Project 3 Discussion

Robb T. Koether

Hampden-Sydney College

Fri, Oct 16, 2009
Outline

1. Introduction
2. User Interface
3. Discussion
   - Surfaces
   - The Mesh Class
   - Lighting
4. Design
5. Due Dates
Outline

1. Introduction
2. User Interface
3. Discussion
   - Surfaces
   - The Mesh Class
   - Lighting
4. Design
5. Due Dates
This **program** will create a complex object as an assembly of meshes.

The object will be assembled from basic shapes:
- Squares
- Disks
- Cylinders
- Cones
- Spheres
- Paraboloids
- Hyperboloids (of one or two sheets)
- Toruses
Outline

1. Introduction
2. User Interface
3. Discussion
   - Surfaces
   - The Mesh Class
   - Lighting
4. Design
5. Due Dates
# The User Interface

<table>
<thead>
<tr>
<th>Event</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Press +</td>
<td>Zoom in</td>
</tr>
<tr>
<td>Press −</td>
<td>Zoom out</td>
</tr>
<tr>
<td>Press ←</td>
<td>Rotate the camera left</td>
</tr>
<tr>
<td>Press →</td>
<td>Rotate the camera right</td>
</tr>
<tr>
<td>Press ↑</td>
<td>Rotate the camera up</td>
</tr>
<tr>
<td>Press ↓</td>
<td>Rotate the camera down</td>
</tr>
<tr>
<td>Press &gt;</td>
<td>Increase fineness of mesh</td>
</tr>
<tr>
<td>Press &lt;</td>
<td>Decrease fineness of mesh</td>
</tr>
<tr>
<td>Right-click</td>
<td>Pop-up menu</td>
</tr>
</tbody>
</table>
The pop-up menu will contain four items:
- Point light source
- Directional light source
- Local viewer
- Infinite viewer
Outline

1. Introduction
2. User Interface
3. Discussion
   - Surfaces
   - The Mesh Class
   - Lighting
4. Design
5. Due Dates
Outline

1. Introduction
2. User Interface
3. Discussion
   - Surfaces
     - The Mesh Class
     - Lighting
4. Design
5. Due Dates
The interesting object that you render must use at least four different surfaces, at least three of which are curved.

In other words, do not build something (like a brick) entirely out of rectangles.

You will most likely want to use the parametric form of the equations of the basic shapes.
Outline

1. Introduction
2. User Interface
3. Discussion
   - Surfaces
   - The Mesh Class
   - Lighting
4. Design
5. Due Dates
A Mesh object represents an array of vertices and their normals, arranged in rows and columns.

The array of vertices is constructed from a function $f(s, t)$ of two parameters $s$ and $t$.

$s$ and $t$ each vary over a specified range of values.

The array of normals is constructed from the derivatives $\frac{\partial f}{\partial s}$ and $\frac{\partial f}{\partial t}$.
The **Mesh** Class

- The data members of a **Mesh** object are
  - `int rows` – Number of rows
  - `int cols` – Number of columns
  - `Point3D** pt` – Array of vertices
  - `Vector3D** norm` – Array of normals

- The number of rows is the size of the partition (number of *vertices*) in the *s* direction.

- The number of columns is the size of the partition (the number of *normals*) in the *t* direction.
There are three `Mesh` constructors: the default constructor, the copy constructor, and one with prototype

```cpp
Mesh(
    Point3D (*f)(double, double),
    Vector3D (*dfds)(double, double),
    Vector3D (*dfdt)(double, double),
    double smin,
    double smax,
    double tmin,
    double tmax,
    int num_s,
    int num_t
);
```
You will be responsible for two `Mesh` class functions:
- `create()`
- `draw()`
The `create()` Function

- The `create()` function has the same prototype as the constructor described above.
- It will use the function $f()$ to calculate the vertices in the array $pt$.
- It will use the functions $dfds()$ and $dfdt()$ to calculate the normals in the array $norm$.
- Of course, your application program must provide the functions $f()$, $dfds()$, and $dfdt()$ outside of the `Mesh` class. They are application-specific.
The `draw()` function will render the mesh as a series of either quad strips or triangle strips (your choice).
Other Mesh Functions

- Most of the remaining functions in the Mesh class are routine CS II exercises.
- However, the functions `translate()`, `rotate()`, and `scale()` are not.
- Each of these functions performs a transformation on the vertices.
- Note that these functions do *not* use the modelview matrix; they recalculate the vertices themselves.
Other Mesh Functions

- Also, notice how they handle the normal vectors.
- The function `translate()` leaves them unchanged.
- The function `rotate()` rotates them along with the vertices.
- The function `scale()` scales them by *dividing* by the scale factors.
This program will use the standard OpenGL lighting model.

You will need to set up (at least) one light and give your meshes material properties.

Depending on the object that you are drawing, your meshes may or may not all have the same material properties.

The material properties are not part of the Mesh class.
Outline

1. Introduction
2. User Interface
3. Discussion
   - Surfaces
   - The Mesh Class
   - Lighting
4. Design
5. Due Dates
Your design should include

- A detailed drawing of the object that you intend to render, including the necessary meshes, their equations, dimensions, and positions.
- A detailed description of how the `Mesh` class function `create()` will work, given an analytical description of the surface.
- A detailed description of how the `Mesh` class function `draw()` will work.
## The Due Dates

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>Fri, Oct 23</td>
</tr>
<tr>
<td>Program</td>
<td>Fri, Oct 30</td>
</tr>
<tr>
<td>Corrections</td>
<td>Return date + one week</td>
</tr>
</tbody>
</table>