NAME

glEvalMesh1, glEvalMesh2 – compute a one- or two-dimensional grid of points or lines

C SPECIFICATION

```
void glEvalMesh1( GLenum mode,
                  GLint i1,
                  GLint i2 )
```

delim $$

PARAMETERS

mode In **glEvalMesh1**, specifies whether to compute a one-dimensional mesh of points or lines. Symbolic constants **GL_POINT** and **GL_LINE** are accepted.

i1, *i2* Specify the first and last integer values for grid domain variable i.

C SPECIFICATION

```
void glEvalMesh2( GLenum mode,
                  GLint i1,
                  GLint i2,
                  GLint j1,
                  GLint j2 )
```

PARAMETERS

mode In **glEvalMesh2**, specifies whether to compute a two-dimensional mesh of points, lines, or polygons. Symbolic constants **GL_POINT**, **GL_LINE**, and **GL_FILL** are accepted.

i1, *i2* Specify the first and last integer values for grid domain variable i.

j1, *j2* Specify the first and last integer values for grid domain variable j.

DESCRIPTION

glMapGrid and **glEvalMesh** are used in tandem to efficiently generate and evaluate a series of evenly-spaced map domain values. **glEvalMesh** steps through the integer domain of a one- or two-dimensional grid, whose range is the domain of the evaluation maps specified by **glMap1** and **glMap2**. *mode* determines whether the resulting vertices are connected as points, lines, or filled polygons.

In the one-dimensional case, **glEvalMesh1**, the mesh is generated as if the following code fragment were executed:

```
glBegin (type);
for (i = i1; i <= i2; i += 1)
   glEvalCoord1(i . DELTA u + u sub 1)
glEnd();
```

where

DELTA u = $(u_2 - u_1) / 1$

and n, u_1, and u_2 are the arguments to the most recent **glMapGrid1** command. *type* is **GL_POINTS** if *mode* is **GL_POINT**, or **GL_LINES** if *mode* is **GL_LINE**.

The one absolute numeric requirement is that if i = n, then the value computed from

$i . \mathrm{DELTA}\ u + u$

is exactly u.

In the two-dimensional case, **glEvalMesh2**, let

$\mathrm{DELTA}\ u = (u_2 - u_1)/n$

$\mathrm{DELTA}\ v = (v_2 - v_1)/m,$

where n, u_1, u_2, m, v_1, and v_2 are the arguments to the most recent **glMapGrid2** command. Then, if *mode* is **GL_FILL**, the **glEvalMesh2** command is equivalent to:

```
for (j = j1;  j < j2; j += 1) {
    glBegin (GL_QUAD_STRIP);
    for (i = i1; i <= i2; i += 1) {
        glEvalCoord2($i . \mathrm{DELTA}\ u + u_1$, $j . \mathrm{DELTA}\ v + v_1$);
        glEvalCoord2($i . \mathrm{DELTA}\ u + u_1$, $(j+1) . \mathrm{DELTA}\ v + v_1$);
    }
    glEnd();
}
```

If *mode* is **GL_LINE**, then a call to **glEvalMesh2** is equivalent to:

```
for (j = j1;  j <= j2; j += 1) {
    glBegin(GL_LINE_STRIP);
    for (i = i1; i <= i2; i += 1)
        glEvalCoord2($i . \mathrm{DELTA}\ u + u_1$, $j . \mathrm{DELTA}\ v + v_1$);
    glEnd();
}
for (i = i1;  i <= i2; i += 1) {
    glBegin(GL_LINE_STRIP);
    for (j = j1; j <= j1; j += 1)
        glEvalCoord2)($i . \mathrm{DELTA}\ u + u_1$, $j . \mathrm{DELTA}\ v + v_1$);
    glEnd();
}
```

And finally, if *mode* is **GL_POINT**, then a call to **glEvalMesh2** is equivalent to:

```
glBegin (GL_POINTS);
for (j = j1;  j <= j2; j += 1) {
    for (i = i1; i <= i2; i += 1) {
        glEvalCoord2($i . \mathrm{DELTA}\ u + u_1$, $j . \mathrm{DELTA}\ v + v_1$);
    }
}
glEnd();
```

In all three cases, the only absolute numeric requirements are that if $i˜=˜n$, then the value computed from

i . DELTA u + u is exactly u ,

              1   2

and if $j˜=˜m$,

then the value computed from

j . DELTA v + v is exactly v .

              1   2

ERRORS

GL_INVALID_ENUM is generated if *mode* is not an accepted value.

GL_INVALID_OPERATION is generated if **glEvalMesh** is executed between the execution of **glBegin** and the corresponding execution of **glEnd**.

ASSOCIATED GETS

glGet with argument **GL_MAP1_GRID_DOMAIN**
glGet with argument **GL_MAP2_GRID_DOMAIN**
glGet with argument **GL_MAP1_GRID_SEGMENTS**
glGet with argument **GL_MAP2_GRID_SEGMENTS**

SEE ALSO

glBegin, **glEvalCoord**, **glEvalPoint**, **glMap1**, **glMap2**, **glMapGrid**