1 (Practice 5) Mesh loader

In this practice we will see the creation of a mesh from a file containing 3D faces, and its visualization with OpenGL.

The goal is to load an ASCII file of extension .off (for **O**bject **F**ile **F**ormat) containing a 3D mesh.

The file look like this

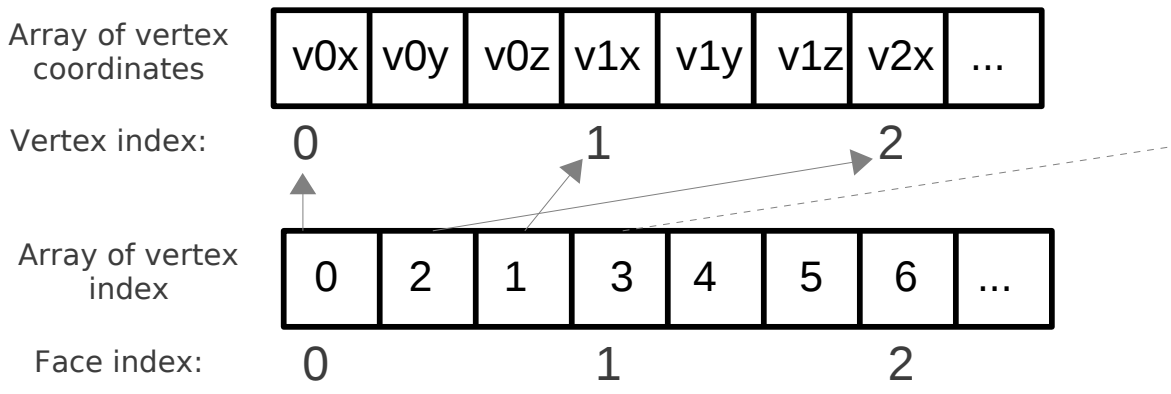
```
OFF
8 12 0
0.256 0.365 0.569
2.365 2.654 8.145
...
...
3 0 1 2
3 2 5 9
3 6 0 4
...
...
```

where:

- The three first characters correspond to the file extension, here: **OFF**
- The three next integers are:
  - number of vertices of the mesh, here: **8**.
  - number of faces of the mesh, here: **12**.
  - number of edges which is **0** (usually the value is zero, because it hasn't been updated since the file creation)
- The list of floats are the vertices coordinates (three first floats are the (x, y, z) coordinates of the vertex, every line designate a single vertex)
- Finally, the faces are given by the list of integers. Every line designate a single face. There is four integers, the first integer indicates how many vertices a face is made of. Then the last integers defines the vertices index in the coordinate list.

In this example, the first face is made of the 3 vertices of index 0, 1 and 2, which corresponds to the three vertices [[0.256 0.365 0.569], [2.365 2.654 8.145], ...].

Later you will load these data into arrays and the memory layout will look like this:



1.1 Drawing the mesh

Download the materials associated to this practice. Take a look to the few '.off' files in the archive.

Integrate the file 'mesh.c' and 'mesh.h' into your project. A structure is already define to store the mesh:

```
typedef struct {
    Materials mat;

    int nb_faces;   // Total number of vertices
    int nb_vertices;// Total number of faces

    int*   index;   // List of triangles [ t00 t01 t02 | t10 t11 t12 | etc.]
    float* verts;   // List of vertices coordinates [ v0x v0y v0z | v1x v1y v1z | etc. ]
    float* normals; // List of normals coordinates [ n0x n0y n0z | n1x n1y n1z | etc. ]

    int minY;       // Minimal y coordinate
}Mesh;

Mesh* mesh_init(char* filename);
void mesh_delete(Mesh* mesh);

void mesh_draw(Mesh* mesh);
```

You must read the function 'Mesh* mesh_init(char* filename);' which has been partially filled. The function contains some code to read an OFF file. For now it only reads the beginning of the file but not the face index.

After understanding what mesh_init() does complete it in order to read and allocate memory for the face index (for the moment don't try to compute the normals and leave the line mesh->normals = 0; as is)

Fill the field 'mat' with these values:

ambient = {0.4, 0.2, 0.08, 0.0}
diffuse = {0.4, 0.2, 0.08, 0.0}
specular = {1.0, 0.5, 0.2, 0.0}
shininess = 90.

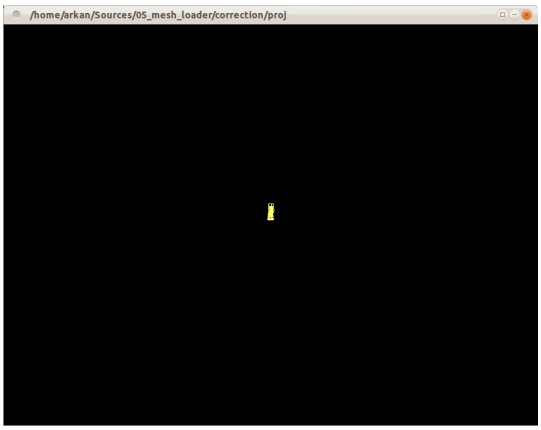
Now complete the function void mesh_draw(Mesh* mesh); You'll need the following OpenGL commands to draw the mesh:

- glEnableClientState(GL_VERTEX_ARRAY); to activate drawing with an array of vertex coordinates
- glVertexPointer() To give the pointer to the array of vertex coordinates (c.f. documentation).
- glDrawElements(primitive_type, size_array, type_array, array) To draw a list of face index (c.f. Documentation)
- glDisableClientState(GL_VERTEX_ARRAY);

Don't forget to set the OpenGL materials before drawing with glDrawElements().

Now we are ready to test this new module. Add a global variable mesh in main.c initialize it in 'main()' and draw the mesh in the 'display()' (you can comment the other drawings we'll reactivate them later).

When loading buddha.off you should see something like this:



The mesh seems to be very small this is because we have no means to know the size of the mesh when loading the file. We also don't know if the mesh points are centered about the origin.

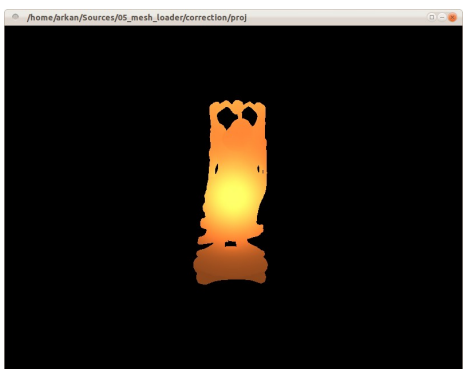
- ✔ Modify mesh_init() to compute the center of gravity of the mesh and substract it to each vertex. This way the mesh will be centered.

- ✔ compute the biggest absolute coordinate of the vertex list with 'fabsf()' and divide the mesh coordinates with this value. After this operation vertex coordinates are guaranteed to be between [-1 1]

Note: center of gravity c is the sum of every vertex v_i divided by the number of vertices:

$$c = \sum_{i}^{N} \frac{v_i}{N}$$

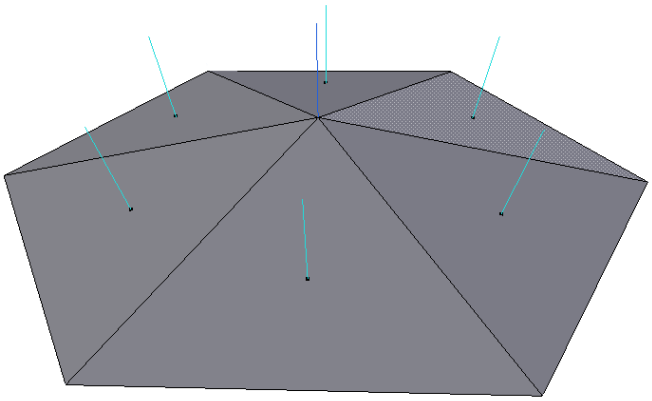
You can now safely scale your drawing every mesh you'll load will be more or less the same size at the center of the scene. This is what you should have with a glScalef(3.f, 3.f, 3.f); before drawing:



If lighting is incorrect it's because we draw without normals.

1.2 Normals

The goal here is to compute the normals of the mesh at each vertex. The more convenient way to do it is to compute the normal at each triangle and then average them at each vertex:

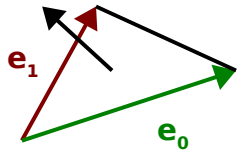


The dark blue normal $\vec{N}_v$ is the average sum of all the triangles normals $\vec{N}_i$ in light blue:

$$\vec{N}_v = \frac{\sum_I^N N_i}{\|\sum_I^N N_i\|}$$

Computing the normal to a face is by doing a cross product between to edge of the triangle:

$$\vec{N}_i = \frac{cross(e_0, e_1)}{\|cross(e_0, e_1)\|}$$

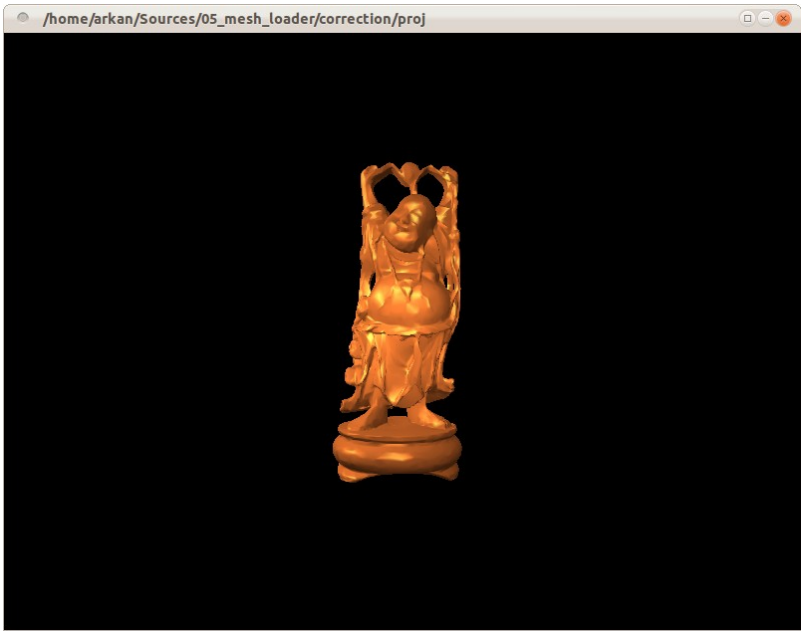


- ✔ Modify 'mesh_init()' to allocate and compute the normals of the mesh at each vertex. You don't need a temporary array to store the normals to the faces. Instead:
  - ✔ initialize to (0.f, 0.f, 0.f) the array of normals at each vertex.
  - ✔ look up the array of faces
    - ✔ compute the normal
    - ✔ add the normal to three vertices belonging to the current face.

- ✔ Modify 'mesh_draw()' to pass to OpenGL the pointer to the normal array. You will need:

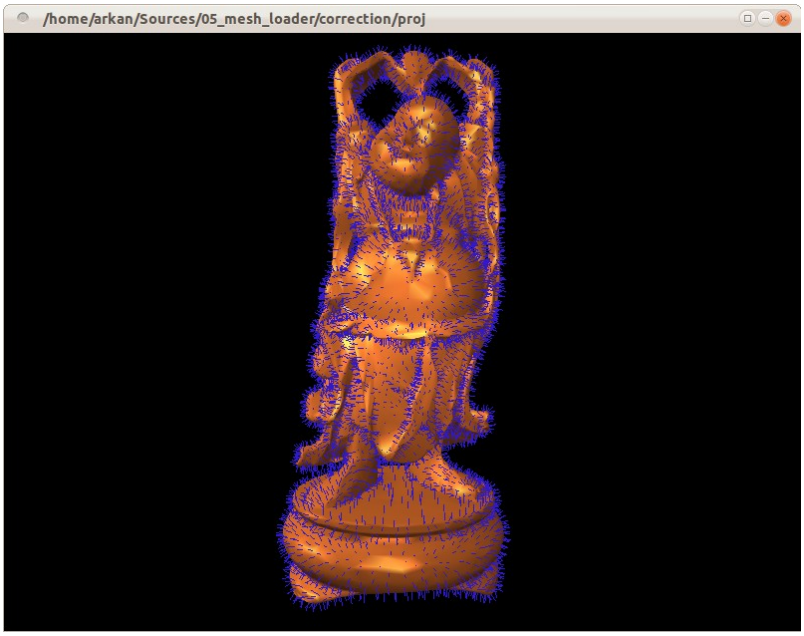
- ✔ glEnableClientState(GL_NORMAL_ARRAY);
- ✔ glNormalPointer(...) To give the pointer to the array of normals coordinates (c.f. Documentation).
- ✔ glDisableClientState(GL_NORMAL_ARRAY);

This is what you should have:



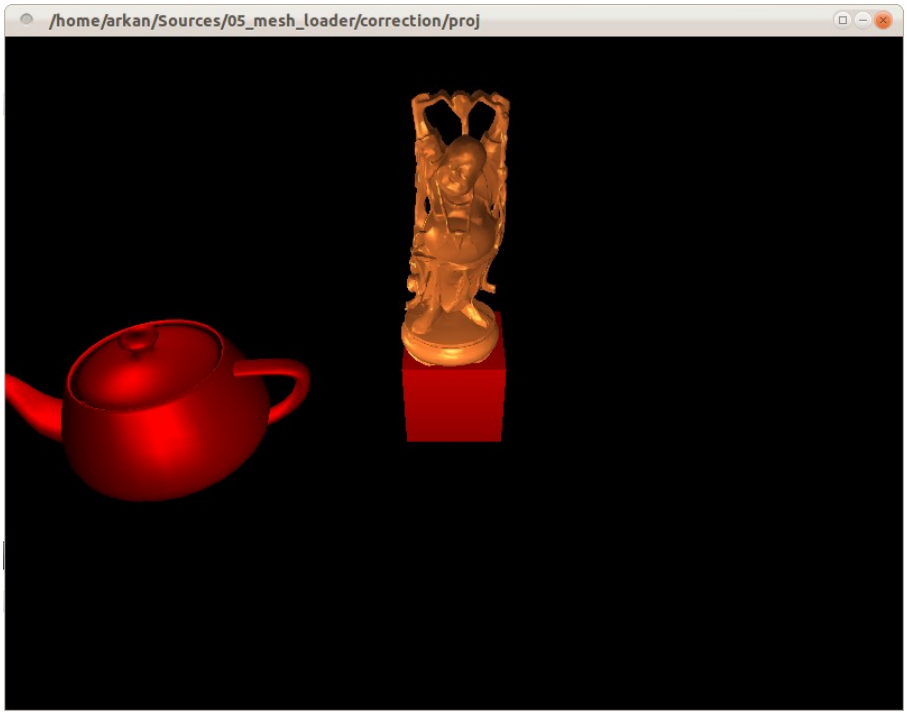
To help you debug the normals and check everything is fine implement a new function void mesh_draw_normals(Mesh* mesh, float size); which draws the normals at each vertex of the mesh. Use the direct mode with glBegin(GL_LINES); and draw blue normals (N.B: size is the length of the OpenGL line representing the segment).

Bound the key 'n' to activate/deactivate normal drawing:

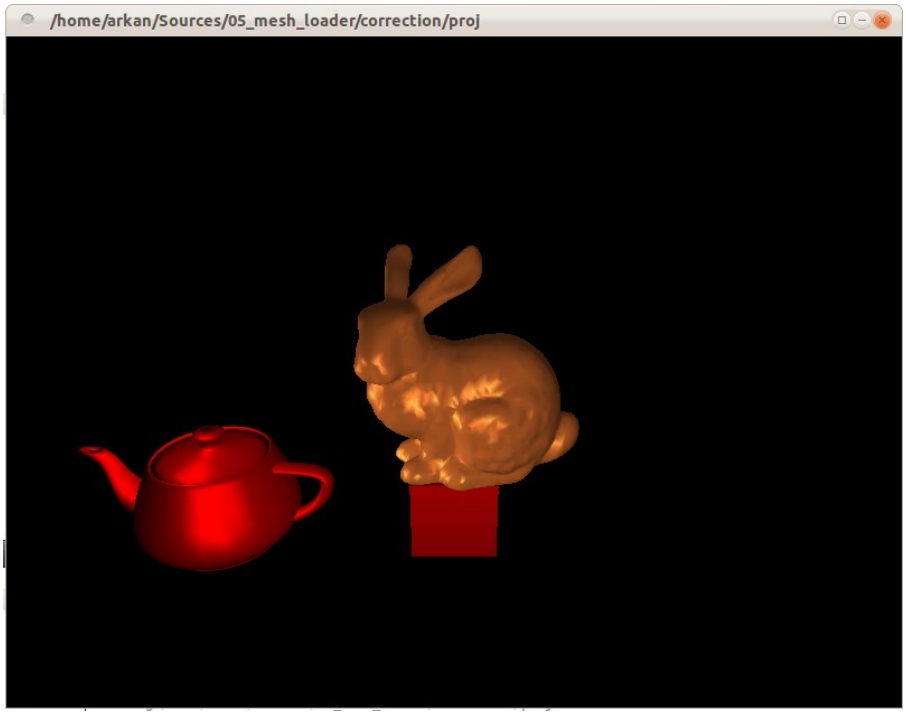


1.3 Bonus

Find a way to put any mesh on top of the cube.



Buddah god of glut teapots.



Bunny celebrating its un-anniversary