UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL

FABRÍCIO LORENI DA SILVA

ADAPTAÇÃO DO CÓDIGO GEANT4 PARA CONVERSÃO DE IMAGENS *DICOM* EM *PHANTOM* VIRTUAL

DISSERTAÇÃO

CURITIBA
2013

FABRÍCIO LORENI DA SILVA

ADAPTAÇÃO DO CÓDIGO GEANT4 PARA CONVERSÃO DE IMAGENS *DICOM* EM *PHANTOM* VIRTUAL

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de “Mestre em Ciências” - Área de Concentração: Engenharia Biomédica.

Orientador: Prof. Dr. Sergei Anatolyevich Paschuk

Coorientador: Prof. Dr. Valeriy Viktorovich Denyak

CURITIBA
2013

Dados Internacionais de Catalogação na Publicação

S586 Silva, Fabrício Loreni da

Adaptação do código Geant4 para conversão de imagens DICOM em *phantom* virtual / Fabrício Loreni da Silva. – 2013.

71 f. : il. ; 30 cm

Orientador: Sergei Anatolyevich Paschuk.

Coorientador: Valeriy Denyak.

Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial. Curitiba, 2013.

Bibliografia: f. 58-61.

1. Tomografia computadorizada. 2. Feixes de prótons. 3. Imagens e fantasmas (Radiologia). 4. Diagnóstico por imagem – Técnicas digitais. 5. Comunicação na medicina. 6. Monte Carlo, Método de. 7. Simulação (Computadores). 8. Engenharia elétrica – Dissertações. I. Paschuk, Sergei Anatolyevich, orient. II. Denyak, Valeriy, coorient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial. IV. Título.

Biblioteca Central da UTFPR, Câmpus Curitiba

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

Título da Dissertação Nº. 633

"Adaptação do Código Geant4 Para Conversão de Imagens Dicom em *Phantom* Virtual"

por

Fabrício Loreni da Silva

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM CIÊNCIAS – Área de Concentração: Engenharia Biomédica, pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial – CPGEI – da Universidade Tecnológica Federal do Paraná – UTFPR, às 14h do dia 01 de abril de 2013. O trabalho foi aprovado pela Banca Examinadora, composta pelos doutores:

Prof. Sergei Anatolyevich Paschuk, Dr.
(Presidente – UTFPR)

Prof. Ricardo Tadeu Lopes, Dr.
(UFRJ)

Prof. Ivan Gennadievitch Evseev, Dr.
(UTFPR)

Prof. Edney Milhoretto, Dr.
(UTFPR)

Visto da coordenação:

Prof. Ricardo Lüders, Dr.
(Coordenador do CPGEI)

Dedico este trabalho a Deus e a meus pais, Loreni e Aneli.

AGRADECIMENTOS

Agradeço a Deus por essa grande conquista.

Agradeço ao Prof. Dr. Sergei Anatolyevich Paschuk, Prof. Dr. Valeriy Denyak, Prof. Dr. Edney Milhoretto, Prof. Dr. Ricardo Tadeu Lopes e ao Prof. Dr. Ivan Evseev pela contribuição cientifica e apoio.

Agradeço ao Prof. Dr. Hugo Reuters Schelin pela inestimável contribuição cientifica, apoio e compreensão.

Agradeço aos amigos do mestrado, Ana Paula Bunick e Flávia Aparecida Franck, e aos colegas de laboratório, Allan Felipe Nunes Perna, Carla Kozuki, Flávia Del Claro, Janine Nicolosi Corrêa, Jaqueline Kappke, Jose Carlos de Camargo Lourenco, Luis Filipe Colla, Rita de Cássia de Lima Silva, aos amigos, Gabriela Pelissari Machado e Jorge Luís Corrêa da Silva, a CAPES pelo apoio financeiro e a todos da UTFPR que colaboraram diretamente ou indiretamente com este trabalho.

Gostaria, ainda, de agradecer aos meus pais, Loreni da Silva e Aneli Izabel da Silva, meus irmãos, Priscilla Carolina da Silva e Lucas Gabriel da Silva Cerutti, e a minha namorada, Elisandre Caroline dos Santos, pelo incentivo, apoio e compreensão.

Reitero aqui a minha calorosa consideração e profunda gratidão a todos, que ao longo desses anos, colaboraram para a realização desse trabalho.

RESUMO

SILVA, Fabricio Loreni. Adaptação do Código Geant4 para Conversão de Imagens *DICOM* em *Phantom* Virtual. 2013. 71f. Dissertação (Mestrado em Engenharia Biomédica) – Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

Este trabalho apresenta a adaptação do código Geant4 para conversão de imagens *DICOM* (*Digital Imaging and Communications in Medicine*) de crânio, obtidas em tomografia convencional (*CT*), em um *phantom* antropomórfico virtual. O trabalho foi baseado no exemplo médico denominado "*Código Dicom"*, disponibilizado pelos desenvolvedores do código Geant4. Durante a execução do trabalho foram feitas reestruturações no exemplo "*Código Dicom"* para a conversão direta de imagens tomográficas em um *phantom* virtual. Foram retirados do código todos os passos referentes aos eventos físicos nucleares. Foi reformulado o arquivo *DicomHandler.cc* para não realizar a compressão dos *pixels* da imagem de *CT*. Em seguida foi realizada a conversão direta de imagens tomográficas, de um *phantom* físico de polietileno (PEAD) com núcleo central de acrílico e de um crânio real humano, em *phantoms* virtuais para o código Geant4. Os resultados demonstraram que com este código é possível a reconstrução de áreas anatômicas com geometrias complexas, partindo do uso de imagens tomográficas reais.

Palavras chave: Tomografia Computadorizada, *Phantom, DICOM, pCT,* Método Monte Carlo.

ABSTRACT

SILVA, Fabricio Loreni. ADAPTATION OF GEANT4 CODE FOR CONVERSION OF *DICOM* IMAGES INTO A VIRTUAL *PHANTOM*. 2013. 71f. Dissertação (Mestrado em Engenharia Biomédica) – Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

This work presents the adaptation of the Geant4 code for converting *DICOM* (Digital Imaging and Communications in Medicine) images of a skull, obtained in conventional tomography (*CT*), into a virtual anthropomorphic phantom. The work was based on the medical example named "*Dicom Code*" provided by the developers of the code Geant4. During the execution, restructurings using the "*Dicom Code*" example were made to achieve the direct conversion of tomographic images into a virtual phantom. All the steps referring to nuclear physical events were removed. The file *DicomHandler.cc* was reformulated in order to avoid the pixels compression of the *CT* image. The *CT* images of a physical polyethylene (PEAD) phantom with acrylic core and a real human skull were then converted into virtual phantoms for the code Geant4. The results showed that with this code, it may be possible the reconstruction of anatomical areas with complex geometries, based on the use of real tomographic images.

Key-words: Computed Tomography, *Phantom*, *DICOM*, *pCT*, Monte Carlo Method.

LISTA DE FIGURAS

Figura 1 - Phantom simples de uma caixa sólida..... 21
Figura 2 - Phantom de polipropileno (azul), colimador (vermelho), detector (verde) e o feixe de prótons..... 22
Figura 3 - Representação do protótipo. 23
Figura 4 - Câmara de espalhamento CV-28 e mecanismo de movimentação do phantom..... 23
Figura 5 - Estrutura dos phantoms, à esquerda da figura com núcleo em barra e à direita com esfera..... 24
Figura 6 - Imagens de CT do phantom de PEAD. Corte sagital, coronal e a reconstrução 3D. 25
Figura 7 - Imagem do phantom físico de tórax com diferentes tecidos..... 26
Figura 8 - Reconstrução tomográfica do phantom físico de tórax..... 27
Figura 9 - Intensidade (Z) por quantidade de pixel (X e Y) da imagem de CT de tórax..... 28
Figura 10 - Imagem DICOM de um corte axial de crânio. (A) Septo Nasal, (B) Osso Zigomatico, (C) Protuberância do Occipital, (D) Canal Auditivo..... 37
Figura 11 - Intensidade (Z) por quantidade de pixel (X e Y) da imagem de CT de crânio..... 38
Figura 12 - Imagem dos 20 cortes que compõem o crânio..... 39
Figura 13 - (A) Reconstrução 3D. (B) Terceiro corte axial da base do crânio..... 40
Figura 14 - Imagens de CT (A), voxel (B), conjunto de voxel formado a partir das imagens de CT..... 43
Figura 15 - Terminal com os comandos necessários para carregar env.sh..... 45
Figura 16 - Phantom compilado..... 45
Figura 17 - Phantom pronto para ser visualizado. 46
Figura 18 - Visualização da base phantom de crânio através do programa Dawn. 46
Figura 19 - Matriz 512 X 512 do phantom de crânio. 48
Figura 20 - Região A da matriz da Figura 19 ampliada para melhor visualização..... 48
Figura 21 - Matriz 512x512 do phantom de crânio selecionada em Excel..... 49
Figura 22 - Matriz 512x512 do phantom de crânio em escala de cinza..... 49
Figura 23 - (A) Imagem Dicom de CT, (B) matriz do phantom de crânio em escala de cinza..... 50
Figura 24 - O phantom antropomórfico parcial virtual reconstruído no código Geant4. Visualização da base do phantom pelo programa Dawn (A) Septo Nasal, (B) Osso Zigomatico, (C) Protuberância do Occipital, (D) Canal Auditivo..... 50
Figura 25 - Imagem 3D do phantom de PEAD..... 53
Figura 26 - (A) Corte axial de crânio (B) Imagem da tomografia do phantom de tórax..... 54
Figura 27 - Intensidade (Z) por quantidade de pixel (X e Y) da imagem de (A) Tórax e da imagem de (B) Crânio..... 55

LISTA DE TABELAS

Tabela 1 - Lista com os materiais, elemento dos materiais, densidade dos materiais e quantidade de elementos que compõem os tecidos do corpo humano. 36
Tabela 2 - Faixe de densidade em g/cm³ e os respectivos materiais do tórax. 42
Tabela 3 - Tecido do tórax e seu número de identificação. 43
Tabela 4 - Tecido do tórax e seu número de identificação. 47
Tabela 5 - Materiais do phantom de PEAD e seus respectivo números de identifcação. 51
Tabela 6 - Matriz 128 X 128 do phantom de PEAD. 52

LISTA DE SIGLAS E SÍMBOLOS

ACR	*American College of Radiology*
"Código Dicom"	Exemplo médico do código Geant4.
CT	Tomografia Computadorizada por Feixes de Raios X.
CERN	Organização Europeia para a Pesquisa Nuclear *(Europen Organization for Nuclear Research).*
DICOM	Comunicação de Imagens Digital em Medicina *(Digital Imaging and Communications in Medicine).*
IEN	Instituto de Energia Nuclear.
LLUMC	Centro Médico da Universidade de Loma Linda *(Loma Linda University Medical Center).*
MeV	Mega Elétron-volt.
NEMA	*National Electronics Manufactures's Association*
OSI	*System Interconnection*
pCT	Tomografia Computadorizada por Feixes de Prótons.
Pixel	*Picture Element*
PEAD	Polietileno de Alta Densidade.
TCP/IP	*Transmission Control Protocol - Internet Protocol*
UERJ	Universidade Estadual do Rio de Janeiro.
UFRJ	Universidade Federal do Rio do Janeiro.
UTFPR	Universidade Tecnológica Federal do Paraná.
Voxel	*Volumetric Picture Element*

SUMÁRIO

1 INTRODUÇÃO 9
1.1 MOTIVAÇÕES 9
1.2 OBJETIVOS 11
1.2.1 Objetivo Geral 11
1.2.2 Objetivos Específicos 11
1.3 ESTRUTURA DO TRABALHO 12
2 FUNDAMENTAÇÃO TEÓRICA 13
2.1 REVISÃO DA LITERATURA 13
2.1.1 Imagem. 13
2.1.2 Breve Histórico da Tomografia Computadorizada 14
2.1.3 Tomografia Computadorizada por Transmissão de Raios X 15
2.1.4 Considerações Gerais Sobre Imagem de Tomografia Computadorizada 17
2.1.5 *DICOM* 18
2.2 CONSTRUÇÃO DE GEOMETRIAS NO CÓDIGO GEANT4 20
2.2.1 Protótipo do IEN/CNEN 22
2.3 *PHANTOM* SIMPLES DE POLIETILENO 24
2.3.1 Protótipo da LLUMC 25
2.4 CONVERSÃO DE IMAGENS DICOM DE UM *PHANTOM* FÍSICO DE TÓRAX EM UM *PHANTOM* VIRTUAL 26
3 MATERIAIS E MÉTODOS 28
3.1 PROGRAMAS COMPUTACIONAIS 28
3.2 INSTALAÇÃO DO GEANT4 RELEASE 4.9.3.P01 EM LINUX 29
3.3 TOMOGRAFIA DE CRÂNIO 35
3.4 *"CÓDIGO DICOM"* 40
3.4.1 Estrutura do *"Código Dicom"* 40
3.4.2 Funcionamento do *"Código Dicom"* 41
3.4.3 Modificações no *"Código Dicom"* 43
3.4.4 Comandos para a Conversão das Imagens de *CT* em um *Phantom* de Crânio 44
4 RESULTADOS E DISCUSSÕES 47
4.1 APRESENTAÇÃO DO *PHANTOM* VIRTUAL DE CRÂNIO 47
4.1.1 Arquivo de Saída *DICOM.gdcm* 47
4.1.2 Arquivo de Saída *DICOM.eps* 50
4.2 APRESENTAÇÃO DO *PHANTOM* VIRTUAL DE PEAD 51
4.3 DISCUSSÕES SOBRE OS PROGRAMAS UTILIZADOS 53
4.4 COMPARAÇÃO DO *PHANTOM* FÍSICO DE TÓRAX E A IMAGEM DICOM DE CRÂNIO 54
5 CONCLUSÕES 56
SUGESTÕES PARA TRABALHOS FUTUROS 57
REFERÊNCIAS 58
ANEXO A – CÓDIGOS PARA CONSTRUÇÃO DO *PHANTOM* DE POLIPROPILENO, COLIMADOR E DETECTOR. 62
ANEXO B – PARTE DOS CÓDIGOS DO ARQUIVO *DICOMHANDLER.CC*. 66

1 INTRODUÇÃO

1.1 Motivações

A radioterapia por feixe de prótons é considerada uma alternativa expressivamente precisa no tratamento de cânceres localizados. Este fato se deve às características da interação do próton com a matéria, que possibilita depositar a maior dose na área tumoral, preservando bem mais os tecidos adjacentes, do que com outras técnicas de tratamento (SCHULTE *et al*, 2004).

O primeiro a pesquisar sobre a utilização do feixe de prótons para uso terapêutico foi o Dr. Robert Rathbun Wilson. Em 1946, Wilson, publicou um artigo, intitulado "*Radiological Use of Fast Protons*", que estabeleceu os principais fundamentos e técnicas que estão sendo seguidos atualmente nos principais centros de terapia por prótons no mundo (NAPT, 2000). Alguns anos mais tarde, em 1954, o pesquisador Dr. Cornelius A. Tobias submeteu o primeiro paciente ao tratamento de câncer por prótons (LOMA LINDA UNIVERSITY, 2013).

O primeiro centro de terapia por prótons foi construído na década de 90, na Universidade de Loma Linda na Califórnia, Estados Unidos da América (SLATER, 1992). Atualmente existem 28 centros e aproximadamente 84.000 pacientes já foram tratados com essa técnica no mundo (PTCOG SECRETARY, 2012).

Para localizar e identificar a região tumoral a ser irradiada pelo feixe de prótons, o paciente é submetido a exames de tomografia computadorizada e ressonância magnética nuclear (SCHULTE et al, 2004). As imagens adquiridas por esses exames servem de base para o planejamento da terapia por prótons. Imagens de tomografia por emissão de pósitrons (PET-CT *Positrons Emission Tomography*) também são utilizadas para auxiliar nesse tipo de tratamento, porém é, até o presente momento, a técnica menos utilizada (SCHULTE, 2007).

No intuito de aumentar a eficiência do tratamento por prótons, surgiu a idéia de construir um equipamento de tomografia computadorizada que utilize feixes de prótons (*pCT*). A grande motivação em se construir um *pCT* é unir no mesmo equipamento de terapia por prótons a capacidade de gerar imagens tomográficas utilizando o mesmo feixe. O resultado da união desses dois equipamentos é obter

dados para o planejamento da terapia e posicionamento do tumor de forma simultânea, aumentando a precisão e rapidez do exame (SCHULTE, 2004).

No Centro Médico da Universidade de Loma Linda (*Loma Linda University Medical Center* - LLUMC) está sendo desenvolvido um protótipo de *pCT* com capacidade para trabalhar com valores de energia de até 250 MeV (PENFOLD, 2010).

Pesquisadores da Universidade Tecnológica Federal do Paraná (UTFPR), do Instituto de Energia Nuclear (IEN), da Universidade Federal do Rio do Janeiro (UFRJ) e da Universidade Estadual do Rio de Janeiro (UERJ) uniram-se para desenvolver um protótipo de um *pCT* no Brasil com capacidade para trabalhar com valores de energia de até 24 MeV. O protótipo, já em teste, trouxe informações a respeito da precisão do mecanismo de movimentação do *phantom*, energia máxima do feixe de prótons e funcionamento do sistema de detecção com o detector de *SiLi*. Além do protótipo, o projeto aborda também áreas de interesse mais específicas, como simulações computacionais por código de Monte Carlo (KOZUKI, 2012).

As simulações computacionais se tornaram uma ferramenta indispensável na práxis científica. Há algumas décadas, permitiu aos cientistas gerar modelos, estabelecer relações e simular as mais variadas hipóteses.

O funcionamento da *pCT* pode ser explorada através de simulações de Monte Carlo, dando informações essenciais que servirão de referências à parte experimental. As simulações podem ser vistas como representações ou modelagens de objetos específicos reais ou imaginados, de sistemas ou fenômenos (MEDEIROS, 2002). Elas podem ser bastante úteis, particularmente quando a experiência original for inviável financeiramente, pois é possível concluir, com códigos bem estruturados, se o investimento terá o retorno esperado ou não. Experimentos perigosos ou que envolvam fenômenos muito lentos ou extremamente rápidos estão, também, dentro da classe de eventos que podem ser explorados pelas simulações computacionais (SNIR *et al*, 1988).

O Geant4 Release 4.9.3, programa utilizado para simulações de Monte Carlo, possui em seus exemplos médicos um código, denominado "*Código Dicom"*, desenvolvido por pesquisadores do *Centre Hospitalier Universitaire de Quebec* no Canadá, que converte diretamente a imagem *DICOM* de um *phantom* físico de tórax em um *phantom* virtual de tórax (BIENVENUE AU LABORATOIRE DE PHYSIQUE NUCLÉAIRE EXPÉRIMENTALE ET MÉDICALE, 2013).

Essa técnica de conversão direta de um objeto físico em um *phantom* virtual faz com que não seja necessária a programação da geometria desejada, economizando tempo e diminuindo a estimativa de erros inerentes da programação de *phantoms,* que possuem várias geometrias que darão forma a um único objeto virtual.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral do presente trabalho é adaptar o código Geant4 para conversão de imagens *DICOM* de tomografia computadorizada (*CT*) de crânio real humano em um *phantom* virtual, representado por uma matriz de diferentes tipos de tecidos.

1.2.2 Objetivos Específicos

Os objetivos específicos são:

a) Configuração do código Geant4 em *Linux* e *Microsoft Windows XP* para as simulações pretendidas.
b) Configuração do programa *Dawn* para a visualização dos arquivos de saída do Geant4.
c) Reestruturar o exemplo denominado "*Código Dicom"* do código Geant4, que é programado em C++, para que se torne possível a conversão direta de imagens de *CT* em um *phantom* virtual.
d) Retirar dos comandos do "*Código Dicom"* todos os passos que são referentes aos eventos físicos nucleares, da geração de feixes de partícula e o código que monitora e registra a trajetória das partículas.
e) Reformular parte dos comandos do "*Código Dicom"* que faz compressão dos *pixels* da imagem de *CT*.
f) Fazer a conversão direta de um *phantom* físico com geometria e materiais conhecidos, para validação das alterações no "*Código Dicom"*.

1.3 Estrutura do Trabalho

O trabalho foi organizado em cinco capítulos. O primeiro capítulo apresenta as motivações, o objetivo geral e os objetivos específicos propostos pelo trabalho.

O segundo capítulo refere-se à fundamentação teórica. É apresentada uma breve revisão da literatura sobre Imagem, *CT*, aquisição de imagens em *CT*, padrão *DICOM*, *pCT*, programação de objetos no Geant4, construção do *"Código DICOM"* e o *phantom* de PEAD.

O terceiro capítulo retrata a metodologia, expondo os principais programas utilizados, bem como o caminho necessário para realizar cada etapa do trabalho.

No quarto capítulo são apresentadas as discussões sobre os programas utilizados e os resultados obtidos no trabalho.

Por fim, as conclusões são apresentadas no quinto capítulo.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Revisão da Literatura

2.1.1 Imagem

O termo imagem monocromática, ou simplesmente imagem, pode ser definida matematicamente como uma função bidimensional de intensidade da luz *f*(*x,y*), onde *x* e *y* são as coordenadas espaciais. O valor de *f* em qualquer par de coordenadas (*x,y*) é a intensidade da imagem nesse ponto. Convencionou-se atribuir proporcionalmente valores mais altos para áreas de maior brilho fazendo com que a altura dos componentes da figura seja proporcional ao brilho correspondente na imagem (GONZALEZ, 1992).

Imagem digital é uma imagem *f*(*x,y*) discretizada tanto em coordenadas espaciais quanto em brilho. A imagem digital pode ser considerada como sendo uma matriz, onde os índices de linhas e de colunas identificam um ponto na imagem, e o valor correspondente do elemento da matriz identifica o nível de cinza naquele ponto. Os elementos dessa matriz digital são chamados de elementos da imagem, elementos da figura "*pixel*"ou "*pixels*", abreviação das palavras em inglês "*Picture Elements*", que significa elementos da figura (GONZALEZ, 2002).

Os *pixels* são os menores pontos distribuídos em linhas e colunas que uma imagem pode obter. Cada um dos *pixels* possui um valor inteiro nas coordenadas *x* e *y*, que representam medidas dependendo de variáveis, que podem ser, por exemplo, o nível de quantificação. Imagens com grande número de *pixels* tem resolução melhor do que outras com menor número de *pixels*. Podemos dizer que quanto maior o numero de *pixels* em uma imagem de dimensões constante melhor é sua resolução e, assim, mais fácil será diferenciar as estruturas que compõem a imagem. A resolução espacial da imagem ou a quantidade de *pixels* varia de acordo com a sua aplicação (GONZALEZ, 2002).

Quando temos imagens que possuem valores no eixo *z*, dizemos que temos uma imagem tridimensional. A menor unidade de uma imagem tridimensional é chamada de *voxel*. Em outras palavras, o v*oxel* é a menor unidade de uma imagem que compõem um volume digital composto pelos eixos *x*, *y* e *z*. Na *CT* os *voxels* possuem largura, altura e profundidades iguais, ou seja, são isotrópicos (HATCHER, 2010).

2.1.2 Breve Histórico da Tomografia Computadorizada

No início do século XX foram desenvolvidas as bases matemáticas de construção de imagens que fundamentaram os cálculos matriciais da tomografia computadorizada (*CT*). O matemático Johann Radon desenvolveu as bases algébricas de projeções espaciais em 1917 (BUKOVICS, 1994). Estas projeções espaciais constituem-se de uma série de integrais em duas dimensões que, ao serem projetadas ao longo de uma linha definida, considerando um número delimitado de linhas, fornecem uma fórmula de inversão que possibilitam a construção algébrica da imagem. As integrais conhecidas como integrais de Radon transformam as informações bidimensionais em algoritmos para imagem plana.

Nos anos 30 já se conseguia obter imagens bidimensionais a partir de objetos tridimensionais fazendo uso de uma fonte de raios X e detectores de radiação. Os detectores eram colocados no lado oposto ao objeto e se movimentavam durante a formação da imagem fazendo uma rotação completa. A reconstrução da imagem fazia uso, nesta época, de cálculos matriciais, que além de mais demorados, formava imagens com muitos ruídos e com pouca nitidez. Sendo assim, impraticáveis para o diagnóstico médico (CORMACK, 1963).

Nos anos 70, com a introdução dos sistemas computacionais associados às técnicas tomográficas, que já vinham sendo desenvolvidas, foram introduzidos no mercado os aparelhos tomográficos para uso clínico que foram chamados de *Computed Axial Tomograph*. As teorias matemáticas necessárias para formação de imagens tomográficas se fundamentaram inicialmente nas técnicas de projeções bidimensionais de Radon. Cormack (1963) ganhou o prêmio Nobel de Medicina e Fisiologia em 1979 junto com Hounsfield pelo desenvolvimento da *CT*. Ele

pressupôs que um feixe planar de radiação penetrante, ao ser projetado em diferentes ângulos sobre um determinado corpo, pode fornecer uma imagem muito melhor do que as então conhecidas imagens radiográficas convencionais (CORMACK,1963). A técnica mostrou-se mais eficiente porque, com múltiplas tomadas de dados, as imagens são mais confiáveis, uma vez que são formadas com maiores quantidades de informações. Hounsfield desenvolveu o primeiro tomógrafo comercial com as bases fundamentadas no trabalho de Cormack.

As técnicas tomográficas propostas naquela época para obtenção de imagens consagraram-se no uso médico diagnóstico e em outras áreas de pesquisas, uma vez que elas permitem uma visão espacial do objeto estudado. Estas imagens foram se tornando cada vez mais nítidas e com a vantagem de ser uma técnica não invasiva (SEERAM, 2001).

Denominou-se tomografia por transmissão, a técnica em que a fonte de radiação é posicionada externamente ao corpo a ser estudado. A fonte também pode ser introduzida no corpo a ser estudado utilizando-se as mesmas técnicas; esta versão tomográfica foi denominada de tomografia por emissão. Conforme foi demonstrado por Edwards e Kuhl (KUHL, 1963) é possível obter imagens morfológicas e funcionais usando-se o SPECT (*Single Photon Emission Computed Tomography*). Depois, outro tomógrafo foi desenvolvido utilizando emissores de pósitrons, por exemplo, o Flúor 18 (^{18}F), o sistema foi denominado de PET *(Positron Emission Tomography).* As imagens morfológicas e funcionais mostradas pelo SPECT e PET, são de ampla aplicação nos campos da neurologia, oncologia, cardiologia, urologia, músculo esquelético, dentre outros.

Nos últimos anos, as técnicas de obtenção de imagens por *CT* vêm sendo desenvolvidas para campos de pesquisa científica e seu uso estendido nos diversos setores industriais.

2.1.3 Tomografia Computadorizada por Transmissão de Raios X

A *CT*, nos dias atuais, é um método de aquisição de imagem não invasiva, rápida, fidedigna e de alta precisão diagnóstica. A imagem gerada é uma

apresentação da anatomia de uma fatia do corpo desenvolvida por múltiplas medidas de absorção do feixe de raios X, feitas ao redor da periferia do corpo.

O Sistema da *CT* é dividido em três sistemas menores de modo a otimizar a aquisição, o processamento (reconstrução), a visualização e a manipulação da imagem (SEERAM, 2001). Os três sistemas são: unidade de varredura (*gantry*), o computador e o console do operador.

O *gantry* inclui o tubo de raios X, o conjunto de detectores, o gerador de alta tensão, colimadores pré-paciente, filtros, conversores eletrônicos e o suporte para o paciente. O suporte do paciente é constituído por uma plataforma na qual o paciente deita durante o exame e é ajustado no *gantry*, de forma que a região a ser examinada fique posicionada entre o tubo de raios X e os detectores. Ele ainda contém componentes eletrônicos e mecânicos que facilitam a movimentação e o posicionamento do paciente no campo de varredura. O tubo de raios X e o gerador de alta tensão, que fornece tensões de 120 a 140 kVp, são responsáveis pela produção de raios X. O feixe de radiação, proveniente do tubo, passa por filtros para proteger o paciente dos fótons de baixas energias. A função dos colimadores é restringir o volume de tecido irradiado, definir a espessura da fatia de interesse que será varrida e minimizar a radiação espalhada. Os detectores medem a energia do feixe de raios X que chegam até eles. Os sinais provenientes do conjunto de detectores são convertidos em sinais elétricos e enviados para o sistema do computador, que fará todo o processo de reconstrução de imagem através de complexos cálculos matemáticos.

Pelo console do operador é possível monitorar e controlar a tensão e a corrente do tubo. São selecionados também o tempo de varredura e a espessura de corte. A movimentação do paciente acontece por meio de um controle automático de movimentação do suporte, juntamente com a inclinação do *gantry*. O console do operador contém um ou mais monitores de vídeo, onde o tecnólogo indica os dados do paciente que será examinado (nome do paciente; idade; sexo; identificação do hospital), além dos dados da varredura (tempo de varredura, espessura de corte, número varredura, posição do paciente, técnica empregada, corrente, tensão e filtros). O console também permite analisar as imagens e manipulá-las de forma a obter o exame desejado.

As imagens são normalmente registradas na memória do computador e em unidades de filme (impressas) sendo também possível gravar estas imagens em discos removíveis para manipulação em computadores pessoais.

2.1.4 Considerações Gerais Sobre Imagem de Tomografia Computadorizada

A imagem de *CT* é uma apresentação da anatomia de uma fatia do corpo desenvolvida por múltiplas medidas de absorção de raios X feitas ao longo do corpo. O conceito fundamental da *CT* é que a estrutura interna de um objeto pode ser reconstruída a partir de múltiplas projeções do objeto. Embora este grupo de números contenha todas as informações do processo, é difícil de interpretar e, portanto, é convertido em imagem por atribuição de uma escala de cinza aos números. Números elevados são representados por tons claros de cinza, e números baixos são representados por tons escuros de cinza. A imagem resultante pode ser então manipulada para realçar determinadas áreas.

Na *CT*, o método de obtenção do arranjo de números é mais complexo, e o número de projeções obtidas é muito maior. O processo de escolha do número de tons de cinza para uma imagem é denominado seleção de uma janela (HAAGA, 1991).

Tanto a imagem formada por radiografia convencional como por *CT* baseiam-se na equação de atenuação dos raios X:

$$I=I_o.e^{-\mu x} \qquad (1)$$

Sendo o I_o a intensidade incidente de um feixe de raios X sobre a superfície de um objeto de espessura x, e I é a intensidade transmitida. O coeficiente de atenuação linear (μ) é uma propriedade dependente do número atômico, da densidade do material e do espectro de energia do feixe de raios X. Apresentar dados de atenuação (seja I ou μ) em cada ponto de todo o corpo seria ideal em um exame por raios X. O grau de alcance depende da forma na qual as intensidades medidas, I e I_o, são registradas ou manipuladas. Em radiografia convencional, a intensidade transmitida (I) é observada como o escurecimento de um filme. Como a

exposição do filme aos raios X o escurece, a imagem de um objeto denso é mais clara no filme que a imagem de um material menos denso. Em uma radiografia do tórax típica, por exemplo, a imagem é clara nos locais onde muitos raios X são absorvidos ou dispersos, tal como no osso, e escuras onde muitos raios X são transmitidos em virtude de baixa absorção, como nas regiões do parênquima pulmonar. Duas áreas diferentes desta radiografia do tórax podem mostrar o mesmo escurecimento e demonstrar igual atenuação total do feixe nas duas posições. O perfil de atenuação através do corpo pode ser muito diferente. Na radiografia convencional os diferentes tons de cinza observados no filme representam as diferenças na transmissão de um feixe de raios X quando atravessa o corpo. A *CT* coloca em ordem a informação de atenuação do feixe de raios X e a apresenta de forma quantitativa com uma precisão muito maior que a obtida por técnicas convencionais. Isso é equivalente a fornecer os valores individuais, em lugar do volume total descrito para radiografia convencional (FERREIRA, 2002).

A *CT* também apresenta a vantagem de que a imagem gerada pode ser armazenada no computador do tomógrafo e/ou gravada em discos removíveis. Uma vez gravado o exame em CD ou em *Digital Versatile Disc* (DVD) é possível visualizar as imagens em computadores pessoais tornando possível, desde que a qualidade do monitor permita, emitir um laudo do exame sem necessariamente estar na clínica ou hospital que possui o tomógrafo. A extensão ou formato salvo das imagens é o padrão .dcm ou *DICOM*.

2.1.5 *DICOM*

Com a intensificação da utilização de equipamentos de imagens digitais para diagnóstico médico na década de 70, notou-se a necessidade de formular um padrão para transferência de imagens médicas e outras informações relevantes para o diagnóstico entre equipamentos de diferentes fabricantes (NEMA, 2011).

O *American College of Radiology* (ACR) em conjunto com o *National Electronics Manufactures's Association* (*NEMA*) desenvolveram o primeiro padrão para ser utilizado em equipamentos hospitalares. O padrão foi denominado de *ACR-NEMA 1.0* (HORII, 2002). Devido a vários erros da primeira versão, foi lançada, em

seguida, a segunda versão que contemplou algumas correções. No entanto, devido à rápida evolução da internet e dos meios de comunicação, essa segunda versão teve que ser revista para adequação aos novos padrões de tecnologia. Foram então adotados métodos de linguagem orientada a objeto para desenvolver a terceira versão. E para acompanhar os avanços das redes de computadores, a terceira versão foi construída para ser compatível com a arquitetura TCP/IP (*Transmission Control Protocol - Internet Protocol*) (PRIOR, 2001).

A terceira versão, lançada em 1993, foi batizada de *DICOM 3.0* ou simplesmente *DICOM* (*Digital Imaging and Communications in Medicine*). Este padrão está continuamente sendo revisado e teve um acentuado grau de aceitação, visto que já em 1995, esse formato foi aceito formalmente pela União Européia. Posterior a esse fato, a maioria dos fabricantes de equipamentos hospitalares passaram a utilizar o padrão estabelecido pelo *DICOM* (PRIOR, 2001).

O *DICOM* é um padrão detalhado que descreve um meio de formatação e comunicação de imagens médicas com informações associadas, suprindo as necessidades de integração dos equipamentos médicos. Seus objetivos são alcançar total compatibilidade e aperfeiçoar a eficiência do fluxo de trabalho entre equipamentos que utilizem imagens e outras informações de saúde (NEMA, 2011). O *DICOM* define: a estrutura de dados para imagens médicas; a forma de registrar as informações relacionadas ao exame; os serviços relacionados à comunicação (transmissão de imagens) e o formato para armazenamento das informações (NEMA, 2011).

O *DICOM* refere-se a vários níveis do modelo de rede *Open System Interconnection* (*OSI*) e fornece a sustentação para a troca de informações em intercâmbio de mídias (NEMA, 1985). O modelo *OSI* é formado por 7 camadas, sendo elas:

1. Camada Física: responsável pelas características mecânicas, elétricas e funcionais;
2. Camada Enlace de Dados: responsável pela transmissão de informações sincronizadas;
3. Camada Rede: responsável por estabelecer, manter e terminar conexões;
4. Camada Transporte: responsável pela correção de fluxo na transmissão de dados entre dois pontos;

5. Camada Sessão: responsável por estabelecer, gerencia e termina conexões entre as aplicações;
6. Camada Apresentação: responsável por realizar a sintaxe entre os dados;
7. Camada Aplicação: acessa o ambiente OSI e sistemas distribuídos.

Atualmente o *DICOM* especifica um protocolo de camadas chamado de *Upper Layer Protocol* que é utilizado sobre a arquitetura TCP/IP.

O Comitê de Padronização *DICOM* (*DICOM Standards Committee*) está focalizando sua atenção na evolução dos padrões ligados à internet. O comitê pretende continuar a integração do padrão *DICOM* nas Recomendações de Internet (*Internet Recommendations*). Nesta evolução, muita cautela tem sido tomada para garantir a consistência do *DICOM*.

2.2 Construção de Geometrias no Código Geant4

As simulações feitas no código Geant4 devem ser escritas em código de programação C++, orientado a objeto, contendo todos os parâmetros a serem processados. Uma ampla biblioteca com estruturas pré-definidas é usada, tais como geometria dos objetos, geração das partículas carregadas, interações nucleares etc.

Para construir geometrias sólidas e de apenas um material a programação é simples. Por exemplo, o comando para obtermos uma caixa sólida com X igual a 30, Y igual a 40 e Z igual a 60 cm está representado a baixo, e a imagem da caixa está representado na Figura 1:

```
#include "G4Box.hh"

G4BOX (G4String const & pName,
        G4double pX,
        G4double PY,
        G4double pZ)

        G4Box* aBox = new G4Box("BoxA", 30*cm, 40*cm, 60*cm);
```

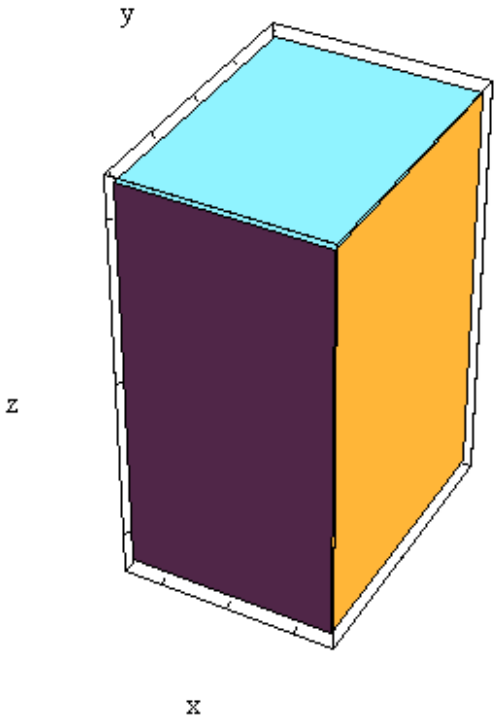


Figura 1 - *Phantom* simples de uma caixa sólida.

Para construir a caixa sólida foram necessárias cinco linhas de comando. No entanto, quando temos que modelar situações com várias geometrias e com diferentes materiais a programação passa a ser onerosa. Em trabalhos anteriores (KOZUKI, 2012) foi desenvolvido em Geant4 um *phantom* em forma de tubo com camadas de polipropileno e água destilada, um colimador de alumínio e um detector de silício. No Anexo A está o código para construir essas geometrias e determinar seus respectivos materiais e na Figura 2 a representação gráfica desse sistema.

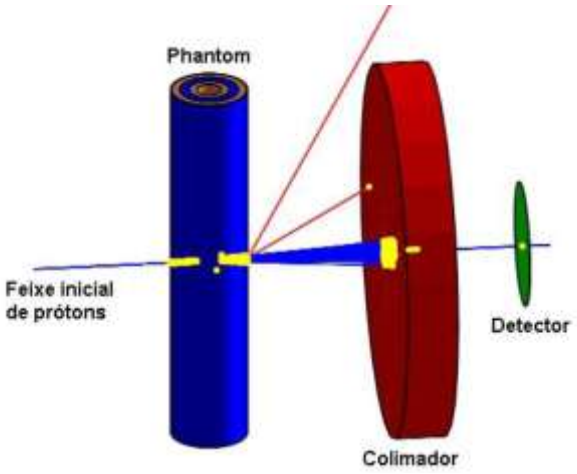


Figura 2 - *Phantom* de polipropileno (azul), colimador (vermelho), detector (verde) e o feixe de prótons.
Fonte: KOZUKI, 2012

Comparando o código da caixa sólida com os códigos (Anexo A) do *phantom* de polipropileno e água, podemos concluir que para modelar sistemas que contenham dois ou mais objetos geométricos, a programação pode se tornar um trabalho de algumas semanas. Ao se modelar estruturas anatômicas, a complexidade exige a necessidade de agrupar inúmeras geometrias, para obter a máxima aproximação da estrutura real, o que pode consumir alguns meses ou anos de esforços para se construir um *phantom* antropomórfico de crânio, tórax ou membros.

Este conjunto de *phantom* de polipropileno e água destilada, colimador e detector foram construído para simular as condições do protótipo de *pCT* do IEN/CNEM.

2.2.1 Protótipo do IEN/CNEN

A UTFPR em parceria com o IP/UERJ e a UFRJ está desenvolvendo um protótipo de *pCT*. Porém, este projeto de *pCT* no Brasil tem dimensões significativamente menores quando comparado com o projeto de *LLUMC*. O principal fator limitante aqui no Brasil é a energia do feixe de prótons disponível no cíclotron do IEN/CNEN no Rio de Janeiro, que é de no máximo 24 MeV. Há também as pequenas dimensões da câmara de espalhamento, que torna necessário um protótipo com tamanho reduzido. Sua configuração pode ser vista na Figura 3.

Os elementos que compõem o esquema do protótipo mostrado na Figura 1 são identificados através dos números que serão descritos a seguir: (1) Feixe inicial tem sua intensidade reduzida a um nível adequado (2) por espalhamento elástico no elemento alvo (3); a barra de suporte existente na câmara de espalhamento (4) é usada na montagem de dois colimadores (5), detector de prótons (6) e mecanismo de movimentação (7); dois motores de passo (8) fornecem a rotação e translação ao longo da guia (9); o sinal do detector é tratado por uma interface eletrônica de saída (10), os movimentos do mecanismo são gerenciados por um controle eletrônico CLP (Controlador Lógico Programável) (11) sob supervisão de um programa (12).

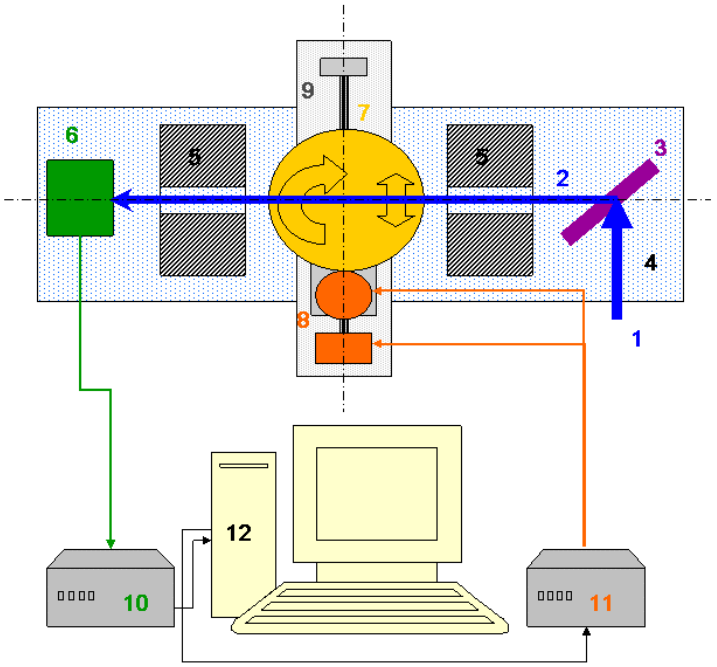


Figura 3 - Representação do protótipo.
Fonte**:** SETTI, 2006.

A Figura 4 mostra a câmara de espalhamento CV-28 e o mecanismo fixado para a sessão de testes:

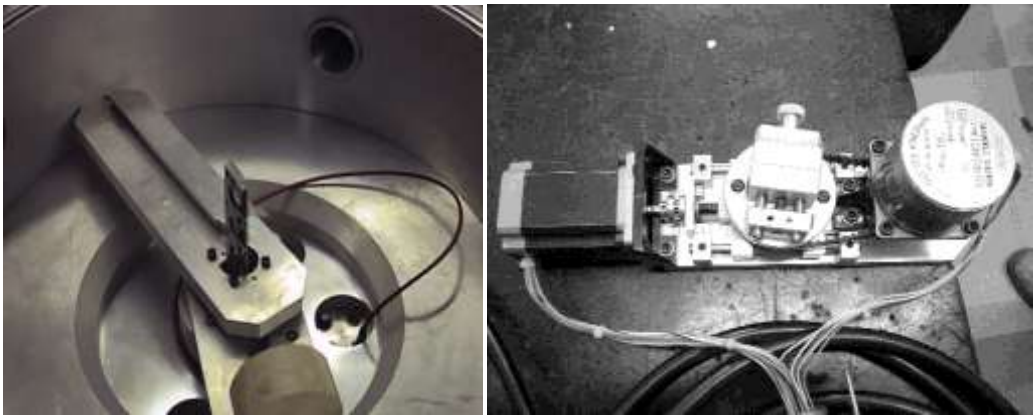


Figura 4 - Câmara de espalhamento CV-28 e mecanismo de movimentação do *phantom*.
Fonte**:** SETTI, 2006.

2.3 *Phantom* Simples de Polietileno.

Em trabalhos anteriores, foram construídos dois *phantoms* com características compatíveis com o suporte do protótipo da Universidade de Loma

Linda, LLUMC, utilizando materiais de fácil aquisição e geometria simples. Estes *phantoms* foram testados no protótipo do LLUMC com feixe de 200 MeV para fazer o processamento das informações obtidas pelos detectores do protótipo. Em seguida, foram feitas simulações em Geant4, a fim de comparar o desempenho do código Geant4 nas condições do LLUMC (MILHORETTO, 2012).

Os *phantoms* são de polietileno de alta densidade (PEAD), densidade de 0,9 g/cm³, com diâmetro externo de 150 mm e núcleo de material acrílico, densidade de 1,18g/cm3, em formato de esfera de 24 mm de diâmetro em um deles e um pino central com 27 mm de diâmetro no outro, conforme a Figura 5.

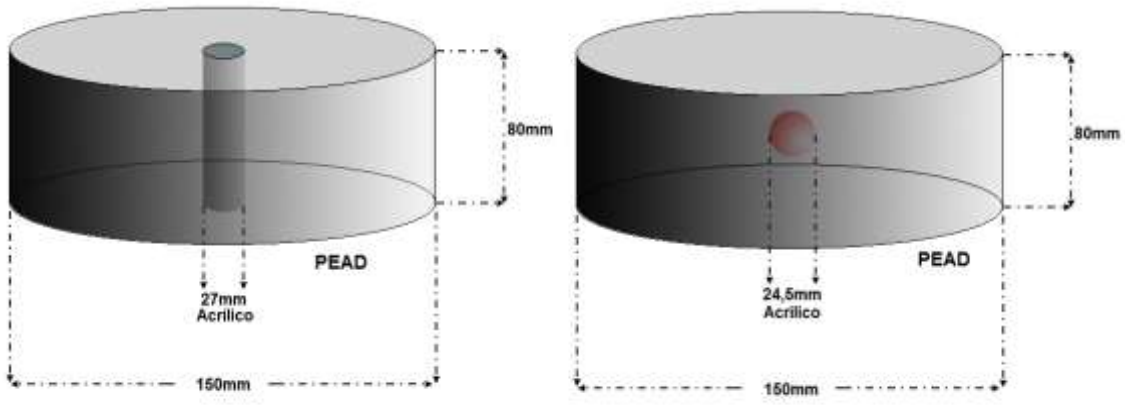


Figura 5 - Estrutura dos *phantoms*, à esquerda da figura com núcleo em barra e à direita com esfera.
Fonte: MILHORETTO, 2012.

Como estes *phantoms* possuem geometria simples e materiais conhecidos, serão utilizados para testar as adaptações feitas no "*Código Dicom*". Para isso será realizado a conversão direta das imagens de *CT* do *phantom*, com o pino central, para um *phantom* virtual. Na Figura 6 é possível observar na esquerda superior um dos cortes, na esquerda inferior o corte coronal, na direita superior corte sagital e na direita inferior a reconstrução 3D do *phantom* de PEAD.

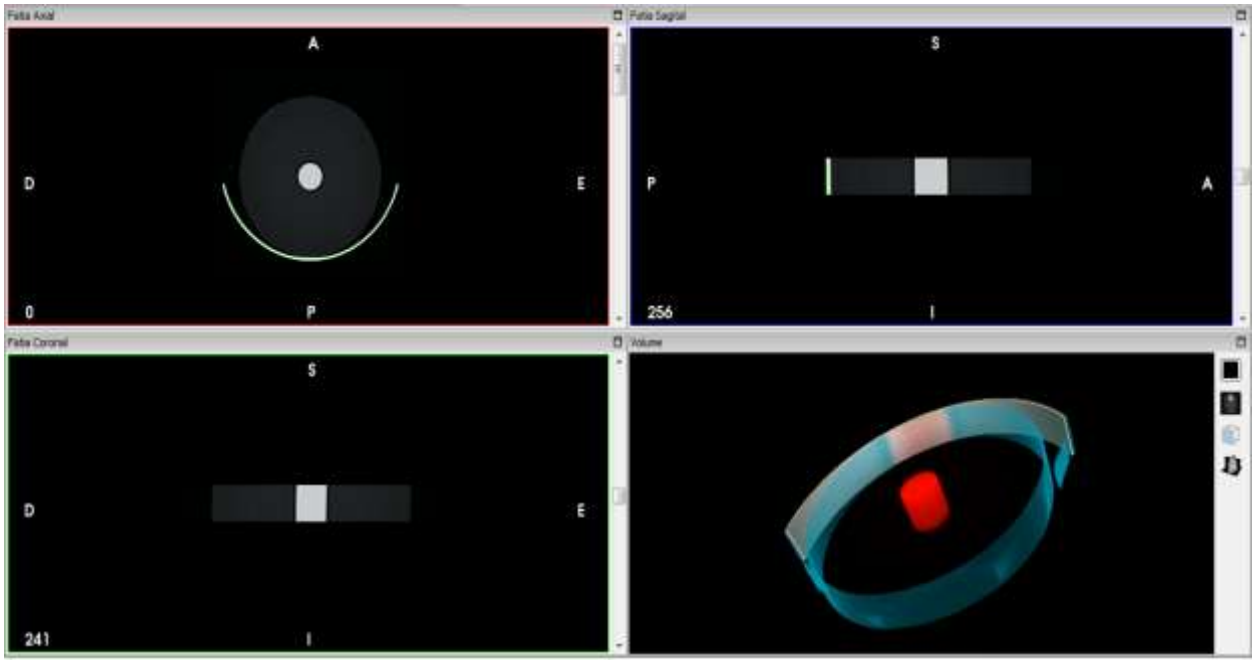


Figura 6 - Imagens de *CT* do *phantom* de PEAD. Corte sagital, coronal e a reconstrução 3D.

2.3.1 Protótipo da LLUMC

O Centro Médico da LLUMC nos Estados Unidos, é um conceituado estabelecimento e merece destaque pelo seu protótipo de *pCT* que está em desenvolvimento, pois engloba as mais recentes tecnologias (SCHULTE, 2004).

Diante dos intensos trabalhos com *pCT*, os pesquisadores do LLUMC, perceberam que técnicas de reconstrução do caminho percorrido próton por próton aumentam consideravelmente a resolução espacial do imageamento com prótons, minimizando os problemas como o espalhamento múltiplo de Coulomb, que no passado, eram os grandes limitadores do desenvolvimento da radiografia e da tomografia com prótons. Após muitos anos de desenvolvimento para experimentos de altas energias e física nuclear, a captação de dados próton por próton é agora disponível com precisão e em altas taxas de contagem necessária para a *pCT*. (MILHORETTO, 2006).

O centro dispõe de uma energia média de 250 MeV para os testes, portanto, uma alta energia (SCHULTE, 2004).

2.4 Conversão de Imagens *DICOM* de um *Phantom* Físico de Tórax em um *Phantom* Virtual

Vários centros de pesquisa estão trabalhando para desenvolver *phantoms* antropomórficos virtuais, com o intuito de ampliar as pesquisas de dosimetria.

No Canadá, três grandes centros de pesquisa, construíram um *phantom* físico de tórax com o objetivo de transformá-lo em um *phantom* virtual a partir da sua tomografia. Os grupos de pesquisas são *Centre Hospitalier Universitaire de Quebec*, *Université Laval* e o Departamento de Física da *University Lund*. Até o momento do início de suas pesquisas a única ferramenta que se conhecia, para reconstruir geometrias, era através de códigos de programação. Porém os canadenses queriam transformar a tomografia de um *phantom* físico em um *phantom* virtual e, para isso, fizeram a tomografia do *phantom* de tórax (Figura 7) e salvaram sua imagem no formato *DICOM* (Figura 8).

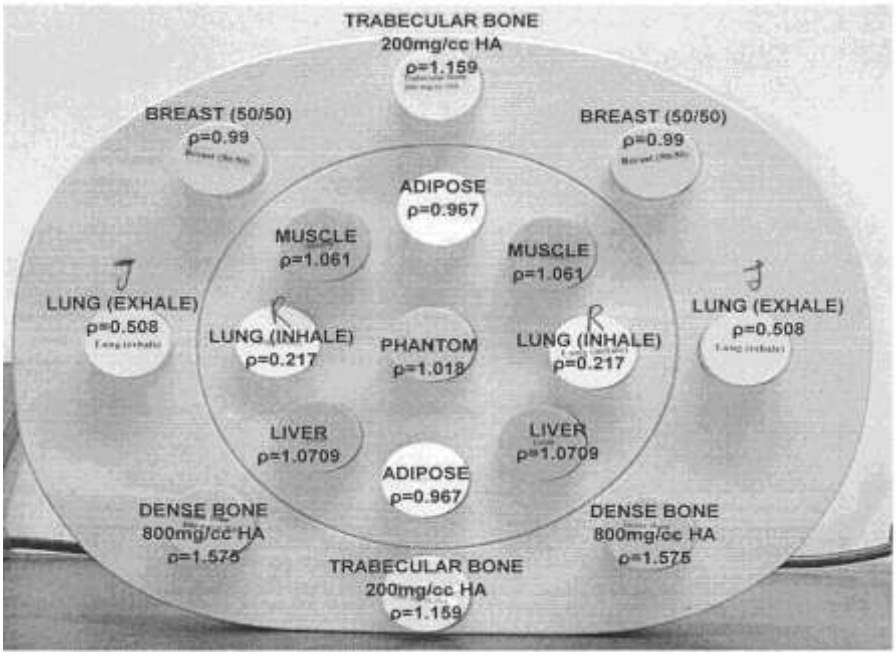


Figura 7 - Imagem do *phantom* físico de tórax com diferentes tecidos.
Fonte: Geant4, 2010.

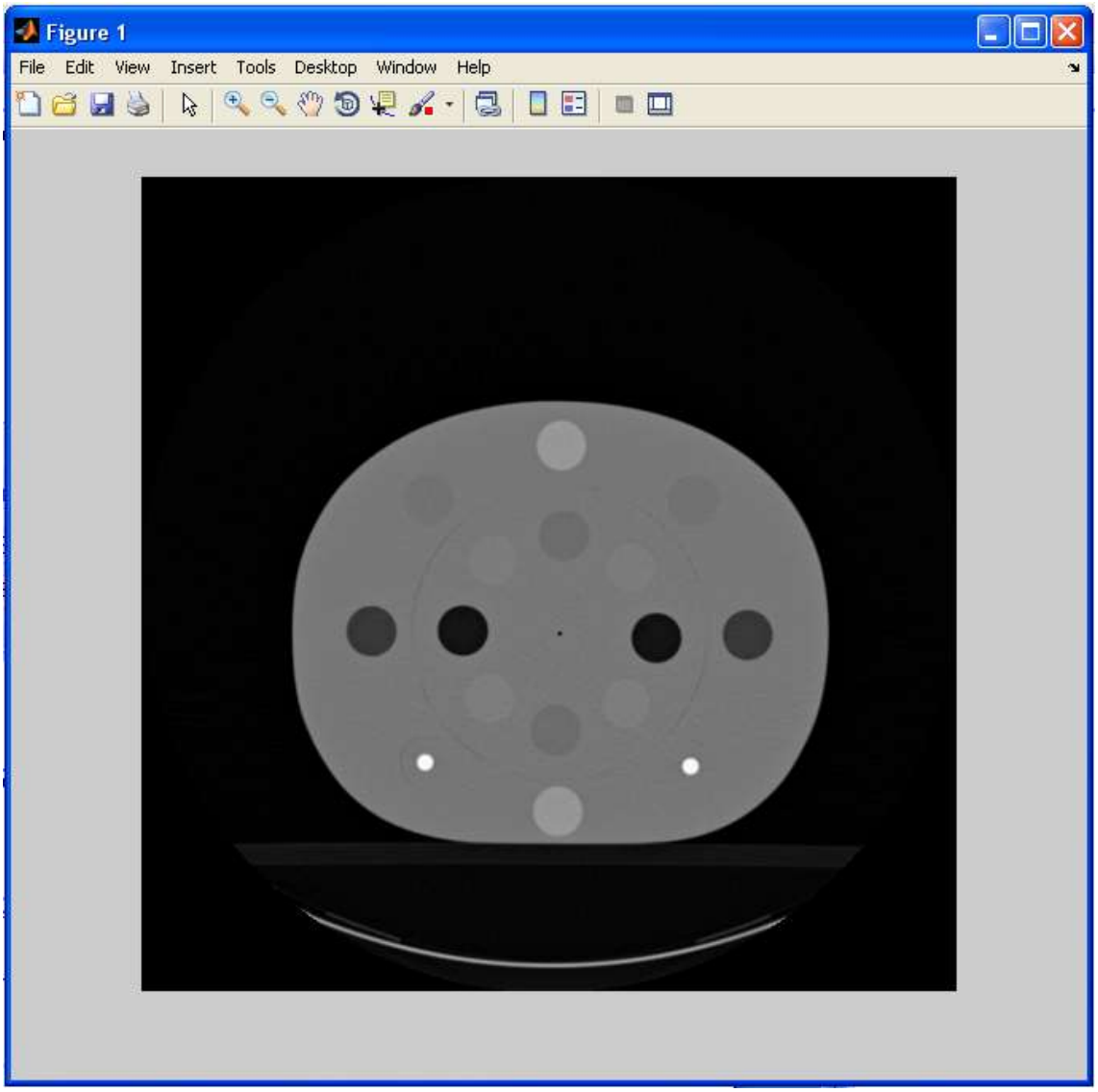


Figura 8 - Reconstrução tomográfica do *phantom* físico de tórax.

Com isso, os centros de pesquisas canadenses mostraram ser possível converter uma imagem de tomografia de um *phantom* físico simples em um *phantom* virtual, diminuindo o tempo e o trabalho para programar a geometria do *phantom* desejado, pois apenas compilando a imagem tomográfica foi possível criar o *phantom* virtual.

O *phantom* físico construído é formado por diferentes estruturas que simulam os mais variados tecidos que compõem o tórax humano. Como cada tecido tem sua respectiva densidade, o *phantom* virtual apresenta diferente intensidade para cada *pixel*. A intensidade está diretamente ligada à densidade que o tecido apresenta. A Figura 9 faz referência a esta diferença de intensidade (Z) por quantidade de *pixel* (X e Y) do *phantom*.

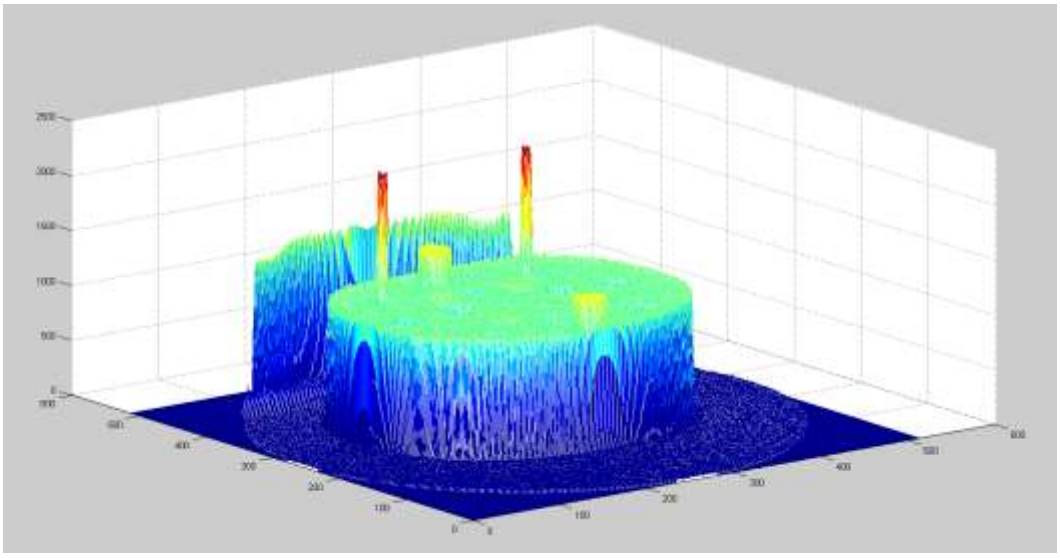


Figura 9 - Intensidade (Z) por quantidade de *pixel* (X e Y) da imagem de *CT* de tórax.

3 MATERIAIS E MÉTODOS

Por se tratar de um trabalho computacional, nesta seção serão apresentados os principais programas utilizados, bem como o caminho necessário para realizar cada etapa do trabalho.

3.1 Programas Computacionais

O código Geant4 (*Geometry and Track*) é uma ferramenta para simulação da interação de partículas com a matéria. É usado em um grande número de experimentos e projetos em vários domínios de aplicação, incluindo, física de alta energia, astrofísica e ciência espacial, física médica e proteção radiológica (GILAT, 2004).

Para gerar a simulação Monte Carlo, o usuário pode alterar o código, que é extremamente flexível e provido de um conjunto de sub-rotinas para definição de materiais, geometrias e propriedades de partículas de acordo com as necessidades.

Esse programa está em constante aprimoramento e é amplamente usado no meio acadêmico e científico. Pode ser instalado nos sistemas *Linux, Unix e em Microsoft Windows*® utilizando um programa chamado *Cygwin*.

Como se trata de programa bem abrangente no âmbito da física sua instalação é complexa e será descrita abaixo.

O código Geant4 depende de programas externos para mostrar a simulação elaborada. São três os programas que podem ser usados para essa finalidade: *Wired*, *Dawn* e *OpenGL*. Optamos por usar o *Dawn*, uma vez que se conseguiu uma melhor compatibilidade com o sistema operacional *Windows XP* e se mostrou prático na escolha do ângulo de visualização do *phantom*.

Fukui Renderer DAWN (*Drawer for Academic WritiNgs*) foi desenvolvido por Satoshi Tanaka com um processador vetorial 3D pós-escrito, com opção de remoção de linhas/superfícies analítico para desenhos técnicos de objetos de geometrias complexas. Especificamente projetado para o Geant4, o programa *Dawn* calcula partes visíveis dos dados 3D antes de desenhar para então produzir gráficos vetoriais de dispositivos independentes de alta qualidade em aplicações técnicas (ALLISON *et al*, 2006).

O programa e outras informações técnicas podem ser obtidas no *web site* dos desenvolvedores.

Já para gerar as imagens de intensidade por quantidade de pixel foi utilizado o *MatLab (MATrix LABoratory).* O *MatLab* é um *software* interativo de alta performance voltado para o cálculo numérico. Integra análise numérica, cálculo com matrizes, processamento de sinais, vetores e construção de gráficos. Atualmente tem sido empregado como uma poderosa ferramenta de programação (GILAT, 2004). A sua utilização neste trabalho foi para gerar gráficos e visualizar imagens *DICOM*.

3.2 Instalação do Geant4 Release 4.9.3.P01 em Linux

A instalação do Geant4 em *Linux* é uma tarefa mais simples do que em *Windows*®, pois não requer a instalação do *Cygwin*. No mercado existem dezenas

de distribuições *Linux* como *RedHat, Ubuntu, SuSE, Debian etc*. O sucesso da instalação em diferentes versões *Linux* vai depender diretamente de pacotes de desenvolvimento instalados e da versão dos mesmos. Para melhor compatibilidade entre o programa Geant4 e o sistema operacional foi escolhido trabalhar com *Scientific Linux CERN 5*, pois ambos são da mesma organização *CERN - Europen Organization for Nuclear Research.* Além disso, este programa pode ser adquirido gratuitamente no *web site* do *CERN*.

O *Linux* possui nativamente compilador similar ao *C++* que é o *G++.* O manual de instalação do Geant4 recomenda instalar *gcc 4.1.2* ou *gcc 4.3.2*, porém com outras versões também foi possível ter êxito. Para saber a versão do *G++* instalado, é necessário digitar no terminal *gcc-v* e pressionar *enter*. A versão do *gcc* utilizada para compilar o código foi a *gcc 4.3.2*, sendo ela a mais compatível com a versão 4.9.3.P01 do código Geant4.

O primeiro programa a ser instalado é o *CLHEP – Class Library For High Energy Physics* que pode ser obtido gratuitamente no *web site* do fabricante, fazer o *download* da versão compatível com o sistema operacional. O arquivo tem aproximadamente 3,4 MB. É necessário abrir o terminal de comandos, navegar até o diretório onde o programa foi salvo e descompactá-lo usando o seguinte comando:

```
[root@ ~] # tar -zxvf clhep-2.0.4.5.tgz <Enter>
```

O comando tar -zxvf descompacta o arquivo clhep-2.0.4.5. O processo de descompactação será iniciado, começando com:

```
[root@ ~] # 2.0.4.5/
[root@ ~] # 2.0.4.5/CLHEP/
[root@ ~] # 2.0.4.5/CLHEP/CVS/
[root@ ~] # 2.0.4.5/CLHEP/CVS/Root
[root@ ~] # 2.0.4.5/CLHEP/CVS/Repository
```

E termina com:

```
[root@ ~] # 2.0.4.5/CLHEP/Vector/configure
```

```
[root@ ~] # 2.0.4.5/CLHEP/config.guess
[root@ ~] # 2.0.4.5/CLHEP/config.sub
[root@ ~] # 2.0.4.5/CLHEP/Makefile.in
[root@ ~] # 2.0.4.5/CLHEP/configure
```

Depois de descompactado deve-se ir para o diretório CLHELP para iniciar o processo de configuração, através do comando:

```
[root@ ~] #. / Configure <Enter>
```

A resposta começa com:

```
[root@ ~] # checking build system type... i686-pc-linux-gnu
[root@ ~] # checking host system type... i686-pc-linux-gnu
[root@ ~] # checking target system type... i686-pc-linux-gnu
[root@ ~] # checking for a BSD-compatible install... /usr/bin/install -c
[root@ ~] # checking whether build environment is sane... Yes
```

E que termina com:

```
[root@ ~] # config.status: creating test/exctest4.sh
[root@ ~] # config.status: creating test/exctestNothrow.sh
[root@ ~] # config.status: creating test/excDblThrow.sh
[root@ ~] # config.status: creating Exceptions/defs.h
[root@ ~] # config.status: executing depfiles commands
```

Em seguida deve ser executado o processo de compilação, através do comando:

```
[root@ ~] # make <Enter>
```

A resposta iniciará com:

```
[root@ ~] # Making all in Units
```

```
[root@ ~] # make[1]: Entering directory
 `/afs /slac.stanford.edu /u /ey /perl /CLHEP /2.0.4.5 /CLHEP /Units'
Making all in Units
[root@ ~] # make[2]: Entering directory `/afs /slac.stanford.edu /u /ey /perl /CLHEP /2.0.4.5 /CLHEP /Units /Units'
[root@ ~] # make  all-am
```

E termina com:

```
[root@ ~] # rm -f libCLHEP-2.0.4.5.so
[root@ ~] # liblist=`./getObjectList -shared Units Vector Evaluator GenericFunctions Geometry Random Matrix RandomObjects RefCount Cast Exceptions`; \
[root@ ~] # g++ -O -ansi -pedantic -Wall -D_GNU_SOURCE -g -O2 -o libCLHEP-2.0.4.5.so -shared -Wl,-soname,libCLHEP-2.0.4.5.so $liblist -o libCLHEP-2.0.4.5.so
[root@ ~] #./build-header
[root@ ~] # make[1]: Leaving directory `/afs /slac.stanford.edu /u /ey /perl /CLHEP /2.0.4.5 /CLHEP'
```

Para finalizar a instalação do CLHP se faz necessário digitar:

```
[root@ ~] # make install <Enter>
```

A resposta iniciará com:

```
[root@ ~] # Making install in Units
[root@ ~] # make[1]: Entering directory /afs /slac.stanford.edu /u /ey /p erl/ CLHEP /2.0.4.5 /CLHEP /Units'

[root@ ~] # Making install in Units
[root@ ~] # make[2]: Entering directory `/afs /slac.stanford.edu /u /ey /perl /CLHEP /2.0.4.5 /CLHEP /Units /Units'
```

E terminará com:

```
[root@ ~] # test -z "/u/ey/perl/CLHEP/include/CLHEP" || mkdir -p --
"/u/ey/perl/CLHEP/include/CLHEP"
[root@ ~] # if test -f ClhepVersion.h; then \
[root@ ~] # echo " /usr /bin /install -c -m 644 'ClhepVersion.h' '/u /ey /perl
/CLHEP /include /CLHEP /ClhepVersion.h'"; \
[root@ ~] # /usr/bin/install -c -m 644 "ClhepVersion.h"
"/u/ey/perl/CLHEP/include/CLHEP/ClhepVersion.h"; \
[root@ ~] # else :; fi
 [root@ ~] # /usr/bin/install -c -m 644 'ClhepVersion.h' '/u /ey /perl /CLHEP
/include /CLHEP /ClhepVersion.h'
[root@ ~] # make[2]: Nothing to be done for `install-data-am'.
[root@ ~] # make[2]: Leaving directory `/afs /slac.stanford.edu /u /ey /perl
/CLHEP /2.0.4.5 /CLHEP'
[root@ ~] # make[1]: Leaving directory ` /afs /slac.stanford.edu /u /ey /perl
/CLHEP /2.0.4.5 /CLHEP'
```

Até este ponto todos os preparativos para instalar o código Geant4 estão feitos, nenhuma mensagem de erro deve ter sido reportada, caso contrário a instalação não terá sucesso. Para maiores informações sobre a instalação do *CLHEP* pode ser acessado o *web site* do *CERN*.

O código Geant4 pode ser obtido gratuitamente da página de *downloads* do *web site* de seus desenvolvedores. A versão compactada tem aproximadamente 20 MB. Deve ser feita a descompactação no diretório raiz (/), através do comando:

```
[root@ ~] # tar -zxvf geant4.9.3.p01.tar.gz <Enter>
```

Após o descompactação é necessário instalar bibliotecas adicionais necessários para a execução das simulações. Estes arquivos são disponibilizados no mesmo *web site* do código Geant4 e devem ser baixados e descompactados dentro da pasta *geant4.9.3.p01* em um subdiretório que deverá ser criado com o nome data. A subpasta data deve conter:

- ✓ Neutron data files with thermal cross sections - version 3.13
- ✓ Neutron data files without thermal cross sections - version 0.2
- ✓ Data files for low energy electromagnetic processes - version 6.9
- ✓ Data files for photon evaporation - version 2.0
- ✓ Data files for radioactive decay hadronic processes - version 3.2
- ✓ Data files for nuclear shell effects in INCL/ABLA hadronic model - version 3.0

[root@ ~] # Data files for measured optical surface reflectance - version 1.0

Uma vez montada a pasta data se faz necessário retornar ao diretório /geant4/geant4.9.3.p01, onde deve-se digitar o comando:

```
[root@ ~] #./ Configure-build <Enter>
```

Iniciará uma série de perguntas, cuja maioria deve ser respondida com o padrão que *Geant4* oferece, exceto para as:

```
[root@ ~] # Enable building of the X11 OpenGL visualization driver? [n] <Enter>
[root@ ~] # Enable building of the X11 RayTracer visualization driver? [n] <Enter>
```

Uma vez que todas as perguntas forem respondidas, será informado que a instalação esta pronta para prosseguir, então basta pressionar *enter*.

Ao fim da instalação, caso não ocorra nenhum erro, será informado:

```
[root@ ~] # Building library management utility liblist ...
[root@ ~] # Libraries installation completed !
[root@ ~] # # Your Geant4 installation seems to be successful!
[root@ ~] # # To be sure please have a look into the log file:
[root@ ~] # /u /ey /perl /geant4 /geant4.9.3.p01 /.config /bin /Linux-g++ /g4make.log
```

Caso ocorra algum erro durante a instalação, o fórum dos usuários do código Geant4 pode ser de grande valia para auxiliar na correção.

3.3 Tomografia de Crânio

Na realização do exame de tomografia convencional, ou seja, tomografia computadorizada por transmissão de raios X, o tubo de raios X faz uma translação ou rotação em torno do alvo, as estruturas internas do corpo atenuam o feixe de raios X de acordo com a densidade e número atômico de cada tecido. A intensidade da radiação detectada pelos sensores de raios X varia de acordo com a atenuação de cada parte do alvo e forma uma lista de intensidades para cada projeção. No fim da translação ou rotação, o conjunto tubo de raios X e detectores retorna para a posição inicial, a mesa com o paciente se movimenta em alguns milímetros, e o tomógrafo começa uma nova varredura. Este processo é repetido inúmeras vezes, gerando uma grande quantidade de dados.

Os dados obtidos, intensidade de raios X, valores de atenuação, a posição da mesa e a posição do *gantry* quando da obtenção dos dados, são armazenados em um computador. Com auxílio de equações matemáticas aplicadas sobre estes valores, é possível determinar as relações espaciais entre as estruturas internas de uma região selecionada do corpo de prova ou de um corpo humano.

A imagem apresentada na tela consiste em uma matriz de valores de atenuação, ou, em cálculo inverso, uma matriz com valores de dose absorvida. Visualmente, para o diagnóstico, os valores de atenuação são apresentados na forma de tons de cinza, criando assim uma imagem espacial do objeto varrido. No fim deste processo é possível deixar armazenada esta imagem. Para isso se faz necessário salvá-la no formato *DICOM* (.dcm).

Como a imagem foi gerada por diferenças de atenuação de cada material que compõem o corpo de prova/humano é possível, de acordo com as imagens, determinar quais são os materiais que compõem o alvo. Seguindo estas técnicas, o *“Código Dicom”* possui em seu código uma lista com alguns materiais do corpo humano. Na Tabela 1 são listados alguns materiais com a composição, densidade e quantidade de elementos que serão utilizados por este trabalho.

Tabela 1 - Lista com os materiais, elemento dos materiais, densidade dos materiais e quantidade de elementos que compõem os tecidos do corpo humano.
Fonte: *"Geant4 Release 4.9.3.P01"*

Material	Elementos dos materiais			Densidade do material	Quantidade de elementos
Água	Hidrogênio	1.01*g/mole	Z=1	1.018*kg/m^3	0.112
	Oxigênio	16.00*g/mole	Z=8		0.888
Osso	Hidrogênio	1.01*g/mole	Z=1	1575*kg/m^3	0.085
	Carbono	12.01*g/mole	Z=6		0.404
	Nitrogênio	14.007 *g/mole	Z=7		0.058
	Oxigênio	16.00*g/mole	Z=8		0.434
	Sódio	22.98977*g/mole	Z=11		0.001
	Magnésio	24.30050*g/mole	Z=12		0.001
	Fósforo	30.973976*g/mole	Z=30		0.072
	Enxofre	32.065*g/mole	Z=16		0.003
	Cloro	35.453*g/mole	Z=16		0.001
	Potássio	30.0983*g/mole	Z=19		0.146
	Cálcio	40.078*g/mole	Z=20		0.146
Músculo	Hidrogênio	1.01*g/mole	Z=1	1061*kg/m³	0.102
	Carbono	12.01*g/mole	Z=6		0.143
	Nitrogênio	14.007 *g/mole	Z=7		0.034
	Oxigênio	16.00*g/mole	Z=8		0.710
	Sódio	22.98977*g/mole	Z=11		0.001
	Fósforo	30.973976*g/mole	Z=30		0.002
	Enxofre	32.065*g/mole	Z=16		0.003
	Cloro	35.453*g/mole	Z=16		0.001
	Potássio	30.0983*g/mole	Z=19		0.004
Tecido Adiposo	Hidrogênio	1.01*g/mole	Z=1	967*kg/m³	0.114
	Carbono	12.01*g/mole	Z=6		0.598
	Nitrogênio	14.007 *g/mole	Z=7		0.007
	Oxigênio	16.00*g/mole	Z=8		0.278
	Sódio	22.98977*g/mole	Z=11		0.001
	Enxofre	32.065*g/mole	Z=16		0.001
	Cloro	35.453*g/mole	Z=16		0.001
Ar	Nitrogênio	14.007 *g/mole	Z=7	1.290*mg/cm³	0.7
	Oxigênio	16.00*g/mole	Z=8		0.3

A Figura 10 é a representação de uma das imagens *DICOM* que será utilizada para determinar os elementos que irão compor o *phantom*, assim como a sua densidade e quantidade de elementos.

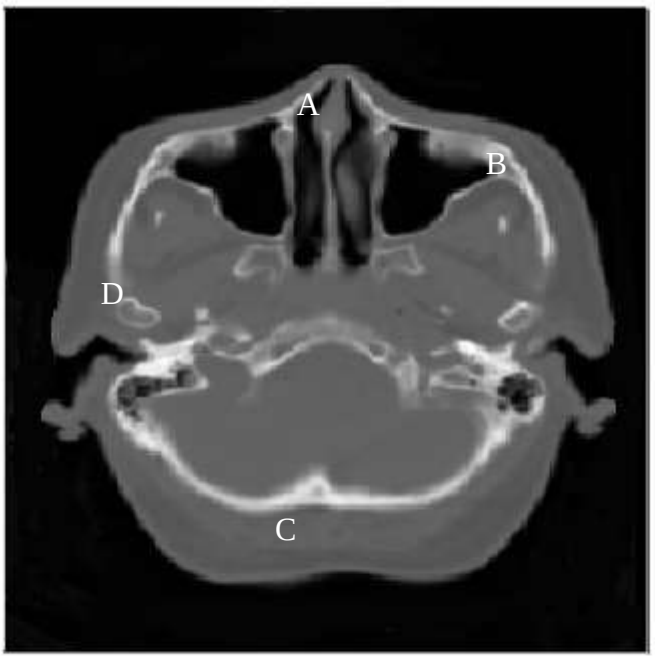


Figura 10 - Imagem *DICOM* de um corte axial de crânio. (A) Septo Nasal, (B) Osso Zigomatico, (C) Protuberância do Occipital, (D) Canal Auditivo.

Com auxílio do *MatLab* também é possível gerar o gráfico de intensidade (Z) por quantidade de *pixel* (X e Y), como é demonstrado na Figura 11.

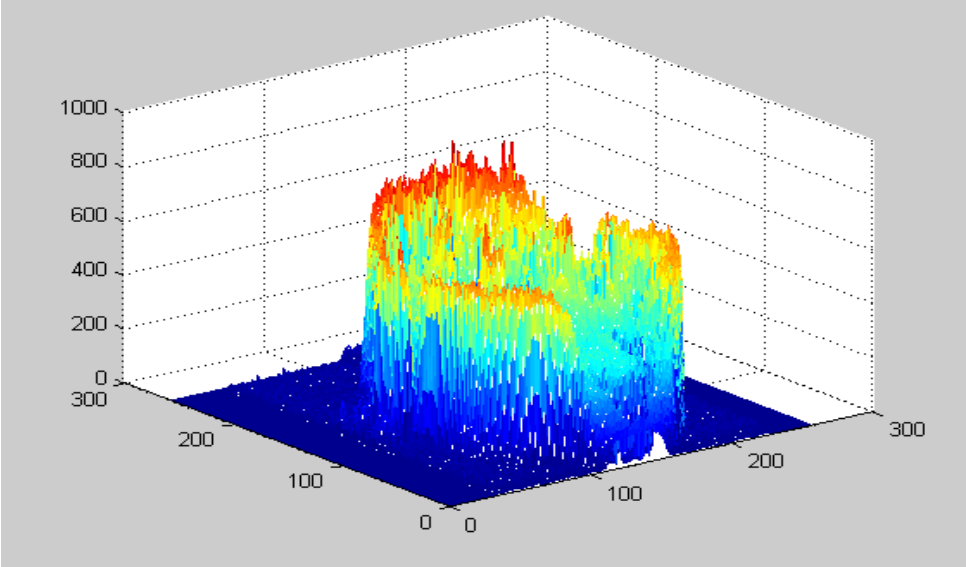


Figura 11 - Intensidade (Z) por quantidade de *pixel* (X e Y) da imagem de *CT* de crânio.

Na realização do exame de crânio rotina mais base de crânio são adquiridas 20 imagens *DICOM* (formato *.dcm*), que podem ser visualizadas na Figura 12. As imagens foram produzidas por um aparelho de *CT* da marca *Siemens modelo Somatom Emotion* (*Multislice* – 16 canais). Essas imagens servirão de base para a elaboração do *phantom* virtual.

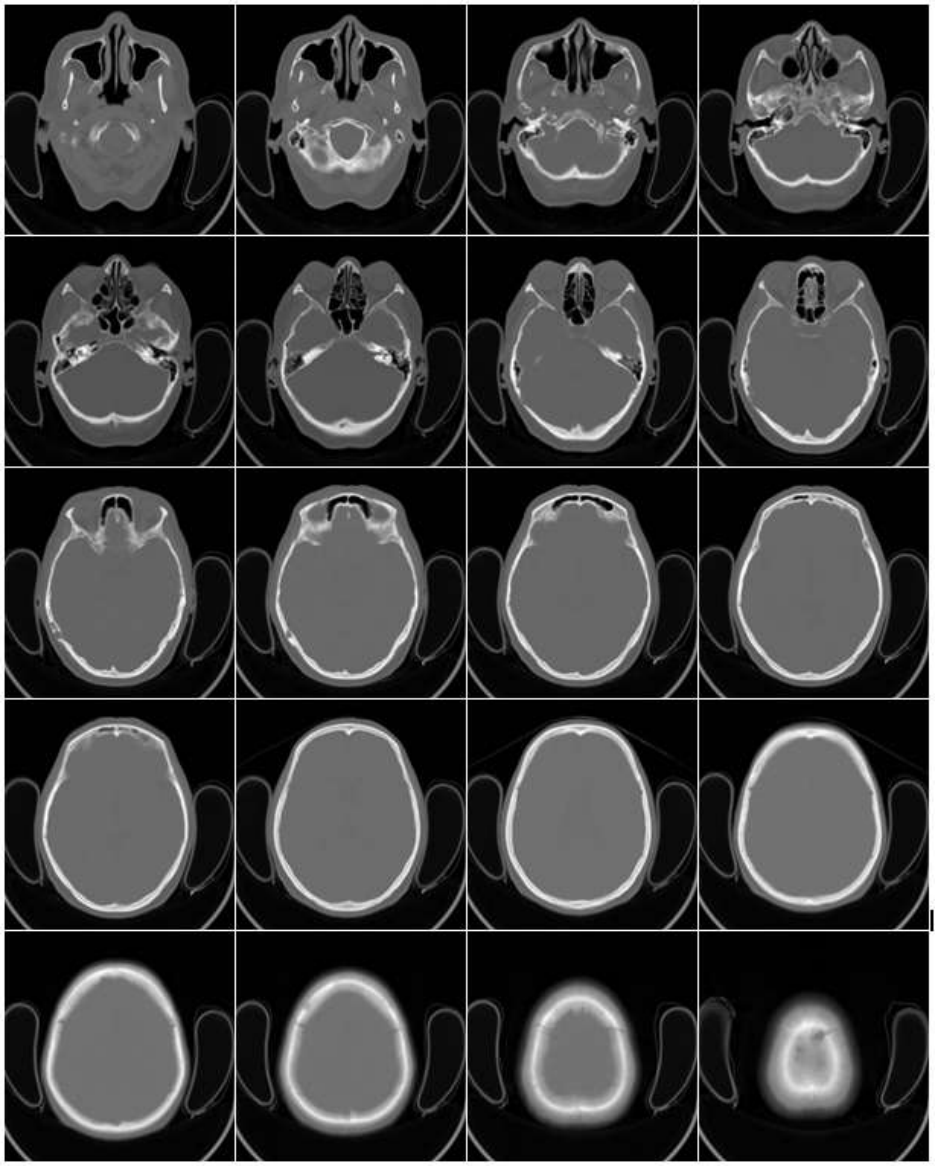


Figura 12 - Imagem dos 20 cortes que compõem o crânio.

Juntando todas as imagens da *CT* de crânio é possível realizar a reconstrução 3D, como está representado na região A da Figura 13. Na região B da Figura 13 podemos observar o terceiro corte da base do crânio. O terceiro corte da base do crânio foi escolhida por este trabalho para ser comentada nos próximos tópicos.

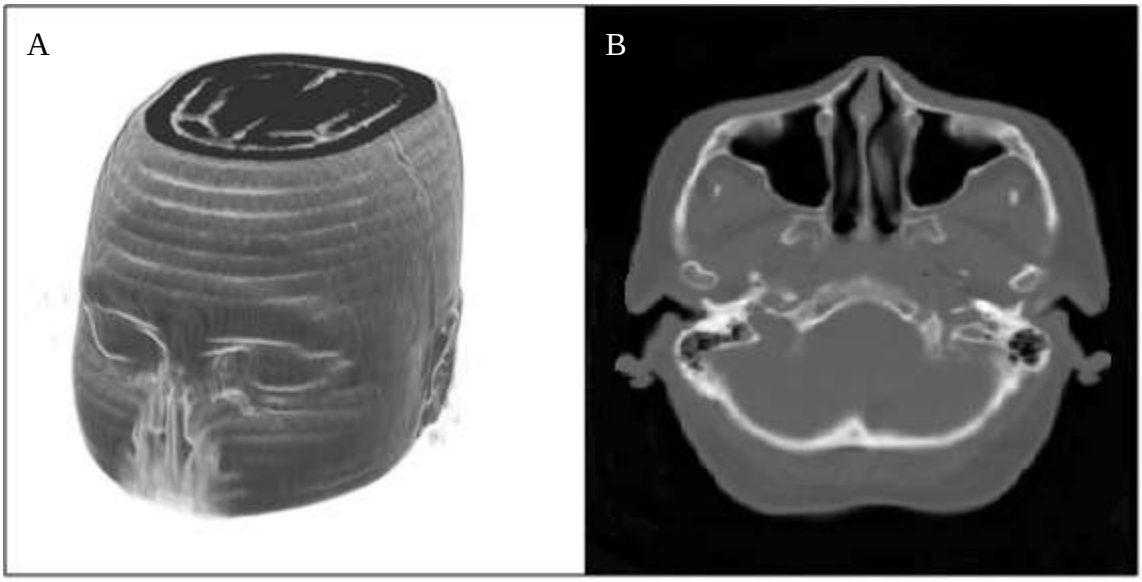


Figura 13 - (A) Reconstrução 3D. (B) Terceiro corte axial da base do crânio.

3.4 "Código Dicom"

Este trabalho utiliza a estrutura de um código feito em trabalhos anteriores (CERN, 2010) por um grupo de pesquisa canadense, que faz parte da biblioteca de exemplos médicos do Geant4, chamado "*Código Dicom"*. Para a conversão de imagens *DICOM* de *CT* de crânio real humano em um *phantom* virtual foi necessário realizar alterações nos comandos do "*Código Dicom"*.

3.4.1 Estrutura do "*Código Dicom"*

O "*Código Dicom"* foi construído para transformar as imagens do *phantom* físico de tórax, mencionado no item 2.3, em um *phantom* virtual para Geant4. O "*Código Dicom"* é composto por vários códigos fontes cada qual com sua finalidade. Esses códigos fontes são:

- ✓ DicomDetectorConstruction.cc: Código principal que fornece a estrutura dos objetos, composição e posicionamento.

- ✓ DicomEventAction.cc: Código que monitora e registra a trajetória das partículas.
- ✓ DicomHandler.cc: Código que manipula o arquivo *DICOM* para gerar uma matriz densidade correspondente ao tipo de tecido lido na imagem.
- ✓ DicomNestedPhantomParameterisation.cc: Código que transforma cada *pixel* da Imagem *DICOM* em um *Voxel* e sua posição dentro do volume.
- ✓ DicomPatientZSliceHeader.cc: Código que estrutura o cabeçalho de sequência de fatias para o objeto 3D
- ✓ DicomPhantomParameterisationColour.cc: Define a cor do *phantom* e das partículas/ondas incidentes, assim como a cor do resultado destas radiações com a matéria.
- ✓ DicomPhysicsList.cc: Código responsável pelos eventos físicos nucleares, interação entre partículas.
- ✓ DicomPrimaryGeneratorAction.cc: Código responsável pela geração do feixe de partículas carregadas, prótons, elétrons e íons.
- ✓ DicomRun.cc : Código responsável pela ação inicial de cada evento.
- ✓ DicomRunAction.cc: Código responsável pela ação final de cada evento.
- ✓ G4PSDoseDeposit_RegNav.cc: Código responsável por registrar a energia depositada.
- ✓ NestedParamDicomDetectorConstruction.cc: Código que estrutura o detector de radiação (gama, raios-X).

Cada um desses códigos fonte possui um arquivo .hh correspondente, que são conhecidos por classes. Nessas classes são declaradas as estruturas que foram incluídas no código .cc.

3.4.2 Funcionamento do "*Código Dicom"*

O "*Código Dicom"* associa o número de cada *pixel* da imagem *DICOM* a uma densidade, e fazendo uso da ICRU 46, associa a respectiva densidade a um material. Para formar uma geometria tridimensional são acoplados os *pixel*s de duas

imagens ou dois cortes de *CT*, formando assim os *voxels*. Os valores dos *pixels* da imagem *DICOM* de *CT* representam números, Escala Hounsfield, que são convertidos para uma determinada densidade.

Uma vez conhecida a densidade podemos associá-la aos materiais que compõem o corpo. A Comissão Internacional de Medidas e Unidades de Radiação (ICRU), relatório 46, foi usada para determinar qual a densidade que compõem cada estrutura. As associações de materiais utilizadas no *"Código Dicom"* foram para materiais que compõem o tórax humano, como demonstra a Tabela 2. O arquivo responsável para determinar a faixa de densidade no *"Código Dicom"* é o *CT2Density.dat*.

Tabela 2 - Faixe de densidade em g/cm³ e os respectivos materiais do tórax.

Faixa de Densidade (g/cm³)	**Materiais**
[0.100 ; 0.351]	Pulmão com Ar
[0.351 ; 0.800]	Pulmão sem Ar
[0.919 , 0.979]	Tecido Adiposo
[0.979 , 1.004]	Seio
[1.004 , 1.043]	Água
[1.043 , 1.109]	Fígado
[1.109 , 1.113]	Musculo
[1.113 , 1.400]	Osso
[1.496 , 1.654]	Osso Denso

Concluindo a conversão da intensidade de cada *pixel* para uma determinada densidade, associando cada uma a um material temos um arquivo de saída com números de 1 a 9, representando cada um dos tecidos do tórax, como é descrito na Tabela 3:

Tabela 3 - Tecido do tórax e seu número de identificação.

Número de Identificação	Materiais
1	Pulmão com Ar
2	Pulmão sem Ar
3	Tecido Adiposo
4	Tecido Mamário
5	Água
6	Fígado
7	Musculo
8	Osso
9	Osso denso

Na Figura 14 temos a exemplificação da associação dos *pixels* de duas em duas imagens de *CT* (A), para a construção dos *voxels* (B) e na região C da figura temos a representação final do conjunto de *voxels* que formam o *phantom*. A letra i é a distancia entre os dois cortes, L é o comprimento do voxel no eixo x, d é o comprimento no eixo Z e P é o comprimento do *voxel* no eixo y.

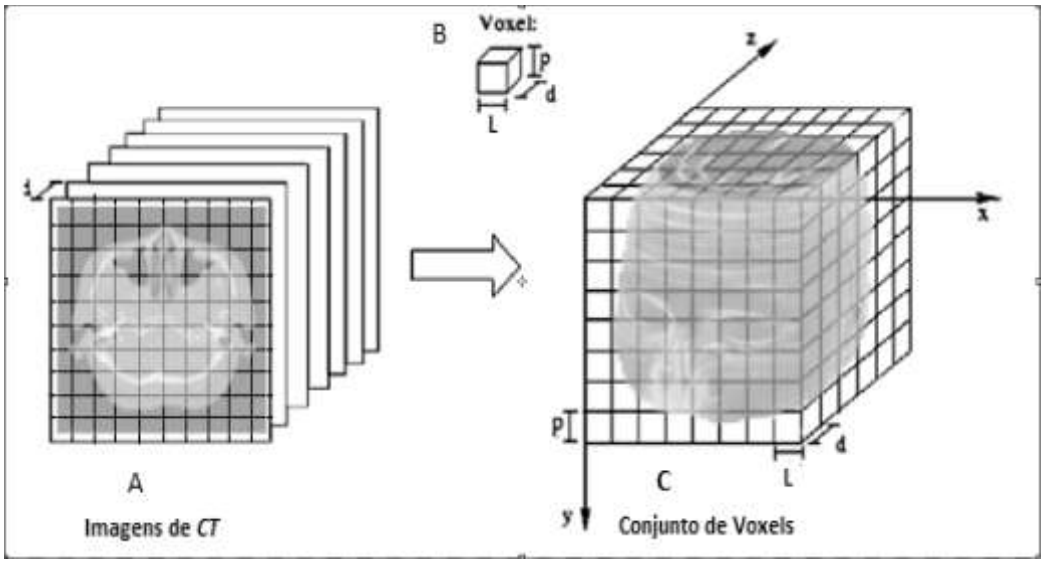


Figura 14 - Imagens de *CT* (A), *voxel* (B), conjunto de *voxel* formado a partir das imagens de *CT*.

3.4.3 Modificações no "*Código Dicom*"

O *"Código Dicom"* foi desenvolvido para rodar apenas com a imagem de *CT* do *phantom* físico de tórax, Figura 5. Por esse motivo, se faz necessário alterar os

materiais para tecidos que compõem o crânio humano. Deverá ser retirado do código o tecido pulmão com ar e pulmão sem ar e substituí-los apenas por Ar. Também se faz necessário substituir o elemento água por cérebro, tendo em vista que suas densidades são próximas, e retirar o material fígado e a sua faixa de densidade acrescentar ao do tecido muscular. Esses parâmetros são alterados no arquivo *Data.dat.*

Tendo em vista que a etapa em que se encontra esse trabalho é a conversão de imagens *DICOM* em um *phantom* de crânio, deve-se retirar do código todos os passos que são referentes aos eventos físicos nucleares, de interação entre partícula, da geração de feixes de partícula e o código que monitora e registra a trajetória das partículas. Os arquivos *DicomPhysicsList.cc, DicomPhysicsList.hh, DicomPrimaryGeneratorAction.cc, DicomPrimaryGeneratorAction.hh, DicomEventAction.cc* e o *DicomEventAction.hh* devem ter parte de seus códigos desativados.

Quando as duas imagens *DICOM* de *CT* são acopladas para formar os *voxels*, o *"Código Dicom"* faz uma média entre dois *pixels* vizinhos e forma um *voxel*. Isso faz com que se tenha um *phantom* com a metade das informações da imagem inicial. Neste trabalho foram retirados do código os comandos que fazem a média dos *pixels*, mantendo assim mais informações da imagem de *CT* e com isso reduzindo distorções entre a imagem real de crânio e o *phantom* de crânio. Para suspender essa compressão é necessário desativar parte dos comandos do arquivo *DicomHandler.cc.* Os códigos do arquivo DICOMHandler.cc estão no Anexo B.

3.4.4 Comandos para a Conversão das Imagens de *CT* em um *Phantom* de Crânio

Antes de compilar o código é necessário carregar o env.sh e para isto deve-se abrir o terminal do *Linux* e digitar o comando:[root@ ~] # source env.sh <Enter>

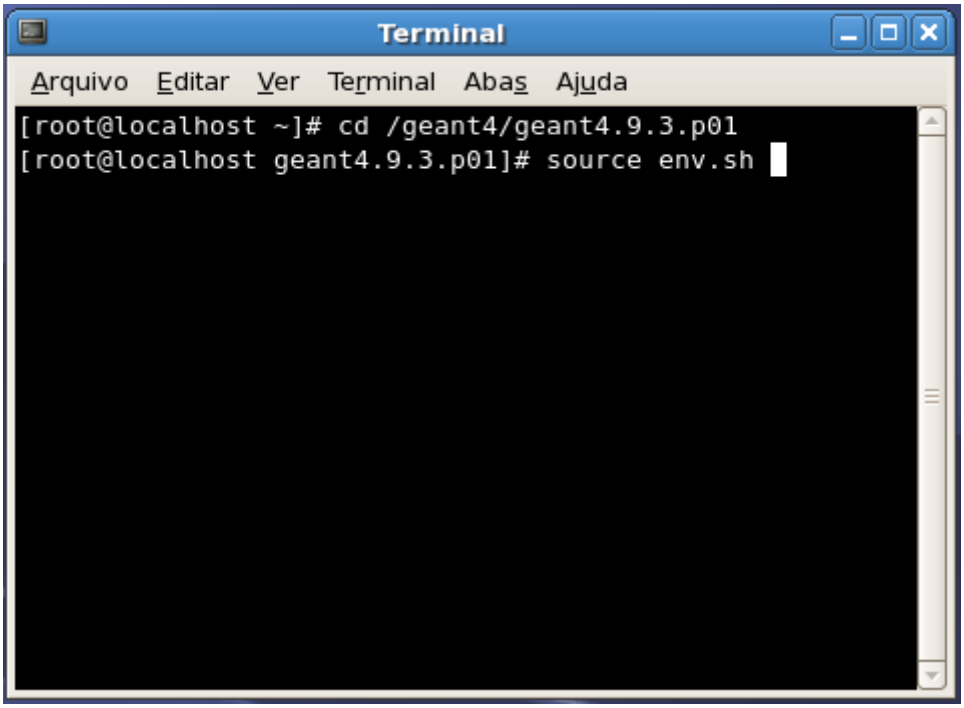


Figura 15 - Terminal com os comandos necessários para carregar *env.sh*.

Após carregar o *env.sh*, deve-se ir até a pasta que contem os códigos do *phantom* para carregar seu conteúdo, para isto deve-se digitar o comando:

```
[root@ ~] # make <Enter>
```

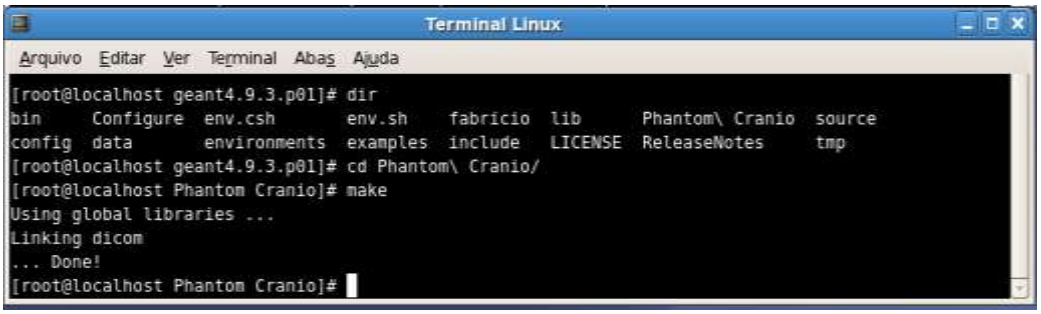


Figura 16 - *Phantom* compilado.

Ao carregar os códigos do *phantom* é gerado um arquivo executável *PhatomCranio.exe* dentro da pasta */geant4/geant4.9.3.p01/Bin/Linux*. Para executá-lo é necessário digitar *./PhatomCranio.exe*. Após executar, é formada uma imagem *.prim* e para visualizar o *phantom* é utilizado o programa *Dawn*.

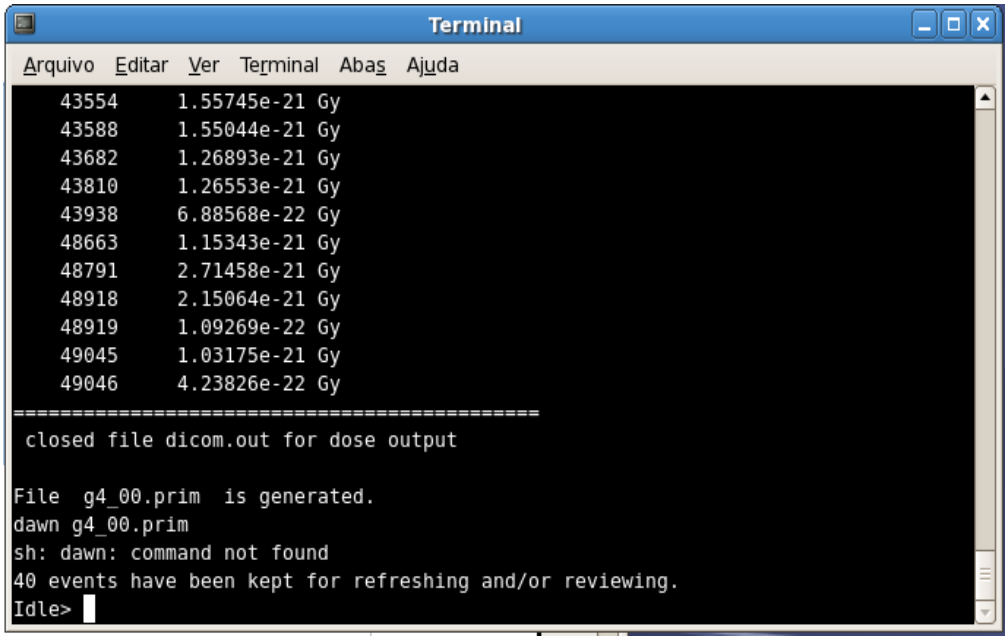


Figura 17 - *Phantom* pronto para ser visualizado.

Para abrir a imagem do *phantom* basta dar dois cliques e escolher qual é o ângulo que se deseja visualizar.

A imagem do *phantom* antropomórfico de crânio está representada na Figura 18.

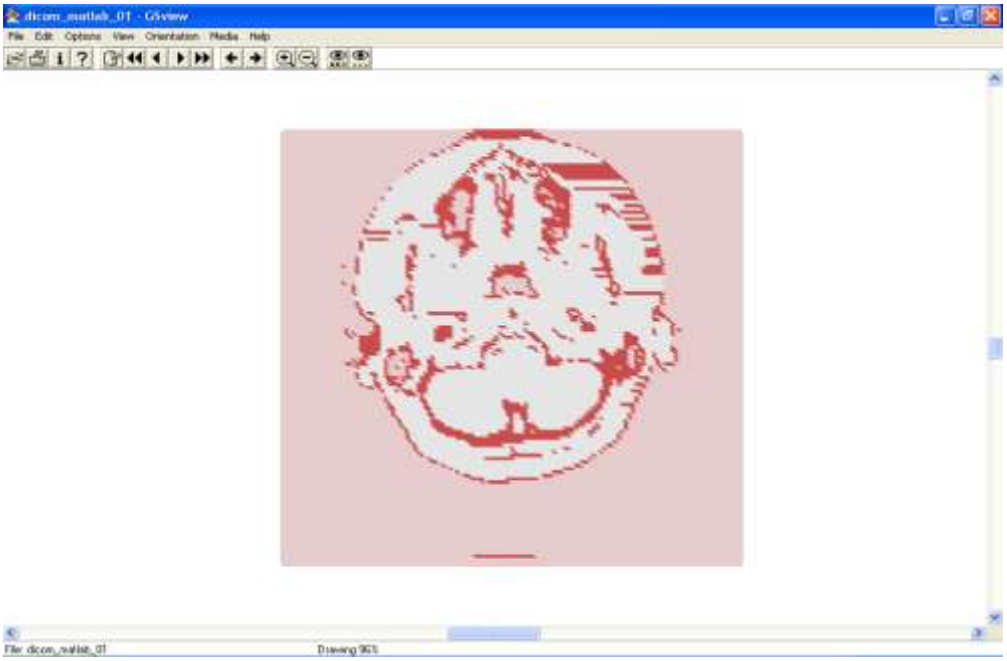


Figura 18 - Visualização da base *phantom* de crânio através do programa *Dawn*.

4 RESULTADOS E DISCUSSÕES

4.1 Apresentação do *Phantom* Virtual de Crânio

Depois de compilar o código e converter cada intensidade de *pixel* para uma determinada densidade e associá-lo a um determinado material obtemos os arquivos de saída em duas extensões distintas, uma *.eps* e o outra no formato *.gdcm*. O formato *.eps* é a imagem formada pelos 512x512 *voxels* de cada um dos 20 cortes. Já os arquivos *.gdcm* são a representação numérica dos materiais presentes em cada um dos cortes. O número de arquivos de saída *.gdcm* depende do número de cortes a ser compilados.

4.1.1 Arquivo de Saída *DICOM.gdcm*

O arquivo *DICOM.g4dcm* é uma matriz de 512x512 composta pelos números 0, 3, 5, 6, 8 e 9 que correspondem respectivamente ao ar, tecido adiposo, musculo, cérebro, osso e osso denso, como demonstra a Tabela 4. Ao abrir essa matriz em uma planilha eletrônica é possível visualizar os números que representa cada um dos materiais que compõem o *phantom*. Na Figura 19 é possível observar os 262.144 números da matriz. Na Figura 20 está representada a região A da Figura 19 aproximada, para melhor visualização dos números que compõem a matriz.

Tabela 4 - Número de identificação de cada material do *phantom* de crânio.

Número de Identificação	**Materiais**
0	Ar
3	Tecido Adiposo
5	Tecido Muscular
6	Cérebro
8	Osso
9	Osso denso

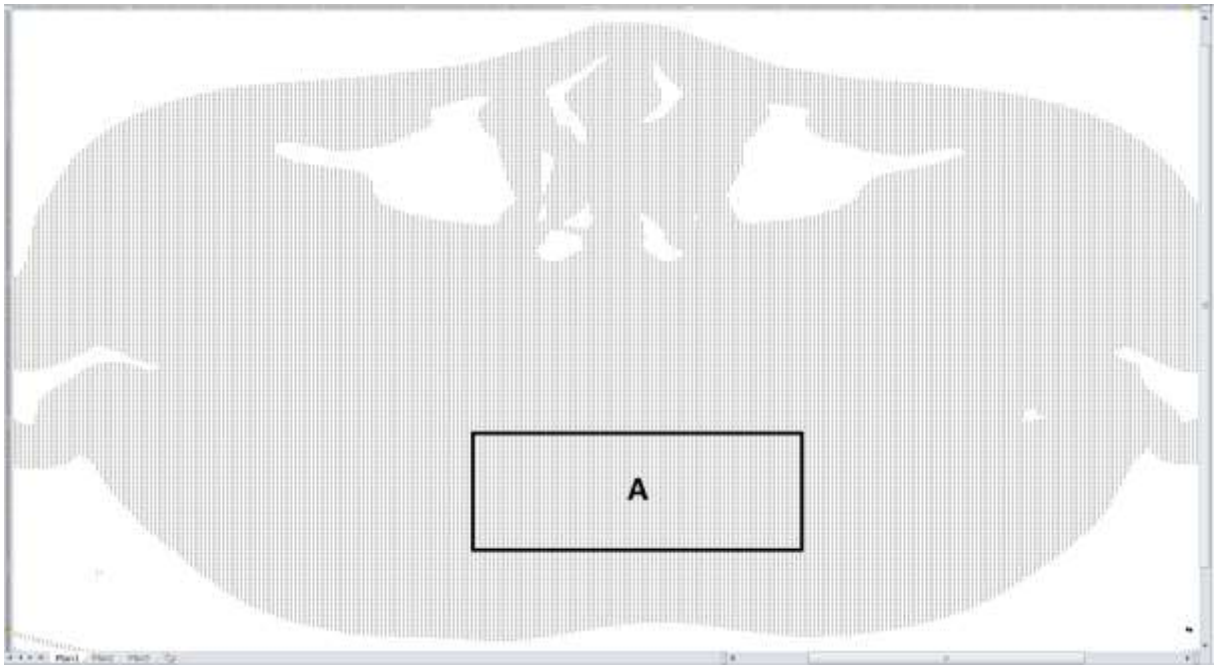


Figura 19 - Matriz 512 X 512 do *phantom* de crânio.

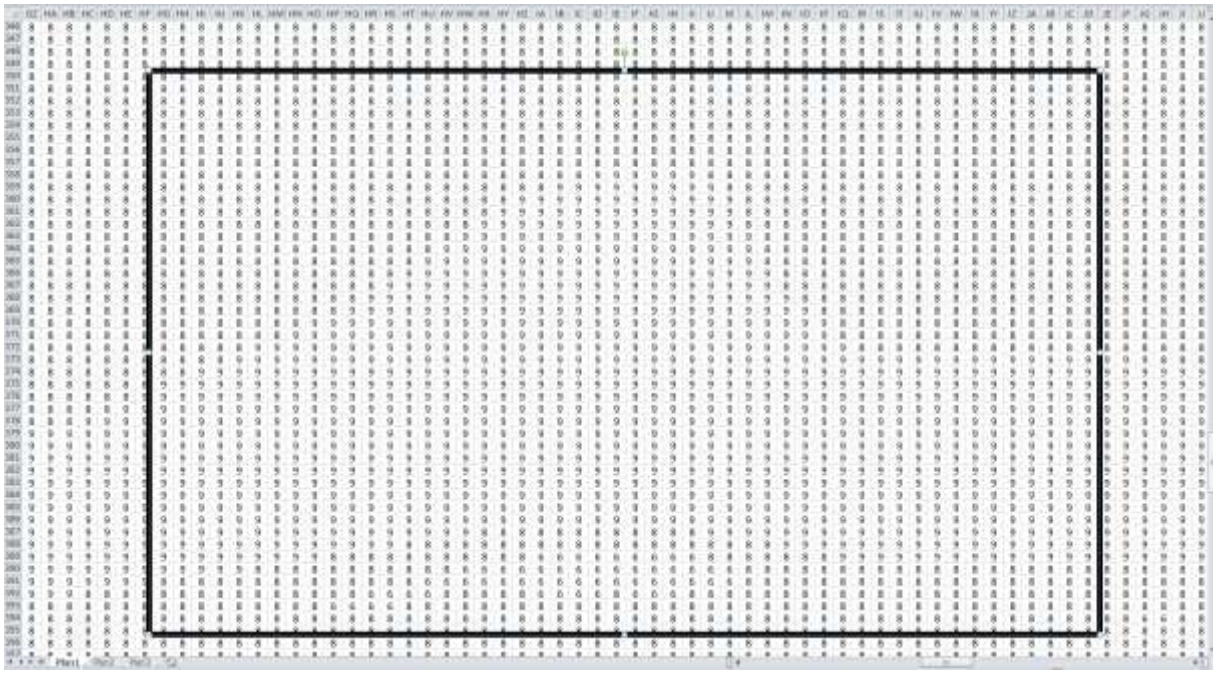


Figura 20 - Região A da matriz da Figura 19 ampliada para melhor visualização.

Adotando uma escala de cinza para os valores que compõem a matriz do *phantom* é possível comparar semelhança do *phantom* de crânio com a imagem de crânio de *CT*, como demonstra a Figura 23. Para adotar uma escala de cinza basta selecionar a matriz, ativar a opção Formatação Condicional do Excel e escolher Escala Tricolor, como é demonstrado na Figura 21 e 22.

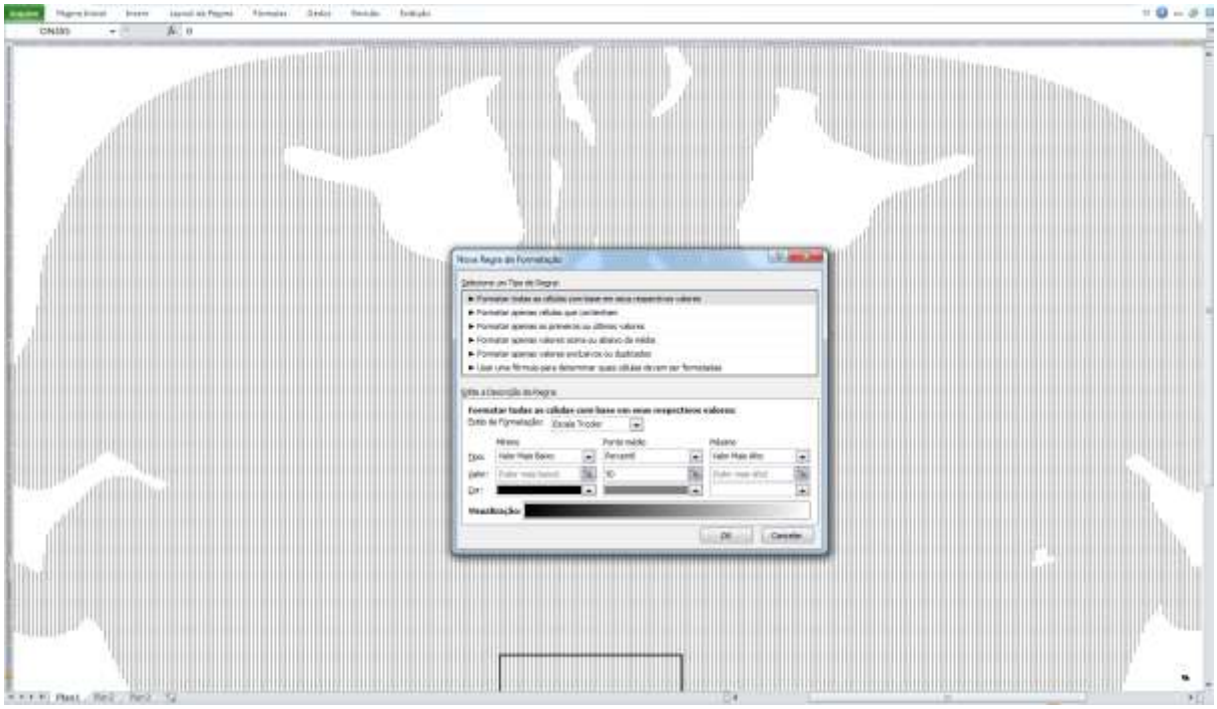


Figura 21 - Matriz 512x512 do *phantom* de crânio selecionada em Excel.

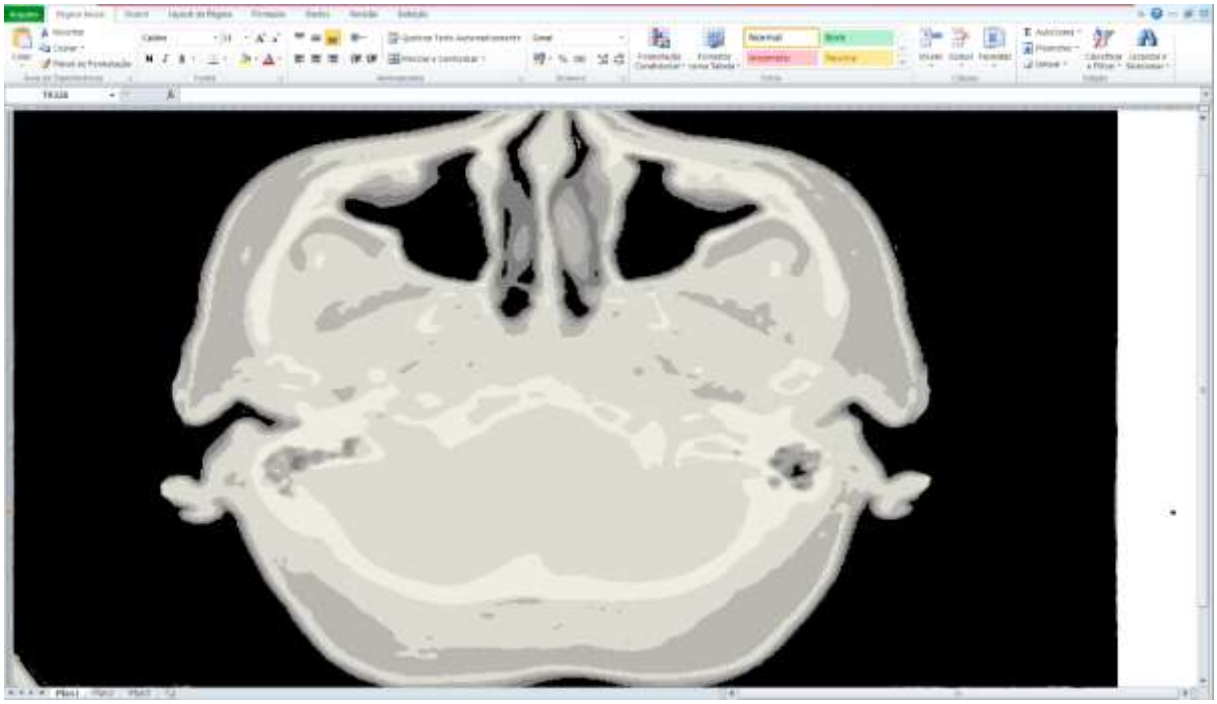


Figura 22 - Matriz 512x512 do *phantom* de crânio em escala de cinza.

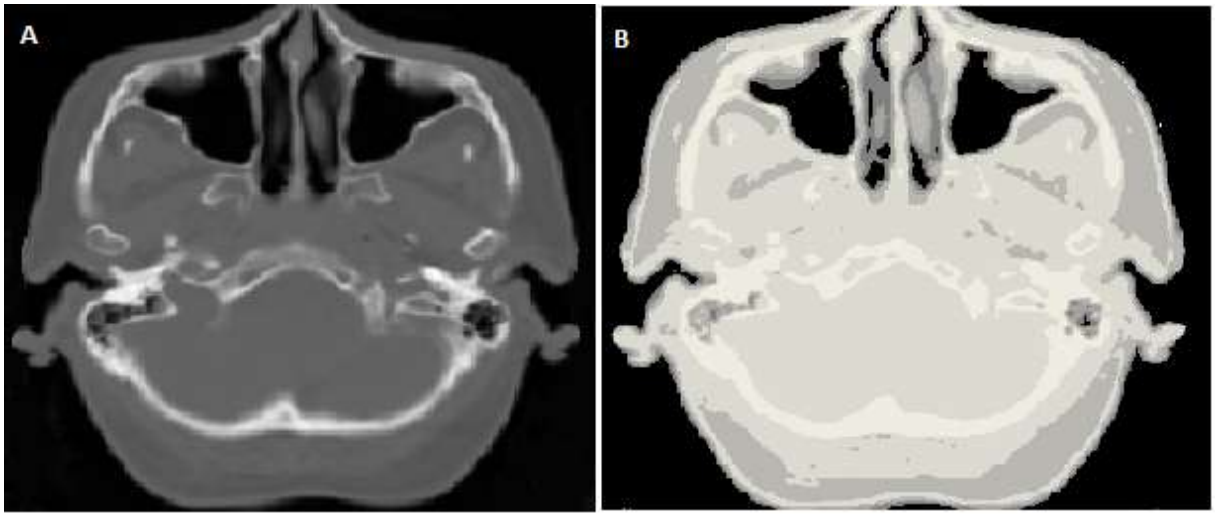


Figura 23 - (A) Imagem *Dicom* de CT, (B) matriz do *phantom* de crânio em escala de cinza.

4.1.2 Arquivo de Saída *DICOM.eps*

O arquivo *DICOM.eps* é uma imagem vetorial. A visualização do *phantom* de crânio construído para código de Monte Carlo está representado na Figura 24.

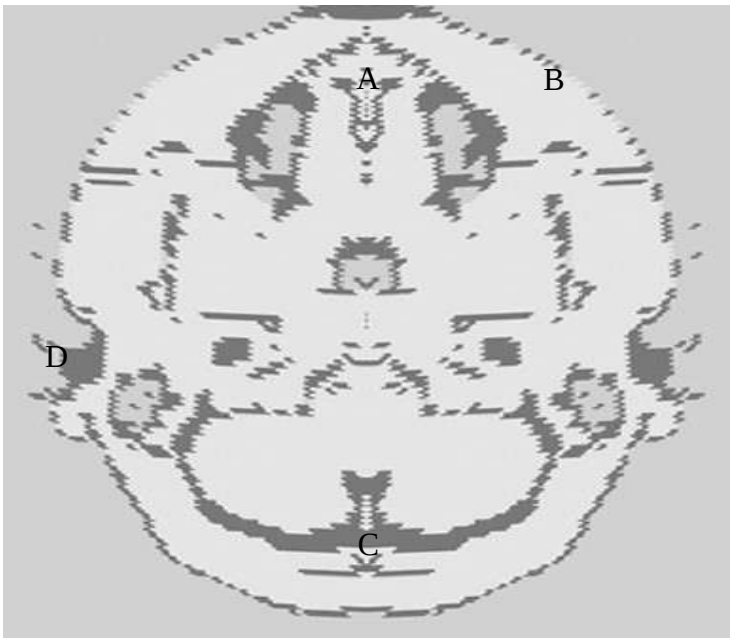


Figura 24 - O *phantom* antropomórfico parcial virtual reconstruído no código Geant4. Visualização da base do *phantom* pelo programa *Dawn* (A) Septo Nasal, (B) Osso Zigomatico, (C) Protuberância do Occipital, (D) Canal Auditivo.

4.2 Apresentação do *Phantom* Virtual de PEAD

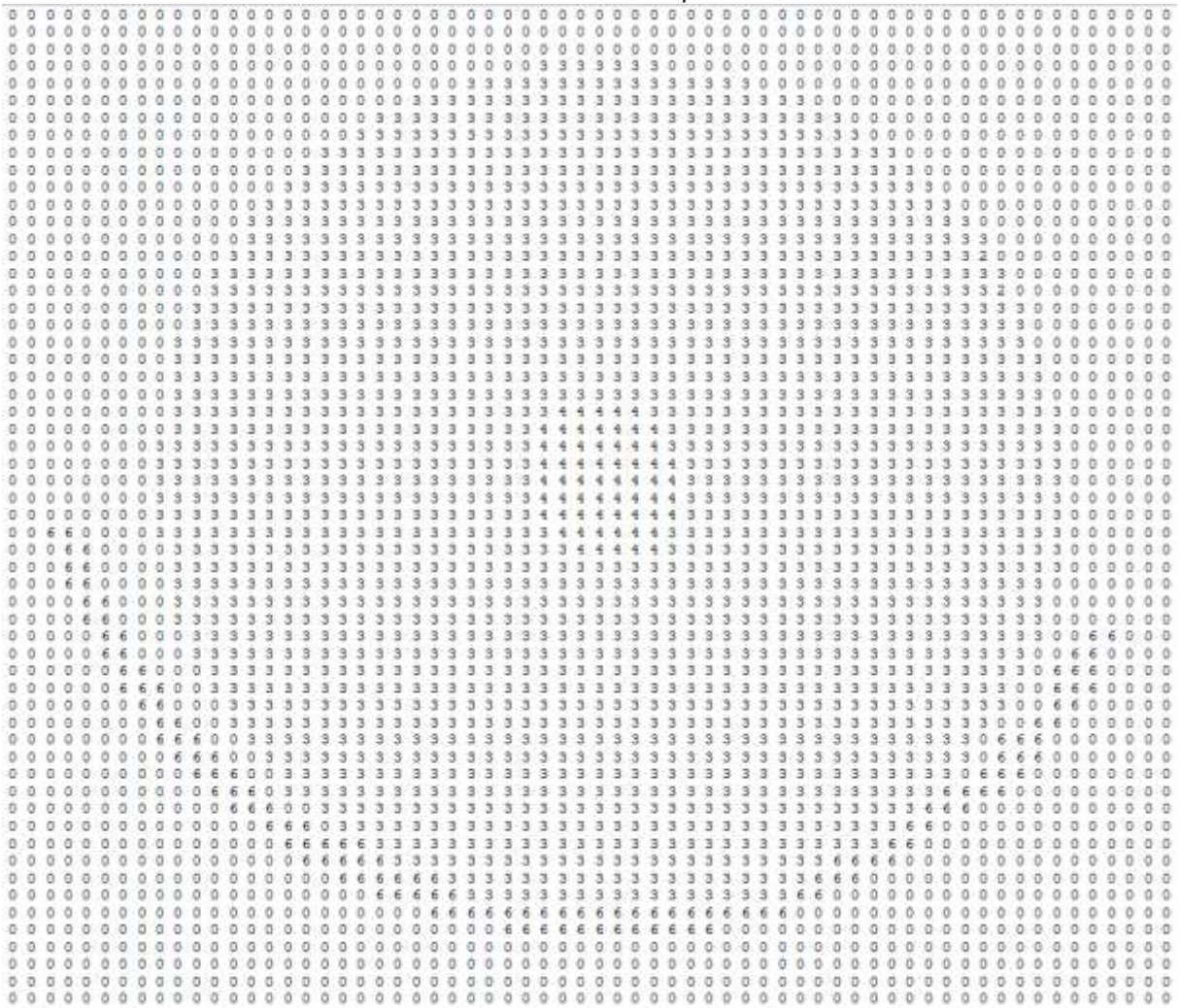
Depois de compilar o código, com três cortes de *CT* do *phantom* de PEAD, converter cada intensidade de *pixel* para uma determinada densidade e associá-lo a uma determinada faixa de densidade, temos os dois arquivos de saída, o *PEAD.gdcm* e outro *PEAD.eps*. O arquivo *PEAD.g4dcm* é uma matriz de 128x128 composta pelos números de 0, 3, 4, e 6 que correspondem respectivamente: ar, PEAD, acrílico e outro material que foi usado como suporte do *phantom* de PEAD para a realizado da *CT*. Na Tabela 5 temos os materiais e seus respectivos números de identificação.

Tabela 5 - Materiais do *phantom* de PEAD e seus respectivo números de identifcação.

Materiais	Número
Ar	0
Polietileno	3
Acrílico	4
Suporte da CT	6

Ao abrir um o arquivo *PEAD.g4dcm* podemos visualizar os números que representam os materiais do *phantom* de PEAD, como pode ser observado na Tabela 6.

Tabela 6 - Matriz 128 X 128 do *phantom* de PEAD.



Os três cortes do *phantom* físico de PEAD foram utilizados para construir o *phantom* virtual de PEAD. O primeiro e o segundo corte formam a primeira camada de *voxels* e o segundo corte com o terceiro corte formão a segunda camada de *voxels*. Na Figura 25 temos a imagem tridimensional, arquivo .*eps*, do *phantom* de PEAD.

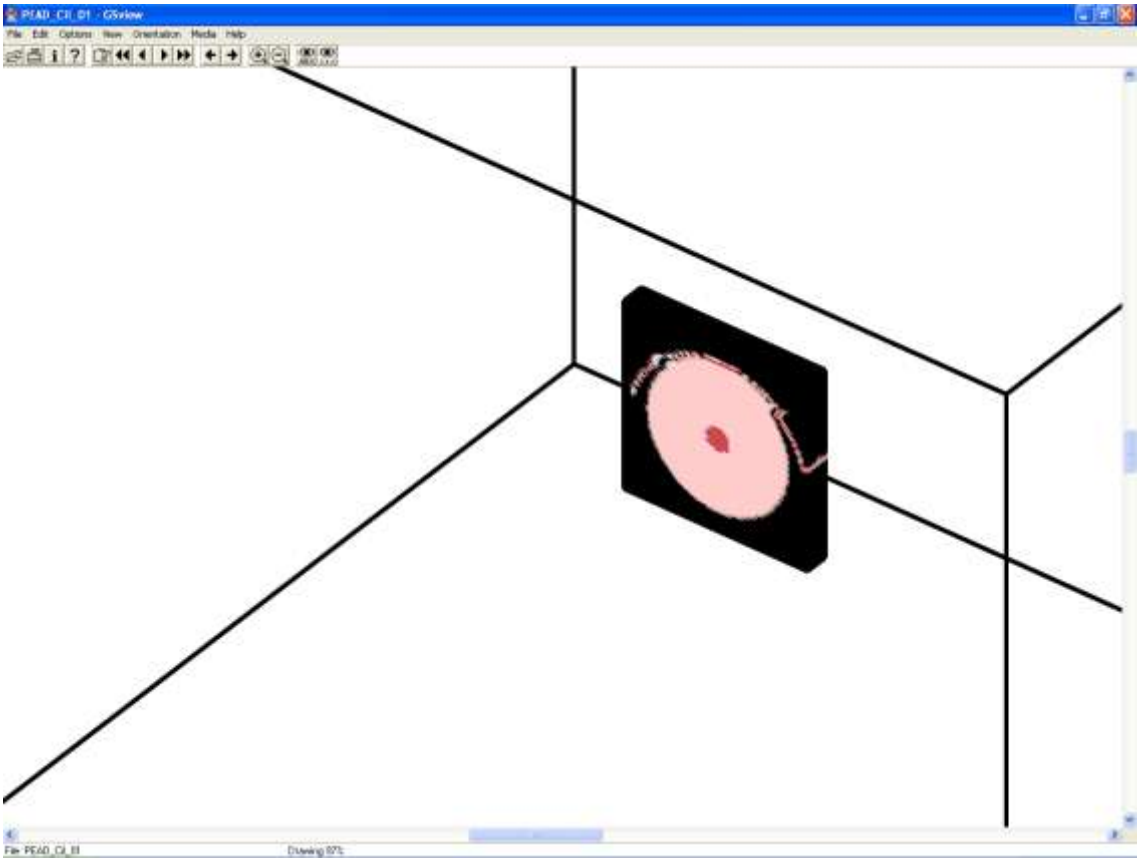


Figura 25 - Imagem 3D do *phantom* de PEAD

4.3 Discussões Sobre os Programas Utilizados

O sistema operacional *Scientific Linux CERN 5* mostrou-se mais compatível com a versão utilizada do Geant4 do que o sistema operacional da Microsoft Windows XP. Para visualizar as imagens em Linux foi usado o programa *OpenGL*, já no Windows se fez uso do *Dawn*. Por alguma incompatibilidade de programação na visualização em Linux com o programa *OpenGL* ocorreu erro, sendo necessário fazer a visualização pelo *Dawn* em Windows. Esta incompatibilidade até o momento do término deste trabalho é desconhecida. Com geometrias menos complexas no *OpenGL* não ocorreram erros.

Para gerar as figuras de intensidade por quantidade de *pixel* no *MatLab* foram usados os seguintes comandos:

```
[root@ ~] # Y = dicomread('c:\Imagem.dcm');
[root@ ~] # mesh(double(Y(:,:,1)));
[root@ ~] # waterfall(double(Y(:,:,1)));
```

A primeira linha de comando é a especificação do endereço onde se encontra a imagem *DICOM*. Tanto o *mesh* quanto o *waterfall* tem a mesma finalidade de gerar o gráfico. A única diferença entre os dois é que com o *waterfall* tem-se a possibilidade de rotação na visualização do gráfico, já o *mesh* não tem esta opção. Porém com o comando *mesh* a obtenção da figura tem processamento mais leve.

4.4 Comparação do *Phantom* Físico de Tórax e a Imagem *DICOM* de Crânio

O *phantom* físico (Figura 8) de tórax apresenta geometria mais simples do que a da imagem *DICOM* de crânio. Com isto, nota-se que quanto mais complexa for a geometria, mais necessário se faz ter um código aprimorado, ou seja, voltado especificamente para a região anatômica em reconstrução. A diferença de geometria, entre o *phantom* físico de tórax e o *DICOM* de crânio, pode ser observada na Figura 26:

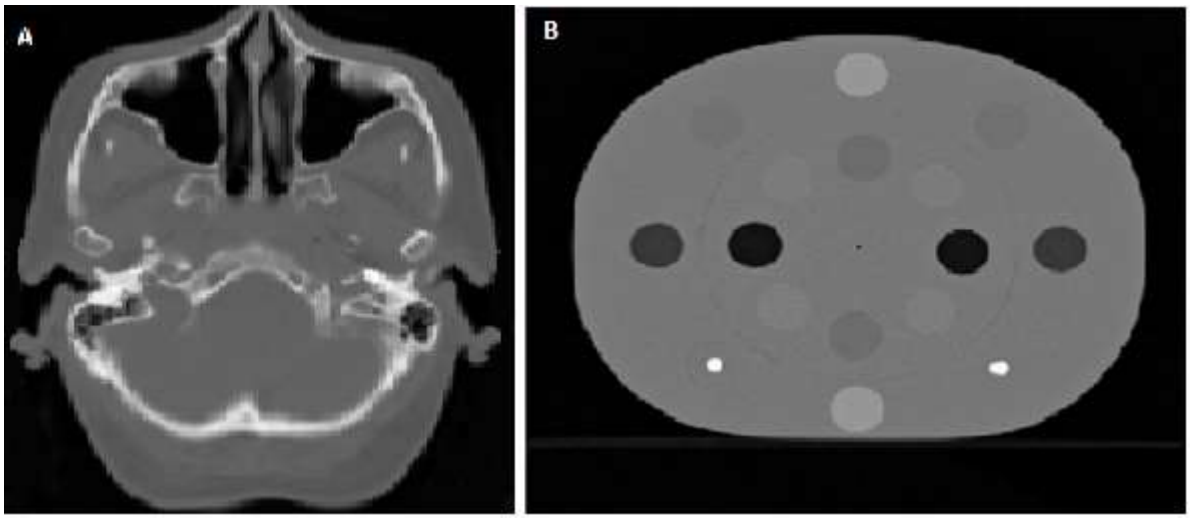


Figura 26 - (A) Corte axial de crânio (B) Imagem da tomografia do *phantom* de tórax.

Na Figura 27 podemos observar a intensidade por quantidade de *pixel* da imagem *DICOM* do *phantom* físico de tórax e da imagem *DICOM* do terceiro corte de Crânio.

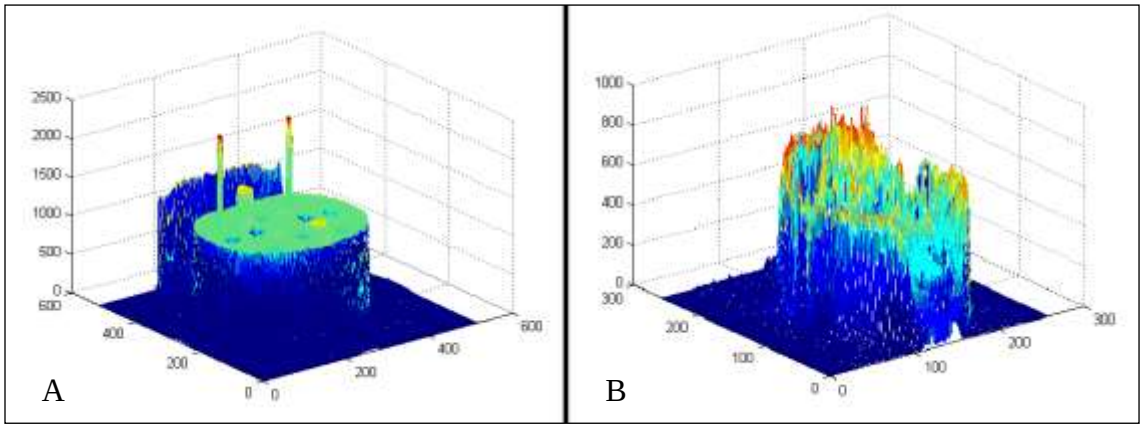


Figura 27 - Intensidade (Z) por quantidade de pixel (X e Y) da imagem de (A) Tórax e da imagem de (B) Crânio.

5 CONCLUSÕES

Neste trabalho foi demonstrado ser possível a construção direta de *phantoms* virtuais para Geant4, partindo do uso de imagens tomográficas (*DICOM*) reais, com o tempo de construção de aproximadamente um minuto.

Após a configuração do código Geant4 e do programa *Dawn*, foram realizadas reestruturações no *"Código Dicom"* para a conversão direta de imagens tomográficas em um *phantom* virtual para o código Geant4.

Foram retirados dos códigos do exemplo *"Código Dicom"* todos os passos que são referentes aos eventos físicos nucleares, de interação entre partícula, da geração de feixes de partícula, e o código que monitora e registra a trajetória das partículas. O procedimento anterior foi realizado para a redução do tempo de processamento, uma vez que são passos desnecessários para a atual etapa em que se encontra este trabalho.

O arquivo *DicomHandler.cc* foi reformulado para não realizar a compressão dos *pixels* da imagem de *CT*. Portanto, o número de *voxels* na saída é igual ao número de *pixels* da imagem *DICOM* (512x512) na entrada, mantendo assim, mais informações da imagem de *CT* e, com isso, reduzindo distorções entre a imagem real de crânio e o *phantom* virtual de crânio.

Foi realizada a conversão direta de imagens tomográficas de um *phantom* físico de PEAD com núcleo central de acrílico, em um *phantom* virtual para o código Geant4.

Finalmente foi realizada a conversão direta de imagens tomográficas de um crânio real humano em um *phantom* virtual para o código Geant4.

SUGESTÕES PARA TRABALHOS FUTUROS

- Construir um *phantom* que contemple maior região anatômica do corpo humano.
- Acrescentar ao código um feixe de radiação, colimadores e detectores para simulação da interação da radiação com o *phantom*.
- Realizar comparação entre dados obtidos em simulação com dados não computacionais.

REFERÊNCIAS

ALLISON, J. ET AL, **Geant4 developments and applications, IEEE Transactions on Nuclear Science 53**, 2006.

BIENVENUE AU LABORATOIRE DE PHYSIQUE NUCLEAIRE EXPERIMENTALE ET MEDICALE. Disponível em: <http://www.nucleaire.phy.ulaval.ca/>. Último acesso em 20/01/2013

BUKOVICS B., **Biography of Johann Radon, in: 75 Years of Radon Transform**, International Press Incorporated, 1994.

CERN: ***European Laboratory for Particle Physics***. Web site, http://geant4.web.cern.ch/geant4/UserDocumentation/Doxygen/examples_doc/html/ExampleDICOM.html, acesso em 20/01/2013.

CORMACK, A.M.; KOEHLER,A.M., **Quantitative Proton Tomography: Preliminary Experiments,** Phys. Med. Biol.,1976.

CRISTÓVÃO, M. T., CAMPOS, T. P. R., **Investigações em modulação do pico de Bragg para terapia de prótons**. 21º Congresso Brasileiro de Engenharia Biomédica, 2008.

DAWN web site, http://geant4.kek.jp/~tanaka/DAWN/About_DAWN.html, acesso em 20/01/2013.

GEANT4. Web site, http://geant4.web.cern.ch/geant4/, acesso em 20/01/2013.

GILAT, A., MATLAB: **AN INTRODUCTION WITH APPLICATIONS 2ND EDITION**, 2004.

GONZALEZ, RAFAEL C.; WOODS, RICHARD E**.** ***DIGITAL IMAGE PROCESSING***, ADDISON-WESLEY. PUBLISHING COMPANY, INC. 1992.

GONZALES, R. C.; WOODS, R.E., *,DIGITAL IMAGE PROCESSING*. SND ED., PRENTICE HALL, UPPER SADDLE RIVER, NJ, 2002.

HAAGA, J. R., ***The effect of mAs variation upon CT image quality as evaluated by in vivo studies***. Radiology, n. 138, p. 449-454, 1991.

HANSON, K. M. et al. **Computed tomography using proton energy loss**. Los Alamos Scientific Laboratory, 1981.

HATCHER D. C. ***Operational Principles for Cone-beam Computed Tomography***. J Am Dent Assoc. 2010.

HORII S. C.; STEVEN Y. K; ***Handbook of Medical Imaging*** Volume 3*. Display and PACS*. Washington; ©2000.

KOZUKI, C. Análise Comparativa Entre Modelagem Computacional e Dados Experimentais para Tomografia Computadorizada com Feixe de Prótons, Dissertação de Mestrado, UTFPR, 2012.

KOZUKI, C.; LIMA, R. C., **Elaboração de um phantom e de um sistema de colimação baseados em modelagem computacional por código de Monte Carlo para o protótipo de um tomógrafo por feixe de prótons**, Trabalho de Conclusão de Curso de Graduação, 2010.

KLOCK, M. C. L., **Desenvolvimento de um detector de energia para tomografia computadorizada com feixe de prótons de alta energia,** Tese de doutorado, UTFPR, 2006.

LOMA LINDA UNIVERSITY web site, http://www.llu.edu, acesso em 20/01/2013

MATLAB web site, http://www.mathworks.com, acesso em 20/01/2013

MEDEIROS, A. et al. **Possibilidades e Limitações das Simulações Computacionais no Ensino da Física**, Revista Brasileira de Ensino de Física, vol. 24 n. 2, São Paulo June 2002.

MILHORETTO, E. et al. **Modelagem de *phantom* por método de Monte Carlo para um protótipo de um tomógrafo por feixe de prótons**. 21º Congresso Brasileiro de Engenharia Biomédica, 2008.

MILHORETTO, E., **Deteminação da Influência de Fatores Físicos no Espectro de Energia de um Protótipo de Tomógrafo por Feixe de Prótons por Simulação de Monte Carlo,** Dissertação de mestrado, UTFPR, 2007.

MILHORETTO, E. et al.; **Desenvolvimento de um Sistema Mecânico de Movimentação para um Tomógrafo Computadorizado com Feixe de partículas Carregadas –** PIBIC-CNPQ, 2004.

NEMA: *National Electrical Manufacturers Association.* ***Digital Imaging and Communications***. 1985. Disponível em: < ftp://medical.nema.org/medical/dicom/1985/ACR-NEMA_300-1985.pdf>. Acesso em: 20/01/2013

NEMA: *National Electrical Manufacturers Association.* ***Digital Imaging and Communications in Medicine (DICOM)***. 2011. Disponível em: <ftp://medical.nema.org/medical/dicom/2011/11_01pu.pdf>. Acesso em: 20/01/2013

NAPT: National Association for Proton Therapy. **Robert R. Wilson: Remembered as "Father of Proton Therapy" and Achievements in Physics and Medicine**. 2000. Disponível em: <http://www.proton-therapy.org/pr10.htm>. Acesso em: 20/01/2013.

PENFOLD, S. L, **Image Reconstruction and Monte Carlo Simulations in the Development of Proton Computed Tomography for Applications in Proton Radiation Therapy**. Tese de Doutorado, Wollongong University, 2010.

PENFOLD, S. L; et al. **Development of Proton Computed Tomography for Applications in Proton Therapy.** 2010. Disponível em: <http://niptrc.org/pdfs/4b%20aip_595_1.pdf>. Acesso em: 20/01/2013.

PRIOR, F.; HORRI, S.; et al. **DICOM: An Introduction to the Standard. University of Pennsylvania**. 2001.

PROTON TREATMENT CENTER – **Loma Linda University Adventist Health Sciences Center**, USA. Disponível em: <http://www.llu.edu/proton/index.html > Ultimo acesso em 10 jan. 2011.

PTCOG SECRETARY, **Hadron Therapy Patient Statistics** (data received per March 2012; PTCOG50) Disponivel em: http://ptcog.web.psi.ch/ptcentres.html. Acesso em 20/01/2013.

SEERAM, E. **Computed tomography: Physical principles, Clinical applications, and Quality control**. 2nd ed, p. 222. Philadelphia, PA: Saunders, 2001.

SETTI, J.; SCHELIN, H., **Tomografia computadorizada por feixe de prótons de baixa energia**, Tese de Doutorado, UTFPR, 2006.

SLATER J. M., ARCHAMBEAU,J. O., MILLER, D. W., NOTARUS, M. I., PRESTON, W., SLATER, J. D., **The proton treatment center at Loma Linda University Medical Center: rationale for and description of its development**, International Jornal of Radiation Oncology, Biology, Physics, Vol. 22, No. 2, p.383, 1992.

SOARES, F. A.; LOPES, H. B**. Tomografia computadorizada**. Curso técnico de radiologia. Florianópolis, 2000.

SCHULTE, R.; BASHKIROV, V.; LI, T., LIANG, Z. MUELLER, K.; HEIMANN, J., JOHNSON, L.R.; KEENEY, B.; SADROZINSKI, H.F.W.; SEIDEN, A.; WILLIAMS, D.C.; ZHANG, L.; LI, Z.; PEGGS, S.; SATOGATA, T.; WOODY, C.; **Conceptual Design of a Proton Computed Tomography System for Applications in Proton Radiation Therapy,** IEEE Trans. Nuclear Science, 2004.

SCHULTE, R. **Strategies for Image-guided Proton Therapy of Cancer. Touch Briefings**, 2007. Disponível em: http://www.touchbriefings.com/pdf/2812/Schulte.pdf Acesso em: 20/01/2013.

SLATER, J. M. et al., **The proton treatment center at Loma Linda University Medical Center: rationale for and description of its development**, International Journal of Radiation Oncology, Biology, Physics, Vol.22, n. 2, p.383, 1992.

SNIR, J. et al. **An Essay on Building a Conceptually Enhanced Computer Simulation for Science Teaching**, Draft Article, Technical Report 88-18, Cambridge, USA, 1988.

YEVSEYEVA,O., **Estudo de Restrições Em Tomografia com Feixe de Partículas Carregadas Através de Modelagem Computacional,** Dissertação de Mestrado, IP/UERJ, Nova Friburgo, 2005.

ANEXO A – CÓDIGOS PARA CONSTRUÇÃO DO *PHANTOM* DE POLIPROPILENO, COLIMADOR E DETECTOR.

Sequência de códigos para construção do detector:

```
  G4double DetectorPosition_x =  0.0   *mm;
  G4double DetectorPosition_y =  0     *mm;
  G4double DetectorPosition_z =  49.0  *mm;
  G4double innerRadiusDetector   = 0.0*mm;
  G4double outerRadiusDetector   = 5.0*mm;
  G4double hightDetector         = 0.00025*mm;
  G4double startAngleDetector    = 0.*deg;
  G4double spanningAngleDetector = 360.*deg;
  solidDetector = new G4Tubs("Detector",
                                        innerRadiusDetector,
                                        outerRadiusDetector,
                                        hightDetector,
                                        startAngleDetector,
                                        spanningAngleDetector);
  logicDetector = new G4LogicalVolume(solidDetector, Silicon,
              "Detector");
  physiDetector = new G4PVPlacement(G4Transform3D(rm,
G4ThreeVector(DetectorPosition_x,DetectorPosition_y, DetectorPosition_z)),
"Detector", logicDetector, physiExperimentalHall, false, 0);
```

Sequência de códigos para determinar a posição do *phantom*:

```
  G4double PhantomPosition_x = 3.2   *mm;
  G4double PhantomPosition_y = 0.0     *mm;
  G4double PhantomPosition_z = 30.0     *mm;
```

Sequência de códigos para construir o *phantom* de polipropileno:

```
  G4double innerRadiusOfTheGlassPhantom = 0.10*mm;
  G4double outerRadiusOfTheGlassPhantom = 0.15*mm;
  G4double hightOfTheGlassPhantom = 0.2*mm;
  G4double startAngleOfTheGlassPhantom = 0.*deg;
  G4double spanningAngleOfTheGlassPhantom = 360.*deg;
  solidGlassPhantom = new G4Tubs("GlassPhantom",
                                        innerRadiusOfTheGlassPhantom,
                                        outerRadiusOfTheGlassPhantom,
                                        hightOfTheGlassPhantom,
                                        startAngleOfTheGlassPhantom,
                                        spanningAngleOfTheGlassPhantom);
 logicGlassPhantom = new G4LogicalVolume(solidGlassPhantom,
                                              Glass, "GlassPhantom", 0, 0, 0);
```

```
 physiGlassPhantom = new G4PVPlacement(G4Transform3D(rmzP,
G4ThreeVector(90, 90, PhantomPosition_z)),  "GlassPhantom",
logicGlassPhantom, physiExperimentalHall, false, 0);
```

Sequência de códigos para construir o *phantom* de água:

```
  G4double innerRadiusOfTheWaterPhantom = 1.6*mm;
  G4double outerRadiusOfTheWaterPhantom = 2.35*mm;
  G4double hightOfTheWaterPhantom = 15.0*mm;
  G4double startAngleOfTheWaterPhantom = 0.*deg;
  G4double spanningAngleOfTheWaterPhantom = 360.*deg;
  solidWaterPhantom = new G4Tubs("WaterPhantom",
                                          innerRadiusOfTheWaterPhantom,
                                          outerRadiusOfTheWaterPhantom,
                                          hightOfTheWaterPhantom,
                                          startAngleOfTheWaterPhantom,
                                          spanningAngleOfTheWaterPhantom);
logicWaterPhantom = new G4LogicalVolume(solidWaterPhantom, Water,
 "WaterPhantom", 0,0,0);
physiWaterPhantom = new G4PVPlacement(G4Transform3D(rmzP,
G4ThreeVector(PhantomPosition_x, 0, PhantomPosition_z)),
                                           "WaterPhantom",
                                           logicWaterPhantom,
                                           physiExperimentalHall,
                                           false, 0);
```

Sequência de códigos para construção do *phantom* externo de polietileno:

```
  G4double innerRadiusOfThePolyethylenePhantom_ext = 2.35*mm;
  G4double outerRadiusOfThePolyethylenePhantom_ext = 3.07*mm;
  G4double hightOfThePolyethylenePhantom_ext = 15.0*mm; 25.*mm;
  G4double startAngleOfThePolyethylenePhantom_ext = 0.*deg;
  G4double spanningAngleOfThePolyethylenePhantom_ext = 360.*deg;
  solidPolyethylenePhantom_ext = new G4Tubs("PolyethylenePhantom_ext",
innerRadiusOfThePolyethylenePhantom_ext,
outerRadiusOfThePolyethylenePhantom_ext,
hightOfThePolyethylenePhantom_ext,
startAngleOfThePolyethylenePhantom_ext,
spanningAngleOfThePolyethylenePhantom_ext);
logicPolyethylenePhantom_ext = new G4LogicalVolume(solidPolyethylenePhantom_ext,
Polyethylene,
"PolyethylenePhantom",0, 0,0);
 physiPolyethylenePhantom_ext = new G4PVPlacement(G4Transform3D(rmzP,
                                      G4ThreeVector(PhantomPosition_x,0,
                                 PhantomPosition_z)),
                                           "PolyethylenePhantom",
                                           logicPolyethylenePhantom_ext,
                                           physiExperimentalHall,
                                           false, 0);
  G4double innerRadiusOfThePolyethylenePhantom = 0.99*mm;
```

```
  G4double outerRadiusOfThePolyethylenePhantom = 1.6*mm;
  G4double hightOfThePolyethylenePhantom = 15.0*mm;
  G4double startAngleOfThePolyethylenePhantom = 0.*deg;
  G4double spanningAngleOfThePolyethylenePhantom = 360.*deg;
  solidPolyethylenePhantom = new G4Tubs("PolyethylenePhantom",
                                        innerRadiusOfThePolyethylenePhantom,
                                        outerRadiusOfThePolyethylenePhantom,
                                        hightOfThePolyethylenePhantom,
                                        startAngleOfThePolyethylenePhantom,
                                    spanningAngleOfThePolyethylenePhantom);
logicPolyethylenePhantom = new G4LogicalVolume(solidPolyethylenePhantom,
                                               Polyethylene,"PolyethylenePhantom",0,0,0);
physiPolyethylenePhantom = new G4PVPlacement(G4Transform3D(rmzP,
G4ThreeVector(PhantomPosition_x,0, PhantomPosition_z)),
                                         "PolyethylenePhantom",
                                         logicPolyethylenePhantom,
                                         physiExperimentalHall, false,0);
  G4double innerRadiusOfTheAluminium = 0.0*mm;
  G4double outerRadiusOfTheAluminium = 15.0*mm;
  G4double hightOfTheAluminium = 2.0*mm;
  G4double startAngleOfTheAluminium = 0.0*deg;
  G4double spanningAngleOfTheAluminium = 180.0*deg;
  solidAluminium = new G4Tubs("Aluminium",
                                        innerRadiusOfTheAluminium,
                                        outerRadiusOfTheAluminium,
                                        hightOfTheAluminium,
                                        startAngleOfTheAluminium,
                                        spanningAngleOfTheAluminium);
logicAluminium = new G4LogicalVolume(solidAluminium,
Aluminium,"Aluminium",0,0,0);
physiAluminium = new G4PVPlacement(G4Transform3D(rmz,
G4ThreeVector(-0.1,0.0,46.0)),                    "Aluminium",logicAluminium,
physiExperimentalHall,
false, 0);
```

Sequência de códigos para construção do colimador de alumínio:

```
G4double innerRadiusOfTheAluminium1 = 0.0*mm;
G4double outerRadiusOfTheAluminium1 = 15.0*mm;
G4double hightOfTheAluminium1 = 2.0*mm;
G4double startAngleOfTheAluminium1 = -180.0*deg;
G4double spanningAngleOfTheAluminium1 = 180.0*deg;
  solidAluminium1 = new G4Tubs("Aluminium",
                                        innerRadiusOfTheAluminium1,
                                        outerRadiusOfTheAluminium1,
                                        hightOfTheAluminium1,
                                        startAngleOfTheAluminium1,
                                        spanningAngleOfTheAluminium1);
logicAluminium1 = new G4LogicalVolume(solidAluminium1,
```

```
Aluminium,"Aluminium",0,0,0); physiAluminium1 = new
G4PVPlacement(G4Transform3D(rmz,  G4ThreeVector(0.1,0.0,46.0)), "Aluminium",
logicAluminium1, physiExperimentalHall, false,0);
G4SDManager* SDman = G4SDManager::GetSDMpointer();
G4String detectorChamberSDname = "PWG/DetectorChamberSD";
PWGDetectorSD* aDetectorSD = new PWGDetectorSD( detectorChamberSDname);
SDman->AddNewDetector( aDetectorSD );
logicDetector->SetSensitiveDetector( aDetectorSD );
}
```

ANEXO B – PARTE DOS CÓDIGOS DO ARQUIVO DICOMHANDLER.CC.

```
unsigned int rowsC = rows/compression;
unsigned int columnsC = columns/compression;
unsigned int planesC = 1;
G4float pixelLocationXM = -pixelSpacingX*rows/2.;
G4float pixelLocationXP = pixelSpacingX*rows/2.;
G4float pixelLocationYM = -pixelSpacingY*rows/2.;
G4float pixelLocationYP = pixelSpacingY*rows/2.;
G4float sliceLocationZM = sliceLocation-sliceThickness/2.;
G4float sliceLocationZP = sliceLocation+sliceThickness/2.;
std::fwrite(&rowsC, sizeof(unsigned int), 1, fileOut);
std::fwrite(&columnsC, sizeof(unsigned int), 1, fileOut);
std::fwrite(&planesC, sizeof(unsigned int), 1, fileOut);
std::fwrite(&pixelLocationXM, sizeof(G4float), 1, fileOut);
std::fwrite(&pixelLocationXP, sizeof(G4float), 1, fileOut);
std::fwrite(&pixelLocationYM, sizeof(G4float), 1, fileOut);
std::fwrite(&pixelLocationYP, sizeof(G4float), 1, fileOut);
std::fwrite(&sliceLocationZM, sizeof(G4float), 1, fileOut);
std::fwrite(&sliceLocationZP, sizeof(G4float), 1, fileOut);
std::printf("%8i   %8i\n",rows,columns);
std::printf("%8f   %8f\n",pixelSpacingX,pixelSpacingY);
std::printf("%8f\n", sliceThickness);
std::printf("%8f\n", sliceLocation);
std::printf("%8i\n", compression);
G4int compSize = compression;
G4int mean;
G4float density;
G4bool overflow = false;
G4int cpt=1;
if(compSize == 1) {
  for( G4int ww = 0; ww < rows; ww++) {
       for( G4int xx = 0; xx < columns; xx++) {
         mean = tab[ww][xx];
         density = Pixel2density(mean);
         unsigned int mateID = GetMaterialIndex( density );
         std::fwrite(&mateID, sizeof(unsigned int), 1, fileOut);      }      }

} else {
  for(G4int ww = 0; ww < rows ;ww += compSize ) {
       for(G4int xx = 0; xx < columns ;xx +=compSize ) {
         overflow = false;
         mean = 0;
         for(int sumx = 0; sumx < compSize; sumx++) {
           for(int sumy = 0; sumy < compSize; sumy++) {
             if(ww+sumy >= rows || xx+sumx >= columns) overflow = true;
             mean += tab[ww+sumy][xx+sumx];          }
           if(overflow) break;   }
         mean /= compSize*compSize;
         cpt = 1;
```

```
            if(!overflow) {
            density = Pixel2density(mean);
            unsigned int mateID = GetMaterialIndex( density );
            std::fwrite(&mateID, sizeof(unsigned int), 1, fileOut);
          }       }      }    }
    if(compSize == 1) { // no compression: each pixel has a density value)
     for( G4int ww = 0; ww < rows; ww++) {
          for( G4int xx = 0; xx < columns; xx++) {
           mean = tab[ww][xx];
           density = Pixel2density(mean);
           std::fwrite(&density, sizeof(G4float), 1, fileOut);
          }      }          } else {
     for(G4int ww = 0; ww < rows ;ww += compSize ) {
          for(G4int xx = 0; xx < columns ;xx +=compSize ) {
           overflow = false;
           mean = 0;
           for(int sumx = 0; sumx < compSize; sumx++) {
            for(int sumy = 0; sumy < compSize; sumy++) {
             if(ww+sumy >= rows || xx+sumx >= columns) overflow = true;
             mean += tab[ww+sumy][xx+sumx];           }
            if(overflow) break;   }
           mean /= compSize*compSize;
           cpt = 1;
            if(!overflow) {
            density = Pixel2density(mean);
            std::fwrite(&density, sizeof(G4float), 1, fileOut);
           }       }          }    }
       std::fclose(fileOut);
       delete [] nameProcessed;
    /*    for ( G4int i = 0; i < rows; i ++ ) {
      delete [] tab[i];    }
    delete [] tab;    */
    return returnvalue;}
/*
  G4int DicomHandler::displayImage(char command[300])  {
  char commandName[500];
  std::sprintf(commandName,"display  %s",command);
  std::printf(commandName);
  G4int i = system(commandName);
  return (G4int )i;  }*/
 G4float DicomHandler::Pixel2density(G4int pixel){
    G4float density = -1.;
    G4int nbrequali = 0;
    G4double deltaCT = 0;
    G4double deltaDensity = 0;
    std::ifstream calibration("CT2Density.dat");
    calibration >> nbrequali;
    G4double * valuedensity = new G4double[nbrequali];
    G4double * valueCT = new G4double[nbrequali];
    if(!calibration) {
```

```
        G4cerr << "@@@ No value to transform pixels in density!" << G4endl;
        exit(1);
    } else { // calibration was successfully opened
        for(G4int i = 0; i < nbrequali; i++) { // Loop to store all the pts in CT2Density.dat
            calibration >> valueCT[i] >> valuedensity[i];      }    }
    calibration.close();
    for(G4int j = 1; j < nbrequali; j++) {
        if( pixel >= valueCT[j-1] && pixel < valueCT[j]) {
            deltaCT = valueCT[j] - valueCT[j-1];
            deltaDensity = valuedensity[j] - valuedensity[j-1];
            density = valuedensity[j] - ((valueCT[j] - pixel)*deltaDensity/deltaCT );
            break;         }    }
    if(density < 0.) {
        std::printf("@@@ Error density = %f && Pixel = %i (0x%x) &&
deltaDensity/deltaCT = %f\n",density,pixel,pixel, deltaDensity/deltaCT);    }
     delete [] valuedensity;
    delete [] valueCT;
    return density;}
void DicomHandler::CheckFileFormat(){
    std::ifstream checkData("Data.dat");
    char * oneLine = new char[128];
    if(!(checkData.is_open())) { //Check existance of Data.dat
        G4cout << "\nDicomG4 needs Data.dat :\n\tFirst line: number of image pixel for a "
            << "voxel (G4Box)\n\tSecond line: number of images (CT slices) to "
            << "read\n\tEach following line contains the name of a Dicom image except "
            << "for the .dcm extension\n";
        exit(0);    }
    checkData >> compression;
    checkData >> nFiles;
    G4String oneName;
    checkData.getline(oneLine,100);
    std::ifstream testExistence;
    G4bool existAlready = true;
    for(G4int rep = 0; rep < nFiles; rep++) {
      checkData.getline(oneLine,100);
      oneName = oneLine;
      oneName += ".g4dcm"; // create dicomFile.g4dcm
      G4cout << nFiles << " test file " << oneName << G4endl;
      testExistence.open(oneName.data());
      if(!(testExistence.is_open())) {
        existAlready = false;
        testExistence.clear();
        testExistence.close();      }
      testExistence.clear();
      testExistence.close();    }
    ReadMaterialIndices( checkData );
    checkData.close();
    delete [] oneLine;
    if( existAlready == false ) { // The files *.g4dcm have to be created
        G4cout << "\nAll the necessary images were not found in processed form, starting "
```

```
            << "with .dcm images\n";
        FILE * dicom;
        FILE * lecturePref;
        char * compressionc = new char[LINEBUFFSIZE];
        char * maxc = new char[LINEBUFFSIZE];
        //char name[300], inputFile[300];
        char * name = new char[FILENAMESIZE];
        char * inputFile = new char[FILENAMESIZE];
        lecturePref = std::fopen("Data.dat","r");
        std::fscanf(lecturePref,"%s",compressionc);
        compression = atoi(compressionc);
        std::fscanf(lecturePref,"%s",maxc);
        nFiles = atoi(maxc);
        G4cout << " nFiles " << nFiles << G4endl;
        for( G4int i = 1; i <= nFiles; i++ ) { // Begin loop on filenames
           std::fscanf(lecturePref,"%s",inputFile);
           std::sprintf(name,"%s.dcm",inputFile);
           std::cout << "check 1: " << name << std::endl;
           std::printf("### Opening %s and reading :\n",name);
           dicom = std::fopen(name,"rb");
           if( dicom != 0 ) {
                ReadFile(dicom,inputFile);
           } else {
                G4cout << "\nError opening file : " << name << G4endl;       }
           std::fclose(dicom); }
        std::fclose(lecturePref);
        delete [] compressionc;
        delete [] maxc;
        delete [] name;
        delete [] inputFile;   } }
template <class Type>
void DicomHandler::GetValue(char * _val, Type & _rval) {
#if BYTE_ORDER == BIG_ENDIAN
   if(littleEndian) {
#else
   if(!littleEndian) {
#endif
        const int SIZE = sizeof(_rval);
        char ctemp;
        for(int i = 0; i < SIZE/2; i++) {
           ctemp = _val[i];
           _val[i] = _val[SIZE - 1 - i];
           _val[SIZE - 1 - i] = ctemp;
        }    }    _rval = *(Type *)_val;
}
```