The Mesh Class

Lecture 23

Robb T. Koether

Hampden-Sydney College

Mon, Oct 28, 2019

Outline

Outline

The `Mesh` Class

- The `Mesh` class will construct a mesh of any specified refinement for
  - Squares
  - Disks
  - Spheres
  - Cylinders
  - Cones
  - Paraboloids
  - Hyperbolic paraboloids
  - Hyperboloids of one sheet
  - Toruses

The Constructors

The Constructors

```
Mesh();
```

- The default constructor creates an empty mesh (no vertices).
- The copy constructor creates a copy of the specified mesh.

The `MeshType` Enumerated Type

The `MeshType` Enumerated Type

```
enum MeshType {MESH_SQUARE, MESH_DISK, MESH_SPHERE,
    MESH_CYLINDER, MESH_CONE, MESH_PARABOLOID,
    MESH_HYPERBOLOID, MESH_HYPERBOLIC_PARABOLOID,
    MESH_TORUS};
```

- The class uses the enumerated `MeshType` to specify the type of object.

The Data Members

The Data Members

```
GLuint vbo;                 // Vertex Buffer Object
GLuint vao;                 // Vertex Array Object

GLint num_s;                // Number of subdivisions in s
GLint num_t;                // Number of subdivisions in t

vec3 m_amb;                 // Material ambient property
vec3 m_diff;                // Material diffuse property
vec3 m_spec;                // Material specular property
GLfloat m_shiny;            // Material shininess

VertexMeshData3D* data; // The array of vertices
                                and normals
```

The Data Members

The Data Members

```
static GLint m_amb_loc;     // Material amb location
static GLint m_diff_loc;    // Material diff location
static GLint m_spec_loc;    // Material spec location
static GLint m_shiny_loc;   // Material shiny location
```

The Parameterized Mesh

- Each mesh uses parameters s and t.
- The programmer specifies
  - The type of object to create.
  - The coarseness of the mesh (number of subintervals) in each of the s and t directions.
  - The starting and ending values for each of s and t.
  - The direction in which the normals are to point (inward or outward).
  - Any other necessary parameters, such as a radius, depending on the object.

The Parameterized Mesh

- The `Mesh` class contains functions for the parameterization of each of the types of object specified above.
- To create a new type of object, the programmer would need to
  - Parameterize the surface in s and t.
  - Provide the formulas for the vertices.
  - Provide the formulas for the normals.

The `create()` Functions

The First `create()` Function

```
void create(MeshType type,
    int n_s, float start_s, float end_s,
    int n_t, float start_t, float end_t,
    bool dir = true);
```

- For s:
  - **`int`** `n_s` – number of cells in the s direction.
  - **`int`** `start_s` – initial value of s.
  - **`int`** `end_s` – final value of s.
- Similarly for t.
- **`bool`** `dir` – direction of the normal.
  - **`true`** uses $\mathbf{n} = \frac{\partial P}{\partial s} \times \frac{\partial P}{\partial t}$.
  - **`false`** uses $\mathbf{n} = \frac{\partial P}{\partial t} \times \frac{\partial P}{\partial s}$.

The `create()` Functions

The Second `create()` Function

```
void create(MeshType type,
    int n_s, float start_s, float end_s,
    int n_t, float start_t, float end_t,
    float rad, bool dir = true);
```

- **`float`** `rad` – a radius (for circles, paraboloids, etc.)

The `create()` Functions

The Third `create()` Function

```
void create(MeshType type,
    int n_s, float start_s, float end_s,
    int n_t, float start_t, float end_t,
    float rad1, float rad2, bool dir = true);
```

- **`float`** `rad1` – first radius
- **`float`** `rad2` – second radius
- A torus has two radii.

Spheres

Spheres

```
Mesh sphere;                        // Define globally

sphere.create(MESH_SPHERE,          // Create in init()
    40, 0.0f, 2.0f*PI,
    20, -0.5f*PI, 0.5f*PI);

sphere.draw();                      // Draw in display()
```

- This will draw a sphere of (default) radius 1.0.

Spheres

Spheres

```
Mesh sphere;

sphere.create(MESH_SPHERE,
    40, 0.0f, 2.0f*PI,
    20, -0.5f*PI, 0.5f*PI, 3.0f);

sphere.draw();
```

- This will draw a sphere of radius `3.0`.

Paraboloids

Paraboloids

```
Mesh paraboloid;

paraboloid.create(MESH_PARABOLOID,
    40, 0.0f, 2.0f*PI,
    20, 0.0f, 3.0f);

paraboloid.draw();
```

- This will draw a “standard” paraboloid of height `3.0`.

Paraboloids

Paraboloids

```
Mesh paraboloid;

paraboloid.create(MESH_PARABOLOID,
    40, 0.0f, 2.0f*PI,
    20, 0.0f, 3.0f, 1.0f);

paraboloid.draw();
```

- This will draw a paraboloid of height `3.0` and with a top radius of `1.0`.

Hyperboloids of One Sheet

Hyperboloids of One Sheet

```
Mesh hyperboloid;

hyperboloid.create(MESH_HYPERBOLOID,
    40, 0.0f, 2.0f*PI,
    20, -3.0f, 3.0f);

hyperboloid.draw();
```

- This will draw a “standard” hyperboloid of one sheet ranging from `-3.0` to `3.0` vertically.

Hyperboloids of One Sheet

Hyperboloids of One Sheet

```
Mesh hyperboloid;

hyperboloid.create(MESH_HYPERBOLOID,
    40, 0.0f, 2.0f*PI,
    20, -3.0f, 3.0f, 2.0f);

hyperboloid.draw();
```

- This will draw a hyperboloid of one sheet ranging from `-3.0` to `3.0` vertically with a minimum radius of `2.0`.

Hyperboloids of One Sheet

Hyperboloids of One Sheet

```
Mesh hyperboloid;

hyperboloid.create(MESH_HYPERBOLOID,
    40, 0.0f, 2.0f*PI,
    20, -3.0f, 3.0f, 2.0f, 3.0f);

hyperboloid.draw();
```

- This will draw a hyperboloid of one sheet ranging from `-3.0` to `3.0` vertically with a minimum radius of `2.0` and a top radius of `3.0`.

Other Member Functions

Other Member Functions

```
void clear();
void draw(bool fill = true);
```

- `clear()` – Return the object to the empty state.
- `draw()` – Draw the object.
- If `fill` is true, then the mesh is filled in (solid).
- If `fill` is false, then the mesh is rendered as a wire frame.
- The default value is **`true`**.