL Directory Structure



Name	Date Modified
▶ dependencies	i dag 14.28
▼ GEL	i dag 15.08
CMakeLists.txt	2. sep. 2013 16.00
▼ doc	3. sep. 2013 15.23
CodingConventions.txt	1. sep. 2013 22.20
Doxyfile	1. sep. 2013 22.20
intro.pdf	27. aug. 2013 10.07
GEL_MAC	i dag 15.08
GEL_WIN	i dag 14.30
GELConfig.cmake.in	3. jun. 2013 09.29
GELUse.cmake	3. jun. 2013 09.29
▶ src	3. sep. 2013 15.23
▼ GEL_demos	i dag 15.08
▶ Mesh_Viewer	i dag 15.08
Installing GLEW and FreeGLUT.doc	2. sep. 2013 13.40
README.txt	3. sep. 2013 15.02

CGLA

CGLA headers and namespace

```
#include <iostream> // For input output
#include <CGLA/Vec3f.h>
#include <CGLA/Mat4x4f.h>

using namespace std; // For input output
using namespace CGLA;
```

Constructors

```
Vec3f p0(10,10,10);
Vec3f p1(20,10,10);
Vec3f p2(10,20,10);
Vec3f p(1);
Vec3f q;
```

Functions

- You will find functions for dot, cross, normalize etc. e.g:

  ```
  Vec3f n = normalize(cross(p1-p0, p2-p0));
  ```

- Assignment operators

  ```
   p += x;
  ```

- Input and output

  ```
  cout << n << endl;
  ```

Index Operators

```
float x = n[0];
Vec4f v4 = m[0];
float c = m[0][3];
```

Matrices

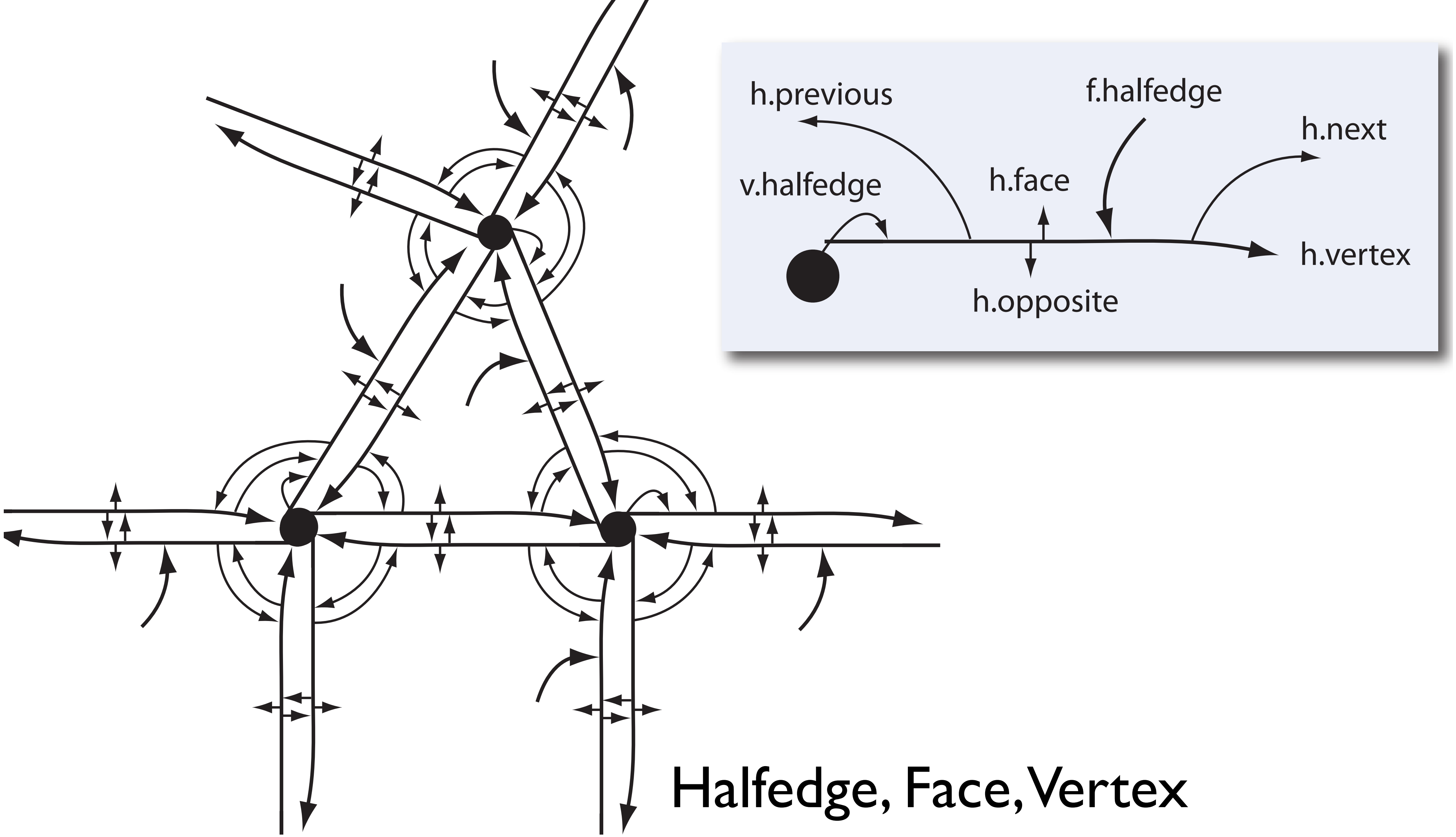
```
Mat4x4f m =
     translation_Mat4x4f(Vec3f(1,2,3));
   m *= q.get_mat4x4f();
   Vec3f p2 = m.mul_3D_point(p);
```

HMesh

```
#include <HMesh/Manifold.h>
using namespace HMesh;

//...

Manifold m;

vector<Vec3f> vertices;
vector<int> faces;
vector<int> indices;

// Fill vectors and then:
m.clear();
m.build(vertices.size(), (float*)&vertices[0],
faces.size(), &faces[0], &indices[0]);
```

Incremental building

```
for(int k=0;k<N;++k)
{
   vector<Vec3d> pts;
   Vec3i f = faces[k];
   for(int i=0;i<3;++i)
      pts.push_back(vertices[f[i]]);
   m.add_face(pts);

   stitch_mesh(m, 1e-30);
}
```



Halfedge, Face, Vertex

HalfEdgeID, FaceID, VertexID

Walker

~~**HalfedgeIDIterator ...**~~

HMesh Example I

```
int n=1;
VertexAttributeVector<int> vidx;
for(VertexID v: m.vertices())
{
    cout << "v " << m.pos(v) << endl;
    vidx[v] = n++;
}
for(FaceID f: m.faces())
{
    cout << "f ";
    circulate_face_ccw(m, f, [&](VertexID vn){
        cout << vidx[vn] << " ";
    });
    cout << endl;
}
```

HMesh Example 1

```
int n=1;
VertexAttributeVector<int> vidx;
for(VertexID v: m.vertices())
{
    cout << "v " << m.pos(v) << endl;
    vidx[v] = n++;
}
for(FaceID f: m.faces())
{
    cout << "f ";
    Walker w = m.walker(f);
    while(!w.full_circle()) {
        cout << vidx[w.vertex()] << " ";
        w = w.next();
    }
    cout << endl;
}
```

Less scary?

HMesh Example 2

```
VertexAttributeVector<Vec3d> npos(m.no_vertices(),
                                  Vec3d(0));
for(VertexID v: m.vertices())
  npos[v] /= circulate_vertex_ccw(m, v, [&](VertexID vn)
    {
        npos[v] += m.pos(vn);
    });
m.positions_attribute_vector() = npos;
```

HMesh Example 2

```
VertexAttributeVector<Vec3d> npos(m.no_vertices(),
                                  Vec3d(0));
for(VertexID v: m.vertices()) {
  Walker w = m.walker(v);
  while(!w.full_circle()) {
    npos[v] += m.pos(w.vertex());
    w = w.circulate_vertex_ccw();
  }
  npos[v] /= w.no_steps();
}
for(VertexID v: m.vertices())
  m.pos(v) = npos[v];
```

Less scary?

Eigen

Eigen is not a part of GEL - it is a separate linear algebra library

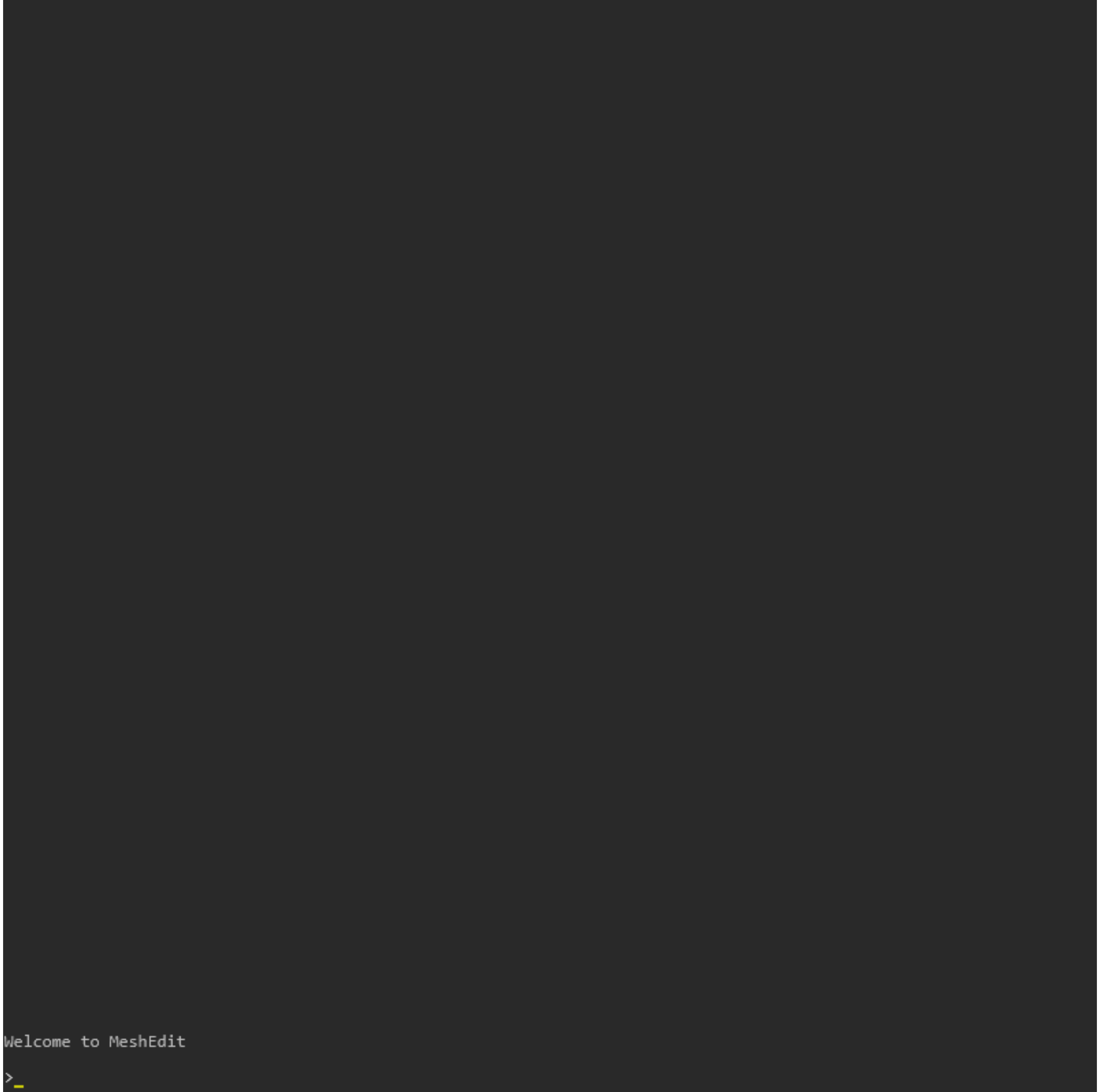
Eigen Example

```
#include <Eigen/Sparse>

int main(int argc, char** argv)
{
    using namespace Eigen;
    using EigMat = SparseMatrix<double>;
    using EigVec = VectorXd;

    EigMat A(6,6);
    for(int i=0;i<6;++i)
        A.insert(i, i) = 1.0/i;


    SimplicialLLT<EigMat> solver(A);

    EigVec b(6);
    b << 1,2,3,4,5,6;
    EigVec X = solver.solve(b);

    cout << X << endl;

    // …
```

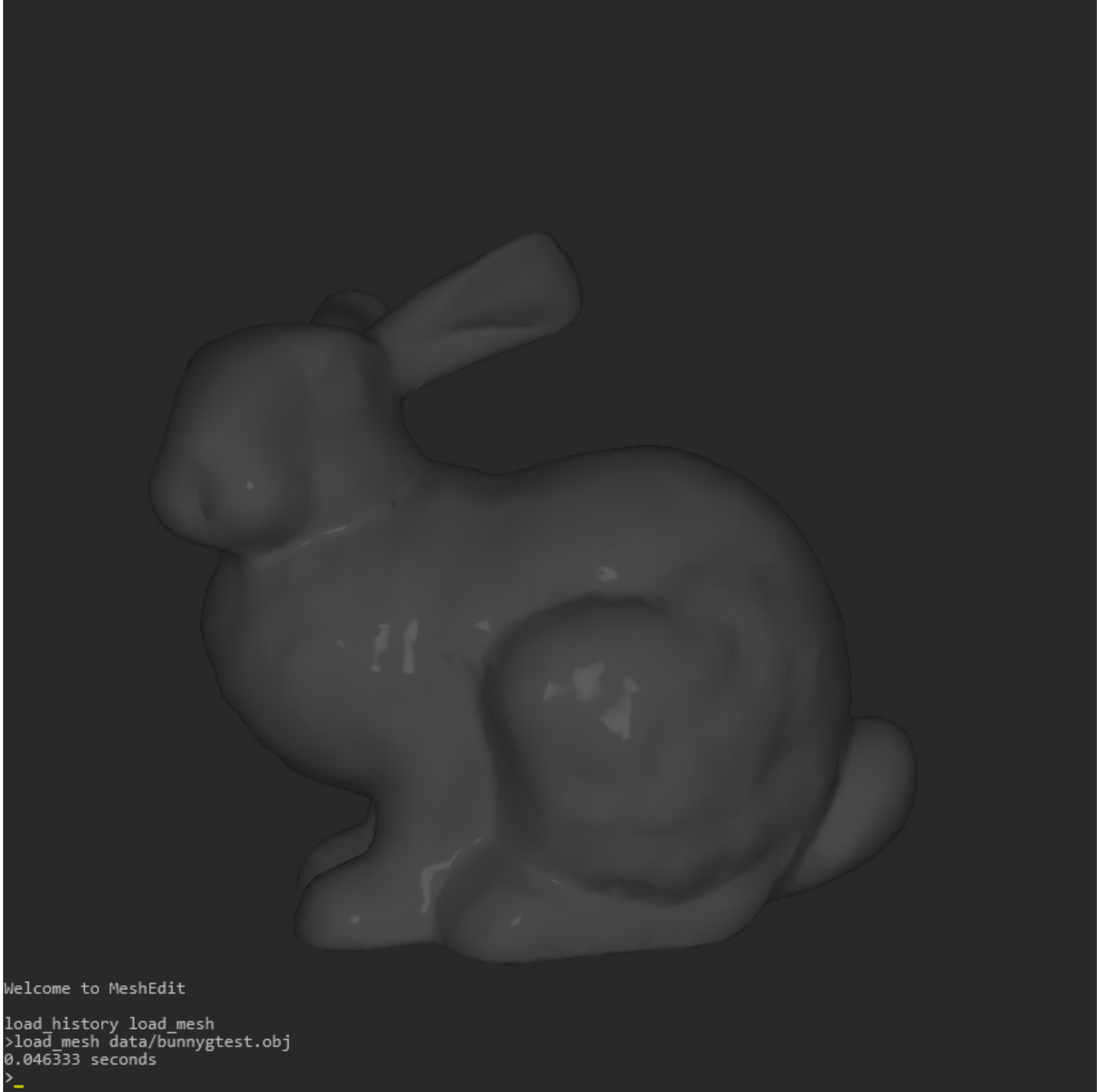
Today’s exercise: Computing the dual of a polygonal mesh

- Getting started:
  - –Download GEL and the example program
  - –Start up Visual Studio
  - –Compute the dual!
- What is the dual anyway?
  - –For each face create a new vertex
  - –For each vertex create a new face

Esc



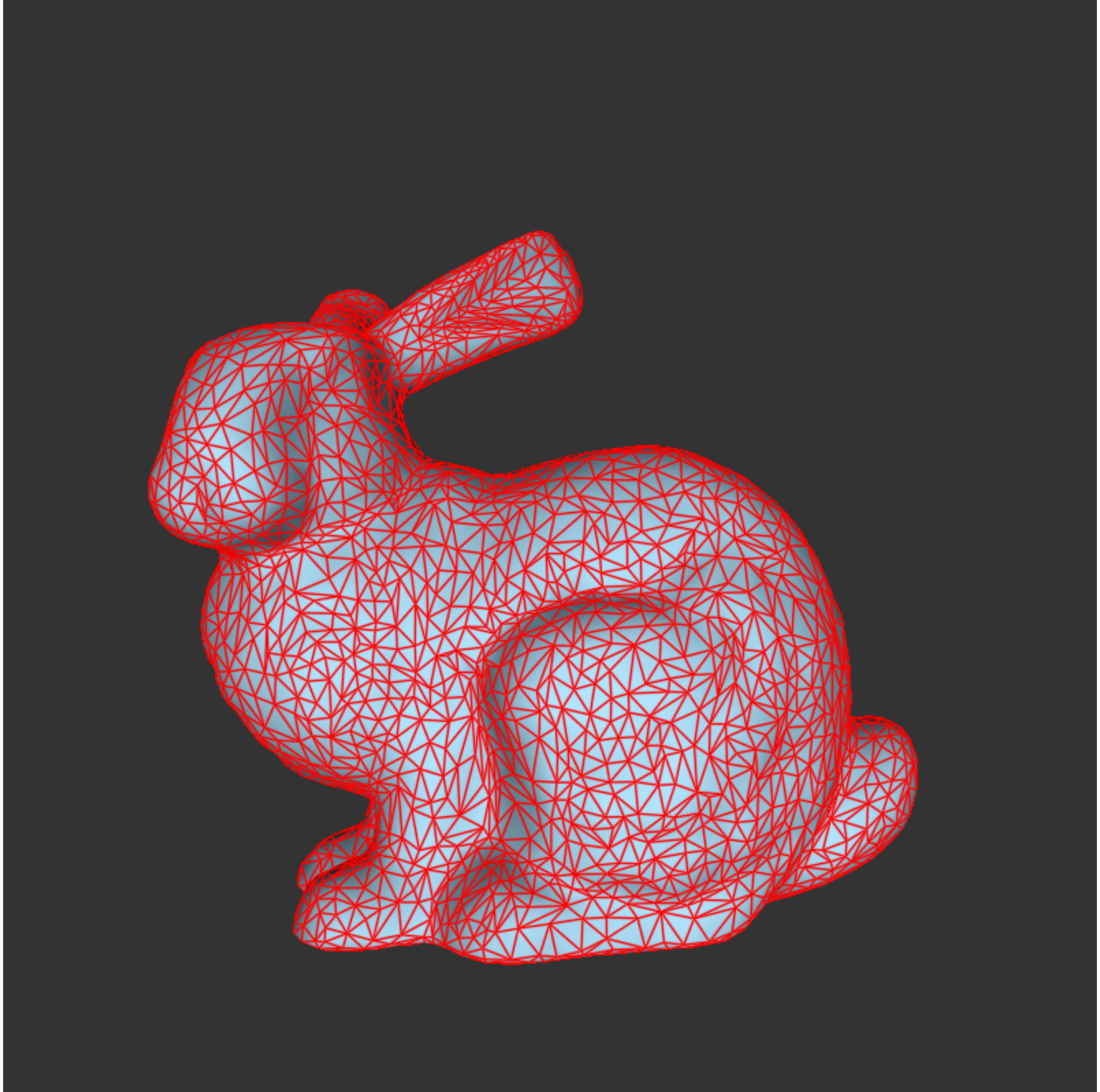
```
Welcome to MeshEdit

>_
```

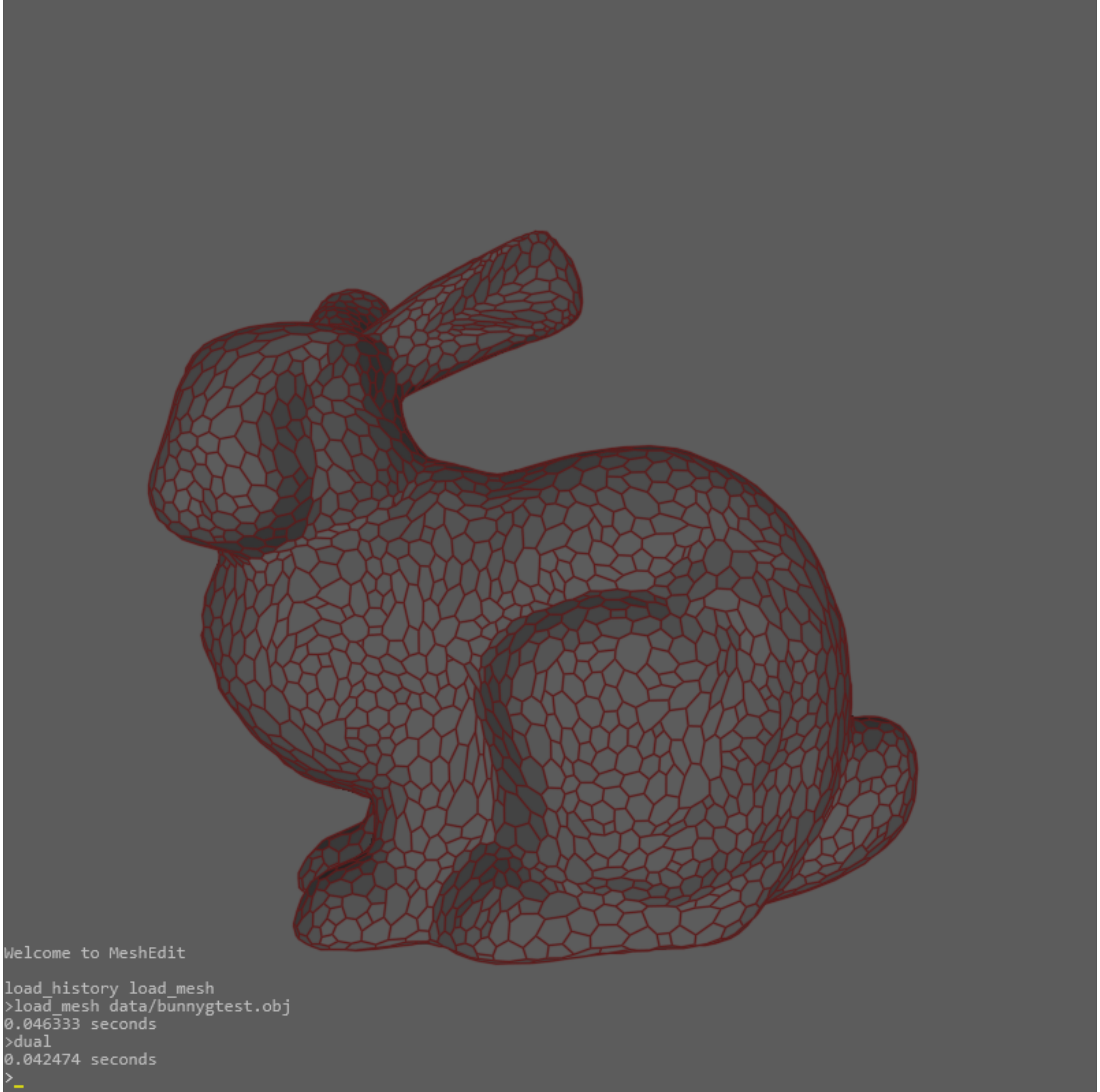


```
Welcome to MeshEdit

load_history load_mesh
>load_mesh data/bunnygtest.obj
0.046333 seconds
>_
```

Esc

‘w’





```
Welcome to MeshEdit

load_history load_mesh
>load_mesh data/bunnygtest.obj
0.046333 seconds
>dual
0.042474 seconds
>
```