


UNISINOS

Modelagem com quádricas

Funções para definição de estados:

- ↓ Estilo de desenho:
 - ↓ `gluQuadricDrawStyle (GLUquadricObj *obj, GLenum drawStyle)`
 - ↓ `GLU_FILL`, `GLU_LINE`, `GLU_POINT` e `GLU_SILHOUETTE`
- ↓ Normais da superfície:
 - ↓ `gluQuadricNormals (GLUquadricObj *obj, GLenum normals)`
 - ↓ `GLU_NONE`, `GLU_FLAT` (normais dos vértices perpendiculares as faces) e `GLU_SMOOTH` (pondera normais com as das faces adjacentes)

8/10/2008 Leandro Tonietto 7




UNISINOS

Modelagem com quádricas

Funções para definição de estados:

- ↓ Orientação das normais:
 - ↓ `gluQuadricOrientation (GLUquadricObj *obj, GLenum orientation)`
 - ↓ `GLU_OUTSIDE` e `GLU_INSIDE`
- ↓ Considerar texturização ou não:
 - ↓ `gluQuadricTexture (GLUquadricObj *obj, GLenum texture)`
 - ↓ `GL_TRUE` (considera mapeamento de texturas) e `GL_FALSE` (não considera o mapeamento de textura).

8/10/2008 Leandro Tonietto 8

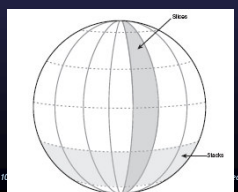
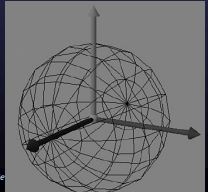


UNISINOS


Modelagem com quádricas

Desenhando quadricas:

- ↓ Esferas:
 - ↓ `gluSphere (GLUquadricObj *obj, GLdouble radius, GLint slices, GLint stacks)`
 - ↓ Slices == gomos. Stacks == "rodelas"

8/10/2008 Leandro Tonietto





UNISINOS


Modelagem com quádricas

Desenhando quadricas:

- ↓ Cilindros:
 - ↓ `gluCylinder (GLUquadricObj *obj, GLdouble baseRadius, GLdouble topRadius, GLdouble height, GLint slices, GLint stacks)`
 - ↓ `topRadius == zero`, permite criar cone.

8/10/2008 Leandro Tonietto

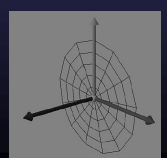



UNISINOS


Modelagem com quádricas

Desenhando quadricas:

- ↓ Discos:
 - ↓ `gluDisk (GLUquadricObj *obj, GLdouble innerRadius, GLdouble outerRadius, GLint slices, GLint loops)`
 - ↓ `innerRadius != zero`, permite criar discos com furos.

8/10/2008 Leandro Tonietto 11



UNISINOS

Modelagem com quádricas


Vantagens:

- ↓ Facilidade de modelagem (criação, propriedades, composição e desenho)

Desvantagens:

- ↓ Limitação de interação com o usuário
- ↓ Pode-se tornar difícil a criação de um cenário grande.
- ↓ Limitação de formas matemáticas

8/10/2008 Leandro Tonietto 12



Modelagem com quádricas

Programação:

- ↓ Variável de classe: *Exemplo Snowman*
`GLUQuadriObj *obj;`
- ↓ No init:
`obj = gluNewQuadric();`
- ↓ No RenderScene (ou display):
`gluQuadricNormals(obj, GLU_SMOOTH);`
`glTranslatef(x, y, z);`
`gluSphere(obj, 5, 5, 5);`
- ↓ No final da execução do main:
`gluDeleteQuadric (obj);`

8/10/2008 Leandro Tonietto 13

Tarefa para ser feita em aula

Utilizando quádricas e formas primitivas modele os seguintes objetos:



Não é necessário modelar todos os detalhes, apenas os objetos que dão a forma básica do objeto

8/10/2008 Leandro Tonietto 14

Curvas e Superfícies Paramétricas

- Alguns objetos são muito complexos para serem modelados com combinações de “primitivas” matemáticas e com geometria descritiva.
- Supondo que momento não há meios ou não se tem uma ferramenta de modelagem disponível para criação de uma malha poligonal.
- Uma solução seria o uso de curvas, porém também não há como memorizar todas as curvas possíveis necessárias para formar objetos mais complexos.
- As vezes é necessário algum tipo de suporte ou controle para permitir interação com usuário, no sentido de modificar completamente a forma geométrica aproximada.
- Solução: superfícies paramétricas.

8/10/2008 Leandro Tonietto 15

Curvas e Superfícies Paramétricas

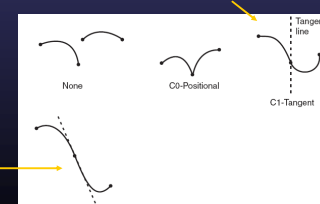
- OpenGL dá suporte a curvas paramétricas.
- Conceitos:
 - ↓ Para uma curva o parâmetro paramétrico é u .
 - ↓ Para uma superfície são u e v .
 - ↓ As curvas são representadas por pontos de controle e são avaliadas a cada instante u de tempo.
 - ↓ Ordem da curva é dada pelo número de pontos; por exemplo, 2 pontos é uma de 2ª ordem, de 3 é de 3ª ordem, ...
 - ↓ O grau é determinado por: ordem da curva - 1. Por exemplo: uma de 4ª ordem é uma de grau 3 (cúbica).
 - ↓ Curvas cúbicas são de longe as mais utilizadas, pois as de maior ordem podem produzir oscilações indesejadas ou até incontroláveis (muita variação com mudanças mínimas).

8/10/2008 Leandro Tonietto 16

Curvas e Superfícies Paramétricas

Continuidade:

- ↓ Problemas:
 - ↓ Curvas com grau maior que três podem produzir comportamento inesperado para a modelagem.
 - ↓ Entretanto, uma curva cúbica não é suficiente para representar uma curva “interessante”.
- ↓ Solução: conectar curvas. *Mesma tangente na conexão*
- ↓ Tipos ou graus de conexão (continuidade):



Mesma tangente e mesma taxa de variação da tangente

8/10/2008 Leandro Tonietto 17

Curvas em OpenGL

Desenho de curvas em OpenGL é feito com **Evaluators** (avaliadores)

- ↓ Uma função avaliadora (*evaluator function*) é executada sobre os pontos de controle da curva para cada instante u .
 - ↓ Elas avaliam pontos da curva dado um instante u da curva.
- ↓ Desenharam curvas e superfícies Bézier.
- ↓ Processo é simples:
 1. Definir pontos de controle
 2. Criar uma matriz de cálculo (função para a curva): `glMap1f()`.
 3. Habilitar *evaluator* e avaliar (desenhar) pontos da curva a cada instante u . Comandos: `glEnable(GL_MAP1_VERTEX_3)` e `glEvalCoord(u)`


Ver código das páginas 391–393 do livro *OpenGL Superbible* [1].

8/10/2008 Leandro Tonietto 18

Curvas em OpenGL

Alternativa para desenhar uma curva:

- ❗ Criar uma grade e avaliar
 - ❗ // 100 pontos de 0.0 a 100.0
 - `glMapGrid1d(100, 0.0, 100.0);`
 - ❗ // desenha linhas na grade
 - `glEvalMesh1(GL_LINE, 0, 100);`
- ❗ Abordagem mais simples, do que avaliar a curva ponto-a-ponto.



8/10/2008 Leandro Tonietto 19


Curvas em OpenGL

Evaluator X grid:

```
glMapGrid1d(100, 0.0, 100.0);
glEvalMesh1(GL_LINE, 0, 100);
```

Equivalência a:

```
glBegin(GL_LINE_STRIP);
  for(i = 0; i <= 100; i++) {
    glEvalCoord1f((GLfloat) i);
  }
glEnd();
```



8/10/2008 Leandro Tonietto 20

Superfícies em OpenGL

Uma superfície é uma mistura ou combinação de curvas.


OpenGL fornece suporte a superfícies Bézier.

Desenho é feito com uma grade bidimensional e com um *evaluator* para cada curva.

❗ Processo:

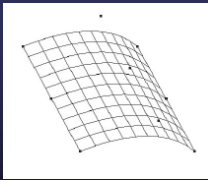

1. Definir pontos de controle das curvas
2. Criar uma matriz de cálculo (função para as curvas): `glMap2f()`.
3. Habilitar *evaluator* e criar grade para as curvas. Comandos: `glEnable(GL_MAP2_VERTEX_3)`, `glMapGrid2()` e `glEvalMesh2()`;
4. `glEvalMesh2(GL_LINE)` desenha superfícies com linhas e `glEvalMesh2(GL_FILL)` desenha preenchida.
5. `glEnable(GL_AUTO_NORMAL)` computa normais automaticamente.


Ver código das páginas 397–399 do livro *OpenGL Superbible* [1].



8/10/2008 Leandro Tonietto 21

Superfícies em OpenGL



8/10/2008 Leandro Tonietto 22

NURBS


Problema: desenhar curvas e superfícies Bézier e garantir continuidade C2 é complexo.

Solução: utilizar NURBS

Suporte a NURBS é feito pela GLU

Conceito: *Knots*

- ❗ Sequência de valores que influenciam os pontos de controle manter uma suavidade da continuidade.
- ❗ 2 *knots* por ponto de controle, devem estar no mesmo domínio que *u* e *v* e devem estar em ordem crescente.




8/10/2008 Leandro Tonietto 23

NURBS

Criação de NURBS:

❗ Segue lógica de orientação a objetos:

1. “Criar” objeto NURBS
`GLUnurbsObj *pNurb = NULL;`
2. Inicializar objeto
`pNurb = gluNewNurbsRenderer();`
3. Definir propriedades, pontos de controle e *knots*
// próximos slides!
4. Eliminar memória alocada.
`gluDeleteNurbsRenderer(pNurb);`



8/10/2008 Leandro Tonietto 24

UNISINOS

NURBS

Propriedades para objeto NURBS:

- Sempre deve ser passado sobre qual objeto NURBS será definida a propriedade.
- Espessura da superfície:**

```
gluNurbsProperty(pNurb, GLU_SAMPLING_TOLERANCE, 25.0f);
```
- Preenchimento**

```
gluNurbsProperty(pNurb, GLU_DISPLAY_MODE, (GLfloat)GLU_FILL);
```

// pode ser usado também GLU_OUTLINE_POLYGON

8/10/2008 Leandro Tonietto 25

UNISINOS

NURBS

Definindo a superfície:

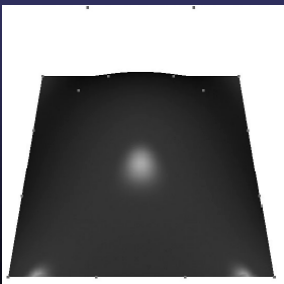
```
gluBeginSurface(pNurb);
gluNurbsSurface(pNurb,
8, knots, // knots para u
8, knots, // knots para v
12, // distância entre pontos u
3, // distância entre pontos v
&ctrl[0][0][0], // pontos de controle
4, 4, // ordem da superfície em u e v
GL_MAP2_VERTEX_3); // tipo de superfície
gluEndSurface(pNurb);
// Pontos de controle:
// x,y,z para u e v em quatro pontos
GLfloat ctrl[4][4][3];
// 2 knots por ponto de controle
GLfloat knots[8];
```

8/10/2008 Leandro Tonietto 26

UNISINOS

NURBS

Exemplo: páginas 404–405 do livro [1]



8/10/2008 Leandro Tonietto 27

UNISINOS

Trimming

- Aparar ou tirar excessos de uma superfícies.
- Comum para aparar bordas da superfície. Também pode-se fazer buracos na superfície.
- Definir uma sequência de pontos no domínio u e v no sentido horário (superfícies são desenhadas em sentido anti-horário)

```
// valores entre 0 e 1 para u e v
GLfloat points[1][1];
```

Colocar dentro do desenho da NURBS:

```
glBeginTrim(pNurb);
gluPwlCurve(pNurb, 4, &points[0][0], 2, GLU_MAP1_TRIM_2);
glEndTrim(pNurb);
```

Objeto Num. pontos pontos pontos dimensão

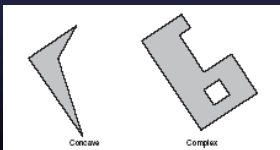
Ver código da páginas 407–408 do livro OpenGL Superbible [1].

8/10/2008 Leandro Tonietto 28

UNISINOS

Tessellation

- Por questões de complexidade performance, OpenGL garante apenas o tratamento de polígonos convexos. Os côncavos e complexos (com cruzamento de arestas ou com buracos), “*não são manipulados*” no OpenGL.
- Porém, as vezes é necessário...



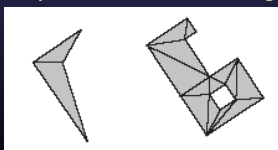
Côncavo Complex

8/10/2008 Leandro Tonietto 29

UNISINOS

Tessellation

- Solução trivial: quebrar em triângulos.
- Tarefa pode se tornar complexa de se fazer manualmente.
- GLU fornece o processo *tessellation*. Um *tessellator* é um objeto que subdivide um polígono complexo em vários triângulos.



8/10/2008 Leandro Tonietto 30

UNISINOS

Tessellation


Assim como NURBS, deve-se criar um objeto *Tessellator*:

```

GLUtesselator *pTess;
pTess = gluNewTess();
. . .
// Do some tessellation
. . .
gluDeleteTess(pTess);

```

8/10/2008 Leandro Tonietto 31



UNISINOS


Tessellation

Assim como NURBS, deve-se criar um objeto *Tessellator*:

1. Criação do objeto *Tessellator*
2. Definir propriedades.
3. Iniciar polígono
4. Iniciar contorno
5. Seta vértices *tessellator* para o contorno
6. Fim contorno
7. Retorna ao passo 4 para mais contornos
8. Fim do polígono

Ver código da páginas 414—418 do livro *OpenGL Superbible* [1].

8/10/2008 Leandro Tonietto 32




UNISINOS

Exercícios

- 1. Criar uma superfície Bézier e permitir edição de pontos.
- 2. Criar uma superfície NURBS e permitir a edição dos pontos de controle e dos *knots*.
- 3. Fazer a figura do mapa do RS habilitando 3 possibilidades de desenho: outline, preenchido e usando *tessellation*.

8/10/2008 Leandro Tonietto 33



UNISINOS

Referências bibliográficas

1. WRIGHT Jr., Richard S; LIPCHAK, Benjamin; HAEMEL, Nicholas. **OpenGL Superbible: Comprehensive Tutorial and Reference**. 4 ed. Addison-Wesley, 2007.

8/10/2008 Leandro Tonietto 34

