Lab 14

Petra Isenberg

November 5th

Assignment 2

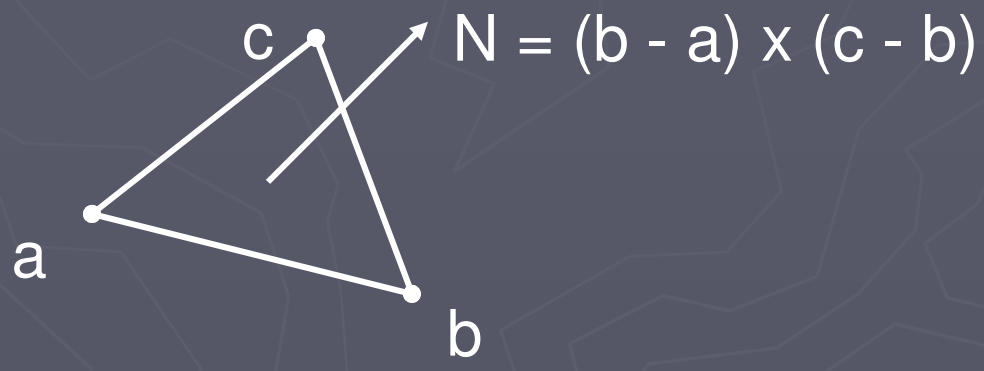
- Rendering (45 marks)
  - Compute and optionally display normal vectors for each triangle and vertex
  - Render the object as OpenGL wire-frame, flat, and smooth shading with directional lightin
  - Upon loading, display the entire object centered in the viewport.
  - Render ground underneath the object, e.g. as a flat checker-board.
- Affine Transformations (20 marks)
  - Translation
  - Uniform and Non-uniform Scaling
  - Rotation around arbitrary vector using virtual trackball object
  - Note: these transformations do not apply to the ground.
- View Changes (20 marks)
  - View location
  - View direction
  - View orientation (up direction)
  - Perspective/Parallel projection
- File open (5 marks)
  - UI for loading models (5 marks).

Assignment 2

- Rendering (45 marks)
  - Compute and optionally display normal vectors for each triangle and vertex
  - Render the object as OpenGL wire-frame, flat, and smooth shading with directional lighting
  - Upon loading, display the entire object centered in the viewport.
  - Render ground underneath the object, e.g. as a flat checker-board.
- Affine Transformations (20 marks)
  - Translation
  - Uniform and Non-uniform Scaling
  - Rotation around arbitrary vector using virtual trackball object
  - Note: these transformations do not apply to the ground.
- View Changes (20 marks)
  - View location
  - View direction
  - View orientation (up direction)
  - Perspective/Parallel projection
- File open (5 marks)
  - UI for loading models (5 marks).

Rendering

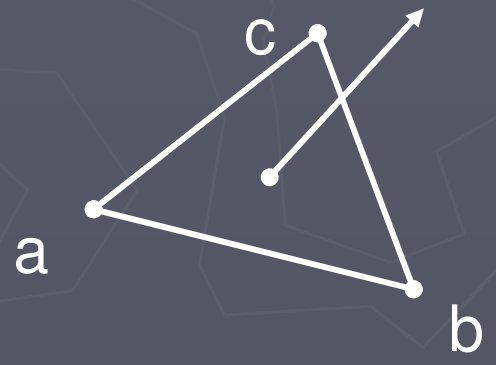
- Normal vectors:
  - Each polygonal face has a normal



  - These are called face normals

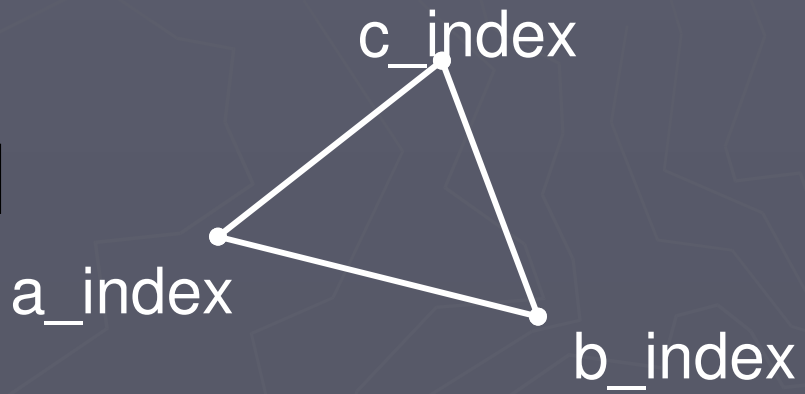
Rendering

- Normal vectors
  - Flat shading: constant color across the polygon
  - Uses face normals

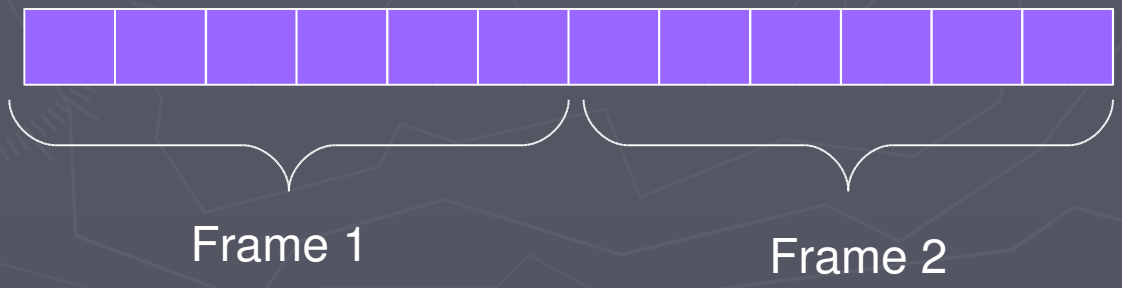


MD2 File & Face Normals

- Structs:
  - triangle_t: index_xyz[3]



  - vec3_t *ptrverts; =pointer to converted vertices
    - ptrverts= &md2File->m_vertices[nrVertices * frame];



MD2 File & Face Normals

```
for (int i = 0; i < nrTriangles; ++i)
    {
        int vertexIndexA = md2File->tris[i].index_xyz[0];
        int vertexIndexB = md2File->tris[i].index_xyz[1];
        int vertexIndexC = md2File->tris[i].index_xyz[2];

        //make sure ptrverts point to the correct frame!
        float *a = ptrverts[vertexIndexA];
        float *b = ptrverts[vertexIndexB];
        float *c = ptrverts[vertexIndexC];

        float normal[3];
        calculateFaceNormal(a,b,c,normal);
            → cross product
            → make sure orientation is correct!
        Save the normal in some data structure (e.g. vec3_t *faceNormals)
}
```

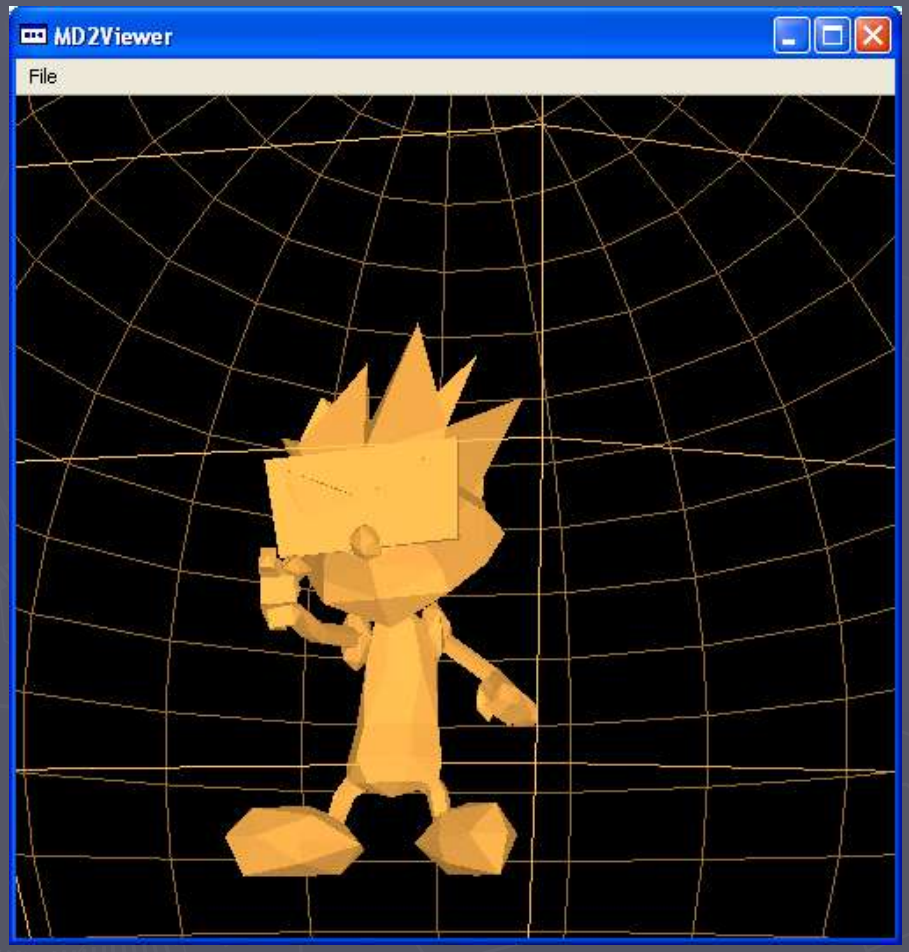
OpenGL & Face Normals

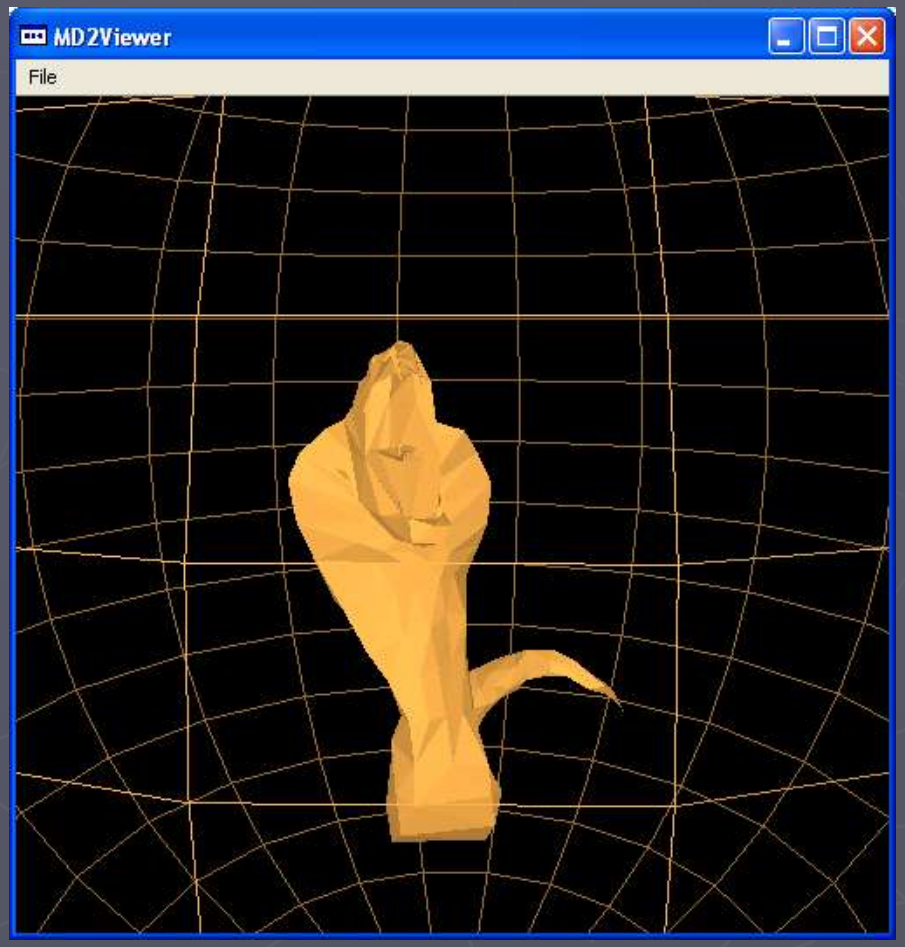
glShadeModel (GL_FLAT);
(call this once not on every redraw)

FOR (all triangles)

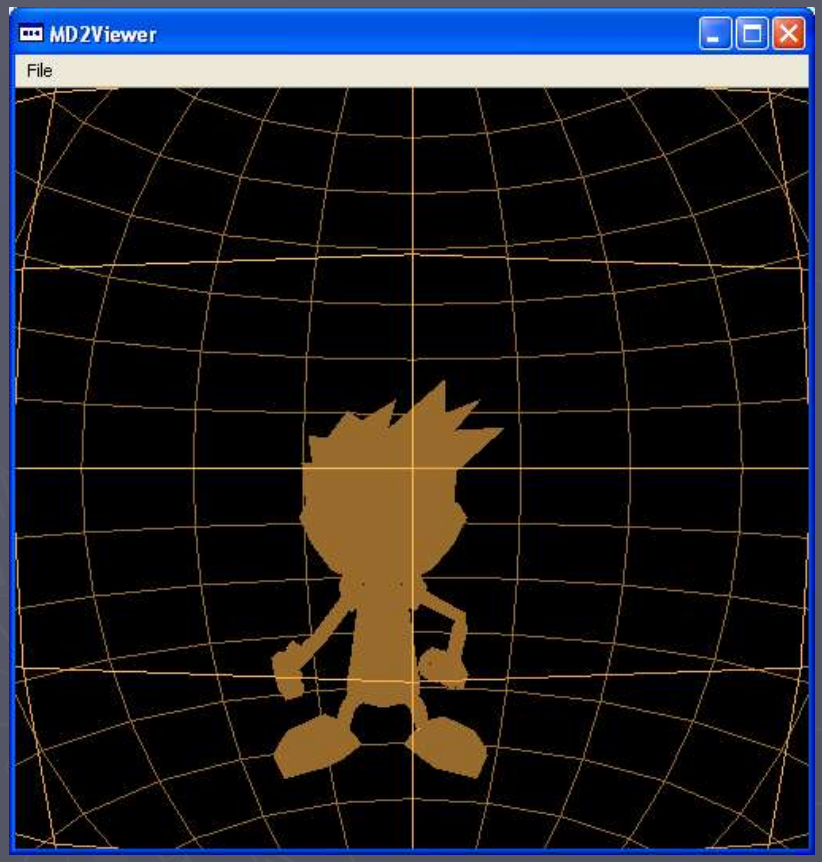
glNormal3fv(faceNormals[triangle]);
glVertex3fv(a);
glVertex3fv(b);
glVertex3fv(c);

OpenGL & Face Normals

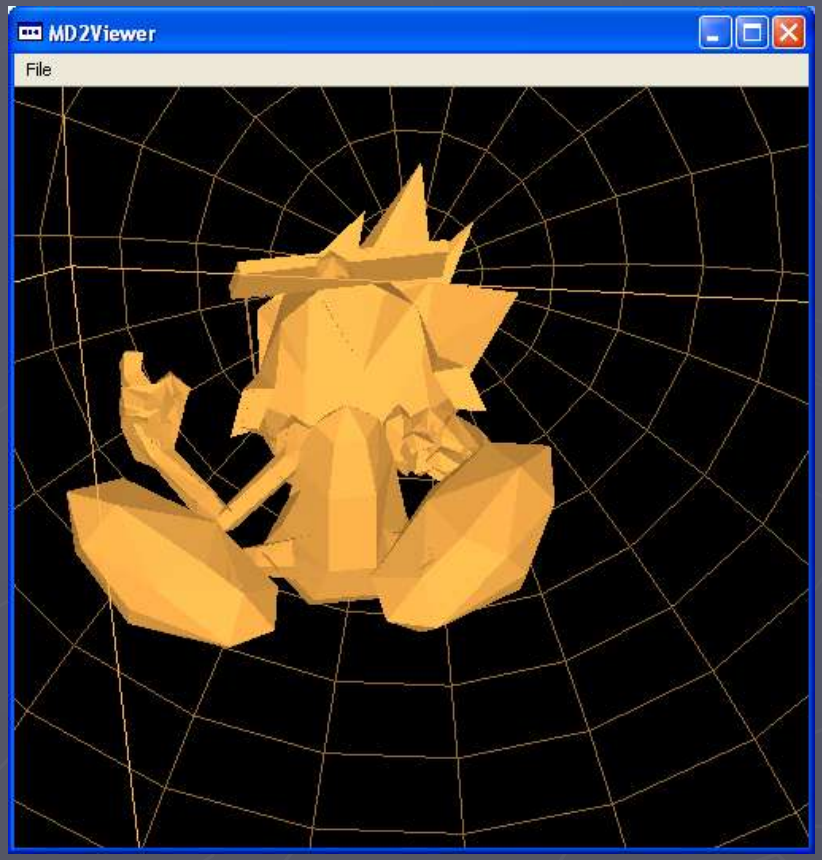




Wrong normals



Normals point inwards

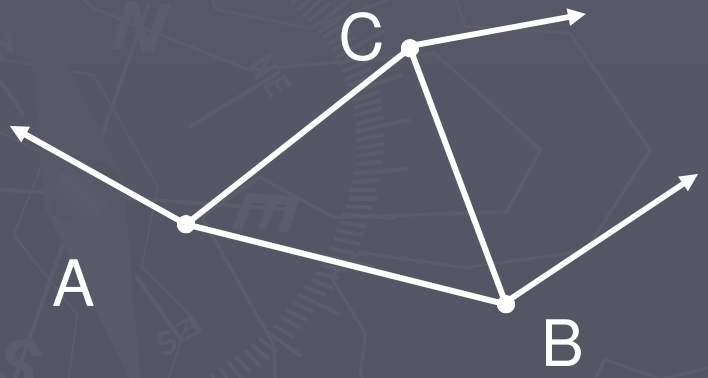


glEnable(GL_CULL_FACE)

Normals point inwards

Rendering

- Normal vectors
  - Smooth shading: Interpolate shading across polygon
  - Uses: vertex normals



MD2 File & Vertex Normals

- triangle_t: index_xyz[3]
- vec3_t *ptrverts

4. For each vertex
   1. Find adjacent triangles
   2. Get the face normal for each adjacent triangle (pre-calculate that)
   3. Linear interpolation of face normals
   4. Normalize the normal
   5. Save in some data structure

OpenGL and Vertex Normals

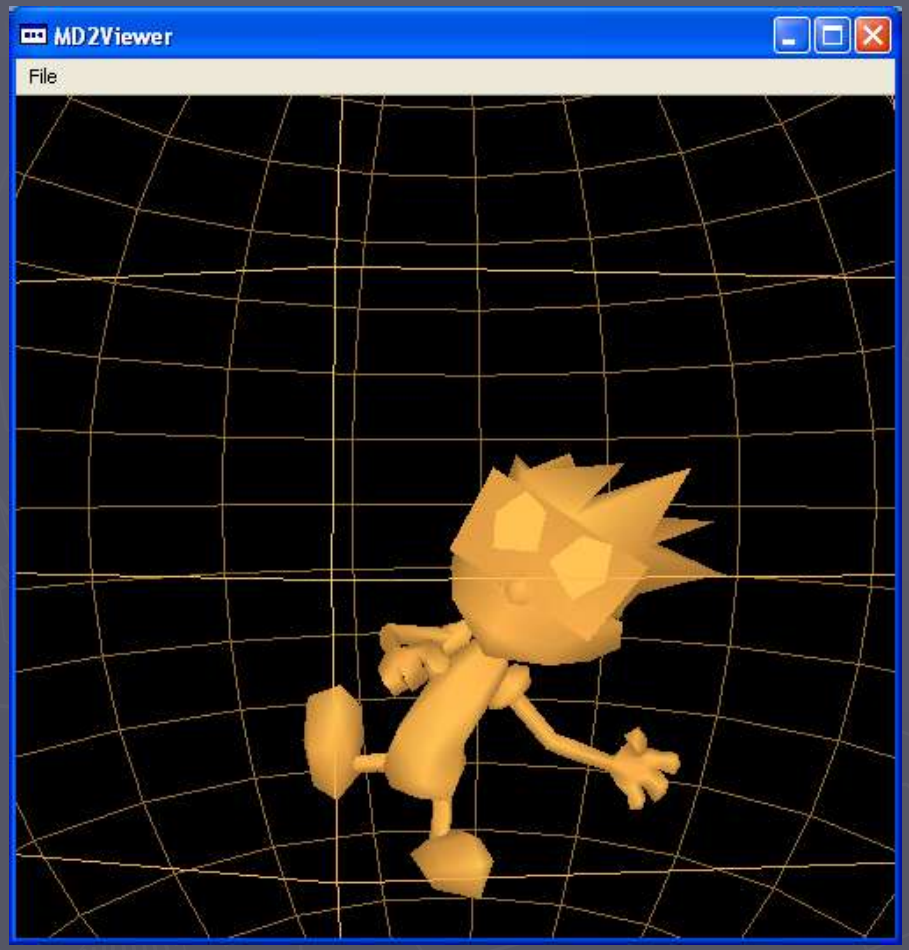
glShadeModel (GL_SMOOTH);

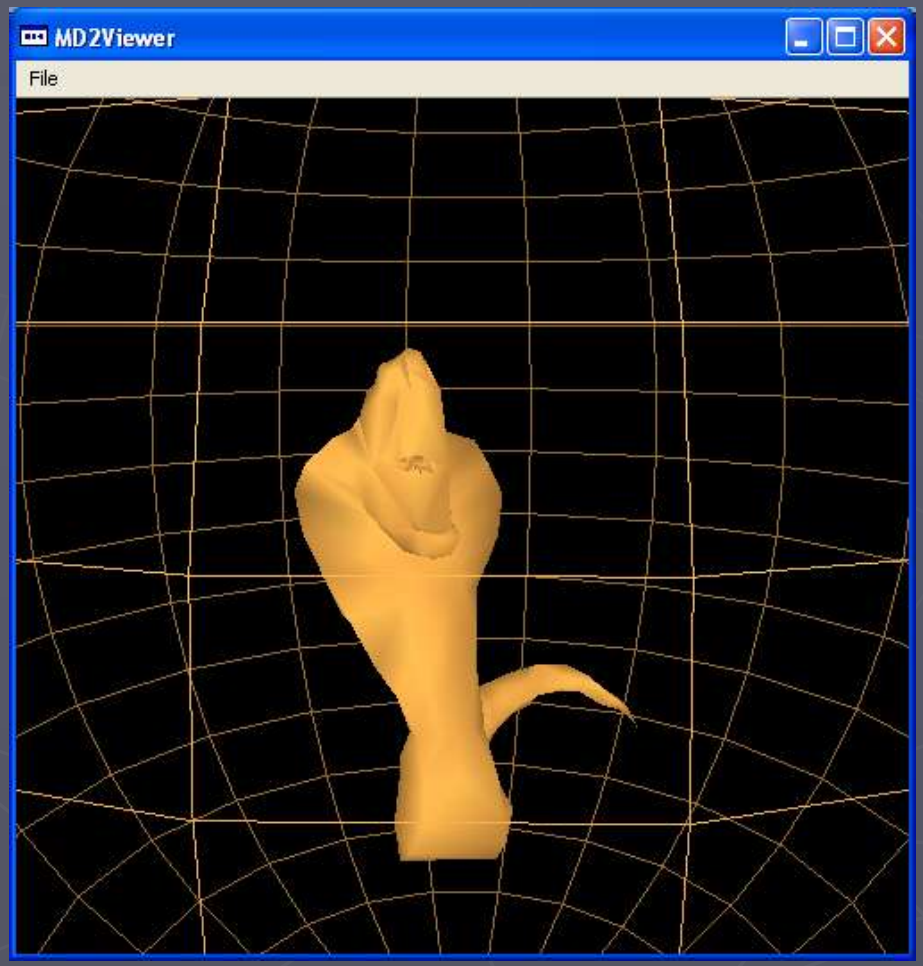
(call this once not on every redraw)

FOR (all triangles)

```
glNormal3fv(normals[vertexIndexA]);
glVertex3fv(a);
glNormal3fv(normals[vertexIndexB]);
glVertex3fv(b);
glNormal3fv(normals[vertexIndexC]);
glVertex3fv(c);
```

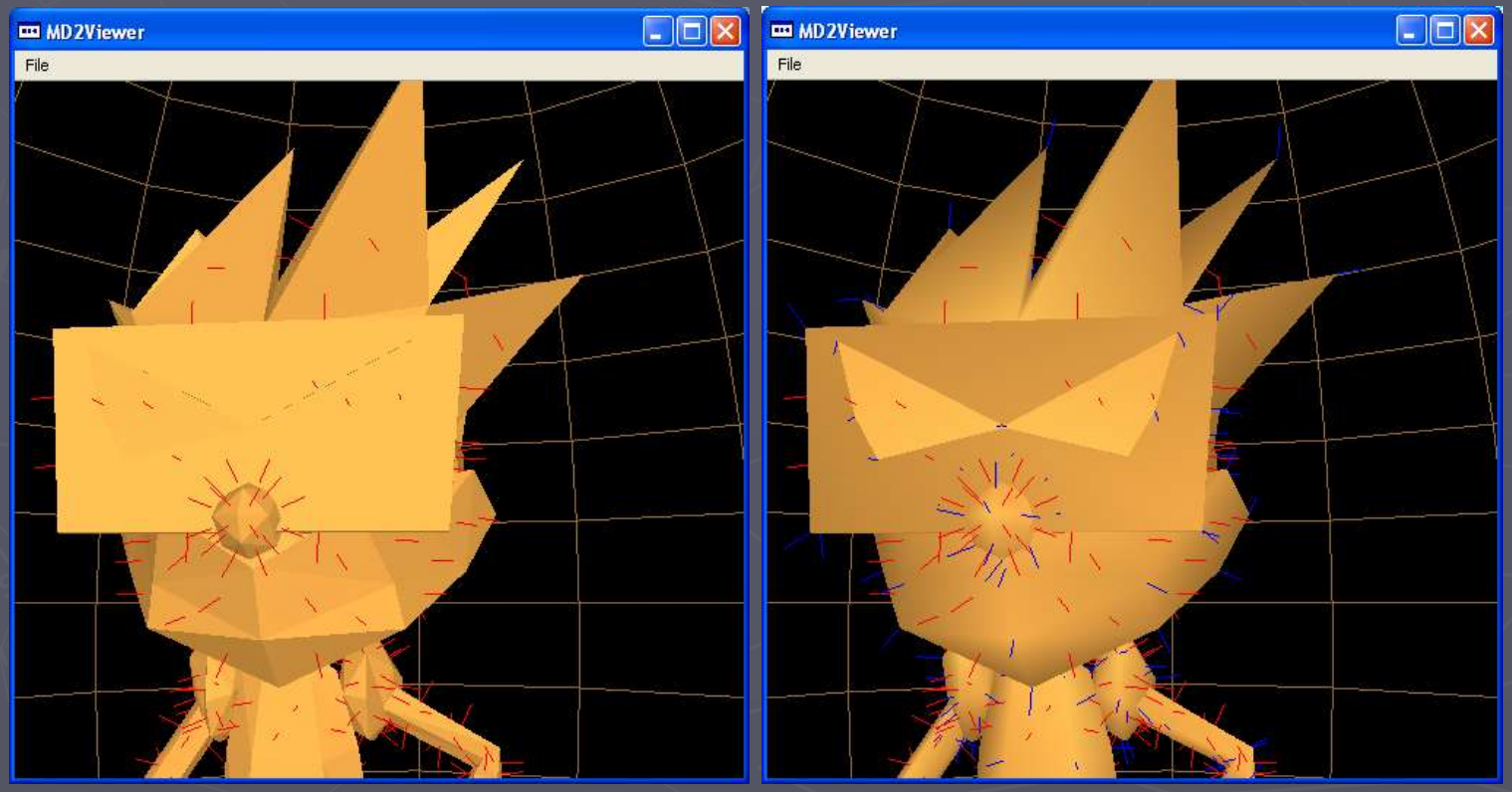
OpenGL and Vertex Normals





Rendering the normals

- GL_LINES
- Turn off lighting before!



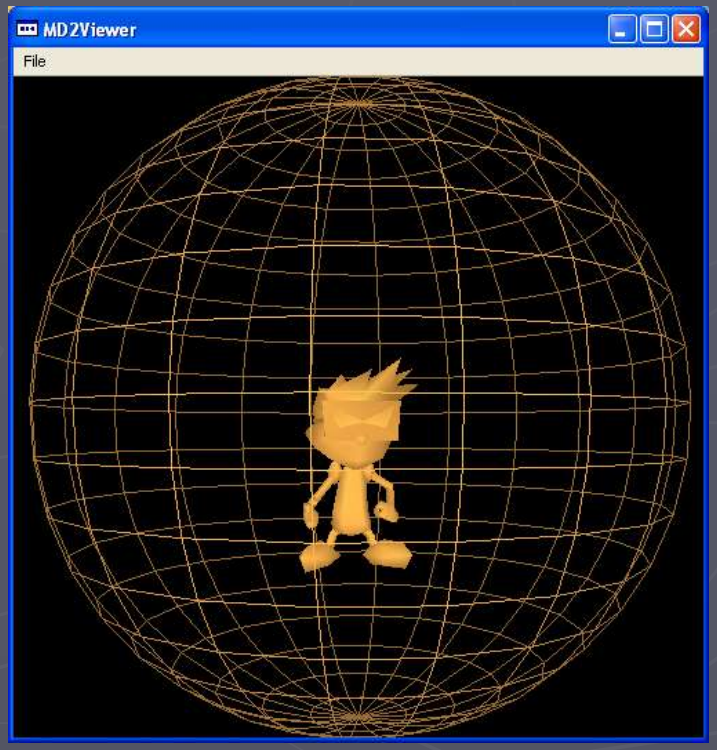
Assignment 2

- Rendering (45 marks)
  - Compute and optionally display normal vectors for each triangle and vertex
  - Render the object as OpenGL wire-frame, flat, and smooth shading with directional lighting
  - Upon loading, display the entire object centered in the viewport.
  - Render ground underneath the object, e.g. as a flat checker-board.
- Affine Transformations (20 marks)
  - Translation
  - Uniform and Non-uniform Scaling
  - Rotation around arbitrary vector using virtual trackball object
  - Note: these transformations do not apply to the ground.
- View Changes (20 marks)
  - View location
  - View direction
  - View orientation (up direction)
  - Perspective/Parallel projection
- File open (5 marks)
  - UI for loading models (5 marks).

Center object in viewport

- = move the camera to the right position

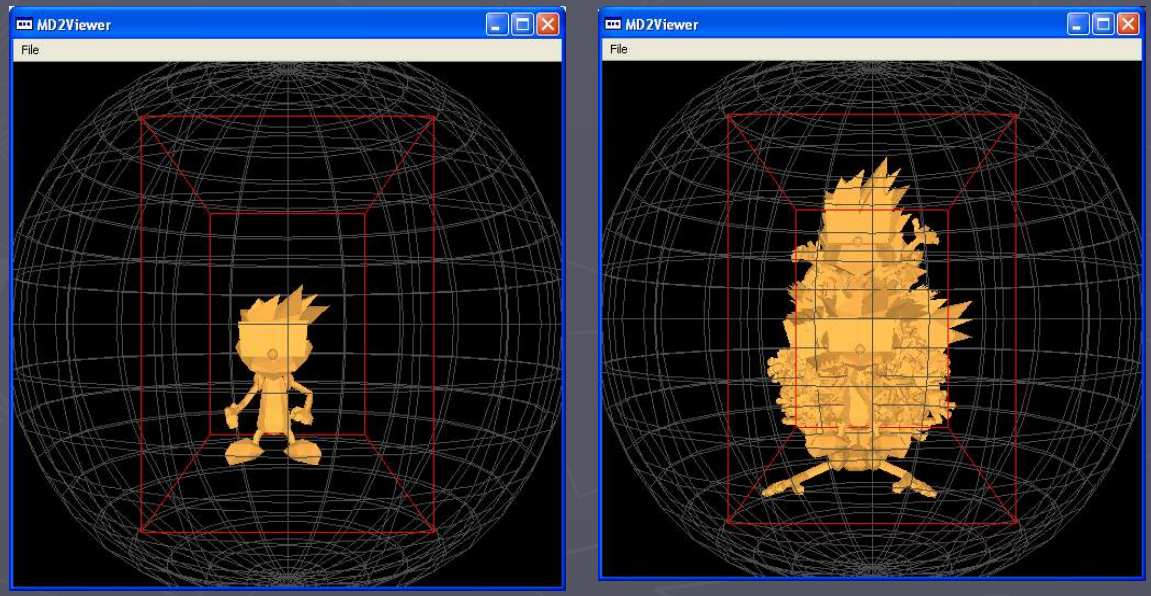
1. Calculate bounding sphere for object



Center object in viewport

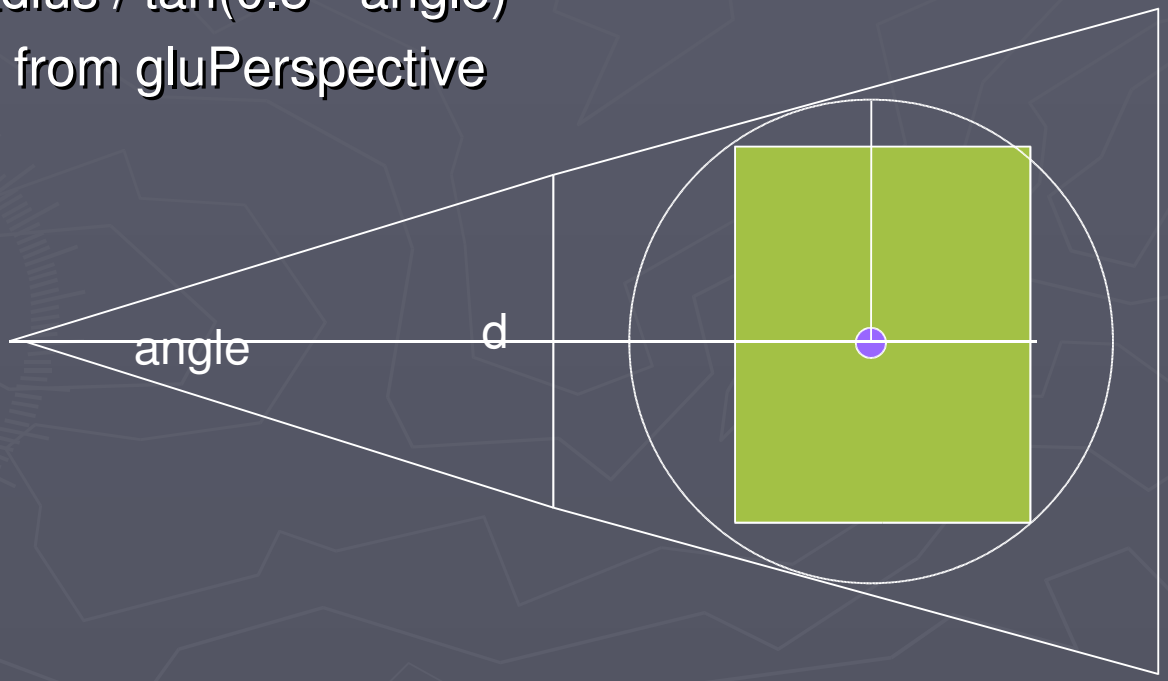
- = move the camera to the right position

1. Calculate bounding sphere for object
   1. Get max and min coordinates for all 3 dimensions
   2. Get those coordinates across all frames!



Center object in viewport

- Calculate center of object
- Calculate radius of sphere (longest diagonal of box)
- Calculate distance from object
  - d = radius / tan(0.5 * angle)
  - Angle from gluPerspective



Assignment 2

- Rendering (45 marks)
  - Compute and optionally display normal vectors for each triangle and vertex
  - Render the object as OpenGL wire-frame, flat, and smooth shading with directional lighting
  - Upon loading, display the entire object centered in the viewport.
  - Render ground underneath the object, e.g. as a flat checker-board.
- Affine Transformations (20 marks)
  - Translation
  - Uniform and Non-uniform Scaling
  - Rotation around arbitrary vector using virtual trackball object
  - Note: these transformations do not apply to the ground.
- View Changes (20 marks)
  - View location
  - View direction
  - View orientation (up direction)
  - Perspective/Parallel projection
- File open (5 marks)
  - UI for loading models (5 marks).

Render ground

- Simple!
- Find out where the bottom is (always the same)
- Render quads for checker board under min coords
- Or render one quad and texture map (later)

Assignment 2

- Rendering (45 marks)
  - Compute and optionally display normal vectors for each triangle and vertex
  - Render the object as OpenGL wire-frame, flat, and smooth shading with directional lighting
  - Upon loading, display the entire object centered in the viewport.
  - Render ground underneath the object, e.g. as a flat checker-board.
- Affine Transformations (20 marks)
  - Translation
  - Uniform and Non-uniform Scaling
  - Rotation around arbitrary vector using virtual trackball object
  - Note: these transformations do not apply to the ground.
- View Changes (20 marks)
  - View location
  - View direction
  - View orientation (up direction)
  - Perspective/Parallel projection
- File open (5 marks)
  - UI for loading models (5 marks).