

Visualização de Curvas de Nível em Pós-Processamento Para Um Sistema de Simulação Eletromagnética

Cátia M. Freire de Pina (IC), Prof. Dr. Marli F. G. Hernandez (PQ)

Resumo

Propósito deste projeto de pesquisa é o desenvolvimento de funcionalidades relacionadas à visualização de curvas de campos eletromagnéticos, que serão integradas a Pós-processamento de um ambiente computacional de simulações eletromagnéticas desenvolvido no Departamento de Micro-ondas e óptica (DMO) da Faculdade de Engenharia Elétrica e de Computação (FEEC) da UNICAMP.

Palavras Chave: Simulações, Eletromagnética

Introdução

Nos dias atuais, diversas áreas do conhecimento fazem uso de recursos computacionais para possibilitar, viabilizar, facilitar e manter o desenvolvimento de estudos, pesquisas, ferramentas e produtos. A diversidade de recursos computacionais é extremamente ampla e estende-se desde o suporte físico, baseado em hardware, à técnicas avançadas de análise, processamento, comunicação e reconhecimento de padrões em dados, baseadas em programas de computador. Neste contexto, a simulação computacional oferece um grande potencial para concretizar os diversos trabalhos referentes à reprodução desses fenômenos, efetuando os cálculos que os regem em tempo finito ou, em certos casos, em tempo real, e tornando-os observáveis aos humanos.

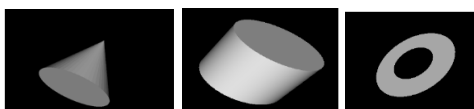
Resultados e Discussão

```
# create a rendering window and renderer # create source
ren = vtk.vtkRenderer()
renWin = vtk.vtkRenderWindow()
renWin.AddRenderer(ren)
WIDTH=640
HEIGHT=480
renWin.SetSize(WIDTH, HEIGHT)
# create a renderwindowinteractor # mapper
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
coneMapper = vtk.vtkPolyDataMapper()
coneMapper.SetInput(cone)
# actor
coneActor = vtk.vtkActor()
coneActor.SetMapper(coneMapper)
# assign actor to the render
ren.AddActor(coneActor)
# enable user interface if
iren.Initialize()
renWin.Render()
iren.Start()

# create cone
cone = vtk.vtkConeSource()
cone.SetResolution(60)
cone.SetCenter(-2, 0, 0)
# create source
source = vtk.vtkCylinderSource()
source.SetCenter(0, 0, 0)
source.SetRadius(5.0)
source.SetHeight(7.0)
source.SetResolution(100, 0)

source = vtk.vtkDiskSource()
source.SetInnerRadius(1)
source.SetOuterRadius(2)
source.SetRadialResolution(10)
source.SetCircumferentialResolution(10)
source.Update()

# actor
coneActor = vtk.vtkActor()
coneActor.SetMapper(coneMapper)
# assign actor to the render
ren.AddActor(coneActor)
# enable user interface if
iren.Initialize()
renWin.Render()
iren.Start()
```



As cores apresentadas no ambiente gráfico do VTK normalmente são definidas pelo mapeamento de dados escalares em cores, em um processo similar ao realizado pelo H5topng. Para isso, o VTK disponibiliza o objeto `vtkLookupTable`, responsável em fornecer os mecanismos de mapeamento de dados em cores. Através do `vtkLookupTable`, é possível configurar a quantidade, os valores e o espaço ou rampa de cores – como o RGB (Red-Green-Blue-Alpha) ou HSVA (Hue-Saturation-Value-Alpha) – e definir os limites inferior e superior de mapeamento dos dados escalares.

Conclusões

O projeto era dividido em várias etapas, mas devido a alguns percalços, da conclusão do curso e na deficiência em obter o software a única etapa concluída com sucesso foi do estudo do sistema VTK, implementação de exemplos para compreensão do pipeline de visualização (transformação de dados-fontes em diretivas gráficas). Gostaria de terminar o projeto que é bastante interessante que visa o desenvolvimento da capacidade de visualização de curvas de nível de campos eletromagnéticos.

Agradecimentos

O meu agradecimento especial a professora Marli pela oportunidade que me deu, ao Adriano pela ajuda com o ambiente computacional e ao CNPQ pela ajuda financeira para execução do projeto.

[1] Schoroder, W. J.; Martin, K.; Lorensen, W., “The Visualization Toolkit - An Object-Oriented Approach to 3D Graphics,” 4ª. Edition. Prentice-Hall, 2006.

[2] Kitware, Inc., “The VTK User’s Guide,” 11ª. Edition. New York, 2010.

[3] <http://www.vtk.org/Wiki/VTK/Examples>. Acesso em 08/12/2014.

[4] Oskooi, A. F.; Roundy, D.; Ibanescu, M.; Bermel, P.; Joannopoulos, J. D.; Johnson, S. G., “MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method,” *Computer Physics Communications* 181, 687-702 (2010).

[5] Lambert, E.; Fiers, M.; Nizamov, S.; Tassaert, M.; Johnson, S.G.; Bienstman, P.; Bogaerts, W., “Python Bindings for the Open Source Electromagnetic Simulator Meep,” *Computing in Science & Engineering* 13, 53-65 (2011).