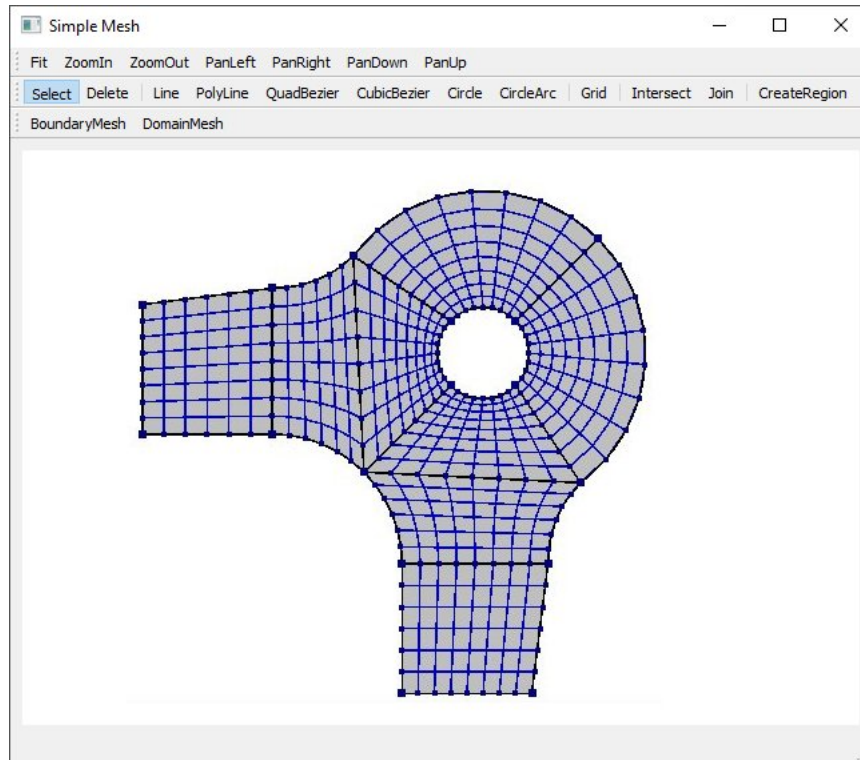


5º Trabalho: Programa gráfico interativo com Qt e OpenGL para geração de malhas por mapeamento em regiões

Implementação de algoritmos de geração de malha por mapeamento transfinito discreto



Complemente o programa gráfico fornecido na *homepage* da disciplina:

<https://www.tecgraf.puc-rio.br/~lfm/compgraf-261> (procure quinto trabalho).

Este programa utiliza sistema de interface Qt e o sistema gráfico OpenGL.

O programa fornecido trata de dois tipos de curva: segmento de linha reta e linha poligonal

Deve-se adicionar ao programa as curvas que foram implementadas no terceiro trabalho.

Também devem ser completados no código os procedimentos feitos no quarto trabalho para interseção de dois segmentos e para verificação de inclusão de ponto em polígono.

O entendimento do código do programa é parte do trabalho.

Solicitado

1. Completar as linhas de código no arquivo `glcanvas.py` que estão indicadas pelos comentários
**** COMPLETE HERE: GLCANVAS_XX ****.
Essas linhas de código podem ser obtidas das soluções do segundo trabalho e do terceiro trabalho.
2. Implementação das mesmas curvas do terceiro trabalho.
3. Completar as linhas de código no arquivo `compgeom.py` que estão indicadas pelos comentários
**** COMPLETE HERE: COMPGEOM_XX ****.
Essas linhas de código foram implementadas no quarto trabalho para interseção de dois segmentos e para verificação de inclusão de ponto em polígono.

4. Os algoritmos de geração de malha dos que estão nos arquivos `transfinbilinear.py` e `transfintrilinear.py` devem ser complementados onde está indicado pelos comentários:
**** COMPLETE HERE: TRANSFINBILINEAR_XX ****.
**** COMPLETE HERE: TRANSFINTRILINEAR_XX ****.
sendo:
`transfinbilinear.py`: geração de malha em região usando o algoritmo de mapeamento transfinito discreto bilinear.
`transfintrilinear.py`: geração de malha em região usando o algoritmo de mapeamento transfinito discreto trilinear.
Esses algoritmos estão descritos nas folhas anexas.
As malhas são retornadas pelos métodos de geração de cada classe em duas listas:
`pts`: lista de pontos do tipo `Pnt2D`;
`conn`: lista de listas de conectividade dos elementos da malha com índices para a lista de pontos.
5. Obrigatoriamente devem ser criados ícones (imagens) para os botões da interface gráfica.
6. Enviar via e-mail em um arquivo `.zip` com os arquivos `myapp.ui`, `myapp.py`, todos os arquivos `*.py`, que correspondem à solução do trabalho, e uma pasta `icons` com os arquivos de imagens dos ícones criados para as ações de interface.
7. Gerar uma malha semelhante à apresentada na figura na primeira página e mostrar uma imagem da tela do programa com essa malha.

Algoritmo de geração de malha em região por triangulação de Delaunay com restrições

A geração de malha pelo método de triangulação Delaunay com restrições usa o algoritmo Triangle de Jonathan Richard Shewchuk – “A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator” (<http://www.cs.cmu.edu/~quake/triangle.html>). Direitos autorais: Jonathan Richard Shewchuk (<http://people.eecs.berkeley.edu/~jrs/>).

Triangle é implementado em C, e a versão adotada usa um *wrapper* Python (<https://rufat.be/triangle>) em torno da implementação C desenvolvido pelo The Block Research Group (BRG) no Institute of Technology in Architecture at ETH Zürich (<https://block.arch.ethz.ch/brg/about>).

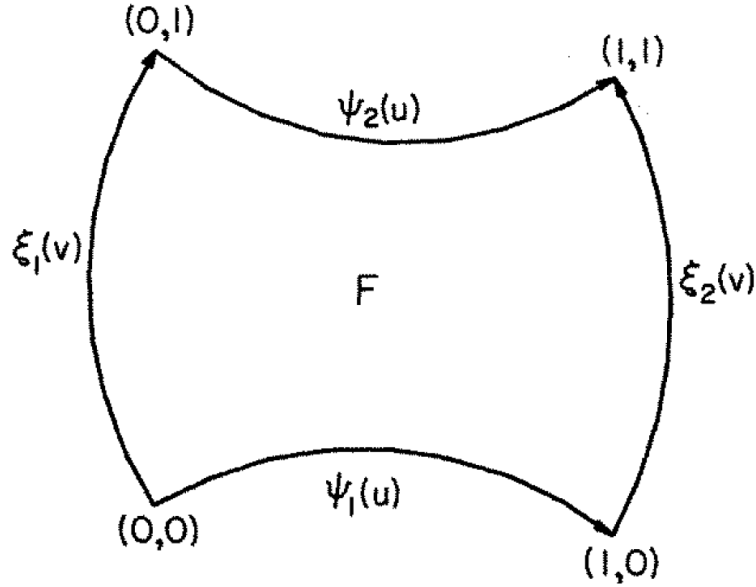
O uso do Triangle é limitado a uma região com apenas um loop externo (sem furos).

Triangle gera triangulações de Delaunay restritas à lista fornecida de pontos de loop. No entanto, para tornar o uso de Triangle mais robusto, o único parâmetro especificado é uma restrição de área máxima do triângulo, calculada como a área de um triângulo equilátero cujo lado é igual ao tamanho máximo do segmento do loop de entrada. Como este é o único parâmetro especificado o algoritmo Triangle, no caso de regiões côncavas, gera triângulos fora do loop dado e dentro de seu fecho convexo. Portanto, esses triângulos externos e os pontos de seus vértices são excluídos na etapa de pós-processamento.

Além disso, Triangle pode gerar novos pontos de triangulação no loop externo. Nenhuma correção é feita para isso.

Instalação: `pip install triangle`

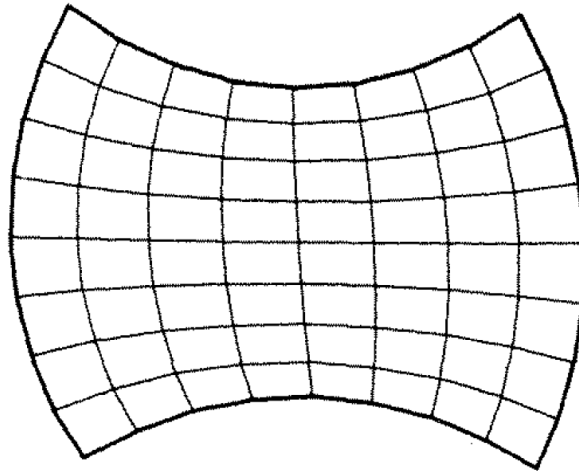
Algoritmo de geração de malha em região por mapeamento transfinite discreto bilinear



Bilinear projector: co-ordinate system and boundary curves

$$\begin{aligned}
 (\mathcal{P}_1 \oplus \mathcal{P}_2)[F] &\equiv \mathcal{P}_1[F] + \mathcal{P}_2[F] - \mathcal{P}_1\mathcal{P}_2[F] \\
 &= P_B(u, v) \\
 &= (1-v)\psi_1(u) + v\psi_2(u) + (1-u)\xi_1(v) + u\xi_2(v) \\
 &\quad - (1-u)(1-v)F(0, 0) - u(1-v)F(0, 1) \\
 &\quad - uvF(1, 1) - (1-u)vF(1, 0) \quad 0 \leq u \leq 1, 0 \leq v \leq 1
 \end{aligned}$$

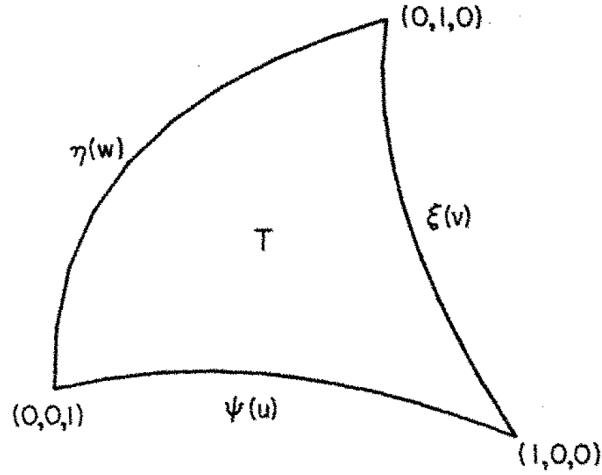
$$\{\xi_1(v_i), \xi_2(v_i)\} i = 1, n, \quad \{\psi_1(u_j), \psi_2(u_j)\} j = 1, m$$



$$\mathcal{P}_1 \oplus \mathcal{P}_2$$

Bilinear projector: $\mathcal{P}_1 \oplus \mathcal{P}_2$

Algoritmo de geração de malha em região por mapeamento transfinite discreto trilinear



Trilinear projector: co-ordinate system and boundary curves

$$\mathcal{N}_1 \equiv N_1(u, v, w) = \left(\frac{u}{1-v} \right) \xi(v) + \left(\frac{w}{1-v} \right) \eta(1-v)$$

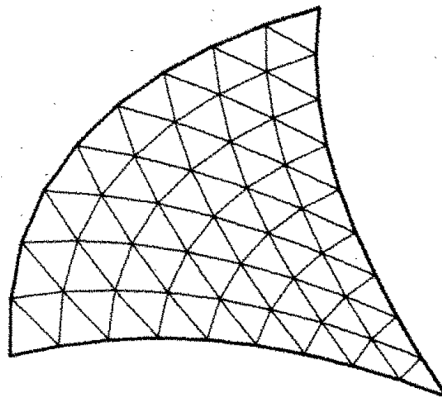
$$\mathcal{N}_2 \equiv N_2(u, v, w) = \left(\frac{v}{1-w} \right) \eta(w) + \left(\frac{u}{1-w} \right) \psi(1-w)$$

$$\mathcal{N}_3 \equiv N_3(u, v, w) = \left(\frac{w}{1-u} \right) \psi(u) + \left(\frac{v}{1-u} \right) \xi(1-u)$$

$$0 \leq u \leq 1, \quad 0 \leq v \leq 1, \quad 0 \leq w \leq 1, \quad u + v + w = 1$$

$$\begin{aligned} \mathcal{Q} \equiv Q(u, v, w) = & \frac{1}{2} \left[\left(\frac{u}{1-v} \right) \xi(v) + \left(\frac{w}{1-v} \right) \eta(1-v) + \left(\frac{v}{1-w} \right) \eta(w) + \left(\frac{u}{1-w} \right) \psi(1-w) \right. \\ & \left. + \left(\frac{w}{1-u} \right) \psi(u) + \left(\frac{v}{1-u} \right) \xi(1-u) - w\psi(0) - u\xi(0) - v\eta(0) \right] \end{aligned}$$

$$\{\psi(u_i), \xi(v_i), \eta(w_i); i = 1, n\}$$



$Q[T]$

Trilinear projector: $\mathcal{Q}$