Reducing the Radiation Dose to Women Receiving Cardiac CT Scans

Michael Hoppe
Marquette University

REDUCING THE RADIATION DOSE TO WOMEN RECEIVING CARDIAC CT SCANS

by
Michael E. Hoppe, B.S.

A Thesis submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin
May 2012

ABSTRACT

REDUCING THE RADIATION DOSE TO WOMEN RECEIVING CARDIAC CT SCANS

Michael E. Hoppe

Marquette University, 2012

This thesis aims to quantify the reduction in radiation dose deposited in glandular breast tissue achieved by using tilted gantry acquisition during cardiac CT scans. Previous work by Halpern et al. suggested using tilted acquisition parallel to the long axis of the patient's heart. However, for a larger portion of the population this is not feasible due to the design of current scanners (which are limited to maximum tilt angles of 30 degrees). This study investigated the potential dose reduction and image quality effects at commercially available tilt angles between 0-30 degrees through simulation and experimental studies.

Upon IRB approval, datasets from 10 female patients from Froedtert Hospital (Milwaukee, WI) were used to create voxelized phantom models for the computer simulation. Experimental measurements were performed with an anthropomorphic phantom on a clinical CT scanner (Discovery CT750HD, GE Healthcare, Chalfont St. Gilates, England). For both simulation and experimental studies, radiation dose to the breast and reconstructed image signal-to-noise ratio (SNR) was quantified for tilt angles between 0-30 degrees in five degree increments.

The simulated and experimental results demonstrate that tilted gantry acquisition reduces the glandular breast dose from cardiac CT scans when compared to conventional (non-tilted) axial scans. Maximum reductions of 33%-81% (mean, 55%) were achieved with a 30-degree gantry tilt. However, a decrease in image quality by approximately 15% when compared to non-tilted images is seen in the simulated results. The image quality is found to remain equivalent, on average, up to a 15 degree tilt. At a 15 degree tilt, a mean breast dose reduction of 12%-64% (mean, 30%) was found.

ACKNOWLEDGEMENTS

Michael E. Hoppe

My adviser, Taly Gilat-Schmidt, Ph.D. (Marquette University, Milwaukee, WI), and committee member, Grant Stevens, Ph.D. (GE Healthcare, Waukesha, WI), for their assistance and guidance.

Dennis Foley, M.D. (Froedtert Hospital, Milwaukee, WI) and Maureen Levenhagen, RT (Froedtert Hospital, Milwaukee, WI) for providing female patient datasets and determination of the patient's long axes.

Lars Olson, Ph.D. (Marquette University, Milwaukee, WI) and David Herzfeld, M.Sc. (Marquette University, Milwaukee, WI) for providing computing support.

Franco Rupcich (Marquette University, Milwaukee, WI) for assisting in Monte Carlo code validation.

Jason Esveld (Marquette University, Milwaukee, WI) for assisting in the phantom segmentation,

Funded in part by the Marion and Alvin Birnshein Foundation.

Computing resources were funded in part by NSF Award OCI-0923037.

TABLE OF CONTENTS

TABLE OF CONTENTS ii

LIST OF TABLES iv

LIST OF FIGURES v

Chapter 1 1

INTRODUCTION 1

Statement of the Problem 1

Specific Aims 2

Chapter 2 4

BACKGROUND 4

X-Rays 4

Chapter 3 25

MATERIALS AND METHODS 25

Anthropomorphic Phantom Experiments 27

Effect on Image Quality 31

Chapter 4 33

RESULTS 33

Simulations with Voxelized Phantoms 33

Anthropomorphic Phantom with the GE Discovery CT750HD 34

Affect of Tilted Acquisition on Image SNR 38

Chapter 5 41

DISCUSSION 41

CONCLUSIONS 44

REFERENCES 45

APPENDIX A 49

The GEANT4 main program: 49

The creation of the world and geometry (physical model): 51

Description of the source and beam: 72

The energy deposition scoring algorithm: 75

The octave script for cumulating the results from every projection angle: 85

The condor submission scripts for distribution of the computation and their called scripts: 85

APPENDIX B 89

The GEANT4 main program: 89

The creation of the world and geometry (physical model): 91

Description of the source and beam: 112

The ray tracing algorithm: 114

The condor submission scripts for distribution of the computation and their called scripts: .. 117

Octave scripts to add noise to ray tracing projections: 120

LIST OF TABLES

Table 1. Tissue weighting factors recommended by the ICRP in Publication 103 10

Table 2. BEIR VII Estimates of the Lifetime Attributable Risk of Incidence and Mortality for Solid Cancers per 100,000 Exposed Persons 12

Table 3. Observed and Expected Solid Cancer Deaths by Dose Group 14

Table 4. Excess Relative Risk Estimates for Selected Dose Ranges Based on Atomic Bomb Survivor Data 14

Table 5. Some common radiologic imaging exams and typical organ doses associated. 16

Table 6. Estimated organ doses and estimated risk for select CT exams. 17

Table 7. The long-axis angles of the ten phantoms and corresponding glandular breast dose reductions at 30 degrees and at patient-specific long axis angle. 32

LIST OF FIGURES

Figure 1. The electromagnetic spectrum. 5

Figure 2. The three predominate interactions of x-rays with matter. 7

Figure 3. An illustration depicting the Beer-Lambert law and beam attenuation. 8

Figure 4. Risk estimation data for individuals exposed to 10 mSv extracted from BEIR VII by Sodickson (2000),. 13

Figure 5. Depicting pencil (left), parallel, cone, and fan (right) beam geometries. 19

Figure 6. A sinogram made of 1000 projections onto a 1,024-pixel detector. 20

Figure 7. A tilted gantry acquisition parallel to the long axis of the heart (left) and a conventional scan (right). Arrows indicate x-ray beam. 22

Figure 8. Voxelized phantom including glandular breast tissue (green), bone (gray), lung (blue), and blood (red). 25

Figure 9. Lungman with RANDO phantom breast models attached and MOSFET dosimeters with leads. 27

Figure 10. The GE CT750HD with its gantry tilted to perform a tilted acquisition. 28

Figure 11. A reconstructed image depicting locations of three 20 x 20 pixel ROIs (squares) selected to calculate the mean value and standard deviation for cardiac tissue in each phantom. 31

Figure 12. The reduction in glandular breast dose for each voxelized phantom with respect to tilt angle. Each marker represents a unique phantom. The dotted line is the mean value across all phantoms. 33

Figure 13. The experimentally measured reduction in dose in the dosimeters placed within the phantom. 34

Figure 14. The dose reductions from a cardiac CT scan in the breast when all dosimeters placed within the breast region of the anthropomorphic phantom are combined (blue, square). For comparison, the average reduction in breast dose calculated across the computer simulations (red, circle) is plotted alongside the anthropomorphic data. The error bars represent +/- one standard deviation. 35

Figure 15. The percent reduction in dose delivered to lung averaged from dosimeters located in directly above and below the heart. 36

Figure 16. Reconstructed images of a voxelized phantom at zero degree tilt (left) and 30 degree tilt (right). 37

Figure 17. SNR ratios at a gantry tilt with respect to zero tilt of gantry. Each marker represents a specific phantom model, corresponding to the markers in Figure 13. The dotted line represents the mean ratio of SNRs across all phantoms at each tilt. 38

Figure 18. The mean SNR ratio per tilt angle (n=5) for the experimental phantom study plotted with the voxelized phantom mean SNR ratio. The error bars represent +/- one standard deviation (not visible for the experimental data, with values less than 0.003). The experimental data exist alongside the simulated data between 0 degrees and 15 degrees. 39

Chapter 1

INTRODUCTION

Statement of the Problem

Computed tomography (CT) is the largest source of exposure to ionizing radiation in medicine, contributing approximately 30% of the radiation dose to the population in the United States. In 2010 an estimated 80 million CT scans were performed, with the use of CT increasing . Due to the increasing role CT serves in patient care, radiation dose mitigation techniques have been received greater attention and implementation. One factor contributing to dose concerns is the exposure of diagnostically irrelevant, radiosensitive tissue to ionizing radiation. Such is the case with coronary CT angiography (cCTA) scans, with 2 million diagnostic cCTA scans performed in 2001 alone. A cCTA scan exposes radiosensitive glandular breast tissue and lung tissue to ionizing radiation when they are not the primary organ or tissue of interest.

Due to the exposure of radiosensitive tissues during a standard cCTA scan, there exists an increase in the risk of cancer incidence for patients. The exposure during cCTA results in an estimated lifetime attributable risk (LAR) for cancer of 1 in 143 for a 20-year old woman, 1 in 284 for 40-year-old woman, and 1 in 466 for a 60-year old woman compared to 1 in 686 for a 20-year-old man, 1 in 1,007 for a 40 year-old man, and 1 in 1,911 for a 60-year-old man. In females, the LAR is approximately 5 times greater at all ages, with the combination of lung and breast cancers contributing 80-85% (~40% each) of this risk.

To mitigate the risk cardiac CT scans, a variety of techniques have been developed, for example, electrographically (ECG) controlled tube current modulation, minimization of the craniocaudal scan length, optimization of the tube current and voltage, and the use of shields. These methods, though effective, may not always be an applicable or a desirable technique. For instance, ECG-gated tube current modulation is best suited for patients with a regular heart rate and rhythm, tube current modulation may have little effect on the dose reduction to glandular breast and lung tissues in larger patients during thoracic CT, shields may negatively affect image quality, and limiting the craniocaudal length is marginally effective when attempting to reduce the dose to breast and lung tissues during cCTA due to their location with respect to the heart. Due to the increased risk in the female population, Halpern et al. proposed tilting the CT gantry and/or patient so that the beam is parallel to the long axis of the heart. This reduces the amount of breast tissue in the irradiated field, however the required tilt angles are beyond the capabilities of conventional scanner geometries and resulting dose reduction was not quantified. The method proposed by Halpern et al also requires an additional low-dose CT acquisition to prospectively determine the long axis.

Specific Aims

This thesis estimated the reduction in dose to glandular breast tissue achieved using tilted gantry acquisition compared to standard axial acquisition. Radiation dose to glandular breast tissue and lung tissue were quantified along with reconstructed image signal-to-noise ratio (SNR) for commercially available tilt angles of 5 to 30 degrees. For comparison, dose and SNR were quantified for tilt angles parallel to the long-axis of the

heart, as proposed by Halpern et al. The energy deposited in the glandular breast tissue and reconstructed image SNR were estimated using Monte Carlo and ray-tracing simulations with voxelized phantoms created from CT datasets of ten female patients and analysis of experimental data collected using one anthropomorphic physical phantom and a clinical scanner (Discovery CT750HD, GE Healthcare, Chalfont St. Gilates, England).

The presented work will demonstrate, after some further research, tilted gantry acquisition may result in meaningful reduction in breast dose. This will be achieved by removing the tissue from the irradiated field during scanning. Across the population, especially for women requiring multiple cardiac scans, tilted acquisition could be an easy way for some clinicians to lower the risk of breast cancer due to these scans.

Chapter 2

BACKGROUND

X-Rays

Two-dimensional medical radiographs, such as x-rays, are familiar to most people due to their frequent use by doctors to inspect bones after trauma, diagnose pneumonia, and to diagnose dental cavities. CT uses a set of digital radiographs acquired around an object to produce a two-dimensional slice image or three-dimensional volume. Therefore, CT allows doctors to view internal patient anatomy non-invasively.

X-rays are a form of electromagnetic radiation similar to visible light and radio waves. However, x-rays have the potential to free an electron from an atom due to their greater energy, thus are a form of ionizing radiation. The electromagnetic spectrum is depicted below in Figure 1.

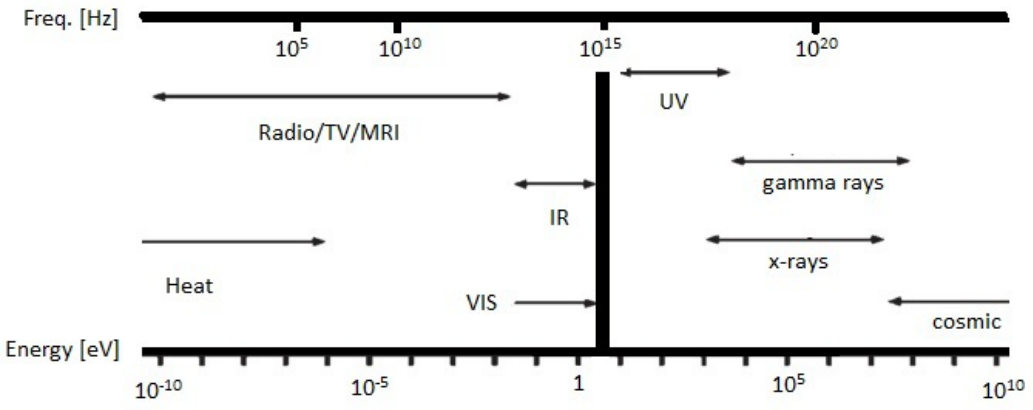

Figure 1. The electromagnetic spectrum.

As x-ray photons travel from their source to a detector, such as radiographic film or a digital detector, they may interact with the matter along their path. The type of interaction, and associated loss of kinetic energy, depends upon the initial kinetic energy of the x-ray photon and properties of the transport material. This interaction may cause photon absorption and the ionization of atoms, and results in the attenuation of the x-ray beam.

X-ray photons within the energy range used for CT, about 20 keV to 150 keV, interact with matter through Rayleigh scattering, the photoelectric effect, and/or Compton scattering.

Rayleigh Scattering

Rayleigh scattering is a “weak” scattering interaction between a low energy x-ray photon and an entire atom. As an x-ray photon’s electromagnetic field nears an atom, it causes the atom’s electrons to absorb some energy and be in a slightly excited state. The

excitation energy is quickly emitted from the atom as an x-ray photon with nearly the same energy, but scattered at a small angle in the forward direction (original direction of the incident photon). This is a low energy process occurring when x-ray photons have energy between 15 to 30 keV and does not cause excitation or ionization of the atom.

Photoelectric Effect

The photoelectric effect is the complete absorption of an x-ray photon by an atom and the ejection of an electron, called a "photoelectron." This process occurs when an incident x-ray photon's energy is equal to, or greater than, an orbital electron's binding energy. It is most likely to cause the ejection of an inner shell (i.e. K shell) electron, which then results in a photoelectron with energy (E_{pe}):

$$E_{pe} = E_{incident} - E_{K\ shell} \qquad \text{Eq. 1}$$

Since an inner orbital electron has been ejected as a photoelectron, higher orbital shell (i.e. L or M shell) electrons release energy to fill the vacancy. This vacancy filling causes the emission of characteristic x-rays or an Auger electron. The Auger electron appears when a higher orbital electron's emitted energy (in the form of a characteristic x-ray) is absorbed by another orbital electron causing it to be ejected from the atom.

The photoelectric effect in medical imaging is the predominate interaction for x-ray photons having energy (E) less than approximately 26 keV and traveling through high atomic number (Z) materials, with approximate probability of occurrence equal to (Z^3/E^3)

.

Compton Scattering

Compton scattering is the predominate interaction in medical imaging above 26 keV . Compton scattering occurs when an x-ray photon collides with an outer orbital electron causing it to be ejected and the incident photon to be scattered. The amount of energy transferred to the ejected electron is dependent on the scattering angle and energy of the incident photon:

$$E_{scatter} = E_{incident}/[1 + (E_{incident}/0.511MeV)\ (1 - \cos(\theta))] \quad \text{Eq. 2}$$

Therefore, to maintain conservation of energy the ejected electron has energy:

$$E_{ejected} = E_{incident} - E_{scatter} \quad \text{Eq. 3}$$

The probability of Compton scattering interaction increases with energy, electron density, and density of the material. However, Compton scattering is independent of atomic number.

The three predominate interactions of x-rays with matter previously discussed are illustrated in Figure 2.

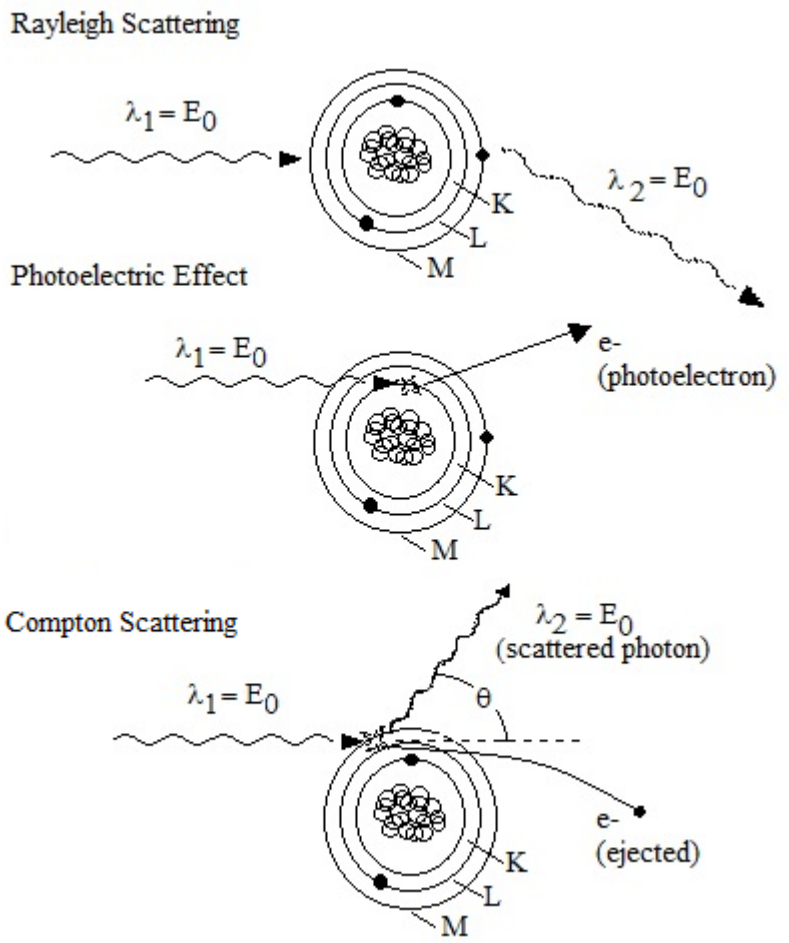

Figure 2. The three predominate interactions of x-rays with matter.

Attenuation

These three types of interaction may cause a photon to be scattered and/or absorbed, resulting in kinetic energy losses. Therefore, the intensity of x-ray photons striking a point on the detector gives an indication of the amount of attenuation due to the material between the source and detector. This phenomenon is described by the Beer-Lambert law, which is a function of initial beam intensity, the attenuation coefficient of the traversed material, beam energy, and thickness of the traversed material. It is important

to note the attenuation coefficient is also a function of incident x-ray photon energy. Where the intensity at the detector, I, of an x-ray at a specific energy exiting an object consisting of one material is:

$$I(E) = I_0(E)e^{-\mu(E)x} \qquad \text{Eq. 4}$$

where, $I_{0,}$ is the original intensity, $\mu(E)$ is the linear attenuation coefficient of the material at a specified energy, and x is the distance travelled in the material. This is illustrated in Figure 3. For polyenergetic sources, such as those used in diagnostic imaging, this discretely becomes a summation of exponentials per energy or an integral over energy if a continuous intensity and attenuation coefficients are known across a given spectrum.

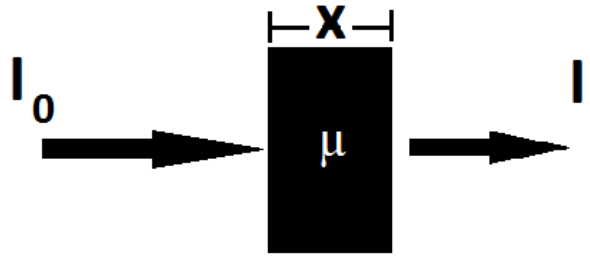

Figure 3. An illustration depicting the Beer-Lambert law and beam attenuation.

The linear attenuation coefficient of a material is representative of the number of atoms per volume and the probability of scattering or absorbing an x-ray photon.

Radiation Dose

As previously discussed, ionizing radiation may interact with individual atoms in the medium through which it travels and deposit enough energy to free an electron causing ionization of the atom. The amount of ionizing radiation an object has received is measured by the amount of ionizing energy deposited in the object per mass of the object. The widely accepted measure in health physics is represented by the unit the Gray (Gy) [J/kg] and is typically expressed per whole organ tissue. For example, a pancreas with mass of 100 grams irradiated by x-rays which deposited 1x10-4 J receives an organ dose of 1 mGy.

The National Research Council's (NRC) Committee on the Biological Effects of Ionizing Radiation (BEIR) report concerning exposure to low levels of ionizing radiation, BEIR VII Phase 2, defines low doses from low linear energy transfer (LET) radiation as dose under 100 mGy deposited by gamma rays or x-rays . The organ doses deposited during a CT scan typically fall into this low-dose, low-LET regime with typical reported organ doses between 10-100 mGy .

Effective Dose

While the amount of energy deposited in a tissue due to ionizing radiation can be quantified in the same manner for every tissue by calculating the dose in Grays, not every tissue's response is similar. Some tissues are more sensitive to radiation, and thus are at greater risk of cancer or developing heritable disease.

Table 1. Tissue weighting factors recommended by the ICRP in Publication 103

Tissue	Tissue weighting factor	Sum of weights
Bone marrow (red), colon, lung, stomach, breast, remainder tissues*	0.12	0.72
Gonads	0.08	0.08
Bladder, esophagus, liver, thyroid	0.04	0.16
Bone surfaces, brain, salivary glands, skin	0.01	0.04
Total		1.00

*Remainder tissues: Adrenals, extrathoracic region, gall bladder, heart, kidneys, lymphatic nodes, muscles, oral mucosa, pancreas, prostate (m), small intestine, spleen, thymus, uterus/cervix (f).

To account for the differing probability of interaction for various types of radiation and the increased sensitivity to ionizing radiation of certain tissues, the metrics equivalent dose and effective dose are used in radiation protection, respectively.

The equivalent dose is calculated by weighting the dose deposited in a tissue by a quality factor, which is dependent on the type of radiation. Since x-rays are photons, they have a quality factor is equal to one. However, for radiation with higher probabilities of interaction, such as beams of protons or alpha particles, the quality factor is greater. For

example, dose deposited in the skin would be multiplied by a factor of one for x-rays and a factor of two for a proton beam to determine the equivalent dose.

$$\text{Equivalent Dose} = \text{Dose [Gy]} * \text{Radiation Quality Factor} \quad \text{Eq. 5}$$

After the dose is weighted for the type of radiation, the sensitivity of the tissue being irradiated is taken into account through a tissue weighting factor, listed in Table 1 above. All equivalent doses, in Grays, are multiplied by respective tissue weighting factors, and summed to calculate the effective dose in Sieverts (Sv). The total effective dose metric is specifically designed for quantifying dose in whole body irradiation scenarios, but is applied to quantify dose due to radiologic imaging in some literature.

$$\text{Effective Tissue Dose} = \text{Equivalent Dose} * \text{Tissue Weighting Factor} \quad \text{Eq. 6}$$

$$\text{Total Effective Dose} = \sum \text{Effective Tissue Doses} \quad \text{Eq. 7}$$

Health Effects and Radiation Dose

As previously discussed, ionizing radiation traveling through tissue has the potential to cause biological effects'. The radiation may affect a biological cell via what are called direct effects and indirect effects. Direct effects are damages to DNA molecules, such as strand breaks, or to other critical cellular components vital to the cell's survival. Indirect effects are interactions causing reactive chemical species in a cell, which then may cause damage to the cell's DNA or other critical cell components. For example, irradiated

water molecules in the human body may produce free electrons, hydrogen ions, hydroxide anions, and peroxide. These may subsequently cause damage to the cell.

If the DNA of a cell is damaged, and not repaired correctly, it may lead to the development of cancer, abnormal cell function, or cell death. The development of cancer attributed to the exposure to ionizing radiation has been studied as a result of the atomic bombs detonated in Japan during World War II and, more recently, as a result of the increased use of radiologic imaging in medicine. In BEIR VII Phase 2 report, the NRC committee provides cancer estimates per exposure to 0. 1 Gy (Table 2) and data estimating the excess cases for individuals receiving 10 mSv, which was made into a plot by Sodickson (2009) below in Figure.4.

Table 2. BEIR VII Estimates of the Lifetime Attributable Risk of Incidence and Mortality for Solid Cancers per 100,000 Exposed Persons

	Males	Females
Excess cases (incl. non-fatal) from exposure to 0.1 Gy	800 (400, 1600)	1300 (690, 2500)
Number of cases in the absence of exposure	45,500	36,900
Excess deaths from exposure to 0.1 Gy	410 (200, 830)	610 (300, 1200)
Number of deaths in the absence of exposure	22,100	17,500

***95% subjective confidence intervals**

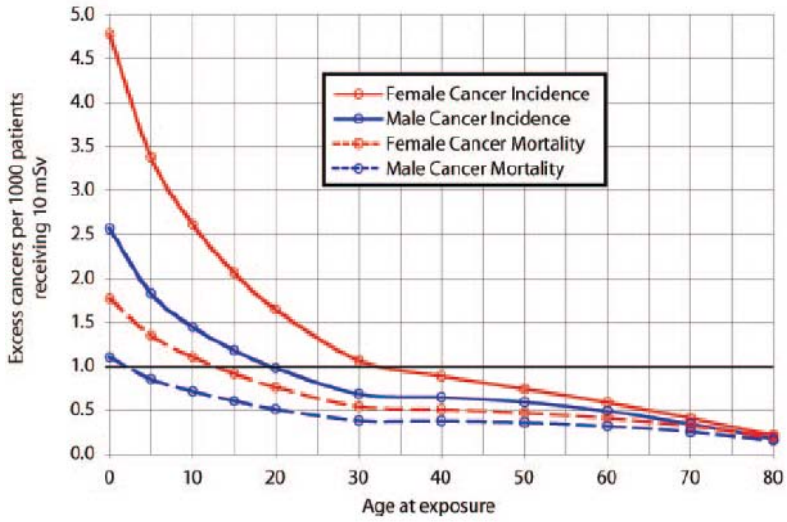

Figure 4. Risk estimation data for individuals exposed to 10 mSv extracted from BEIR VII by Sodickson (2000)[7].

Atomic Bomb Data

The risk of developing cancer due to whole-body ionizing irradiation is periodically updated in the Life Span Study (LSS) cohort of atomic bomb survivors by the Radiation Effects Research Foundation (RERF). The portion of the cohort used for radiation dose effects consists of 86,572 people with estimated dose, age, sex, and distance from the bombs' hypocenters with approximate locations known . Dose estimates in Table 3 show that approximately 80% of this portion of the cohort received less than or equal to 100 mSv. Due to the exposure to mostly gamma radiation, large sample size, and extensive amount of data available, this report is primarily used when estimating the health effects due to low-doses of gamma radiation[7].

Table 3. Observed and Expected Solid Cancer Deaths by Dose Group

		1950–1997			1991–1997		
Dose	People	Deaths	Expected background	Fitted excess	Deaths	Expected background	Fitted excess
<0.005	37,458	3,833	3,844	0	742	718	0
0.005–0.1	31,650	3,277	3,221	44	581	596	12
0.1–0.2	5,732	668	622	39	137	109	10
0.2–0.5	6,332	763	678	97	133	118	24
0.5–1	3,299	438	335	109	75	62	28
1–2	1,613	274	157	103	68	31	27
2+	488	82	38	48	20	8	13
Total	86,572	9,335	8,895	440	1,756	1,642	114

Excess relative risk (ERR) is the rate of cancer death due to a dose divided by the natural rate of cancer death minus 1, accounting for the natural background rate. The background rate is assumed from a Japanese population at enough distance from the detonations to not be affected. In Table 4, a statistically significant increase in the ERR/Sv is observed. These calculations of ERR averaged the risk over sex, age at exposure, and attained age of individuals. This makes it difficult to determine a specific individual's risk due to exposure from a determined dose due to the following factors: increased susceptibility to radiation for females, increased susceptibility with decreasing age, and error in the individual dose estimates due to the unknown shielding from exposure.

Table 4. Excess Relative Risk Estimates for Selected Dose Ranges Based on Atomic Bomb Survivor Data

Dose (Sv)	ERR/Sv (SE)[i]	*P* value[ii]
0-0.05	0.93 (0.85)	0.15
0-0.1	0.64 (0.55)	0.3
0-0.125	0.74 (0.38)	0.025
0-0.15	0.56 (0.32)	0.045
0-0.2	0.76 (0.29)	0.003
0-0.5	0.44 (0.12)	<0.001
0-1	0.47 (0.10)	<0.001
0-2	0.54 (0.07)	<0.001
0-4	0.47 (0.05)	<0.001

i. Sex-averaged estimates at age 70 after exposure at age 30.

ii. One-sided P value for a test of the hypothesis that the slope is 0.

Though the dose range 0-0.125 Sv is shown to be the lowest range in which a statistically significant ERR/Sv is observed, it is noted by the authors that the lower doses in the range do not have a significant impact and when doses from 0-0.02 Sv are excluded in the determination of significance there is no change in the P value for the 0-0.125 range making it difficult to ascertain the true effects at the lower end of the range. This is because effects of radiation at these low levels are likely difficult to distinguish from the natural incidence. The importance of the presented data is not for determining the dose levels at which adverse effects become statistically apparent, but to show the existence of a statistically significant linear slope, or response, with respect to dose (the ERR/Sv).

Though the LSS study shows a relationship between cancer and exposure to ionizing radiation, several factors limit the application of the data for describing the excess relative risk due to radiologic imaging. The living conditions of the Japanese people during exposure were hostile and may have caused adverse health effects that confound the effects of limited radiation exposure. Furthermore, the doses from which statistically significant data can be derived are higher than those typically found in radiologic

imaging and, thus, a linear extrapolation is required. And, the study averages over age at exposure with 41% of the cohort exposed before age 20 which may make it more conservative when assigning risk to an older individual. BEIR VII, however, concludes that no-threshold, linear extrapolation of the LSS cohort data to estimate risk in the regime of radiological imaging is consistent with current evidence. The report concludes that the implementation of a linear-quadratic model for the low-dose regime produced a small change in risk compared to the uncertainty in the risk estimates so did not provide a significantly better model. Also in agreement with the linear, no-threshold model, the ICRP recognizes a 5% increase in fatal cancer risk per Sv for radiological protection purposes.

Diagnostic Cancer Risk

Additional support for the LSS study and the use of a linear, no-threshold model is found in several studies of which the dose due to radiologic imaging to organs can be estimated fairly accurately. The first is an analysis of 5,573 female patients with scoliosis across the United States who received an average dose of 10.8 cGy to the breast before age 20. The result was a standardized mortality ratio of observed breast cancer deaths to expected breast cancer deaths of 1.69 (95% CI, 1.2-14.1), and an excess relative risk per Gy of 5.4 (95% CI, -0.2-9.3). Also supporting the LSS findings is a comparison study by Howe and McLaughlin which compared breast cancer mortality rates between a cohort of 31,710 women who received x-ray fluoroscopy exams along with their tuberculosis treatment and a cohort of women from the atomic bomb LSS study. The results were found to be consistent with one another and indicate a linear relationship between dose and risk ($P < 0.0001$) .

With the current acceptance of the linear, no-threshold response for excess relative risk and exposure to ionizing radiation, studies have been performed to estimate the risk to persons after receiving radiological imaging exams. Typical doses for radiologic imaging exams have been tabulated by Brenner (2007), below in Table 5.

Table 5. Some common radiologic imaging exams and typical organ doses associated.

Study	Relevant Organ	Organ Dose (mGy or mSv)
Dental radiography	Brain	0.005
Posterior-anterior chest radiography	Lung	0.01
Lateral chest radiography	Lung	0.15
Screening mammography	Breast	3
Adult abdominal CT	Stomach	10
Barium enema	Colon	15
Neonatal abdominal CT	Stomach	20

Note: mGy and mSv are equivalent for x-ray radiation.

It is readily apparent the highest doses are attributed to CT exams. Due to this, researchers have focused on the impact the increasing number of CT exams performed each year has on the whole population. Some estimated organ doses and estimate risks are tabulated in Table 6 below.

Though statistically significant evidence is limited due to a limited number of epidemiological studies, there is a growing amount of evidence beginning to support the linear, no-threshold model seen in the atomic bomb cohort as accurate enough to estimate risk due to low doses from radiological imaging exams.

To develop techniques to lower the possible risk of using ionizing radiation involves the understanding of the radiation physics and image formation to produce equivalent images with lower radiation dose.

Table 6. Estimated organ doses and estimated risk for select CT exams.

Study	Organ(s)	Dose (mSv)	Estimated Risk
Multi-slice cardiac CT	Lung	10	Lung cancer mortality 1:8000
Cumulative CT Exams	Whole body	Mean: 54.3	LAR cancer incidence (whole pop.): 0.3% LAR cancer mortality (whole pop.): 0.2%
Coronary CT angiogram	Whole body	Mean: 22	Cancer incidence (female pop.) 1:270 Cancer incidence (male pop.) 1:600
Multi-slice coronary CT angiogram	Lung Breast	42-91 50-80	LAR cancer incidence (20 y.o female) 1:143 LAR cancer incidence (20 y.o. male) 1:686

CT Physics and Image Formation

CT imaging is associated with a non-negligible risk to the patient for developing cancer due to the amount of radiation dose deposited by the x-rays used in CT. X-rays, emitted from the tube, pass through the patient, ultimately being recorded on the detector, which are located on a ring encircling the object of interest, or patient. The source-detector pair rotates 360-degrees around the patient during a standard scan to allow the reconstruction of internal anatomy.

Source and Detector

The source x-ray tube contains an anode (typically made of tungsten) towards which electrons are accelerated to create x-ray photons via Bremsstrahlung interaction. A Bremsstrahlung interaction occurs when an energetic electron passes close to an atomic nucleus. The emission of x-ray photons is caused by the loss of energy due to the

attraction of the electron to the nucleus which alters the electron's trajectory. The intensity and energy of the created x-ray photons can be controlled by the anode tube current, expressed in mA*s, and the tube potential, expressed in kVp. Typical values for a cardiac CT are around 120 kVp and 170 mA*s. Since the emitted x-ray photons travel in somewhat random directions with various energies, beam collimation and filtration are performed. Collimation is the blocking of the peripheral beam to maintain a tight, focused area of irradiation on the patient. Filtration discards the lower energy photons from the beam which would likely not make it to the detector, but attribute to the patient's dose. Since they serve no benefit to image quality yet still deposit dose, it is common practice to filter the beam in accordance with the practice of ALARA (As Low As Reasonably Achievable) when using ionizing radiation.

Typical beam geometries include: pencil beam, parallel beam, cone beam, and fan beam seen below in Figure 5.

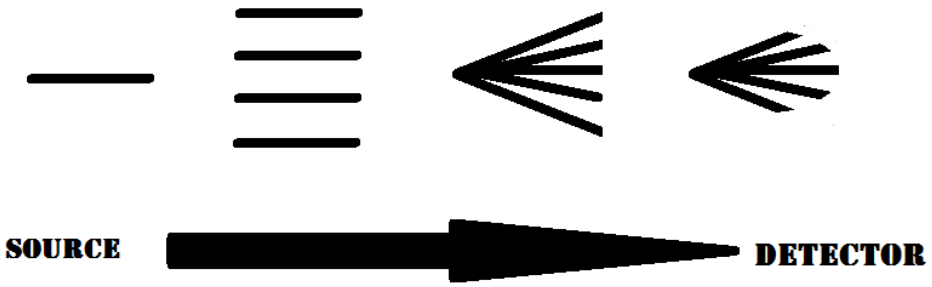

Figure 5. Depicting pencil (left), parallel, cone, and fan (right) beam geometries.

Image Reconstruction

After approximately 1000 projections are acquired around the patient, the image reconstruction process can begin. The projections are typically viewed in what is termed the sinogram, seen below in Figure 6.

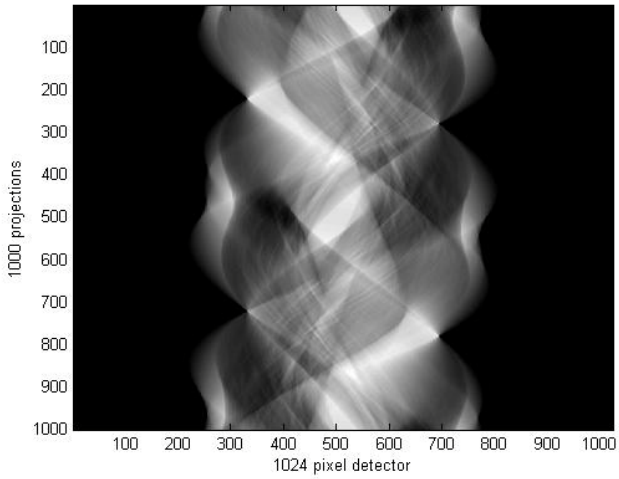

Figure 6. A sinogram made of 1000 projections onto a 1,024-pixel detector.

There are two main reconstruction techniques applied in CT today: filtered backprojection and iterative reconstruction techniques. Filtered backprojection performs image reconstruction based on the central slice theorem by transforming the projection data into the frequency domain, high-pass filtering it (i.e. a ramp filter), inverse transforming to the spatial domain, and backprojecting the filtered projection to the image grid. The high-pass filter is designed to undo the 1/r blurring incurred during backprojection. Iterative techniques sometimes rely on complex statistical models of the noise in an imaging system and the comparison of subsequent iterations to converge on the solution. A main contributor to noise in CT reconstruction is quantum noise due to photon-counting statistics. This, however, is understood to be Poisson distributed and

can be effectively accounted for in statistical reconstruction algorithms. The Poisson distribution is one in which the mean and the variance are equal.

After reconstruction the pixel values are calculated in terms of Hounsfield units (HU):

$$\text{Pixel Value}(x,y) = 1000 \left[(\mu(x,y) - \mu_{water}) / \mu_{water} \right] \quad . \qquad \text{Eq. 8}$$

Note, -1000 HU corresponds to air, and 0 HU to water.

The process is well understood, and thus can be accurately modeled with computers.

Monte Carlo Simulations of Radiation Transport

The Monte Carlo approach to simulate radiation transport is applied across many disciplines to determine the average behavior of particles by simulating their statistical processes of interaction with matter. This often requires many simulated particles or trials to converge on the accurate behavior of the physical phenomena being modeled.

GEANT4 is a Monte Carlo toolkit for simulating radiation transport and has been used to estimate doses due to CT. GEANT4 version 9.3.p01 is used for simulation in this study with the Livermore low energy physics models in place for photons (photoelectric effect, Compton scattering, Rayleigh scattering, and gamma conversion) and electrons (electron ionization and Bremsstrahlung). The Livermore low energy models accurately simulate particles from 250 eV to 100 GeV.

CT Dose Reduction Techniques

Current techniques employed to minimize the dose delivered to patients include: x-ray beam filtration and collimation, tube current modulation, size- or weight-based techniques, in-plane shielding, better detector technology, and noise reduction algorithms. Beam filtration removes low-energy photons that are unlikely to reach the

detector and varies the flux across the beam to normalize the detected flux for the expected elliptical patient shape. Collimation increases dose efficiency (achieving acceptable image quality of the region of interest with the least amount of dose deposited) by limiting the beam width of scan length to irradiate only the anatomy of interest. Tube current modulation involves changing the tube current depending on attenuation, which may vary by projection angle, cranio-caudal position, heart cycle, and patient size. The tube current can be changed during the CT scan to decrease with lower attenuating regions, to decrease during patient/heart movement by having it ECG-controlled, or to decrease when the source is irradiating more radiosensitive tissues such as lung tissue. Tube current modulation has been estimated to reduce the dose to the lungs and breasts by up to 64% during chest CT Patient size techniques may be used to adjust the mA*s, kVp, and scan time to allow for the reconstruction of images with acceptable levels of noise at the lowest dose. For example, some general guidelines are to reduce the mA*s by a factor of 4 or 5 from adult protocols when scanning infants or to increase the mA*s by a factor of 2 for obese patients. In-plane shielding attempts to minimize the dose deposited in radiosensitive organs by being a physical barrier. For example, one technique is to place Bismuth shielding over the breasts during cardiac CT. This method, however, increases image noise and thus has not been recommended because the same level of dose reduction and noise can be achieved by a reduction in tube current without the increased beam-hardening artifacts introduced by the highly attenuating shields. Better detector technology and noise reduction algorithms decrease dose by allowing the reconstruction of images with similar, or better, quality and resolution with fewer photons. Again, fewer photons traveling through the patient equates to less deposited dose.

Proposed Dose Reduction Technique by Halpern

Since all of the previously described reduction techniques leave radiosensitive breast tissue in the field of view during cardiac CT, Halpern et al. proposed the tilted-gantry acquisition protocol, shown Figure 7.

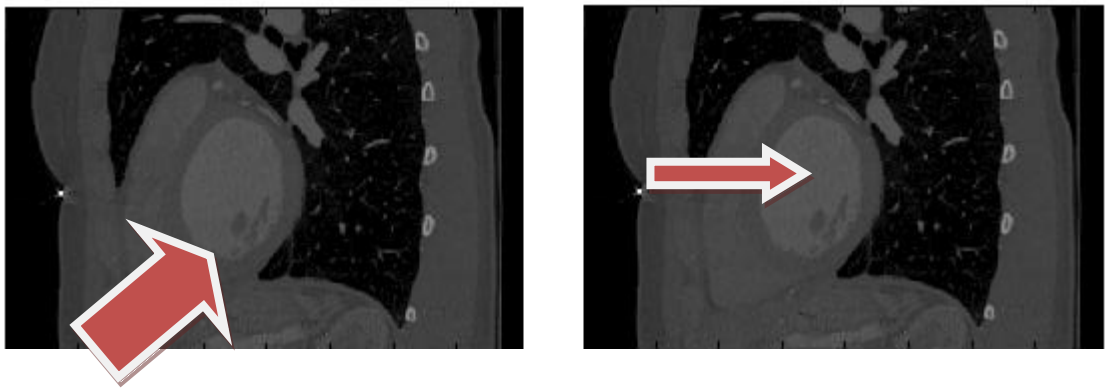

Figure 7. A tilted gantry acquisition parallel to the long axis of the heart (left) and a conventional scan (right). Arrows indicate x-ray beam.

This endeavor to minimize the dose to the breast by removal from the irradiated field is worthwhile as evidence has been shown that about half of 80-85% of the risk associated with a cardiac CT scan for women stems from irradiation of the lung and breast tissue due to their radiosensitivity. This acquisition protocol provides direct acquisition of short-axis cardiac images, and allows the reconstruction of long-axis cardiac images after reformatting. In a retrospective analysis of reconstructed images, Halpern et al estimated at least a 50% reduction in breast volume irradiation, however dose to the breast was not quantified. The study also found required tilt angles ranging from 7-54 degrees to image along the long axis of the left ventricle, with the majority between 20-40 degrees. While Halpern et al. also estimated that 92% of patients could benefit from this protocol with scanners able to tilt 40 degrees, but current scanners are limited both by mechanical and physical limitation to a maximum tilt of 30 degrees. Halpern proposes a low-dose planning CT scan to determine the patient-specific angle, which will incur a dose penalty. Thus the dose reduction obtained by tilted acquisition must outweigh the added dose of the planning CT scan.

Chapter 3

While previous work demonstrated preliminary feasibility of breast dose reduction by tilted-gantry acquisition, the proposed method of acquiring data along the long-axis of the heart is not clinically feasible for many patients. This thesis investigates the reduction of dose to the breast using commercially available gantry-tilt angles. Because the tilt angles are not patient specific, a preplanning CT scan is not required as in the method proposed by Halpern. In the current study, breast dose and reconstructed image SNR were quantified for a range of gantry tilt angles through computer simulations and anthropomorphic phantom experiments.

MATERIALS AND METHODS

Voxelized Phantoms for Simulation Studies

With IRB approval, ten axial cardiac CT datasets of adult, female patients were obtained from Froedtert Hospital. These were used for voxelized phantom creation, illustrated in Figure 8 below. The datasets contained DICOM images with slice thicknesses between 0.625 millimeters to 2.5 millimeters, and axial resolutions between 0.547 millimeters to 0.703 millimeters. These were semi-automatically segmented into seven materials (adipose, air, blood, bone, glandular breast, lung, and muscle), based on Hounsfield number and location, to create voxelized phantoms. Glandular breast tissue was manually segmented in all datasets. Material compositions were taken from the International Commission on Radiological Protection Publication 110.

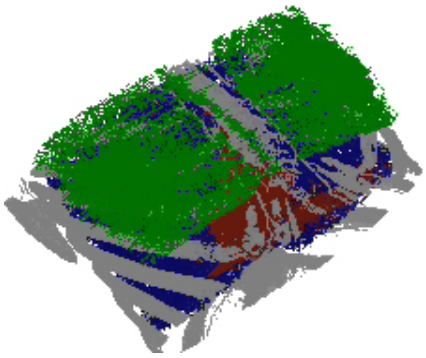

Figure 8. Voxelized phantom including glandular breast tissue (green), bone (gray), lung (blue), and blood (red).

Simulated CT System Specifications

The simulations modeled a CT system with a source-to-iscocenter distance of 54 centimeters, an iscocenter-to-detector distance of 41 centimeters, and a multi-row detector with dimensions 16 centimeters (slice direction) by 80 centimeters (in-plane) and pixel dimensions of 0.078125 mm by 0.078125 mm. A 120 kVp point source with 6.0 millimeters of Aluminum-equivalent filtration was modeled using the SPEC78 software. Simulations of axial trajectory acquisitions were performed at tilt angles of 0-30 degrees in five-degree increments. For comparison, simulations were also performed at a patient-specific tilt angle parallel to the long-axis of the heart. For each dataset, the long-axis of the left atrium-ventricle, which is approximately the longest line that can be drawn through the left atrium-mitral valve-left ventricle complex in the saggital plane, was identified by a trained radiology technologist at Froedtert Hospital.

Radiation Transport Simulation

Monte Carlo simulations to quantify radiation dose were performed with the voxelized phantoms and GEANT4. The point source transmitted 10E5 x-ray photons onto the

detector surface for each view. Each photon and its secondaries were tracked until their total energies were absorbed or they exited the simulation's boundaries. As the photon's travelled through the phantom during a simulation, energy deposited in eV per material/tissue was tallied. A total of 360 views (1 degree/view) were simulated. The number of x-ray photons per view is not representative of realistic output; however, it was chosen to provide a low variance after consecutive runs of the same simulation. This is admissible as the purpose of this study is to calculate a percent reduction in dose and not to estimate absolute dose values.

Anthropomorphic Phantom Experiments

Experiments were performed on a clinical CT system (Discovery CT750HD , GE Healthcare, Chalfont St Giles, England) using a standard chest CT protocol (120 kVp, 250 mAs, 1.25 mm slice thickness, step-and-shoot axial scans). Acquisitions were performed with the anthropomorphic Lungman (Kyoto Kagaku Company, Kyoto, Japan) phantom which contains a well-defined "cardiac tissue" volume and intricate representation of the lungs. To simulate the female population, the breast models were taken from the female Rando Phantom (Alderson Research Laboratories, Stanford, CA) and attached with masking tape to the Lungman phantom (Figure 9). To quantify dose to the breast and lung, MOSFET dosimeters (mobileMOSFET Dosimetry System, Best Medical, Ottawa, Canada) were placed in three "breast regions" and two "lung regions." The three "breast regions" (superior, medial, and inferior) can be seen in the photograph in Figure 9, below, while the two "lung regions" inside the phantom were superior-

posterior and inferior-ventral to the “heart region.” The phantom was placed in the scanner bore as shown in Figure 10.

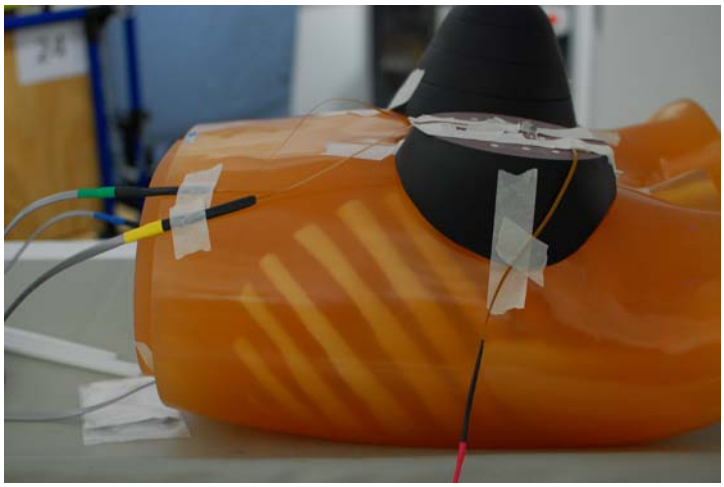

Figure 9. Lungman with RANDO phantom breast models attached and MOSFET dosimeters with leads.

Figure 10. The GE CT750HD with its gantry tilted to perform a tilted acquisition.

Percent Reduction in Dose

For the simulation studies, the total energy deposited in the glandular breast tissue was tallied for each phantom. At 0 degrees and 30 degree tilt, the Monte Carlo simulations were repeated three times to characterize the uncertainty in the deposited energy. At 0 and 30 degrees, the mean, μ, standard deviation, σ, and percent coefficient of variation, $\%COV=100\%*\sigma/\mu$, were calculated for the glandular breast doses across the three trials per tilt angle Only simulations at a 0 degree tilt and 30 degree tilt were run three times due to the assumption that a 0 degree tilt and a 30 degree tilt represent the extrema for dose reduction. One simulation was performed for each voxelized phantom at tilt angles between 0 degrees and 30 degrees.

For the experimental phantom study, the dose deposited in each MOSFET was averaged across five trials for each tilt angle. Further analysis was performed by adding the dose measurements of the three MOSFETS in the breast region to estimate the dose across the breast region and adding the two MOSFETS in the lung region to estimate lung dose.

For all simulations (ten voxelized phantoms, six tilt angles) and all experimental measurements (one phantom, six tilt angles),the percent reduction in the mean deposited energy in the breast, expressed in Eq. 9, was calculated with respect to the energy deposited at 0 degree tilt. The voxelized phantoms, derived from patient CT datasets, did not contain the complete lung organ, thereby preventing quantification of lung dose through simulations. Lung dose was evaluated in the experimental phantom by

calculating the percent change in dose to the lung relative to the zero-degree tilt using Eq. 9.

$$\%Reduction = \left(1 - \frac{Energy_{tilt}}{Energy_{zero\ tilt}}\right) * 100\% \quad \text{Eq. 9}$$

Ray Tracing Simulation

To determine if the investigated titled-gantry acquisitions affect image noise, the same geometries as in the radiation transport simulations were modeled in ray tracing simulations that included quantum noise. The purpose of the study was to quantify relative differences in SNR with varying tilt angle, rather than quantifying absolute image noise. The polyenergetic spectrum and Poisson distribution of quantum noise were modeled. In the ray tracing simulations, non-interacting particles traveled from the point source to the detector. The distance travelled through each material was independently recorded per ray. A polyenergetic sinogram was then constructed assuming Poisson noise and 739,219 photons incident per detector pixel (i.e., a source fluence 75 cm from the source of 4.76E6 photons/mAs*mm^2 as determined by SPEC78) for each of 1,000 views (0.36 degrees/view). The number of detected photons for each ray in the sinogram was normalized by the incident number of photons, followed by the logarithm operation (Eq. 11).

$$-ln\left(\frac{N_{f,noisy}}{N_0}\right) = \mu x \quad \text{Eq. 11}$$

Reconstructing Images with Poisson Noise

Images perpendicular to the axis of rotation were reconstructed from the log-normalized sinograms using a conventional filtered backprojection algorithm. Thus, the angle of the reconstructed images relative to the axial plane was equal to the tilt angle of the gantry. Images were reconstructed onto voxels of size 0.07 cm by 0.07 cm.

Effect on Image Quality

To determine if the proposed acquisition protocols affect SNR, three 20 by 20 pixel regions-of-interest (ROIs) (Figure 11) in the cardiac muscle were extracted in all images reconstructed from the simulated and experimental data. For both the simulated and experimental images, ROIs were extracted in images reconstructed on the central plane of the tilted, reconstructed volume. The mean reconstructed value, standard deviation, and signal-to-noise ratio (SNR), expressed in Eq. 12, was calculated for each ROI . The SNR was averaged over the three regions per tilt angle per phantom for the simulated voxelized phantom studies and five trials with three ROIs per tilt angle for the experimental phantom study.

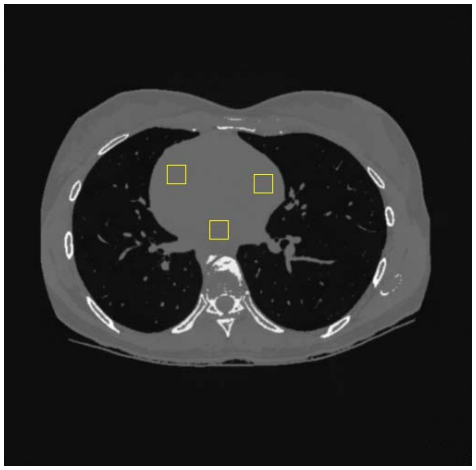

Figure 11. A reconstructed image depicting locations of three 20 x 20 pixel ROIs (squares) selected to calculate the mean value and standard deviation for cardiac tissue in each phantom.

$$SNR = \frac{\sigma}{\mu} \qquad \text{Eq. 12}$$

Chapter 4

RESULTS

Simulations with Voxelized Phantoms

Figure 12 plots the reduction in dose at tilt angles of 0-30 degrees in five degree increments for all simulated phantoms. The mean reduction in dose across all phantoms is plotted as a dotted line. Including all phantoms, the maximum %COV of the glandular breast doses was 0.07% across the trials for a particular tilt angle. For comparison to the tilt method previously proposed by Halpern, Table 7 lists the patient-specific, long-axis tilt angle for each phantom, along with the percent dose reduction at the long-axis tilt angle and at 30 degrees with respect to a zero degree tilt. The long-axis of the heart varied between 10-50 degrees off the axial plane for the set of voxelized phantoms used in the simulations, with a mean angle of 30.5 degrees.

Table 7. The long-axis angles of the ten phantoms and corresponding glandular breast dose reductions at 30 degrees and at patient-specific long axis angle.

Phantom (Long Axis Angle)	Reduction at 30 deg. (%)	Reduction at Long Axis (%)
1 (20°)	**43.8**	**36.7**
2 (10°)	**56.6**	**21.3**
3 (35°)	**55.4**	**61.0**
4 (40°)	**44.2**	**65.7**
5 (20°)	**50.2**	**30.1**
6 (40°)	**81.3**	**84.3**
7 (50°)	**59.4**	**84.9**
8 (40°)	**52.2**	**71.2**
9 (30°)	**32.9**	**32.9**
10 (20°)	**75.6**	**67.5**
Mean	**55.2**	**55.6**
Std. Dev.	**14.5**	**23.3**

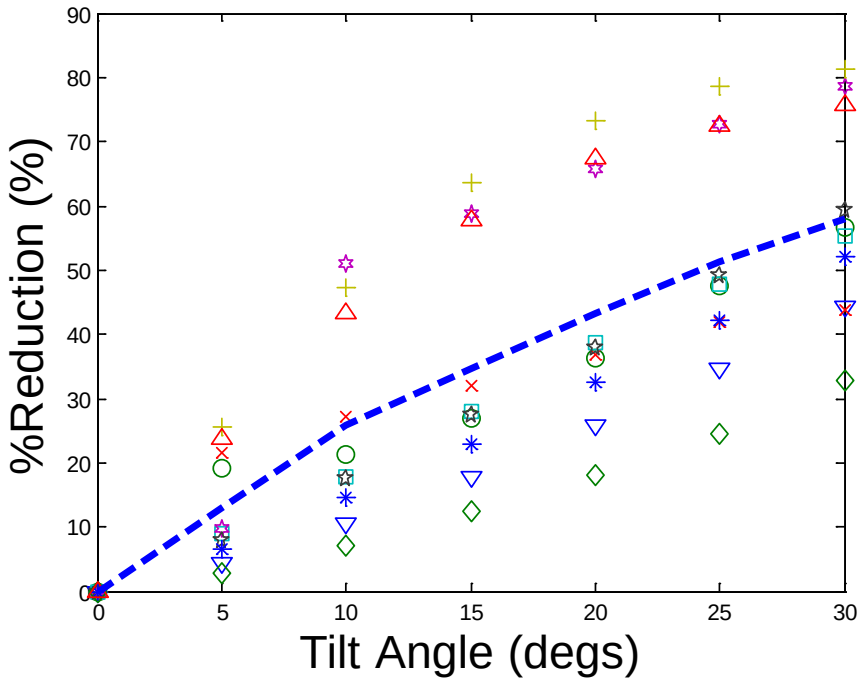

Figure 12. The reduction in glandular breast dose for each voxelized phantom with respect to tilt angle. Each marker represents a unique phantom. The dotted line is the mean value across all phantoms.

Anthropomorphic Phantom with the GE Discovery CT750HD

The percent reduction in dose relative to the zero-degree tilt is plotted in Figure 13 for MOSFETs in the breast and lung regions of the anthropomorphic phantom for all studied gantry-tilt angles.

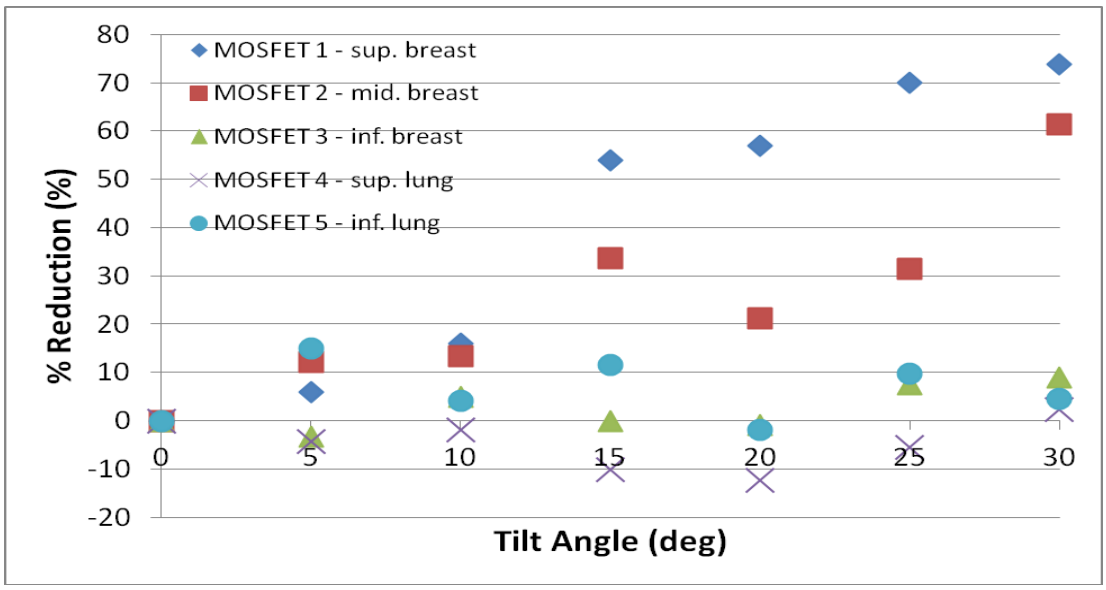

Figure 13. The experimentally measured reduction in dose in the dosimeters placed within the phantom.

The dose readings from the three breast dosimeters and the two lung dosimeters were combined, respectively, to quantify a measure of dose across the organs. The reduction in dose to the combined breast and lung regions are plotted in Figure 14 and Figure 15, respectively.

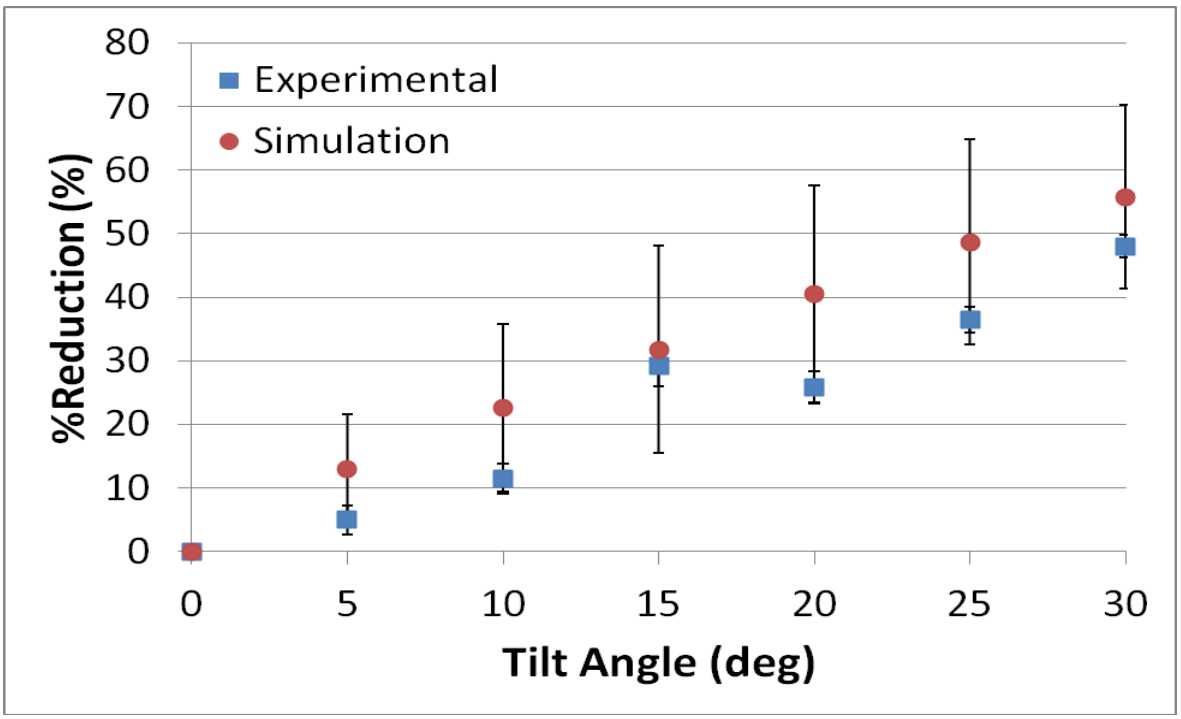

Figure 14. The dose reductions from a cardiac CT scan in the breast when all dosimeters placed within the breast region of the anthropomorphic phantom are combined (blue, square). For comparison, the average reduction in breast dose calculated across the computer simulations (red, circle) is plotted alongside the anthropomorphic data. The error bars represent +/- one standard deviation.

The doses deposited in the lung regions ranged from a 15% dose reduction to a 12% dose increase, displayed in Figure 13. In general, dose reduction occurred in the inferior region of the lung, while the superior region of the lung received increased dose. If both dosimeters are averaged as was done with the breast MOSFETs, the lung region represented receives between a 6% reduction to an 8% increase in dose across all tilt angles, in Figure 15. Dose was reduced for all angles except the 20-degree tilt, where the lung received an 8% increase in dose compared to a 27% reduction in breast dose.

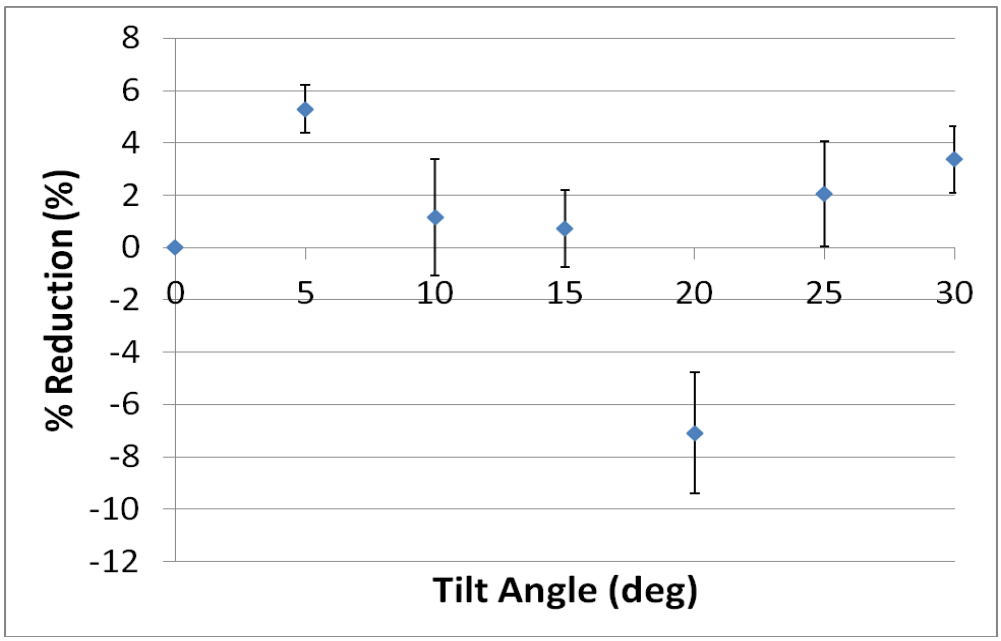

Figure 15. The percent reduction in dose delivered to lung averaged from dosimeters located in directly above and below the heart.

Affect of Tilted Acquisition on Image SNR

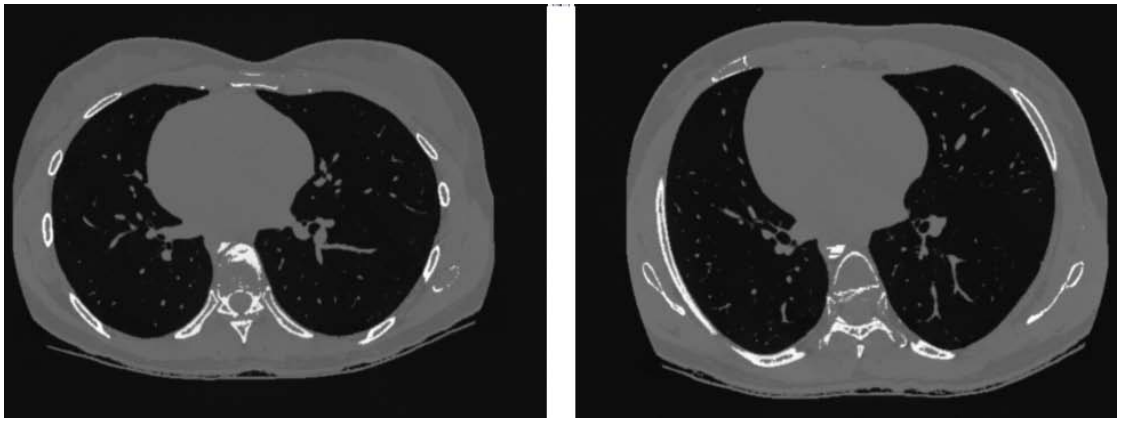

Figure 16. Reconstructed images of a voxelized phantom at zero degree tilt (left) and 30 degree tilt (right).

Figure 16 displays sample reconstructed images of one voxelized phantom acquired at a 0 and 30 degree gantry tilt. Figure 17 displays the mean SNR ratio, ($SNR_{tilt}/SNR_{0degrees}$), at each tilt angle for all phantoms and the mean ratio across all phantoms. A constant regime in the mean ratio, with ratio of ~1 exists up until approximately 15 degrees tilt. After tilting the gantry beyond 15 degrees a slow decline in the ratio occurs with increasing tilt angle. At 30 degree tilt, a mean ($SNR_{30degrees}/SNR_{0degrees}$) of 0.84 was calculated. Three phantoms experienced increased SNR at tilt angles ranging from 5-20 degrees, while three phantoms experienced reduced SNR. The remaining four phantoms showed minimal change in SNR (ratios of 0.9-1) for tilt angles up to 15 degrees.

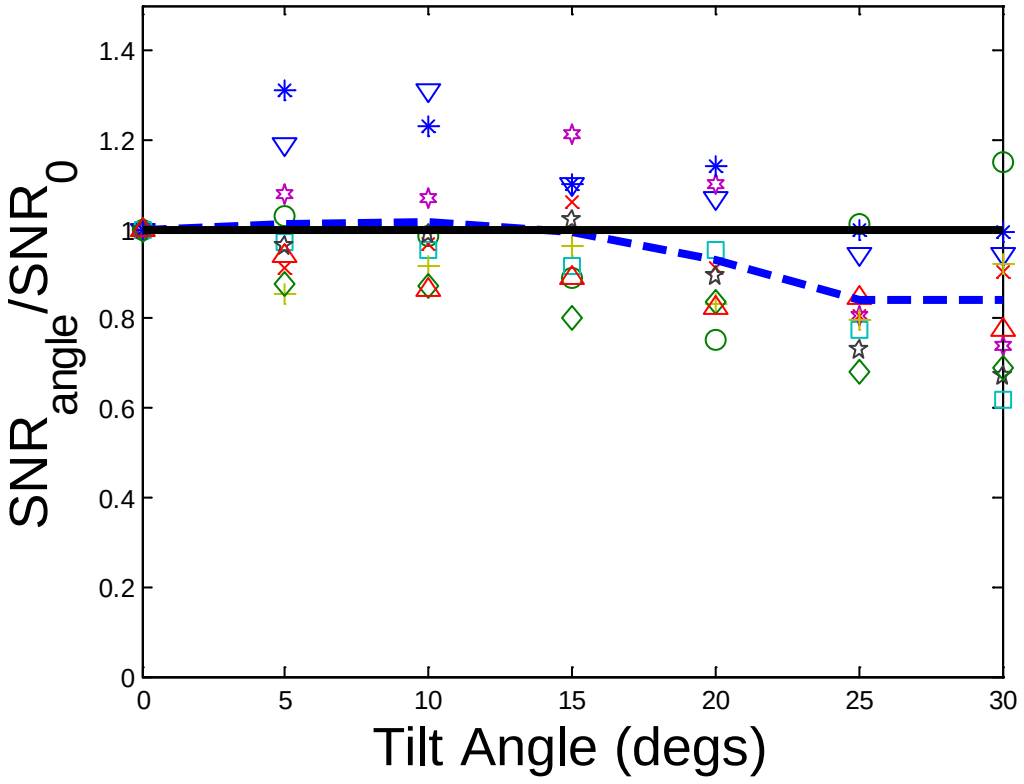

Figure 17. SNR ratios at a gantry tilt with respect to zero tilt of gantry. Each marker represents a specific phantom model, corresponding to the markers in Figure 13. The dotted line represents the mean ratio of SNRs across all phantoms at each tilt.

For the case of the experimental phantom study, the SNR was averaged across 5 trials per tilt angle to calculate the mean SNRs and SNR ratios, plotted in Figure 18. From Figure 18 the ($SNR_{30degrees}/SNR_{0degrees}$) for the experimental phantom is found to be 1.002, compared to the value of 0.84 in the case of the voxelized phantoms. Also, the decrease in the ratio seen after approximately 15 degrees in the voxelized phantoms is not present in the experimental study. At every tilt angle the experimental study maintained a fairly consistent SNR ratio of 1 or slightly higher.

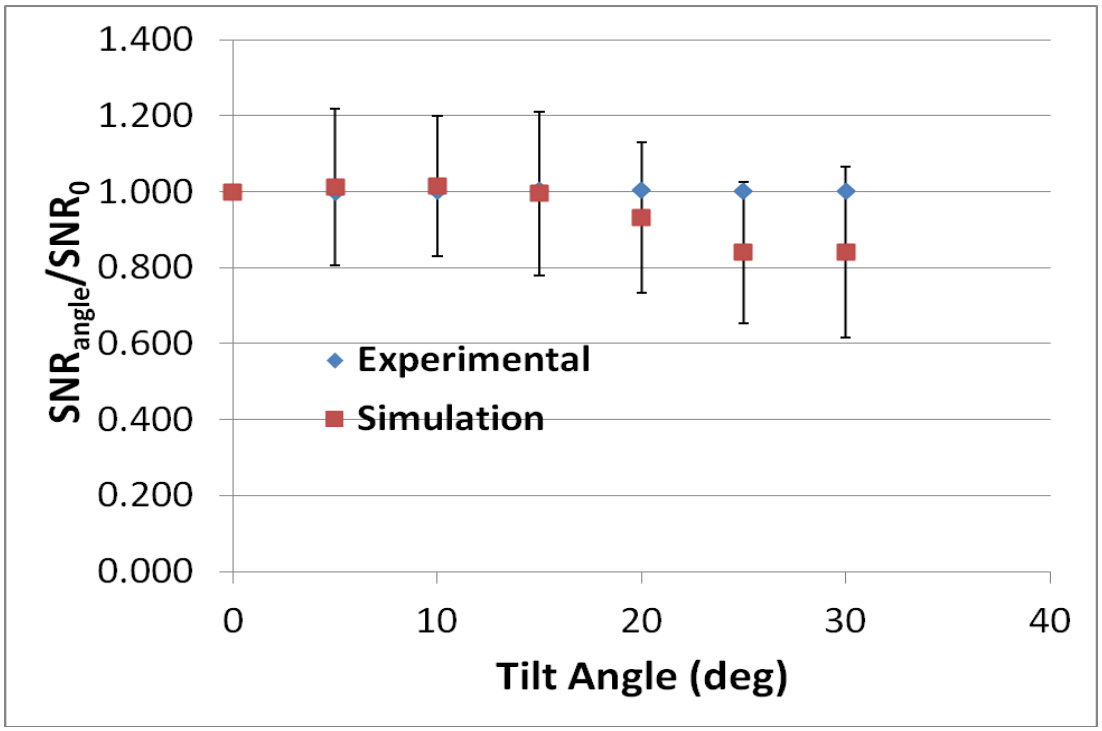

Figure 18. The mean SNR ratio per tilt angle (n=5) for the experimental phantom study plotted with the voxelized phantom mean SNR ratio. The error bars represent +/- one standard deviation (not visible for the experimental data, with values less than 0.003). The experimental data exist alongside the simulated data between 0 degrees and 15 degrees.

Chapter 5

DISCUSSION

To overcome the practical clinical limitations of scanning along the long-axis of the heart, this study investigated cardiac imaging with standard gantry tilt angles between 0-30 degrees. The simulation results demonstrated mean dose reduction for all tilt angles greater than 0 degrees. Within this set of angles, reductions between 12% and 60% (mean, 44%) were achieved in the glandular breast tissue (Figure 12). At 30 degrees, reductions range from 32% to 81% (mean, 55%), which is similar to the reductions seen when scanning at the long axis angle, which provided reductions ranging from 21% to 85% (mean, 55%). This agrees with the predicted 50% dose reduction estimated by Halpern et al . Our study estimated required gantry tilt angles between 10-50 degrees (mean, 30 degrees) for CT scanning parallel to the long axis of the heart of all the voxelized phantom models in this study. In the previous work, tilt angles between 7-54 degrees (mean, 32 degrees) allowed the inclusion of all 100 female patients for cardiac CT scanning at the heart's long axis.

The experimental study demonstrated breast dose reductions similar to those obtained in the simulations at all tilt angles. For the same range of angles, 5-30 degrees, the physical measurements show glandular breast dose reductions between 4% and 49% (mean, 26%) (Figure 15). Stronger corroboration exists between the two studies when tilt angles are between 15 degrees and 30 degrees. Within these angles, the simulated breast dose reduction ranged from 30% to 56% (mean, 44%) and the anthropomorphic phantom study ranged from 29% to 49% (mean, 35%).

In Figure 13 one can see as the gantry tilted away from the most superior breast dosimeter (MOSFET 1), it saw the greatest reduction in dose, while the most inferior breast dosimeter (MOSFET 3) experienced the smallest reduction in dose. Lung dose was found to be quite consistent across all tilt angles using the anthropomorphic phantom and ranged from an 8% increase in dose at 20 degree tilt and a 0-6% reduction in dose for the remaining tilt angles. Reductions in radiation dose must be considered concurrently with effects on image quality, with the goal of reducing dose while maintaining or improving

image quality. For the computer simulations and the experimental study, the SNR behavior with respect to tilt angles was investigated. Both the images reconstructed from the computer simulations and the experimental study showed a fairly consistent SNR across all angles up to 15 degrees (Figure 16, Table 8, and Figure 17). Beyond 15 degrees in the case of the computer simulations the SNR decreased with respect to tilt angle. At 30 degrees the mean ($SNR_{30degrees}/SNR_{0degrees}$) was found to be 0.84 and 1.002 for the computer simulations and experimental study, respectively.

In the case of the computer simulations, the decrease in SNR with increasing tilt angle after 15 degrees is likely attributed to a longer path length through the torso. However, between 0-15 degrees there is likely a greater decrease in path length through the breast than there is an increase through the torso for some phantoms. Once the torso path length is increased to be approximately equal to or greater than the reduction in breast path length, a decrease in SNR is expected and is demonstrated here by a 15% reduction in SNR at 30 degrees tilt. The variation in SNR ratios across voxelized phantoms is likely due to differences in patient anatomy and breast positioning. Further work and additional patient models are required to understand which patients will benefit from tilted gantry acquisition. Breast positioning devices may provide increased uniformity in results across patients. Between 15 degrees and 20 degrees it is noted a marked change in behavior of the trend occurs. It is possible channels hollowed out of the phantom model played a role in skewing the data when the gantry was tilted between these angles.

It should be noted as the gantry tilt is increased, patient positioning is affected. Due to the spatial position of a patient's heart being variable in the scanner, some patients will need to be translated out of the scanner bore. It was found the required distance to move the patient/CT scanner table out of the bore is between 0 cm and 1.5 cm for the voxelized phantoms used here. The necessity was also seen when making physical measurements by examination of the scout scans before the cardiac CT protocol was performed. Also, it has been suggested by Halpern et al. that tilted gantry acquisition can be achieved with a non-tilting gantry by the use of wedges under the patient.

Limitations of this work which may affect the acceptance of the tilted acquisition likely stem from a relatively small sample size of ten patient phantoms and one anthropomorphic phantom, since patient size and anatomy is likely a large factor in the use of tilted gantry. This is due to the fact patient breast size and positioning will affect the percent dose reduction and the maximum tilt angle that allows the patient to be positioned within the gantry.

In current clinical practice, cardiac CT images are reconstructed onto axial planes and reformatted to long and short-axis cardiac views. Conventional CT reconstruction algorithms available on the scanners can be used to reconstructed images that are tilted with respect to the standard axial plane, as was performed in the current study. Post-processing will be required to reformat the tilted images to long or short axis images, or images along the standard anatomical planes, however this processing is similar to what is performed on current scanners when reformatting to cardiac planes. Alternatively, axial images can be reconstructed directly from the tilted gantry acquisitions, using previously proposed algorithms. Overall, clinical translation of this method requires minor changes to the available reconstruction and reformatting algorithms.

Further research is required to quantify the dose deposited in all organs in the irradiated field during tilted gantry acquisition, and such studies are recommended before implementation in the clinic. This is primarily of concern for lung tissue due to its widespread presence in the irradiated field during cardiac scanning and its radiosensitivity. Using a positioning device to assist in removing breast tissue from the irradiated field may enable additional dose reductions. Additionally, even further dose reduction to the breast and lung tissue may be possible by reducing the scan length for the tilted gantry acquisitions. . It is expected that the scan length can be decreased with tilted acquisition since the projected profile of the heart is narrower when viewed off the axial axis. Results of the simulation study suggest that some patients will not benefit from tilted gantry acquisition, as the increase in mAs required to recover the lost SNR may negate the reduced breast dose. However, these patients may experience a net decrease in

dose when the scan length is reduced to adjust for the tilted acquisition. Thus, future studies are required to quantify the total dose in the breast and lung taking into account gantry tilt, reduced scan length, and SNR.

CONCLUSIONS

The simulated and experimental results demonstrated that tilted gantry acquisition has the potential to reduce glandular breast dose from cardiac CT scans while maintaining similar SNR to current axial scans. While maximum reductions of 33%-81% (mean, 55%) were seen with a 30-degree gantry tilt, SNR was also reduced for most simulated phantoms. At a 15-degree tilt angle, the breast dose was reduced by 12%-64% (mean, 30%) compared to a mean 0.3% decrease of SNR. Therefore, a tilt angle of 15 degrees may be most applicable for wide range of the patient population. Additional work is required to understand which patients may benefit from tilted gantry acquisition. Tilted gantry acquisitions can be applied with other currently available dose reduction techniques, such as tube current modulation, to offer additional dose reductions, thus lowering patients' risk for future cancer development.

REFERENCES

1. Linton OW, Mettler F a. National conference on dose reduction in CT, with an emphasis on pediatric patients. *AJR. American journal of roentgenology*. 2003;181(2):321-9. Available at: http://www.ncbi.nlm.nih.gov/pubmed/12876005.

2. Nickoloff EL, Alderson PO. Radiation exposures to patients from CT: reality, public perception, and policy. *AJR. American journal of roentgenology*. 2001;177(2):285-7. Available at: http://www.ncbi.nlm.nih.gov/pubmed/11461846.

3. IMV Medical Information Division. *IMV 2011 CT Market Outlook Report*. Des Plaines, IL; 2011.

4. Hoffmann U, Ferencik M, Cury RC, Pena AJ. Coronary CT angiography. *Journal of nuclear medicine : official publication, Society of Nuclear Medicine*. 2006;47(5):797-806. Available at: http://www.ncbi.nlm.nih.gov/pubmed/17437740.

5. Einstein AJ, Henzlova MJ, Rajagopalan S. Estimating risk of cancer associated with radiation exposure from 64-slice computed tomography coronary angiography. *JAMA : the journal of the American Medical Association*. 2007;298(3):317-23. Available at: http://www.ncbi.nlm.nih.gov/pubmed/17635892. Accessed July 26, 2011.

6. Raff GL, Gentry RE. Tomography Before and After Implementation of Radiation Dose – Reduction Techniques. *Cardiology*. 2009;301(22):2340-2348.

7. Leschka S, Kim C-H, Baumueller S, et al. Scan length adjustment of CT coronary angiography using the calcium scoring scan: effect on radiation dose. *AJR. American journal of roentgenology*. 2010;194(3):W272-7. Available at: http://www.ncbi.nlm.nih.gov/pubmed/20173126. Accessed September 4, 2011.

8. Hurwitz LM, Yoshizumi TT, Goodman PC, et al. Radiation dose savings for adult pulmonary embolus 64-MDCT using bismuth breast shields, lower peak kilovoltage, and automatic tube current modulation. *AJR. American journal of roentgenology*. 2009;192(1):244-53. Available at: http://www.ncbi.nlm.nih.gov/pubmed/19098206. Accessed August 26, 2011.

9. Halpern EJ, Takakuwa KM, Gingold EL, Halpern DJ. A novel approach to reduce breast radiation exposure with coronary CTA: angled axial image acquisition. *Academic radiology*. 2009;16(8):951-6. Available at: http://www.ncbi.nlm.nih.gov/pubmed/19375949. Accessed September 21, 2011.

10. Angel E, Yaghmai N, Jude CM, et al. Dose to radiosensitive organs during routine chest CT: effects of tube current modulation. *AJR. American journal of roentgenology*. 2009;193(5):1340-5. Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2954276&tool=pmcentrez&rendertype=abstract. Accessed July 26, 2011.

11. AAPM Task Group 23 of the Diagnostic Imaging Council CT Committee. *The measurement, reporting, and management of radiation dose in CT*. 2008.

12. Bushberg JT, Seibert JA, Leidholdt EM, Boone JM. *The Essential Physics of Medical Imaging*. Third Edit. (Mitchell CW, ed.). Lippincott Williams & Wilkins; 2012:38-44.

13. National Research Council (US) Committee to Assess Health Risks from Exposure to Low Levels of Ionizing Radiation; Nuclear and Radiation Studies Board, Division of Earth and Life Studies NRC of the NA. Health risks from exposure to low levels of ionizing radiation: BEIR VII Phase 2. *Cancer*. 2006;18(3).

14. Mettler F a, Huda W, Yoshizumi TT, Mahesh M. Effective doses in radiology and diagnostic nuclear medicine: a catalog. *Radiology*. 2008;248(1):254-63. Available at: http://www.ncbi.nlm.nih.gov/pubmed/18566177.

15. Einstein AJ, Moser KW, Thompson RC, Cerqueira MD, Henzlova MJ. Radiation dose to patients from cardiac diagnostic imaging. *Circulation*. 2007;116(11):1290-305. Available at: http://www.ncbi.nlm.nih.gov/pubmed/17846343. Accessed July 26, 2011.

16. Smith-Bindman R, Lipson J, Marcus R, et al. Radiation dose associated with common computed tomography examinations and the associated lifetime attributable risk of cancer. *Archives of internal medicine*. 2009;169(22):2078-86. Available at: http://www.ncbi.nlm.nih.gov/pubmed/20008690.

17. Hall EJ, Brenner DJ. Cancer risks from diagnostic radiology. *The British journal of radiology*. 2008;81(965):362-78. Available at: http://www.ncbi.nlm.nih.gov/pubmed/18440940. Accessed July 29, 2011.

18. 2007 Recommendations of the International Commission on Radiological. ICRP Publication 103. *Ann ICRP*. 2007;37:1-332.

19. United Nations Scientific Committee on the Effects of Atomic Radiation. *Report of the United Nations Scientific Committee on the Effects of Atomic Radiation 2010 UNSCEAR 2010 Report*. 2010.

20. Sodickson A, Andriole KP, Prevedello LM, Nawfel RD. Recurrent CT, Cumulative radiaton exposure, and associated radiation-induced cancer risks from CT of adults. *Radiology*. 2009;251(1):175-184.

21. Preston DL, Shimizu Y, Pierce DA, Suyama A. Studies of Mortality of Atomic Bomb Survivors . Studies of Mortality Report 13□: Solid Cancer and Noncancer Disease Mortality□: 1950-1997. *Radiation Research*. 2003;160(4):381-407.

22. Tubiana M, Feinendegen LE, Kaminski JM. The Linear No-Threshold Relationship Is Inconsistent Experimental Data 1. 2009;251(1).

23. Morin Doody M, Lonstein JE, Stovall M, et al. Breast Cancer Mortality After Diagnostic Radiography: Findings From the U.S. Scoliosis Cohort Study. *Spine*. 2000;25(16):2052-2063.

24. Howe GR, McLaughlin J. Breast Cancer Mortality between 1950 and 1987 after exposure to fractionated moderate-dose-rate ionizing radiation in the Canadian Fluoroscopy Cohort Study and a comparison with breast cancer mortality in the Atomic Bomb Survivors Study. *Radiation Research*. 1996;145(6):694-707.

25. Brenner DJ, Hall EJ. Computed tomography--an increasing source of radiation exposure. *The New England journal of medicine*. 2007;357(22):2277-84. Available at: http://www.ncbi.nlm.nih.gov/pubmed/18046031.

26. Parker MS, Hui FK, Camacho M a, et al. Female breast radiation exposure during CT pulmonary angiography. *AJR. American journal of roentgenology*. 2005;185(5):1228-33. Available at: http://www.ncbi.nlm.nih.gov/pubmed/16247139. Accessed August 4, 2011.

27. Tong S, Alessio AM, Kinahan PE. Image reconstruction for PET/CT scanners: past achievements and future challenges. *Imaging*. 2010;2(5):529-545.

28. Briesmeister JF. *MCNP – A General Monte Carlo N – Particle Transport Code Version 4C*. 2000.

29. Agostinelli S. GEANT4—a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. 2003;506(3):250-303. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0168900203013688. Accessed July 14, 2011.

30. Sztejnberg M, Jevremovic T. Geant4 Based Monte Carlo Dose Calculation Engine for Radiation Therapy. *IEEE Transactions on Nuclear Science*. 2010;57(2):775-781. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5416322.

31. Jiang H, Paganetti H. Adaptation of GEANT4 to Monte Carlo dose calculations based on CT data. *Medical Physics*. 2004;31(10):2811. Available at: http://link.aip.org/link/MPHYA6/v31/i10/p2811/s1&Agg=doi. Accessed September 21, 2011.

32. Amako K, Guatelli S, Ivanchencko V, et al. Validation of Geant4 electromagnetic physics versus protocol data. *IEEE Symposium Conference Record Nuclear Science 2004.* 2004;4(C):2115-2119. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1462680.

33. Geleijns J, Salvadó Artells M, Veldkamp WJH, López Tortosa M, Calzado Cantera a. Quantitative assessment of selective in-plane shielding of tissues in computed tomography through evaluation of absorbed dose and image quality. *European radiology*. 2006;16(10):2334-40. Available at: http://www.ncbi.nlm.nih.gov/pubmed/16604323. Accessed November 22, 2011.

34. International Commission on Radiological Protection. Adult Reference Computational Phantoms. ICRP Publication 110. *Ann. of ICRP*. 2009;39(2).

35. Cranley K, Gilmore B, Fogarty G, Desponds L. *Catalogue of diagnostic x-ray spectra and other data. IPEM Report 78.* 1997.

36. Hein I, Taguchi K, Silver MD, Kazama M, Mori I. Feldkamp-based cone-beam reconstruction for gantry-tilted helical multislice CT. *Medical Physics*. 2003;30(12):3233. Available at: http://link.aip.org/link/MPHYA6/v30/i12/p3233/s1&Agg=doi. Accessed November 22, 2011.

APPENDIX A

Appendix A lists GEANT4 C++ code, octave scripts, shell scripts, and Condor submit scripts used when performing a dose deposition per material simulation. Note that not all GEANT4 code for a simulation is listed, only poignant portions.

The GEANT4 main program:

```
//GEANT4 mail program
#include "DicomGeometry.hh"
#include "LeiTestPhysicsList.hh"
#include "LeiTestPrimaryGeneratorAction.hh"
#include "LeiTestEventAction.hh"
#include "LeiTestTrackingAction.hh"
#include "LeiTestSteppingAction.hh"
#include "LeiTestAnalysisManager.hh"
#include "Randomize.hh"
#include "RegularDicomDetectorConstruction.hh"
#include "LeiTestRunAction.hh"

#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "G4UIterminal.hh"
#include "G4UItcsh.hh"
#include "G4VisExecutive.hh"


#include <iostream>
#include <fstream>
#include <math.h>
#include <time.h>

#define PI 3.1415965

double GlobalAngle=0;
double GlobalPhi=0;

int main(int argc, char** argv)
{
//Read in input arguments (number of projections, projection number, gantry tilt
angle
//double GlobalAngle=90;
       int numProj = std::atoi(argv[1]);//number of projections
       G4double totalAngle=360.0*deg;
       G4double deltaPhi = totalAngle/(numProj);
       int angleNumber = std::atoi(argv[2]);//projection number
       int angleNumber2 = std::atoi(argv[3]);//gantry tilt angle

       G4double phi = angleNumber*deltaPhi;
       GlobalAngle=(phi/deg);
    GlobalPhi=angleNumber2;

       time_t systyme=time(NULL);
       int GlobalAngleSeed= (int) (systyme*GlobalAngle+0.5);
```

```
	CLHEP::HepRandom::setTheEngine(new CLHEP::RanecuEngine);
	CLHEP::HepRandom::setTheSeed(GlobalAngleSeed);
	// Run Manager
	G4RunManager* runManager = new G4RunManager;
//G4cout<<"run manager new"<<G4endl;
	//set mandatory user initialization classes
	DicomGeometry* detector;
	 detector = new RegularDicomDetectorConstruction();
	runManager->SetUserInitialization(detector);
//	G4VUserPhysicsList* physics = new LeiTestPhysicsList;
	runManager->SetUserInitialization(new LeiTestPhysicsList);

	//user action classes
//	G4VUserPrimaryGeneratorAction* gen_action = new
LeiTestPrimaryGeneratorAction;
	runManager->SetUserAction(new LeiTestPrimaryGeneratorAction());

	//LeiTestRunAction* theRunAction = new LeiTestRunAction;
	runManager->SetUserAction(new LeiTestRunAction);

//	LeiTestEventAction* event_action = new LeiTestEventAction;
	runManager->SetUserAction(new LeiTestEventAction);

	/*G4UserTrackingAction* tracking_action = new LeiTestTrackingAction;
	runManager->SetUserAction(tracking_action);
	*/
	//G4UserSteppingAction* stepping_action = new LeiTestSteppingAction;
	//runManager->SetUserAction(stepping_action);

	G4UImanager* UI =G4UImanager::GetUIpointer();

	char buffer[512];
	sprintf(buffer, "/LeiTest/det/setViewAngle %f deg", GlobalAngle);
	UI->ApplyCommand(buffer);

UI->ApplyCommand("/random/setDirectoryName .");
UI->ApplyCommand("/random/setSavingFlag 1");
	//Initialize G4 Kernel
	runManager->Initialize();

G4cout<<"global angle = "<<GlobalAngle<<G4endl;
G4cout<<"global phi = "<<GlobalPhi<<G4endl;
///*

	//Get the pointer to the User Interface manager
	// Initialize Analysis Package
//	LeiTestAnalysisManager* analysis = LeiTestAnalysisManager::getInstance();

//	analysis->book();
//G4cout<<"booked"<<G4endl;
  UI->ApplyCommand("/run/verbose 0");
  UI->ApplyCommand("/event/verbose 0");
  UI->ApplyCommand("/tracking/verbose 0");

  //UI->ApplyCommand("/units/list");
```

```
///*
G4int numberOfEvent = 100000;//100000<--this is multipled by 100 in
src/LeiTestPrimaryGeneratorAction.cc
runManager->BeamOn(numberOfEvent);
//*/

/*
//ADDING VISUALS
G4VisManager* visManager = new G4VisExecutive;
visManager->Initialize();
//

char buffer4[512];
int gAng=(int) (GlobalAngle + 0.5);
sprintf(buffer4,"/vis/rayTracer/trace Laura_%d.jpeg",gAng);
UI->ApplyCommand("/vis/open RayTracer"); //RAY
UI->ApplyCommand(buffer4);
UI->ApplyCommand("/vis/drawVolume");//RAY
//
delete visManager;
//END OF ADDING VISuals
*/
delete runManager;

return 0;
}
```

The creation of the world and geometry (physical model):

```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration.  It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license .  These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use.  Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
```

```
// The code was written by :
//     *Louis Archambault louis.archambault@phy.ulaval.ca,
//      *Luc Beaulieu beaulieu@phy.ulaval.ca
//      +Vincent Hubert-Tremblay at tigre.2@sympatico.ca
//
//
// *Centre Hospitalier Universitaire de Quebec (CHUQ),
// Hotel-Dieu de Quebec, departement de Radio-oncologie
// 11 cote du palais. Quebec, QC, Canada, G1R 2J6
// tel (418) 525-4444 #6720
// fax (418) 691 5268
//
// + Université Laval, Québec (QC) Canada
//*********************************************************//
//*******************************************************
//
// DicomGeometry.hh :
//     - Start the building of the geometry
//     - Creation of the world and other "mother"(middle) volume
//     - Initialisation of patient geometry
//     - Initialisation of HeaD geometry
//     - Functions are in DicomGeometry.cc, PatientConstructor.cc
//*******************************************************

#ifndef DicomGeometry_h
#define DicomGeometry_h 1
#include "DicomPatientZSliceHeader.hh"
#include "globals.hh"
#include "G4VUserDetectorConstruction.hh"
#include <map>

//class DicomConfiguration;
//class DicomPatientConstructor;
class G4Material;
class G4LogicalVolume;
//class G4PhysicalVolume;
class G4Box;
class LeiTestDetectorMessenger;
class DicomGeometry : public G4VUserDetectorConstruction
{
public:
  DicomGeometry();
  ~DicomGeometry();
public:
  G4VPhysicalVolume* Construct();
  void SetViewAngle (G4double angle) {viewAngle=angle;};

private:
  G4double viewAngle;
  G4double sliceThickness2;
  G4double totalNumberOfFile2;
protected:
  void InitialisationOfMaterials();
  void ReadPatientData();
  void ReadPatientDataFile(const G4String& fname);
  void MergeZSliceHeaders();
```

```
  G4Material* BuildMaterialWithChangingDensity(const G4Material* origMate,
G4double density, G4String newMateName);
  G4String ftoa(float flo);
  void ConstructPatientContainer();
  virtual void ConstructPatient()=0;
  void SetScorer(G4LogicalVolume* voxel_logic);
  LeiTestDetectorMessenger* detectorMessenger;
//  DicomPatientConstructor* patientConstructor;

  //Materials ...
 /*
  G4Material* hardBone;
  G4Material* cartilage;
  G4Material* skin;
  G4Material* blood;
  G4Material* muscleTissue;
  G4Material* softTissue;
  G4Material* redBoneMarrow;
  G4Material* glandularBreast;
  G4Material* yellowBoneMarrow;
  G4Material* adiposeTissue;
  G4Material* lungTissue;
 // G4Material* air;
  G4Material* Bodyair;
  G4Material* Pb;
*/
//  G4Material* Pb;
// World ...
protected:
  G4Material* air;
  G4Box* solidWorld;
  G4Box* container_solid;
  G4LogicalVolume* container_logic;
  G4VPhysicalVolume* container_phys;
  G4int fNoFiles;
  std::vector<G4Material*> fOriginalMaterials;
  std::vector<G4Material*> fMaterials;
  G4LogicalVolume* logicWorld;
  G4VPhysicalVolume* physiWorld;
  size_t* fMateIDs;
  std::map<G4int,G4double> fDensityDiffs;
  std::vector<DicomPatientZSliceHeader*> fZSliceHeaders;
  DicomPatientZSliceHeader* fZSliceHeaderMerged;

  G4int nVoxelX, nVoxelY, nVoxelZ;
  G4double voxelHalfDimX,  voxelHalfDimY, voxelHalfDimZ;

};

#endif


//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
```

```
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration.  It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license .  These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use.  Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
// The code was written by :
//     *Louis Archambault louis.archambault@phy.ulaval.ca,
//      *Luc Beaulieu beaulieu@phy.ulaval.ca
//      +Vincent Hubert-Tremblay at tigre.2@sympatico.ca
//
//
// *Centre Hospitalier Universitaire de Quebec (CHUQ),
// Hotel-Dieu de Quebec, departement de Radio-oncologie
// 11 cote du palais. Quebec, QC, Canada, G1R 2J6
// tel (418) 525-4444 #6720
// fax (418) 691 5268
//
// + Université Laval, Québec (QC) Canada
//*****************************************************

#include "globals.hh"
#include "LeiTestDetectorSD.hh"
#include "G4MaterialTable.hh"
#include "G4ElementTable.hh"
#include "G4Box.hh"
#include "G4LogicalVolume.hh"
#include "G4VPhysicalVolume.hh"
#include "G4PVPlacement.hh"
#include "G4PVParameterised.hh"
#include "G4Material.hh"
#include "G4Element.hh"
#include "G4VisAttributes.hh"
#include "G4Colour.hh"
#include "G4ios.hh"
#include "G4ThreeVector.hh"
#include "G4SDManager.hh"
#include "G4UserLimits.hh"
#include "LeiTestDetectorMessenger.hh"
#include "DicomGeometry.hh"
#include "DicomPatientParameterisation.hh"
```

```
//#include "DicomPatientConstructor.hh"
#include "DicomConfiguration.hh"
#include "G4UnitsTable.hh"
#include "G4VUserDetectorConstruction.hh"
#include "G4MultiFunctionalDetector.hh"
#include "G4VPrimitiveScorer.hh"
#include "LeiTestPSEnergyDeposit.hh"
#include "G4Cons.hh"
#include "G4Tubs.hh"
#include "G4Trd.hh"
#include "G4Transform3D.hh"
#include "G4VisAttributes.hh"
#include <math.h>
#include "DicomPatientZSliceHeader.hh"
#define PI 3.14159265

extern double GlobalPhi;

DicomGeometry::DicomGeometry()
{detectorMessenger = new LeiTestDetectorMessenger(this);
//  patientConstructor = new DicomPatientConstructor();
  /*hardBone = 0;
  cartilage = 0;
  skin = 0;
  blood = 0;
  muscleTissue = 0;
  softTissue = 0;
  redBoneMarrow = 0;
  glandularBreast = 0;
  yellowBoneMarrow = 0;
  adiposeTissue = 0;
  lungTissue = 0;
  Bodyair = 0;
  air = 0;
  Pb = 0;
  solidWorld = 0;
  logicWorld = 0;
  physiWorld = 0;
*/
}

DicomGeometry::~DicomGeometry()
{delete detectorMessenger;
  /*
  delete air;
  delete Bodyair;
  delete hardBone;
  delete cartilage;
  delete skin;
  delete blood;
  delete muscleTissue;
  delete softTissue;
  delete redBoneMarrow;
  delete glandularBreast;
  delete yellowBoneMarrow;
  delete Pb;
```

```
  delete adiposeTissue;
  delete lungTissue;
  */
//delete patientConstructor;
G4cout<<"Geometry Deleted"<<G4endl;
}

void DicomGeometry::InitialisationOfMaterials()
{
  // Creating elements :
  G4double z, a, density;
  G4String name, symbol;

  G4Element* elC = new G4Element( name = "Carbon",
                                  symbol = "C",
                                  z = 6.0, a = 12.011 * g/mole );
  G4Element* elH = new G4Element( name = "Hydrogen",
                              symbol = "H",
                              z = 1.0, a = 1.008  * g/mole );
  G4Element* elN = new G4Element( name = "Nitrogen",
                              symbol = "N",
                              z = 7.0, a = 14.007 * g/mole );
  G4Element* elO = new G4Element( name = "Oxygen",
                                  symbol = "O",
                                  z = 8.0, a = 16.00  * g/mole );
  G4Element* elNa = new G4Element( name = "Sodium",
                                   symbol = "Na",
                                   z= 11.0, a = 22.98977* g/mole );
  G4Element* elS = new G4Element( name = "Sulfur",
                                  symbol = "S",
                                  z = 16.0,a = 32.065* g/mole );
  G4Element* elCl = new G4Element( name = "Chlorine",
                                   symbol = "Cl",
                                   z = 17.0, a = 35.453* g/mole );
  G4Element* elK = new G4Element( name = "Potassium",
                                  symbol = "K",
                                  z = 19.0, a = 30.0983* g/mole );
  G4Element* elP = new G4Element( name = "Phosphorus",
                                  symbol = "P",
                                  z = 30.0, a = 30.973976* g/mole );
  G4Element* elFe = new G4Element( name = "Iron",
                                   symbol = "Fe",
                                   z = 26, a = 56.845* g/mole );
  G4Element* elMg = new G4Element( name = "Magnesium",
                                   symbol = "Mg",
                                   z = 12.0, a = 24.3050* g/mole );
  G4Element* elCa = new G4Element( name="Calcium",
                                   symbol = "Ca",
                                   z = 20.0, a = 40.078* g/mole );
  G4Element* elAr = new G4Element( name="Argon",
                                   symbol = "Ar",
                                   z = 18, a = 39.948* g/mole );
  G4Element* elPb = new G4Element(name = "Lead", symbol = "Pb",
z=82,a=207.19*g/mole);
```

```
  G4Element* elI = new G4Element(name = "Iodine", symbol = "I",
z=53,a=126.9045*g/mole);

// Creating Materials :
  G4int numberofElements;


  // Hard Bone
  G4Material* hardBone = new G4Material( "Bone",
                               density = 1920*kg/m3,
                               numberofElements = 9 );
  hardBone->AddElement(elH,.033);
  hardBone->AddElement(elC,0.154);
  hardBone->AddElement(elN,0.041);
  hardBone->AddElement(elO,0.432);
  hardBone->AddElement(elNa,0.01);
  hardBone->AddElement(elP,0.103);
  hardBone->AddElement(elS,0.002);
  hardBone->AddElement(elCa,0.223);
  hardBone->AddElement(elMg,0.002);

// Cartilage
  G4Material* cartilage = new G4Material( "Cartilage",
                              density = 1100*kg/m3,
                              numberofElements = 8 );
  cartilage->AddElement(elH,0.096);
  cartilage->AddElement(elC,0.099);
  cartilage->AddElement(elN,0.022);
  cartilage->AddElement(elO,0.744);
  cartilage->AddElement(elNa,0.005);
  cartilage->AddElement(elP,0.022);
  cartilage->AddElement(elS,0.009);
  cartilage->AddElement(elCl,0.003);


  // Skin
  G4Material* skin = new G4Material( "Skin",
                           density = 1090*kg/m3,
                           numberofElements = 9);
  skin->AddElement(elH,0.1);
  skin->AddElement(elC,0.204);
  skin->AddElement(elN,0.042);
  skin->AddElement(elO,0.645);
  skin->AddElement(elNa,0.002);
  skin->AddElement(elP,0.001);
  skin->AddElement(elS,0.002);
  skin->AddElement(elCl,0.003);
  skin->AddElement(elK,0.001);

  // Blood
  G4Material* blood = new G4Material( "Blood",
                            density = 1060*kg/m3,
                            numberofElements = 10 );
  blood->AddElement(elH,0.102);
  blood->AddElement(elC,0.112);
  blood->AddElement(elN,0.03);
```

```
  blood->AddElement(elO,0.746);
  blood->AddElement(elNa,0.001);
  blood->AddElement(elP,0.001);
  blood->AddElement(elS,0.002);
  blood->AddElement(elCl,0.003);
  blood->AddElement(elK,0.002);
  blood->AddElement(elFe,0.001);

  // Muscle Tissue
  G4Material* muscleTissue = new G4Material( "Muscle",
                            density = 1050*kg/m3,
                            numberofElements = 9 );
  muscleTissue->AddElement(elH,0.102);
  muscleTissue->AddElement(elC,0.142);
  muscleTissue->AddElement(elN,0.034);
  muscleTissue->AddElement(elO,0.711);
  muscleTissue->AddElement(elNa,0.001);
  muscleTissue->AddElement(elP,0.002);
  muscleTissue->AddElement(elS,0.003);
  muscleTissue->AddElement(elCl,0.001);
  muscleTissue->AddElement(elK,0.004);

  // Soft Tissue
  G4Material* softTissue = new G4Material( "SoftTissue",
                           density = 1050*kg/m3,
                           numberofElements = 9 );
  softTissue->AddElement(elH,0.104);
  softTissue->AddElement(elC,0.139);
  softTissue->AddElement(elN,0.029);
  softTissue->AddElement(elO,0.718);
  softTissue->AddElement(elNa,0.001);
  softTissue->AddElement(elP,0.002);
  softTissue->AddElement(elS,0.002);
  softTissue->AddElement(elCl,0.002);
  softTissue->AddElement(elK,0.003);

  // Red Bone Marrow
  G4Material* redBoneMarrow = new G4Material( "RedBoneMarrow",
                                  density = 1030*kg/m3,
                                  numberofElements = 9);
  redBoneMarrow->AddElement(elH,0.106);
  redBoneMarrow->AddElement(elC,0.415);
  redBoneMarrow->AddElement(elN,0.034);
  redBoneMarrow->AddElement(elO,0.439);
  redBoneMarrow->AddElement(elNa,0.0);
  redBoneMarrow->AddElement(elS,0.002);
  redBoneMarrow->AddElement(elCl,0.002);
  redBoneMarrow->AddElement(elFe,0.001);
  redBoneMarrow->AddElement(elP,0.001);

// Breast (mammary gland)
  G4Material* glandularBreast = new G4Material( "glandularBreast",
                                density = 1020*kg/m3,
                                numberofElements = 8 );
  glandularBreast->AddElement(elH,0.106);
  glandularBreast->AddElement(elC,0.332);
```

```
  glandularBreast->AddElement(elN,0.03);
  glandularBreast->AddElement(elO,0.527);
  glandularBreast->AddElement(elNa,0.001);
  glandularBreast->AddElement(elP,0.001);
  glandularBreast->AddElement(elS,0.002);
  glandularBreast->AddElement(elCl,0.001);

  // Yellow Bone Marrow
  G4Material* yellowBoneMarrow = new G4Material( "YellowMarrow",
                               density = 980*kg/m3,
                               numberofElements = 7);
  yellowBoneMarrow->AddElement(elH,0.115);
  yellowBoneMarrow->AddElement(elC,0.644);
  yellowBoneMarrow->AddElement(elN,0.007);
  yellowBoneMarrow->AddElement(elO,0.231);
  yellowBoneMarrow->AddElement(elNa,0.001);
  yellowBoneMarrow->AddElement(elS,0.001);
  yellowBoneMarrow->AddElement(elCl,0.001);


//Adipose Tissue
  G4Material* adiposeTissue = new G4Material("Adipose", density = 950*kg/m3,
                            numberofElements = 7);
  adiposeTissue->AddElement(elH,0.114);
  adiposeTissue->AddElement(elC,0.598);
  adiposeTissue->AddElement(elN,0.007);
  adiposeTissue->AddElement(elO,0.278);
  adiposeTissue->AddElement(elNa,0.001);
  adiposeTissue->AddElement(elP,0.001);
  adiposeTissue->AddElement(elS,0.001);

//Lung Tissue
 G4Material* lungTissue = new G4Material("Lung", density = 260*kg/m3,
                            numberofElements = 9);
  lungTissue->AddElement(elH,0.103);
  lungTissue->AddElement(elC,0.105);
  lungTissue->AddElement(elN,0.031);
  lungTissue->AddElement(elO,0.749);
  lungTissue->AddElement(elNa,0.002);
  lungTissue->AddElement(elP,0.002);
  lungTissue->AddElement(elS,0.003);
  lungTissue->AddElement(elCl,0.003);
  lungTissue->AddElement(elK,0.002);

// Body Air
  G4Material* Bodyair = new G4Material( "BodyAir",
                        1.290*mg/cm3,
                        numberofElements = 4 );
  Bodyair->AddElement(elN, 0.755268);
  Bodyair->AddElement(elAr, 0.012827);
  Bodyair->AddElement(elC, 0.000124);
  Bodyair->AddElement(elO, 0.231781);

// Air
  air = new G4Material( "Air",
                        1.290*mg/cm3,
```

```
                        numberofElements = 4 );
  air->AddElement(elN, 0.755268);
  air->AddElement(elAr, 0.012827);
  air->AddElement(elC, 0.000124);
  air->AddElement(elO, 0.231781);

//Pb = new G4Material("Lead", density=11.35*g/cm3,numberofElements=1);
//Pb->AddElement(elPb,1.0);

//FOR VIRTUAL FAMILY
///*
//G4Material* water=new G4Material("Water", density=11.34*g/cm3,
numberofElements=1);

G4Material* water=new G4Material("Water", density=1.0*g/cm3, numberofElements=2);
water->AddElement(elH,0.112);
water->AddElement(elO,0.888);
//water->AddElement(elPb,1.0);

G4Material* connectiveTissue=new G4Material("ConnectiveTissue",
density=1.120*g/cm3, numberofElements=7);
connectiveTissue->AddElement(elH,0.094);
connectiveTissue->AddElement(elC,0.208);
connectiveTissue->AddElement(elN,0.063);
connectiveTissue->AddElement(elO,0.624);
connectiveTissue->AddElement(elNa,0.006);
connectiveTissue->AddElement(elS,0.002);
connectiveTissue->AddElement(elCl,0.003);


G4Material* eyeLens=new G4Material("EyeLens", density=1.070*g/cm3,
numberofElements=8);
eyeLens->AddElement(elH,0.096);
eyeLens->AddElement(elC,0.195);
eyeLens->AddElement(elN,0.057);
eyeLens->AddElement(elO,0.646);
eyeLens->AddElement(elNa,0.001);
eyeLens->AddElement(elS,0.001);
eyeLens->AddElement(elP,0.003);
eyeLens->AddElement(elCl,0.001);


G4Material* trachea=new G4Material("Trachea", density=1.060*g/m3,
numberofElements=9);
trachea->AddElement(elH,0.101);
trachea->AddElement(elC,0.139);
trachea->AddElement(elN,0.033);
trachea->AddElement(elO,0.713);
trachea->AddElement(elNa,0.001);
trachea->AddElement(elP,0.004);
trachea->AddElement(elS,0.004);
trachea->AddElement(elCl,0.001);
trachea->AddElement(elK,0.004);
```

```
G4Material* stomach=new G4Material("Stomach", density=1.050*g/cm3,
numberofElements=9);
stomach->AddElement(elH,0.104);
stomach->AddElement(elC,0.139);
stomach->AddElement(elN,0.029);
stomach->AddElement(elO,0.721);
stomach->AddElement(elNa,0.001);
stomach->AddElement(elP,0.001);
stomach->AddElement(elS,0.002);
stomach->AddElement(elCl,0.001);
stomach->AddElement(elK,0.002);


G4Material* wholeBreast=new G4Material("WholeBreast", density=1.020*g/cm3,
numberofElements=8);
wholeBreast->AddElement(elH,0.106);
wholeBreast->AddElement(elC,0.332);
wholeBreast->AddElement(elN,0.03);
wholeBreast->AddElement(elO,0.527);
wholeBreast->AddElement(elNa,0.001);
wholeBreast->AddElement(elP,0.001);
wholeBreast->AddElement(elS,0.002);
wholeBreast->AddElement(elCl,0.001);


G4Material* adrenal=new G4Material("Adrenal", density=1.030*g/cm3,
numberofElements=8);
adrenal->AddElement(elH,0.106);
adrenal->AddElement(elC,0.284);
adrenal->AddElement(elN,0.026);
adrenal->AddElement(elO,0.578);
adrenal->AddElement(elP,0.001);
adrenal->AddElement(elS,0.002);
adrenal->AddElement(elCl,0.002);
adrenal->AddElement(elK,0.001);


G4Material* gallBladder=new G4Material("GallBladder", density=1.030*g/cm3,
numberofElements=6);
gallBladder->AddElement(elH,0.108);
gallBladder->AddElement(elC,0.061);
gallBladder->AddElement(elN,0.001);
gallBladder->AddElement(elO,0.822);
gallBladder->AddElement(elNa,0.004);
gallBladder->AddElement(elCl,0.004);


G4Material* wholeHeart=new G4Material("WholeHeart", density=1.050*g/cm3,
numberofElements=9);
wholeHeart->AddElement(elH,0.104);
wholeHeart->AddElement(elC,0.139);
wholeHeart->AddElement(elN,0.029);
wholeHeart->AddElement(elO,0.718);
wholeHeart->AddElement(elNa,0.001);
wholeHeart->AddElement(elP,0.002);
wholeHeart->AddElement(elS,0.002);
```

```
wholeHeart->AddElement(elCl,0.002);
wholeHeart->AddElement(elK,0.003);


G4Material* pancreas=new G4Material("Pancreas", density=1.040*g/cm3,
numberofElements=9);
pancreas->AddElement(elH,0.116);
pancreas->AddElement(elC,0.179);
pancreas->AddElement(elN,0.002);
pancreas->AddElement(elO,0.694);
pancreas->AddElement(elNa,0.002);
pancreas->AddElement(elP,0.002);
pancreas->AddElement(elS,0.001);
pancreas->AddElement(elCl,0.002);
pancreas->AddElement(elK,0.002);


G4Material* spleen=new G4Material("Spleen", density=1.060*g/cm3,
numberofElements=9);
spleen->AddElement(elH,0.103);
spleen->AddElement(elC,0.113);
spleen->AddElement(elN,0.032);
spleen->AddElement(elO,0.741);
spleen->AddElement(elNa,0.001);
spleen->AddElement(elP,0.003);
spleen->AddElement(elS,0.002);
spleen->AddElement(elCl,0.002);
spleen->AddElement(elK,0.003);


G4Material* thyroid=new G4Material("Thyroid", density=1.050*g/cm3,
numberofElements=10);
thyroid->AddElement(elH,0.104);
thyroid->AddElement(elC,0.119);
thyroid->AddElement(elN,0.024);
thyroid->AddElement(elO,0.745);
thyroid->AddElement(elNa,0.002);
thyroid->AddElement(elP,0.001);
thyroid->AddElement(elS,0.001);
thyroid->AddElement(elCl,0.002);
thyroid->AddElement(elK,0.001);
thyroid->AddElement(elI,0.001);


G4Material* bladder=new G4Material("Bladder", density=1.040*g/cm3,
numberofElements=9);
bladder->AddElement(elH,0.105);
bladder->AddElement(elC,0.096);
bladder->AddElement(elN,0.026);
bladder->AddElement(elO,0.761);
bladder->AddElement(elNa,0.002);
bladder->AddElement(elP,0.002);
bladder->AddElement(elS,0.002);
bladder->AddElement(elCl,0.003);
bladder->AddElement(elK,0.003);
```

```
G4Material* kidney=new G4Material("Kidney", density=1.050*g/cm3,
numberofElements=10);
kidney->AddElement(elH,0.103);
kidney->AddElement(elC,0.132);
kidney->AddElement(elN,0.03);
kidney->AddElement(elO,0.724);
kidney->AddElement(elNa,0.002);
kidney->AddElement(elP,0.002);
kidney->AddElement(elS,0.002);
kidney->AddElement(elC,0.002);
kidney->AddElement(elK,0.002);
kidney->AddElement(elCa,0.001);


G4Material* ovary=new G4Material("Ovary", density=1.050*g/cm3,
numberofElements=9);
ovary->AddElement(elH,0.105);
ovary->AddElement(elC,0.093);
ovary->AddElement(elN,0.024);
ovary->AddElement(elO,0.768);
ovary->AddElement(elNa,0.002);
ovary->AddElement(elP,0.002);
ovary->AddElement(elS,0.002);
ovary->AddElement(elCl,0.002);
ovary->AddElement(elK,0.002);

G4Material* testis=new G4Material("Testis", density=1.040*g/cm3,
numberofElements=9);
testis->AddElement(elH,0.106);
testis->AddElement(elC,0.099);
testis->AddElement(elN,0.02);
testis->AddElement(elO,0.766);
testis->AddElement(elNa,0.002);
testis->AddElement(elP,0.001);
testis->AddElement(elS,0.002);
testis->AddElement(elCl,0.002);
testis->AddElement(elK,0.002);

G4Material* intestine=new G4Material("Intestine", density=1.030*g/cm3,
numberofElements=9);
intestine->AddElement(elH,0.106);
intestine->AddElement(elC,0.115);
intestine->AddElement(elN,0.022);
intestine->AddElement(elO,0.751);
intestine->AddElement(elNa,0.001);
intestine->AddElement(elP,0.001);
intestine->AddElement(elS,0.001);
intestine->AddElement(elCl,0.002);
intestine->AddElement(elK,0.001);


G4Material* liver=new G4Material("Liver", density=1.060*g/cm3,
numberofElements=9);
liver->AddElement(elH,0.102);
liver->AddElement(elC,0.139);
```

```
liver->AddElement(elN,0.030);
liver->AddElement(elO,0.716);
liver->AddElement(elNa,0.002);
liver->AddElement(elP,0.003);
liver->AddElement(elS,0.003);
liver->AddElement(elCl,0.002);
liver->AddElement(elK,0.003);

G4Material* redMarrow=new G4Material("RedMarrow", density=1.030*g/cm3,
numberofElements=9);
redMarrow->AddElement(elH,0.105);
redMarrow->AddElement(elC,0.414);
redMarrow->AddElement(elN,0.034);
redMarrow->AddElement(elO,0.439);
redMarrow->AddElement(elP,0.001);
redMarrow->AddElement(elS,0.002);
redMarrow->AddElement(elCl,0.002);
redMarrow->AddElement(elK,0.002);
redMarrow->AddElement(elFe,0.001);


G4Material* aorta=new G4Material("Aorta", density=1.050*g/cm3,
numberofElements=9);
aorta->AddElement(elH,0.099);
aorta->AddElement(elC,0.147);
aorta->AddElement(elN,0.042);
aorta->AddElement(elO,0.698);
aorta->AddElement(elP,0.004);
aorta->AddElement(elS,0.003);
aorta->AddElement(elK,0.001);
aorta->AddElement(elNa,0.002);
aorta->AddElement(elCa,0.004);
  //----- Put the materials in a vector

//*****************************************************
//The following list is for manually segmented phantoms
//*****************************************************
///*

  fOriginalMaterials.push_back(air);
  fOriginalMaterials.push_back(lungTissue);
  fOriginalMaterials.push_back(adiposeTissue);
  fOriginalMaterials.push_back(muscleTissue);
  fOriginalMaterials.push_back(glandularBreast);
  fOriginalMaterials.push_back(blood);
  fOriginalMaterials.push_back(hardBone);
  fOriginalMaterials.push_back(water);

//*/
//*****************************************************
//The following lis is for Helmholtz phantom Laura
//*****************************************************
/*
  fOriginalMaterials.push_back(air);
  fOriginalMaterials.push_back(hardBone);
  fOriginalMaterials.push_back(cartilage);
```

```
  fOriginalMaterials.push_back(skin);
  fOriginalMaterials.push_back(blood);
  fOriginalMaterials.push_back(muscleTissue);
  fOriginalMaterials.push_back(softTissue);
  fOriginalMaterials.push_back(adiposeTissue);
  fOriginalMaterials.push_back(lungTissue);
  fOriginalMaterials.push_back(Bodyair);
  fOriginalMaterials.push_back(water);
  fOriginalMaterials.push_back(connectiveTissue);
  fOriginalMaterials.push_back(eyeLens);
  fOriginalMaterials.push_back(trachea);
  fOriginalMaterials.push_back(stomach);
  fOriginalMaterials.push_back(wholeBreast);//for ella
 // fOriginalMaterials.push_back(testis);//for theo
  fOriginalMaterials.push_back(adrenal);
  fOriginalMaterials.push_back(gallBladder);
  fOriginalMaterials.push_back(wholeHeart);
  fOriginalMaterials.push_back(pancreas);
  fOriginalMaterials.push_back(spleen);
  fOriginalMaterials.push_back(thyroid);
  fOriginalMaterials.push_back(bladder);
  fOriginalMaterials.push_back(kidney);
  fOriginalMaterials.push_back(intestine);
  fOriginalMaterials.push_back(ovary);//for ella and billie
  fOriginalMaterials.push_back(aorta);
  fOriginalMaterials.push_back(liver);
  fOriginalMaterials.push_back(redMarrow);
*/
}

//------------------------------------------------------------
void DicomGeometry::ReadPatientData()
{
  std::ifstream finDF("Data.dat");
  G4String fname;
  if(finDF.good() != 1 ) {
    G4Exception(" DicomGeometry::ReadPatientData.  Problem reading data file:
Data.dat");
  }

  G4int compression, startFile;
  finDF >> compression; // not used here
  finDF >> startFile;
  finDF >> fNoFiles;

  for(G4int t=0; t<startFile; t++){
  finDF >> fname;
  }

  for(G4int i = startFile; i < fNoFiles; i++ ) {
    finDF >> fname;
    //--- Read one data file
    fname += ".g4";
    ReadPatientDataFile(fname);
  }
```

```
  //----- Merge data headers
  MergeZSliceHeaders();

  finDF.close();

}

//-------------------------------------------------------------
void DicomGeometry::ReadPatientDataFile(const G4String& fname)
{
#ifdef G4VERBOSE
  G4cout << " DicomGeometry::ReadPatientDataFile opening file " << fname <<
G4endl;
#endif
  std::ifstream fin(fname.c_str(), std::ios_base::in);
  if( !fin.is_open() ) {
    G4Exception("DicomGeometry::ReadPatientDataFil. File not found " + fname );
  }
  //----- Define density differences (maximum density difference to create a new
material)
  G4double densityDiff = 1.0;
  std::map<G4int,G4double>  fDensityDiffs; // to be able to use a different
densityDiff for each material
  for( unsigned int ii = 0; ii < fOriginalMaterials.size(); ii++ ){
    fDensityDiffs[ii] = densityDiff; //currently all materials with same
difference
  }

  //----- Read data header
  DicomPatientZSliceHeader* sliceHeader = new DicomPatientZSliceHeader( fin );
  fZSliceHeaders.push_back( sliceHeader );

  //----- Read material indices
  G4int nVoxels = sliceHeader->GetNoVoxels();

  //--- If first slice, initiliaze fMateIDs
  if( fZSliceHeaders.size() == 1 ) {
    //fMateIDs = new unsigned int[fNoFiles*nVoxels];
    fMateIDs = new size_t[fNoFiles*nVoxels];

  }

 // unsigned int mateID;
  G4int voxelCopyNo = (fZSliceHeaders.size()-1)*nVoxels; // number of voxels from
previously read slices
/*
 for( G4int ii = 0; ii < nVoxels; ii++, voxelCopyNo++ ){
    fin >> mateID;
    fMateIDs[voxelCopyNo] = mateID;
  }

  //----- Read material densities and build new materials if two voxels have same
material but its density is in a different density interval (size of density
intervals defined by densityDiff)
  G4double density;
  //G4int density;
```

```
  voxelCopyNo = (fZSliceHeaders.size()-1)*nVoxels; // number of voxels from
previously read slices
*/
unsigned int dens;
  for( G4int ii = 0; ii < nVoxels; ii++, voxelCopyNo++ ){
    fin >> dens;
    //G4cout<<"matl number: "<<dens<<G4endl;
    //-- Get material from list of original materials
    int mateID = fMateIDs[voxelCopyNo];
    G4Material* mateOrig  = fOriginalMaterials[dens];

    //-- Get density bin: middle point of the bin in which the current density is
included
    float densityBin = fDensityDiffs[mateID] *
(G4int(dens/fDensityDiffs[mateID])+0.5);
    //-- Build the new material name
    G4String newMateName = mateOrig->GetName()+"__"+ftoa(densityBin);
   G4double myDens = mateOrig->GetDensity();
    //-- Look if a material with this name is already created (because a previous
voxel was already in this density bin)
    unsigned int im;
    for( im = 0; im < fMaterials.size(); im++ ){
      if( fMaterials[im]->GetName() == newMateName ) {
       break;
      }
    }
    //-- If material is already created use index of this material
    if( im != fMaterials.size() ) {
      fMateIDs[voxelCopyNo] = im;
//-- else, create the material
    } else {
      fMaterials.push_back( BuildMaterialWithChangingDensity( mateOrig, myDens,
newMateName ) );
      fMateIDs[voxelCopyNo] = fMaterials.size()-1;
      //G4cout<<density<<G4endl;//me
    }
  }

}
```

```
//-------------------------------------------------------------
void DicomGeometry::ConstructPatientContainer()
{
///*
  //---- Extract number of voxels and voxel dimensions
  nVoxelX = fZSliceHeaderMerged->GetNoVoxelX();
  nVoxelY = fZSliceHeaderMerged->GetNoVoxelY();
```

```
  nVoxelZ = fZSliceHeaderMerged->GetNoVoxelZ();

  voxelHalfDimX = fZSliceHeaderMerged->GetVoxelHalfX();
  voxelHalfDimY = fZSliceHeaderMerged->GetVoxelHalfY();
  voxelHalfDimZ = fZSliceHeaderMerged->GetVoxelHalfZ();
#ifdef G4VERBOSE
  G4cout << " nVoxelX " << nVoxelX << " voxelHalfDimX " << voxelHalfDimX <<G4endl;
  G4cout << " nVoxelY " << nVoxelY << " voxelHalfDimY " << voxelHalfDimY <<G4endl;
  G4cout << " nVoxelZ " << nVoxelZ << " voxelHalfDimZ " << voxelHalfDimZ <<G4endl;
  G4cout << " totalPixels " << nVoxelX*nVoxelY*nVoxelZ <<  G4endl;
#endif

  //----- Define the volume that contains all the voxels
  container_solid = new
G4Box("PhantomContainer",nVoxelX*voxelHalfDimX,nVoxelY*voxelHalfDimY,nVoxelZ*voxel
HalfDimZ);
  container_logic =
    new G4LogicalVolume( container_solid,
                         fMaterials[0],  //the material is not important, it will be
fully filled by the voxels
                         "PhantomContainer",
                         0, 0, 0 );
//my stuff
G4VisAttributes* visualAttribs = new G4VisAttributes();
visualAttribs->SetForceSolid(false);
visualAttribs->SetColour(0.,0.,0.,0.);
container_logic->SetVisAttributes(visualAttribs);

//my stuff
  //--- Place it on the world
  G4double offsetX = (fZSliceHeaderMerged->GetMaxX() + fZSliceHeaderMerged-
>GetMinX() ) /2.;
  G4double offsetY = (fZSliceHeaderMerged->GetMaxY() + fZSliceHeaderMerged-
>GetMinY() ) /2.;
  G4double offsetZ = (fZSliceHeaderMerged->GetMaxZ() + fZSliceHeaderMerged-
>GetMinZ() ) /2.;

 //G4ThreeVector posCentreVoxels(offsetX,offsetY,offsetZ);
 G4ThreeVector posCentreVoxels(0.,0.,-1.0*cm);
//  G4ThreeVector posCentreVoxels(0.,0.,-46.*cm);
#ifdef G4VERBOSE
  G4cout << " placing voxel container volume at " << posCentreVoxels << G4endl;
#endif
G4RotationMatrix containerRot;

containerRot.rotateX(180.*deg+GlobalPhi*deg);
containerRot.rotateZ(viewAngle);// - viewAngle
//containerRot.rotateZ(225.*deg);
  container_phys =
    new G4PVPlacement(G4Transform3D(containerRot,  // rotation
                    posCentreVoxels),
                    container_logic,     // The logic volume
                    "PhantomContainer",  // Name
                    logicWorld,  // Mother
                    false,           // No op. bool.
                    1);              // Copy number
```

```
//*/
/*
G4RotationMatrix srcRotate;
srcRotate.rotateX(90.*deg);

G4Material* matl = fOriginalMaterials[11];
G4Cons *my_src = new G4Cons("mySource",0.25*cm, 0.5*cm,0.5*cm,8.25*cm,
3*cm,0,6.4);
G4LogicalVolume *logicalSource = new G4LogicalVolume(my_src, matl,
"logicalSource",0,0,0);
G4VPhysicalVolume *physicalSource = new G4PVPlacement(G4Transform3D(srcRotate,
G4ThreeVector(0.0*cm,54.0*cm,40.0*cm)),logicalSource,"physicalSource",logicWorld,f
alse,0); //54.0
*/
/*
G4Tubs* cyl=new G4Tubs("cyl",0.*cm,7.5*cm,3.9*m,0.*deg,360.*deg);
G4LogicalVolume* cyl_log= new
G4LogicalVolume(cyl,fOriginalMaterials[1],"cyl_log");
G4VPhysicalVolume* cyl_phys=new
G4PVPlacement(0,G4ThreeVector(0.,0.,0.),cyl_log,"cyl_phys",logicWorld,false,0);
*/
/*
G4RotationMatrix fieldRot;
fieldRot.rotateX(-90.*deg);

G4Trd *irradField = new G4Trd("irradField", 50.0*cm, 0.1*cm, 8.0*cm, 0.1*cm,
50.0*cm);

G4LogicalVolume * logicalField = new G4LogicalVolume(irradField, fMaterials[2],
"logicalField");

G4ThreeVector fieldPos(0.,0.,0.);

G4VPhysicalVolume * physicalField = new G4PVPlacement(G4Transform3D(fieldRot,
fieldPos), logicalField,"logicalField",logicWorld,false,0);
*/

/*
G4Box *solidDetector = new G4Box("solidDetector", 0.5*cm,3.5*m, 0.5*cm);
G4LogicalVolume *logicalDetector = new G4LogicalVolume(solidDetector,
fMaterials[1], "logicalDetector");
G4ThreeVector posi(0.,10.0*cm,0.);
G4VPhysicalVolume *physicalDetector=new
G4PVPlacement(0,posi,logicalDetector,"logicalDetector",logicWorld,false,0);
*/

}
///*
#include "G4SDManager.hh"
#include "G4MultiFunctionalDetector.hh"
#include "G4PSDoseDeposit_RegNav.hh"

//-------------------------------------------------------------
void DicomGeometry::SetScorer(G4LogicalVolume* voxel_logic)
{
```

```
  G4SDManager* SDman = G4SDManager::GetSDMpointer();
  //
  // Sensitive Detector Name
  G4String concreteSDname = "PatientSD";

  //------------------------
  // MultiFunctionalDetector
  //------------------------
  //
  // Define MultiFunctionalDetector with name.
  G4MultiFunctionalDetector* MFDet = new
G4MultiFunctionalDetector(concreteSDname);
  SDman->AddNewDetector( MFDet );                 // Register SD to SDManager

  voxel_logic->SetSensitiveDetector(MFDet);

  G4PSDoseDeposit_RegNav*   scorer = new G4PSDoseDeposit_RegNav("DoseDeposit");
  MFDet->RegisterPrimitive(scorer);

}
//*/




//-------------------------------------------------------------
void DicomGeometry::MergeZSliceHeaders()
{
  //----- Images must have the same dimension ...
  fZSliceHeaderMerged = new DicomPatientZSliceHeader( *fZSliceHeaders[0] );
  for( unsigned int ii = 1; ii < fZSliceHeaders.size(); ii++ ) {
    *fZSliceHeaderMerged += *fZSliceHeaders[ii];
  };
}

//-------------------------------------------------------------
G4Material* DicomGeometry::BuildMaterialWithChangingDensity( const G4Material*
origMate, G4double dens, G4String newMateName )
{
  //----- Copy original material, but with new density
  G4int nelem = origMate->GetNumberOfElements();

G4cout<<(dens/(g/cm3))<<" g/cm3   "<<newMateName<<G4endl;

  G4Material* mate = new G4Material( newMateName, dens, nelem, kStateUndefined,
STP_Temperature );

  for( G4int ii = 0; ii < nelem; ii++ ){
    G4double frac = origMate->GetFractionVector()[ii];
    G4Element* elem = const_cast<G4Element*>(origMate->GetElement(ii));
    mate->AddElement( elem, frac );
  }

  return mate;
}
```

```
//---------------------------------------------------------------------------
G4String DicomGeometry::ftoa(float flo)
{
  char ctmp[100];
  gcvt( flo, 10, ctmp );
  return G4String(ctmp);
}


G4VPhysicalVolume* DicomGeometry::Construct()
{
  InitialisationOfMaterials();
//STAT START START
//
//G4RotationMatrix* rm= new G4RotationMatrix;
//rm-> rotateX(-90*deg

       G4cout<<"Geometry Angle = "<<viewAngle/deg <<G4endl;
       G4cout<<"Phi Angle = "<<GlobalPhi<<G4endl;
       //solids

  G4double worldXDimension = 2.*m;
//  G4double worldXDimension = ((sliceThickness2*totalNumberOfFile2)+0.2)*m;
  G4double worldYDimension = 2.*m;
  G4double worldZDimension = 2.*m;

  solidWorld = new G4Box( "WorldSolid",
                          worldXDimension,
                          worldYDimension,
                          worldZDimension );

  logicWorld = new G4LogicalVolume( solidWorld,
                                    air,
                                    "WorldLogical",
                                    0, 0, 0 );
//My ADD
  logicWorld->SetVisAttributes(G4VisAttributes::Invisible);
  physiWorld = new G4PVPlacement( 0,
                                  G4ThreeVector(0,0,0),
                                  "World",
                                  logicWorld,
                                  0,
                                  false,
                                  0 );


ReadPatientData();
ConstructPatientContainer();
ConstructPatient();

  return physiWorld;
}
```

The description of the x-ray source, position, energy, and beam direction.

```
#ifndef LeiTestPrimaryGeneratorAction_h
#define LeiTestPrimaryGeneratorAction_h 1
#include "globals.hh"
#include "G4VUserPrimaryGeneratorAction.hh"
//#include "LeiTestRunAction.hh"

//class LeiTestDetectorConstruction;
class G4ParticleGun;
//class G4GeneralParticleSource;
class G4Event;
class LeiTestRunAction;

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class LeiTestPrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
  public:
    LeiTestPrimaryGeneratorAction();
    ~LeiTestPrimaryGeneratorAction();

  public:
    void GeneratePrimaries(G4Event* anEvent);

  private:
    G4ParticleGun* particleGun;
//G4GeneralParticleSource* particleGun;
   LeiTestRunAction* runManager;
   G4double sigmaX;
   G4double sigmaZ;
//G4double dir;
   //G4double dirSigma;
};

#endif
```

Description of the source and beam:

```
#ifndef LeiTestPrimaryGeneratorAction_h
#define LeiTestPrimaryGeneratorAction_h 1
#include "globals.hh"
#include "G4VUserPrimaryGeneratorAction.hh"
//#include "LeiTestRunAction.hh"

//class LeiTestDetectorConstruction;
class G4ParticleGun;
//class G4GeneralParticleSource;
class G4Event;
class LeiTestRunAction;

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class LeiTestPrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
  public:
    LeiTestPrimaryGeneratorAction();
    ~LeiTestPrimaryGeneratorAction();
```

```
  public:
    void GeneratePrimaries(G4Event* anEvent);

  private:
    G4ParticleGun* particleGun;
//G4GeneralParticleSource* particleGun;
   LeiTestRunAction* runManager;
   G4double sigmaX;
   G4double sigmaZ;
//G4double dir;
   //G4double dirSigma;
};

#endif


#include "LeiTestPrimaryGeneratorAction.hh"
#include "RegularDicomDetectorConstruction.hh"
#include "G4Event.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "globals.hh"
#include <math.h>
#include "G4ios.hh"
#include "Randomize.hh"
#include "CLHEP/Random/RandFlat.h"
#include "G4UImanager.hh"
#include "G4UIterminal.hh"
#include "G4UItcsh.hh"
#include "LeiTestDataSet.hh"
#include "LeiTestRunAction.hh"
#include <iostream>
#include <fstream>
#define PI 3.1415965


extern double GlobalAngle;
extern double GlobalPhi;

LeiTestPrimaryGeneratorAction::LeiTestPrimaryGeneratorAction()
{
//G4cout<<"prim gen start"<<G4endl;

  G4int n_particle = 1;

  particleGun = new G4ParticleGun(n_particle);

  runManager = new LeiTestRunAction();
}

LeiTestPrimaryGeneratorAction::~LeiTestPrimaryGeneratorAction()
{
  delete particleGun;
}
```

```
void LeiTestPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
//G4cout<<"gen primaries"<<G4endl;

/*
  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  G4String particleName;
  particleGun->SetParticleDefinition(particleTable-
>FindParticle(particleName="gamma"));
 particleGun->SetParticlePosition(G4ThreeVector(0.0*cm,54.0*cm,0.0*cm));
*/
 G4double sigmaX = 100.0*cm;//for scanner sim
 G4double sigmaZ = 16.0*cm;//for scanner sim

// G4double sigmaX = 80.0*cm;//for 15cm dia cyl dose
// G4double sigmaZ = 10.0*mm;//for cyl dose
 //Multiple numberOfEvent in
for (int p=0;p<100;p++)//fire off multiple particles for faster run
{
  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  G4String particleName;
  particleGun->SetParticleDefinition(particleTable-
>FindParticle(particleName="gamma"));
  particleGun->SetParticlePosition(G4ThreeVector(0.0*cm,54.0*cm,0.0*cm));
//Add Spectrum
const LeiTestDataSet* dataSet = runManager->GetGammaSet();
G4int i = 0;
G4int id = 0;
G4double minEnergy = 0.0*keV;
G4double maxEnergy = 123.0*keV;
G4double energyRange = maxEnergy - minEnergy;
G4double particleEnergy = 0.;

while (i ==0)
       {
       G4double random = G4UniformRand();
       G4double randomNum = G4UniformRand();
       particleEnergy = (random*energyRange) + minEnergy;
       if ((dataSet->FindValue(particleEnergy,id)) > randomNum)
              {
              i=1;
              }
       } //end of spectrum

G4double dirX=(G4UniformRand()-0.5)*sigmaX;
G4double dirZ=(G4UniformRand()-0.5)*sigmaZ;

  particleGun->SetParticleEnergy(particleEnergy);//particleEnergy
// G4cout<<"particle energy "<<particleEnergy/keV<<G4endl;
/*
std::ofstream fileout;
fileout.open("energies.txt",std::ios::out | std::ios::app);
fileout<<particleEnergy/keV<<G4endl;
fileout.close();
*/
```

```
  particleGun->SetParticleMomentumDirection(G4ThreeVector(dirX, -95.0*cm,
dirZ));//for scanner sim

//  particleGun->SetParticleMomentumDirection(G4ThreeVector(dirX, -
54.0*cm,dirZ));//for cyl dose

//G4cout<<"dirX "<<dirX/cm<<" dirZ "<<dirZ/cm<<G4endl;
//G4cout<<"sigX "<<sigmaX/cm<<" sigZ "<<sigmaZ/cm<<G4endl;
  particleGun->GeneratePrimaryVertex(anEvent);
}
}
```

The energy deposition scoring algorithm:

```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration.  It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license .  These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use.  Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
//
// $Id: G4PSDoseDeposit_RegNav.hh,v 1.2 2009/11/10 18:52:35 arce Exp $
// GEANT4 tag $Name: geant4-09-03-patch-01 $
//

#ifndef G4PSDoseDeposit_RegNav_h
#define G4PSDoseDeposit_RegNav_h 1

#include "G4VPrimitiveScorer.hh"
#include "G4THitsMap.hh"

////////////////////////////////////////////////////////////////////////////
// (Description)
//   Primitive scorer class for scoring dose deposit in the geometry volume.
//
// Created: 2005-11-14  Tsukasa ASO, Akinori Kimura.
```

```
//
////////////////////////////////////////////////////////////////////////
class G4PhantomParameterisation;
class G4EnergyLossForExtrapolator;
class G4VPhysicalVolume;

class G4PSDoseDeposit_RegNav : public G4VPrimitiveScorer
{
 public: // with description
      G4PSDoseDeposit_RegNav(G4String name, G4int depth=0);
      virtual ~G4PSDoseDeposit_RegNav();

  protected: // with description
      virtual G4bool ProcessHits(G4Step*,G4TouchableHistory*);

  public:
      virtual void Initialize(G4HCofThisEvent*);
      virtual void EndOfEvent(G4HCofThisEvent*);
      virtual void clear();
      virtual void DrawAll();
      virtual void PrintAll();

  void SetNIterations( G4int niter ){
    theNIterations = niter;
  }

  private:
      G4int HCID;
      G4THitsMap<G4double>* EvtMap;

  G4PhantomParameterisation* GetPhantomParam(G4bool mustExist);
  G4bool IsPhantomVolume( G4VPhysicalVolume* pv );

  G4PhantomParameterisation* thePhantomParam;
  G4EnergyLossForExtrapolator* theElossExt;

  G4int theNIterations;
};
#endif

//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration.  It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license .  These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use.  Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
```

```
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
//#define VERBOSE_DOSEDEP
//
// $Id: G4PSDoseDeposit_RegNav.cc,v 1.4 2009/12/16 17:54:21 gunter Exp $
// GEANT4 tag $Name: geant4-09-03-patch-01 $
//
// G4PSDoseDeposit_RegNav
#include "G4PSDoseDeposit_RegNav.hh"
#include "G4VSolid.hh"
#include "G4UnitsTable.hh"
#include "G4PhantomParameterisation.hh"
#include "G4RegularNavigationHelper.hh"
#include "G4EnergyLossForExtrapolator.hh"
#include "G4EmCalculator.hh"
#include "G4PhysicalVolumeStore.hh"
#include "G4PVParameterised.hh"
#include <fstream>

extern double GlobalAngle;
////////////////////////////////////////////////////////////////////////////
// (Description)
//   This is a primitive scorer class for scoring only energy deposit.
//
//
// Created: 2005-11-14  Tsukasa ASO, Akinori Kimura.
//
////////////////////////////////////////////////////////////////////////////

G4PSDoseDeposit_RegNav::G4PSDoseDeposit_RegNav(G4String name, G4int depth)
  :G4VPrimitiveScorer(name,depth),HCID(-1)
{
   theElossExt = new G4EnergyLossForExtrapolator(0);
   thePhantomParam = 0;
   theNIterations = 2;
}

G4PSDoseDeposit_RegNav::~G4PSDoseDeposit_RegNav()
{;}

G4bool G4PSDoseDeposit_RegNav::ProcessHits(G4Step* aStep,G4TouchableHistory*)
{
  G4double edep = aStep->GetTotalEnergyDeposit();
  edep=(edep/eV);
  if ( edep == 0. ) return FALSE;
  G4double volume  = aStep->GetPreStepPoint()->GetPhysicalVolume()
    ->GetLogicalVolume()->GetSolid()->GetCubicVolume();
  G4double density = aStep->GetTrack()->GetMaterial()->GetDensity();
```

```
G4String matty = aStep->GetPreStepPoint()->GetMaterial()->GetName();

int matID=0;
if(matty=="Lung__1.5"){
matID=1;}

if(matty=="Adipose__2.5"){
matID=2;}

if(matty=="Muscle__3.5"){
matID=3;}

if(matty=="glandularBreast__4.5"){
matID=4;}

if(matty=="Blood__5.5"){
matID=5;}

if(matty=="Bone__6.5"){
matID=6;}

if(matty=="Water__7.5"){
matID=7;}
if(matID==0) return FALSE;
double dub_matID=(double) (matID);
/*
if(matty=="Lung__1.5"){
matID=1;}
if(matty=="Water__1.5"){
std::ofstream fileout;
char fname[512];
int ang=(int) (GlobalAngle+0.5);
sprintf(fname, "dose_%d.out",ang);
fileout.open(fname, std::ios::out | std::ios::app);
fileout<<(edep/eV)<<G4endl;
fileout.close();
}
*/



#ifdef VERBOSE_DOSEDEP
  G4bool verbose = 1;
#endif

 if( aStep == 0 ) return FALSE; // it is 0 when called by GmScoringMgr after last
event

#ifdef VERBOSE_DOSEDEP
 if( verbose ) G4cout << "GmG4PSDoseDeposit::FillScorer totalEdepo " << aStep-
>GetTotalEnergyDeposit()
                      << " Nsteps " <<
G4RegularNavigationHelper::theStepLengths.size() << G4endl;
#endif
  //----- Do not distribute dose in voxels
 G4double dose  = edep / ( density * volume );
```

```
 G4double wei = aStep->GetPreStepPoint()->GetWeight();

  if( G4RegularNavigationHelper::theStepLengths.size() <= 1 ||
      aStep->GetTrack()->GetDefinition()->GetPDGCharge() == 0)  { // we are only
counting dose deposit
    dose *= wei;
    G4int  index = GetIndex(aStep);
    EvtMap->add(matID,edep);
#ifdef VERBOSE_DOSEDEP
    if( verbose) G4cout  << "GmG4PSDoseDeposit::FillScorer  RN: energy lost " <<
dose << " index " << index << G4endl;
#endif
  } else {
    //----- Distribute dose in voxels
    std::vector< std::pair<G4int,G4double> > rnsl =
G4RegularNavigationHelper::theStepLengths;
    //      G4double geomSL = rnsl[0].second;
    //      G4double SL = aStep->GetStepLength();
    if( !thePhantomParam ) thePhantomParam = GetPhantomParam(true);
    //      const G4Material* mate = thePhantomParam->GetMaterial( rnsl[0].first
);
    const G4ParticleDefinition* part = aStep->GetTrack()->GetDefinition();
    G4double kinEnergyPreOrig = aStep->GetPreStepPoint()->GetKineticEnergy();
    G4double kinEnergyPre = kinEnergyPreOrig;

    G4double stepLength = aStep->GetStepLength();
    G4double slSum = 0.;
    unsigned int ii;
    for( ii = 0; ii < rnsl.size(); ii++ ){
      G4double sl = rnsl[ii].second;
      slSum += sl;
#ifdef VERBOSE_DOSEDEP
      if(verbose) G4cout  << "GmG4PSDoseDeposit::FillScorer"<< ii << " RN: it\
er1 step length geom " << sl << G4endl;
#endif
    }

#ifdef VERBOSE_DOSEDEP
    if( verbose )
      G4cout << "GmG4PSDoseDeposit RN:  step length geom TOTAL " << slSum
             << " true TOTAL " << stepLength
             << " ratio " << stepLength/slSum
             << " Energy " << aStep->GetPreStepPoint()->GetKineticEnergy()
             << " Material " << aStep->GetPreStepPoint()->GetMaterial()->GetName()
             << " Number of geom steps " << rnsl.size() << G4endl;
#endif
    //----- No iterations to correct elost and msc, distribute dose according to
geometrical step length in each voxel
    if( theNIterations == 0 ) {
      for( unsigned int ii = 0; ii < rnsl.size(); ii++ ){
       G4int index = G4RegularNavigationHelper::theStepLengths[ii].first;
       G4double sl = G4RegularNavigationHelper::theStepLengths[ii].second;
       G4double doseStep = dose * sl/slSum; //divide dose along steps,
proportional to step lengthr
       G4double dosewei = doseStep*wei;
#ifdef VERBOSE_DOSEDEP
```

```
       if(verbose) G4cout  << "GmG4PSDoseDeposit::FillScorer"<< ii
                           << " dose " << dosewei
                           << " in " << index << G4endl;
#endif

       EvtMap->add(matID, edep );

      }
    } else { //  1 or more iterations demanded

#ifdef VERBOSE_DOSEDEP
      // print corrected energy at iteration 0
      if(verbose)  {
       G4double slSum = 0.;
       for( ii = 0; ii < rnsl.size(); ii++ ){
         G4double sl = rnsl[ii].second;
         slSum += sl;
       }

       for( ii = 0; ii < rnsl.size(); ii++ ){
         G4cout  << "GmG4PSDoseDeposit::FillScorer "<< ii
                 << " RN: iter0 corrected energy lost " << aStep-
>GetTotalEnergyDeposit()*rnsl[ii].second/slSum
                 << G4endl;
       }
      }
#endif
      G4double slRatio = stepLength/slSum;
#ifdef VERBOSE_DOSEDEP
      if(verbose) G4cout << "GmG4PSDoseDeposit::FillScorer  RN: iter" << iiter <<
" step ratio " << slRatio << G4endl;
#endif

      //--- energy at each interaction
      G4EmCalculator emcalc;
      G4double totalELost = 0.;
      std::vector<G4double> kinELost;
      std::vector<G4double> stepLengths;
      for( int iiter = 1; iiter <= theNIterations; iiter++ ) {
       //--- iter1: distribute true step length in each voxel: geom SL in each
voxel is multiplied by a constant so that the sum gives the total true step length
       if( iiter == 1 ) {
         for( ii = 0; ii < rnsl.size(); ii++ ){
           G4double sl = rnsl[ii].second;
           stepLengths.push_back( sl * slRatio );
#ifdef VERBOSE_DOSEDEP
           if(verbose) G4cout  << "GmG4PSDoseDeposit::FillScorer"<< ii << " RN:
iter" << iiter << " corrected step length " << sl*slRatio << G4endl;
#endif
         }

         for( ii = 0; ii < rnsl.size(); ii++ ){
           const G4Material* mate = thePhantomParam->GetMaterial( rnsl[ii].first
);
           G4double dEdx = 0.;
           if( kinEnergyPre > 0. ) {  //t check this
```

```
            dEdx = emcalc.GetDEDX(kinEnergyPre, part, mate);
          }
          G4double elost = stepLengths[ii] * dEdx;

#ifdef VERBOSE_DOSEDEP
          if(verbose) G4cout  << "GmG4PSDoseDeposit::FillScorer"<< ii << " RN:
iter1 energy lost "  << elost
                              << " energy at interaction " << kinEnergyPre
                              << " = stepLength " << stepLengths[ii]
                              << " * dEdx " << dEdx << G4endl;
#endif
          kinEnergyPre -= elost;
          kinELost.push_back( elost );
          totalELost += elost;
        }

      } else{
        //------ 2nd and other iterations
        //----- Get step lengths corrected by changing geom2true correction
        //-- Get ratios for each energy
        slSum = 0.;
        kinEnergyPre = kinEnergyPreOrig;
        for( ii = 0; ii < rnsl.size(); ii++ ){
          const G4Material* mate = thePhantomParam->GetMaterial( rnsl[ii].first
);
          stepLengths[ii] = theElossExt->TrueStepLength( kinEnergyPre,
rnsl[ii].second , mate, part );
          kinEnergyPre -= kinELost[ii];

#ifdef VERBOSE_DOSEDEP
          if(verbose) G4cout << "GmG4PSDoseDeposit::FillScorer" << ii
                              << " RN: iter" << iiter << " step length geom " <<
stepLengths[ii]
                              << " geom2true " << rnsl[ii].second / stepLengths[ii]
<< G4endl;
#endif

          slSum += stepLengths[ii];
        }

        //Correct step lengths so that they sum the total step length
        G4double slratio = aStep->GetStepLength()/slSum;
#ifdef VERBOSE_DOSEDEP
        if(verbose) G4cout << "GmG4PSDoseDeposit::FillScorer" << ii << " RN:
iter" << iiter << " step ratio " << slRatio << G4endl;
#endif
        for( ii = 0; ii < rnsl.size(); ii++ ){
          stepLengths[ii] *= slratio;
#ifdef VERBOSE_DOSEDEP
          if(verbose) G4cout  << "GmG4PSDoseDeposit::FillScorer"<< ii << " RN:
iter" << iiter << " corrected step length " << stepLengths[ii] << G4endl;
#endif
          }

        //---- Recalculate energy lost with this new step lengths
        G4double kinEnergyPre = aStep->GetPreStepPoint()->GetKineticEnergy();
```

```
          totalELost = 0.;
          for( ii = 0; ii < rnsl.size(); ii++ ){
            const G4Material* mate = thePhantomParam->GetMaterial( rnsl[ii].first
);
            G4double dEdx = 0.;
            if( kinEnergyPre > 0. ) {
              dEdx = emcalc.GetDEDX(kinEnergyPre, part, mate);
            }
            G4double elost = stepLengths[ii] * dEdx;
#ifdef VERBOSE_DOSEDEP
            if(verbose) G4cout  << "GmG4PSDoseDeposit::FillScorer"<< ii << " RN:
iter" << iiter << " energy lost " << elost
                                << " energy at interaction " << kinEnergyPre
                                << " = stepLength " << stepLengths[ii]
                                << " * dEdx " << dEdx << G4endl;
#endif
            kinEnergyPre -= elost;
            kinELost[ii] = elost;
            totalELost += elost;
          }

        }

        //correct energies so that they reproduce the real step energy lost
        G4double enerRatio = (aStep->GetTotalEnergyDeposit()/totalELost);

#ifdef VERBOSE_DOSEDEP
        if(verbose) G4cout  << "GmG4PSDoseDeposit::FillScorer"<< ii << " RN: iter"
<< iiter << " energy ratio " << enerRatio << G4endl;
#endif

#ifdef VERBOSE_DOSEDEP
        G4double elostTot = 0.;
#endif
        for( ii = 0; ii < kinELost.size(); ii++ ){
          kinELost[ii] *= enerRatio;
#ifdef VERBOSE_DOSEDEP
          elostTot += kinELost[ii];
          if(verbose) G4cout  << "GmG4PSDoseDeposit::FillScorer "<< ii << " RN:
iter" << iiter << " corrected energy lost " << kinELost[ii]
                              << " orig elost " << kinELost[ii]/enerRatio
                              << " energy before interaction " << kinEnergyPreOrig-
elostTot+kinELost[ii]
                              << " energy after interaction " << kinEnergyPreOrig-
elostTot
                              << G4endl;
#endif
        }
      }

      //---- Compute the dose (for N iterations)
      G4double dosesum = 0.;
      G4double volume  = aStep->GetPreStepPoint()->GetPhysicalVolume()
       ->GetLogicalVolume()->GetSolid()->GetCubicVolume();
      G4double density = aStep->GetTrack()->GetMaterial()->GetDensity();
```

```
      for( ii = 0; ii < kinELost.size(); ii++ ){
       G4double dose  = kinELost[ii] / ( density * volume );
       G4double dosewei = dose*wei;
       G4int index = rnsl[ii].first;
#ifdef VERBOSE_DOSEDEP
       if(verbose) G4cout  << "GmG4PSDoseDeposit::FillScorer"<< ii
                           << " dose " << dosewei
                           << " in " << index
                           << " RN: deposited energy " << kinELost[ii]*wei
                           << G4endl;
#endif

       EvtMap->add(matID, edep );
       dosesum += dosewei;

      }

#ifdef VERBOSE_DOSEDEP
      if(verbose) {
       //      G4cout << "DOSE-DOSESUM " << (dose-dosesum)/dose << " DOSE " <<
dose << " DOSESUM " << dosesum << G4endl;
       if( (dose-dosesum)/dose > 1.E-9 ) G4cout << ScoringVerb(debugVerb) <<
"GmG4PSDoseDeposit::FillScorer"<< "ERRORDOSE-DOSESUM " << (dose-dosesum)/dose << "
DOSE " << dose << " DOSESUM " << dosesum << G4endl;
      }
#endif
    }
  }

  return TRUE;
}

void G4PSDoseDeposit_RegNav::Initialize(G4HCofThisEvent* HCE)
{
  EvtMap = new G4THitsMap<G4double>(GetMultiFunctionalDetector()->GetName(),
                                    GetName());
  if(HCID < 0) {HCID = GetCollectionID(0);}
  HCE->AddHitsCollection(HCID, (G4VHitsCollection*)EvtMap);
}

void G4PSDoseDeposit_RegNav::EndOfEvent(G4HCofThisEvent*)
{;}

void G4PSDoseDeposit_RegNav::clear()
{
  EvtMap->clear();
}

void G4PSDoseDeposit_RegNav::DrawAll()
{;}

void G4PSDoseDeposit_RegNav::PrintAll()
{
  G4cout << " MultiFunctionalDet  " << detector->GetName() << G4endl;
  G4cout << " PrimitiveScorer " << GetName() << G4endl;
  G4cout << " Number of entries " << EvtMap->entries() << G4endl;
```

```
  std::map<G4int,G4double*>::iterator itr = EvtMap->GetMap()->begin();
  for(; itr != EvtMap->GetMap()->end(); itr++) {
    G4cout << "  copy no.: " << itr->first
          << "  dose deposit: " << G4BestUnit(*(itr->second),"Dose")
          << G4endl;
  }
}



//--------------------------------------------------------------------
G4PhantomParameterisation* G4PSDoseDeposit_RegNav::GetPhantomParam(G4bool
mustExist)
{
  G4PhantomParameterisation* paramreg = 0;

  G4PhysicalVolumeStore* pvs = G4PhysicalVolumeStore::GetInstance();
  std::vector<G4VPhysicalVolume*>::iterator cite;
  for( cite = pvs->begin(); cite != pvs->end(); cite++ ) {
    //    G4cout << " PV " << (*cite)->GetName() << " " << (*cite)-
>GetTranslation() << G4endl;
    if( IsPhantomVolume( *cite ) ) {
      const G4PVParameterised* pvparam = static_cast<const
G4PVParameterised*>(*cite);
      G4VPVParameterisation* param = pvparam->GetParameterisation();
      //    if( static_cast<const G4PhantomParameterisation*>(param) ){
      //    if( static_cast<const G4PhantomParameterisation*>(param) ){
      //      G4cout << "G4PhantomParameterisation volume found  " << (*cite)-
>GetName() << G4endl;
      paramreg = static_cast<G4PhantomParameterisation*>(param);
    }
  }

  if( !paramreg && mustExist ) G4Exception("GmRegularParamUtils::GetPhantomParam:
No G4PhantomParameterisation found ");

  return paramreg;

}


//--------------------------------------------------------------------
G4bool G4PSDoseDeposit_RegNav::IsPhantomVolume( G4VPhysicalVolume* pv )
{
  EAxis axis;
  G4int nReplicas;
  G4double width,offset;
  G4bool consuming;
  pv->GetReplicationData(axis,nReplicas,width,offset,consuming);
  EVolume type = (consuming) ? kReplica : kParameterised;
  if( type == kParameterised && pv->GetRegularStructureId() == 1 ) {
    return TRUE;
  } else {
    return FALSE;
  }
```

```
}
```

The octave script for cumulating the results from every projection angle:

```
tot=0;
out=load(['dose_',num2str(0),'.out']);
#33 is max number of materials in phantom
doses=zeros(33,2);
files=0;
doses(:,1)=1:33;

for t=0:359
#t=165;
    if(exist(['dose_',num2str(t),'.out'])>0)
    out=load(['dose_',num2str(t),'.out']);
    files=files+1;
    disp(files);
#disp(out);
    for p=1:size(out,1)
    #nrg=sum(out(:,2));
    #tot=tot+nrg;
    doses(out(p,1),2)=out(p,2)+doses(out(p,1),2);
    endfor
    clear out;

    endif
endfor

save -ascii DoseReport.txt doses

#joul=(tot*1.602E-19);
disp("Total number of files read:");
disp(files);
#disp("Total eV deposited:");
#disp(tot);
#disp("Total joules deposited:");
#disp(joul);
disp("Dose List");
for y=1:size(doses,1)
printf('Material %i : %10.2e eV\n',doses(y,1),doses(y,2));
endfor
exit;
```

The condor submission scripts for distribution of the computation and their called scripts:

```
#condor submission script for simulation submissions
Universe = vanilla
#Executable = /home/MARQNET/3935hoppem/geant4/bin/Linux-g++/geantino
Executable = example_script.sh
#Arguments: number of projections, projection number, gantry tilt angle
Arguments = 360 $(Process) 25
Log = logs/dor.log
Output = logs/dor$(Process).out
Error = logs/dor$(Process).error
getenv = True
should_transfer_files = YES
when_to_transfer_output = ON_EXIT_OR_EVICT
transfer_input_files = Edist.dat, ColourMap.dat, CT2Density.dat, Data.dat,
geantino_files.tgz
on_exit_remove = (ExitCode == 0)
RequestMemory=3000
RequestCpus=1
Queue 360
```

#example_script.sh for simulation submission

```
#!/bin/bash

# Where is our executable located?
EXECUTABLE=/home/geant4/bin/Linux-g++/G4RegNav
PROCESS_NUMBER=$2

# Output information
echo "Running on Host: $(hostname)"
echo "Current Time: $(date)"
echo "Preparing to Run: '$EXECUTABLE $@'"

# Make sure this is running as a condor executable!
if [ ! -f .machine.ad ]
then
       echo "ERROR: This script must be run as a Condor job." >&2
       exit 99
fi

# Make sure all our files exist
if [ ! -x $EXECUTABLE ]
then
       echo "ERROR: Unable to find $EXECUTABLE!" >&2
       exit 1
fi

if [ ! -f ColourMap.dat ]
then
       echo "ERROR: Unable to find Colormap.dat" >&2
       exit 2
fi

if [ ! -f CT2Density.dat ]
then
```

```
       echo "ERROR: Unable to find CT2Density.dat" >&2
       exit 3
fi

if [ ! -f Data.dat ]
then
       echo "ERROR: Unable to find Data.dat" >&2
       exit 4
fi

if [ ! -f geantino_files.tgz ]
then
       echo "ERROR: Unable to find geantino_files.tgz" >&2
       exit 5
fi

# Untar our input file (geantino_files.tgz)
tar xzvf geantino_files.tgz >/dev/null
if [ $? -ne 0 ]
then
       echo "ERROR: Unable to untar geantino_files.tgz." >&2
       echo "Abrupt end of tar ball!" >&2
       exit 6
fi

# Run the executable with the passed arguments
$EXECUTABLE $@
RC=$?
# The script exits with the return code of EXECUTABLE

# Clean up
rm -rf *.g4
rm -rf *.tgz *.tar.gz
rm -rf *.dat

ls

#if [ ! -f Laura_${PROCESS_NUMBER}.jpeg ]
#then
#      echo "ERROR: Unable to find Laura_${PROCESS_NUMBER}.jpeg" >&2
#      exit 7
#fi


#if [ ! -f dose_${PROCESS_NUMBER}.out ]
#then
#      echo "ERROR: Unable to find dose output">&2
#      exit 8
#fi
exit $RC
```

#condor submit script for dose summary report

```
Universe = vanilla
```

```
Executable = example_script3.sh
Log = logs/dor.log
Output = logs_nrg/nrg$(Process).out
Error = logs_nrg/nrg$(Process).error
getenv = True
should_transfer_files = YES
when_to_transfer_output = ON_EXIT_OR_EVICT
transfer_input_files = dose.tgz,nrgTally.m
#on_exit_remove = (ExitCode == 0)
Queue 1
```

#example_script3.sh

```
#!/bin/bash

# Where is our executable located?

# Output information
echo "Running on Host: $(hostname)"
echo "Current Time: $(date)"
tar -xzvf dose.tgz
octave nrgTally.m
# Make sure all our files exist

# Clean up
rm -rf *.out
rm -rf *.m
rm -rf *.tgz
rm -rf dose_*
exit
```

APPENDIX B

Appendix B lists GEANT4 C++ code, octave scripts, shell scripts, and Condor submit scripts used when performing a ray tracing simulation per material. Note that not all GEANT4 code for a simulation is listed, only poignant portions.

The GEANT4 main program:

```
//GEANT4 mail program
#include "DicomGeometry.hh"
#include "LeiTestPhysicsList.hh"
#include "LeiTestPrimaryGeneratorAction.hh"
#include "LeiTestEventAction.hh"
#include "LeiTestTrackingAction.hh"
#include "LeiTestSteppingAction.hh"
#include "LeiTestAnalysisManager.hh"
#include "Randomize.hh"

#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "G4UIterminal.hh"
#include "G4UItcsh.hh"
#include "G4VisExecutive.hh"
#include "LeiTestRunAction.hh"

#include <iostream>
#include <fstream>
#include <math.h>
#include <time.h>
#include "RegularDicomDetectorConstruction.hh"
#define PI 3.1415965

int angleNumber=0;
double GlobalAngle=0;
double GlobalPhi=0;
int matlNum=0;
int main(int argc, char** argv)
{
//**********
//**********
//WARNING: Parameters of the detector's dimensions and pixels (line 100)
//MUST match the description in src/LeiTestPrimaryGeneratorAction.cc
//**********
//**********
//The number of projections desired (scans phantoms' L to R)
       int numProj = std::atoi(argv[1]);
//Angular coverage of the scan plan
       G4double totalAngle=360.0*deg;
//Angular difference between projections
       G4double deltaPhi = totalAngle/(numProj);
//Projection number
       angleNumber = std::atoi(argv[2]);
//Gantry tilt angle (top tips toward phantoms' feet)
       int angleNumber2 = std::atoi(argv[3]);
//Number of materials in phantom
```

```
	int totalMatl= std::atof(argv[4]);
	totalMatl=totalMatl+1;

//Angle of projection and gantry tilt set
	G4double phi = angleNumber*deltaPhi;
	GlobalAngle=(phi/deg);
	 GlobalPhi=angleNumber2;

//Initialize random number generator
	time_t systyme=time(NULL);
	int GlobalAngleSeed= (int) (systyme*GlobalAngle+0.5);
	CLHEP::HepRandom::setTheEngine(new CLHEP::RanecuEngine);
	CLHEP::HepRandom::setTheSeed(GlobalAngleSeed);

// Run Manager
	G4RunManager* runManager = new G4RunManager;
//G4cout<<"run manager new"<<G4endl;
//set mandatory user initialization classes
	DicomGeometry* detector;
	 detector = new RegularDicomDetectorConstruction();
	runManager->SetUserInitialization(detector);
	runManager->SetUserInitialization(new LeiTestPhysicsList);

	//user action classes
	runManager->SetUserAction(new LeiTestPrimaryGeneratorAction());
	runManager->SetUserAction(new LeiTestRunAction);
	runManager->SetUserAction(new LeiTestEventAction);
//	runManager->SetUserAction(tracking_action);

G4UserSteppingAction* stepping_action = new LeiTestSteppingAction;
runManager->SetUserAction(stepping_action);

	G4UImanager* UI =G4UImanager::GetUIpointer();

	char buffer[512];
	sprintf(buffer, "/LeiTest/det/setViewAngle %f deg", GlobalAngle);
	UI->ApplyCommand(buffer);

UI->ApplyCommand("/random/setDirectoryName .");
UI->ApplyCommand("/random/setSavingFlag 1");
	//Initialize G4 Kernel
	runManager->Initialize();

G4cout<<"global angle = "<<GlobalAngle<<G4endl;
G4cout<<"global phi = "<<GlobalPhi<<G4endl;
///*


  UI->ApplyCommand("/run/verbose 0");
  UI->ApplyCommand("/event/verbose 0");
  UI->ApplyCommand("/tracking/verbose 0");

  //UI->ApplyCommand("/units/list");
//FOR CONE BEAM IMPLEMENTATION (Not Working)
//Set up detector parameters (SAME as src/LeiTestPrimaryGeneratorAction.cc!!!)
//G4double detectorX=100.0*cm;
```

```
//G4double detectorZ=2.0*cm;
//G4double detectorPixelsX=1024.0;
//G4double del=detectorX/detectorPixelsX;
//G4double detectorPixelsZ=floor(detectorZ/del);

//The number of axial slices in z
G4cout<<" zPixels "<<detectorPixelsZ<<G4endl;
///*
for (matlNum=1;matlNum<totalMatl;matlNum++){
//G4int numberOfEvent = 1024*detectorPixelsZ;//1024 per row
G4int numberOFEvent = 1024;
G4cout<<" Material Number : "<<matlNum<<G4endl;
runManager->BeamOn(numberOfEvent);
}
//*/

/*
//ADDING VISUALS
//The following code outputs a .jpeg of your geometry set-up
//The image is of the form geom_img_%projection(d)
//
G4VisManager* visManager = new G4VisExecutive;
visManager->Initialize();

char buffer4[512];
int gAng=(int) (GlobalAngle + 0.5);
sprintf(buffer4,"/vis/rayTracer/trace geom_img_%d.jpeg",gAng);
UI->ApplyCommand("/vis/open RayTracer"); //RAY
UI->ApplyCommand(buffer4);
UI->ApplyCommand("/vis/drawVolume");//RAY
//
delete visManager;
//END OF ADDING VISuals
*/
delete runManager;

return 0;
}
```

The creation of the world and geometry (physical model):

```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration.  It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license .  These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
```

```
// * regarding  this  software system or assume any liability for its *
// * use.  Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
// The code was written by :
//     *Louis Archambault louis.archambault@phy.ulaval.ca,
//      *Luc Beaulieu beaulieu@phy.ulaval.ca
//      +Vincent Hubert-Tremblay at tigre.2@sympatico.ca
//
//
// *Centre Hospitalier Universitaire de Quebec (CHUQ),
// Hotel-Dieu de Quebec, departement de Radio-oncologie
// 11 cote du palais. Quebec, QC, Canada, G1R 2J6
// tel (418) 525-4444 #6720
// fax (418) 691 5268
//
// + Université Laval, Québec (QC) Canada
//*********************************************************//
//*******************************************************
//
// DicomGeometry.hh :
//     - Start the building of the geometry
//     - Creation of the world and other "mother"(middle) volume
//     - Initialisation of patient geometry
//     - Initialisation of HeaD geometry
//     - Functions are in DicomGeometry.cc, PatientConstructor.cc
//*******************************************************

#ifndef DicomGeometry_h
#define DicomGeometry_h 1
#include "DicomPatientZSliceHeader.hh"
#include "globals.hh"
#include "G4VUserDetectorConstruction.hh"
#include <map>

//class DicomConfiguration;
//class DicomPatientConstructor;
class G4Material;
class G4LogicalVolume;
//class G4PhysicalVolume;
class G4Box;
class LeiTestDetectorMessenger;
class DicomGeometry : public G4VUserDetectorConstruction
{
public:
  DicomGeometry();
  ~DicomGeometry();
public:
```

```
  G4VPhysicalVolume* Construct();
  void SetViewAngle (G4double angle) {viewAngle=angle;};

private:
  G4double viewAngle;
  G4double sliceThickness2;
  G4double totalNumberOfFile2;
protected:
  void InitialisationOfMaterials();
  void ReadPatientData();
  void ReadPatientDataFile(const G4String& fname);
  void MergeZSliceHeaders();
  G4Material* BuildMaterialWithChangingDensity(const G4Material* origMate,
G4double density, G4String newMateName);
  G4String ftoa(float flo);
  void ConstructPatientContainer();
  virtual void ConstructPatient()=0;
  void SetScorer(G4LogicalVolume* voxel_logic);
  LeiTestDetectorMessenger* detectorMessenger;
//  DicomPatientConstructor* patientConstructor;

  //Materials ...
 /*
  G4Material* hardBone;
  G4Material* cartilage;
  G4Material* skin;
  G4Material* blood;
  G4Material* muscleTissue;
  G4Material* softTissue;
  G4Material* redBoneMarrow;
  G4Material* glandularBreast;
  G4Material* yellowBoneMarrow;
  G4Material* adiposeTissue;
  G4Material* lungTissue;
 // G4Material* air;
  G4Material* Bodyair;
  G4Material* Pb;
*/
//  G4Material* Pb;
// World ...
protected:
  G4Material* air;
  G4Box* solidWorld;
  G4Box* container_solid;
  G4LogicalVolume* container_logic;
  G4VPhysicalVolume* container_phys;
  G4int fNoFiles;
  std::vector<G4Material*> fOriginalMaterials;
  std::vector<G4Material*> fMaterials;
  G4LogicalVolume* logicWorld;
  G4VPhysicalVolume* physiWorld;
  size_t* fMateIDs;
  std::map<G4int,G4double> fDensityDiffs;
  std::vector<DicomPatientZSliceHeader*> fZSliceHeaders;
  DicomPatientZSliceHeader* fZSliceHeaderMerged;
```

```
  G4int nVoxelX, nVoxelY, nVoxelZ;
  G4double voxelHalfDimX,  voxelHalfDimY, voxelHalfDimZ;

};

#endif


//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration.  It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license .  These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use.  Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
// The code was written by :
//     *Louis Archambault louis.archambault@phy.ulaval.ca,
//      *Luc Beaulieu beaulieu@phy.ulaval.ca
//      +Vincent Hubert-Tremblay at tigre.2@sympatico.ca
//
//
// *Centre Hospitalier Universitaire de Quebec (CHUQ),
// Hotel-Dieu de Quebec, departement de Radio-oncologie
// 11 cote du palais. Quebec, QC, Canada, G1R 2J6
// tel (418) 525-4444 #6720
// fax (418) 691 5268
//
// + Université Laval, Québec (QC) Canada
//*******************************************************

#include "globals.hh"
#include "LeiTestDetectorSD.hh"
#include "G4MaterialTable.hh"
#include "G4ElementTable.hh"
#include "G4Box.hh"
#include "G4LogicalVolume.hh"
#include "G4VPhysicalVolume.hh"
#include "G4PVPlacement.hh"
```

```
#include "G4PVParameterised.hh"
#include "G4Material.hh"
#include "G4Element.hh"
#include "G4VisAttributes.hh"
#include "G4Colour.hh"
#include "G4ios.hh"
#include "G4ThreeVector.hh"
#include "G4SDManager.hh"
#include "G4UserLimits.hh"
#include "LeiTestDetectorMessenger.hh"
#include "DicomGeometry.hh"
#include "DicomPatientParameterisation.hh"
//#include "DicomPatientConstructor.hh"
#include "DicomConfiguration.hh"
#include "G4UnitsTable.hh"
#include "G4VUserDetectorConstruction.hh"
#include "G4MultiFunctionalDetector.hh"
#include "G4VPrimitiveScorer.hh"
#include "LeiTestPSEnergyDeposit.hh"
#include "G4Cons.hh"
#include "G4Tubs.hh"
#include "G4Trd.hh"
#include "G4Transform3D.hh"
#include "G4VisAttributes.hh"
#include <math.h>
#include "DicomPatientZSliceHeader.hh"
#define PI 3.14159265

extern double GlobalPhi;

DicomGeometry::DicomGeometry()
{detectorMessenger = new LeiTestDetectorMessenger(this);
//  patientConstructor = new DicomPatientConstructor();
  /*hardBone = 0;
  cartilage = 0;
  skin = 0;
  blood = 0;
  muscleTissue = 0;
  softTissue = 0;
  redBoneMarrow = 0;
  glandularBreast = 0;
  yellowBoneMarrow = 0;
  adiposeTissue = 0;
  lungTissue = 0;
  Bodyair = 0;
  air = 0;
  Pb = 0;
  solidWorld = 0;
  logicWorld = 0;
  physiWorld = 0;
*/
}

DicomGeometry::~DicomGeometry()
{delete detectorMessenger;
  /*
```

```
  delete air;
  delete Bodyair;
  delete hardBone;
  delete cartilage;
  delete skin;
  delete blood;
  delete muscleTissue;
  delete softTissue;
  delete redBoneMarrow;
  delete glandularBreast;
  delete yellowBoneMarrow;
  delete Pb;
  delete adiposeTissue;
  delete lungTissue;
  */
//delete patientConstructor;
G4cout<<"Geometry Deleted"<<G4endl;
}

void DicomGeometry::InitialisationOfMaterials()
{
  // Creating elements :
  G4double z, a, density;
  G4String name, symbol;

  G4Element* elC = new G4Element( name = "Carbon",
                                  symbol = "C",
                                  z = 6.0, a = 12.011 * g/mole );
  G4Element* elH = new G4Element( name = "Hydrogen",
                              symbol = "H",
                              z = 1.0, a = 1.008  * g/mole );
  G4Element* elN = new G4Element( name = "Nitrogen",
                              symbol = "N",
                              z = 7.0, a = 14.007 * g/mole );
  G4Element* elO = new G4Element( name = "Oxygen",
                                  symbol = "O",
                                  z = 8.0, a = 16.00  * g/mole );
  G4Element* elNa = new G4Element( name = "Sodium",
                                   symbol = "Na",
                                   z= 11.0, a = 22.98977* g/mole );
  G4Element* elS = new G4Element( name = "Sulfur",
                                  symbol = "S",
                                  z = 16.0,a = 32.065* g/mole );
  G4Element* elCl = new G4Element( name = "Chlorine",
                                   symbol = "Cl",
                                   z = 17.0, a = 35.453* g/mole );
  G4Element* elK = new G4Element( name = "Potassium",
                                  symbol = "K",
                                  z = 19.0, a = 30.0983* g/mole );
  G4Element* elP = new G4Element( name = "Phosphorus",
                                  symbol = "P",
                                  z = 30.0, a = 30.973976* g/mole );
  G4Element* elFe = new G4Element( name = "Iron",
                                   symbol = "Fe",
                                   z = 26, a = 56.845* g/mole );
  G4Element* elMg = new G4Element( name = "Magnesium",
```

```
                                    symbol = "Mg",
                                    z = 12.0, a = 24.3050* g/mole );
  G4Element* elCa = new G4Element( name="Calcium",
                                    symbol = "Ca",
                                    z = 20.0, a = 40.078* g/mole );
  G4Element* elAr = new G4Element( name="Argon",
                                    symbol = "Ar",
                                    z = 18, a = 39.948* g/mole );
  G4Element* elPb = new G4Element(name = "Lead", symbol = "Pb",
z=82,a=207.19*g/mole);


  G4Element* elI = new G4Element(name = "Iodine", symbol = "I",
z=53,a=126.9045*g/mole);

// Creating Materials :
  G4int numberofElements;

  // Hard Bone
  G4Material* hardBone = new G4Material( "Bone",
                              density = 1920*kg/m3,
                              numberofElements = 9 );
  hardBone->AddElement(elH,.033);
  hardBone->AddElement(elC,0.154);
  hardBone->AddElement(elN,0.041);
  hardBone->AddElement(elO,0.432);
  hardBone->AddElement(elNa,0.01);
  hardBone->AddElement(elP,0.103);
  hardBone->AddElement(elS,0.002);
  hardBone->AddElement(elCa,0.223);
  hardBone->AddElement(elMg,0.002);

// Cartilage
  G4Material* cartilage = new G4Material( "Cartilage",
                             density = 1100*kg/m3,
                             numberofElements = 8 );
  cartilage->AddElement(elH,0.096);
  cartilage->AddElement(elC,0.099);
  cartilage->AddElement(elN,0.022);
  cartilage->AddElement(elO,0.744);
  cartilage->AddElement(elNa,0.005);
  cartilage->AddElement(elP,0.022);
  cartilage->AddElement(elS,0.009);
  cartilage->AddElement(elCl,0.003);


  // Skin
  G4Material* skin = new G4Material( "Skin",
                          density = 1090*kg/m3,
                          numberofElements = 9);
  skin->AddElement(elH,0.1);
  skin->AddElement(elC,0.204);
  skin->AddElement(elN,0.042);
  skin->AddElement(elO,0.645);
  skin->AddElement(elNa,0.002);
  skin->AddElement(elP,0.001);
```

```
skin->AddElement(elS,0.002);
skin->AddElement(elCl,0.003);
skin->AddElement(elK,0.001);

// Blood
G4Material* blood = new G4Material( "Blood",
                         density = 1060*kg/m3,
                         numberofElements = 10 );
blood->AddElement(elH,0.102);
blood->AddElement(elC,0.112);
blood->AddElement(elN,0.03);
blood->AddElement(elO,0.746);
blood->AddElement(elNa,0.001);
blood->AddElement(elP,0.001);
blood->AddElement(elS,0.002);
blood->AddElement(elCl,0.003);
blood->AddElement(elK,0.002);
blood->AddElement(elFe,0.001);

// Muscle Tissue
G4Material* muscleTissue = new G4Material( "Muscle",
                          density = 1050*kg/m3,
                          numberofElements = 9 );
muscleTissue->AddElement(elH,0.102);
muscleTissue->AddElement(elC,0.142);
muscleTissue->AddElement(elN,0.034);
muscleTissue->AddElement(elO,0.711);
muscleTissue->AddElement(elNa,0.001);
muscleTissue->AddElement(elP,0.002);
muscleTissue->AddElement(elS,0.003);
muscleTissue->AddElement(elCl,0.001);
muscleTissue->AddElement(elK,0.004);

// Soft Tissue
G4Material* softTissue = new G4Material( "SoftTissue",
                         density = 1050*kg/m3,
                         numberofElements = 9 );
softTissue->AddElement(elH,0.104);
softTissue->AddElement(elC,0.139);
softTissue->AddElement(elN,0.029);
softTissue->AddElement(elO,0.718);
softTissue->AddElement(elNa,0.001);
softTissue->AddElement(elP,0.002);
softTissue->AddElement(elS,0.002);
softTissue->AddElement(elCl,0.002);
softTissue->AddElement(elK,0.003);

// Red Bone Marrow
G4Material* redBoneMarrow = new G4Material( "RedBoneMarrow",
                                density = 1030*kg/m3,
                                numberofElements = 9);
redBoneMarrow->AddElement(elH,0.106);
redBoneMarrow->AddElement(elC,0.415);
redBoneMarrow->AddElement(elN,0.034);
redBoneMarrow->AddElement(elO,0.439);
redBoneMarrow->AddElement(elNa,0.0);
```

```
  redBoneMarrow->AddElement(elS,0.002);
  redBoneMarrow->AddElement(elCl,0.002);
  redBoneMarrow->AddElement(elFe,0.001);
  redBoneMarrow->AddElement(elP,0.001);

// Breast (mammary gland)
  G4Material* glandularBreast = new G4Material( "glandularBreast",
                              density = 1020*kg/m3,
                              numberofElements = 8 );
  glandularBreast->AddElement(elH,0.106);
  glandularBreast->AddElement(elC,0.332);
  glandularBreast->AddElement(elN,0.03);
  glandularBreast->AddElement(elO,0.527);
  glandularBreast->AddElement(elNa,0.001);
  glandularBreast->AddElement(elP,0.001);
  glandularBreast->AddElement(elS,0.002);
  glandularBreast->AddElement(elCl,0.001);

  // Yellow Bone Marrow
  G4Material* yellowBoneMarrow = new G4Material( "YellowMarrow",
                              density = 980*kg/m3,
                              numberofElements = 7);
  yellowBoneMarrow->AddElement(elH,0.115);
  yellowBoneMarrow->AddElement(elC,0.644);
  yellowBoneMarrow->AddElement(elN,0.007);
  yellowBoneMarrow->AddElement(elO,0.231);
  yellowBoneMarrow->AddElement(elNa,0.001);
  yellowBoneMarrow->AddElement(elS,0.001);
  yellowBoneMarrow->AddElement(elCl,0.001);


//Adipose Tissue
  G4Material* adiposeTissue = new G4Material("Adipose", density = 950*kg/m3,
                            numberofElements = 7);
  adiposeTissue->AddElement(elH,0.114);
  adiposeTissue->AddElement(elC,0.598);
  adiposeTissue->AddElement(elN,0.007);
  adiposeTissue->AddElement(elO,0.278);
  adiposeTissue->AddElement(elNa,0.001);
  adiposeTissue->AddElement(elP,0.001);
  adiposeTissue->AddElement(elS,0.001);

//Lung Tissue
 G4Material* lungTissue = new G4Material("Lung", density = 260*kg/m3,
                            numberofElements = 9);
  lungTissue->AddElement(elH,0.103);
  lungTissue->AddElement(elC,0.105);
  lungTissue->AddElement(elN,0.031);
  lungTissue->AddElement(elO,0.749);
  lungTissue->AddElement(elNa,0.002);
  lungTissue->AddElement(elP,0.002);
  lungTissue->AddElement(elS,0.003);
  lungTissue->AddElement(elCl,0.003);
  lungTissue->AddElement(elK,0.002);

// Body Air
```

```
  G4Material* Bodyair = new G4Material( "BodyAir",
                        1.290*mg/cm3,
                        numberofElements = 4 );
  Bodyair->AddElement(elN, 0.755268);
  Bodyair->AddElement(elAr, 0.012827);
  Bodyair->AddElement(elC, 0.000124);
  Bodyair->AddElement(elO, 0.231781);

// Air
  air = new G4Material( "Air",
                        1.290*mg/cm3,
                        numberofElements = 4 );
  air->AddElement(elN, 0.755268);
  air->AddElement(elAr, 0.012827);
  air->AddElement(elC, 0.000124);
  air->AddElement(elO, 0.231781);


G4Material* lead  = new G4Material("Lead",
density=11.35*g/cm3,numberofElements=1);
lead->AddElement(elPb,1.0);

//FOR VIRTUAL FAMILY
///*
//G4Material* water=new G4Material("Water", density=11.34*g/cm3,
numberofElements=1);

G4Material* water=new G4Material("Water", density=1.0*g/cm3, numberofElements=2);
water->AddElement(elH,0.112);
water->AddElement(elO,0.888);
//water->AddElement(elPb,1.0);

G4Material* connectiveTissue=new G4Material("ConnectiveTissue",
density=1.120*g/cm3, numberofElements=7);
connectiveTissue->AddElement(elH,0.094);
connectiveTissue->AddElement(elC,0.208);
connectiveTissue->AddElement(elN,0.063);
connectiveTissue->AddElement(elO,0.624);
connectiveTissue->AddElement(elNa,0.006);
connectiveTissue->AddElement(elS,0.002);
connectiveTissue->AddElement(elCl,0.003);


G4Material* eyeLens=new G4Material("EyeLens", density=1.070*g/cm3,
numberofElements=8);
eyeLens->AddElement(elH,0.096);
eyeLens->AddElement(elC,0.195);
eyeLens->AddElement(elN,0.057);
eyeLens->AddElement(elO,0.646);
eyeLens->AddElement(elNa,0.001);
eyeLens->AddElement(elS,0.001);
eyeLens->AddElement(elP,0.003);
eyeLens->AddElement(elCl,0.001);
```

```
G4Material* trachea=new G4Material("Trachea", density=1.060*g/m3,
numberofElements=9);
trachea->AddElement(elH,0.101);
trachea->AddElement(elC,0.139);
trachea->AddElement(elN,0.033);
trachea->AddElement(elO,0.713);
trachea->AddElement(elNa,0.001);
trachea->AddElement(elP,0.004);
trachea->AddElement(elS,0.004);
trachea->AddElement(elCl,0.001);
trachea->AddElement(elK,0.004);


G4Material* stomach=new G4Material("Stomach", density=1.050*g/cm3,
numberofElements=9);
stomach->AddElement(elH,0.104);
stomach->AddElement(elC,0.139);
stomach->AddElement(elN,0.029);
stomach->AddElement(elO,0.721);
stomach->AddElement(elNa,0.001);
stomach->AddElement(elP,0.001);
stomach->AddElement(elS,0.002);
stomach->AddElement(elCl,0.001);
stomach->AddElement(elK,0.002);


G4Material* wholeBreast=new G4Material("WholeBreast", density=1.020*g/cm3,
numberofElements=8);
wholeBreast->AddElement(elH,0.106);
wholeBreast->AddElement(elC,0.332);
wholeBreast->AddElement(elN,0.03);
wholeBreast->AddElement(elO,0.527);
wholeBreast->AddElement(elNa,0.001);
wholeBreast->AddElement(elP,0.001);
wholeBreast->AddElement(elS,0.002);
wholeBreast->AddElement(elCl,0.001);


G4Material* adrenal=new G4Material("Adrenal", density=1.030*g/cm3,
numberofElements=8);
adrenal->AddElement(elH,0.106);
adrenal->AddElement(elC,0.284);
adrenal->AddElement(elN,0.026);
adrenal->AddElement(elO,0.578);
adrenal->AddElement(elP,0.001);
adrenal->AddElement(elS,0.002);
adrenal->AddElement(elCl,0.002);
adrenal->AddElement(elK,0.001);


G4Material* gallBladder=new G4Material("GallBladder", density=1.030*g/cm3,
numberofElements=6);
gallBladder->AddElement(elH,0.108);
gallBladder->AddElement(elC,0.061);
gallBladder->AddElement(elN,0.001);
gallBladder->AddElement(elO,0.822);
```

```
gallBladder->AddElement(elNa,0.004);
gallBladder->AddElement(elCl,0.004);


G4Material* wholeHeart=new G4Material("WholeHeart", density=1.050*g/cm3,
numberofElements=9);
wholeHeart->AddElement(elH,0.104);
wholeHeart->AddElement(elC,0.139);
wholeHeart->AddElement(elN,0.029);
wholeHeart->AddElement(elO,0.718);
wholeHeart->AddElement(elNa,0.001);
wholeHeart->AddElement(elP,0.002);
wholeHeart->AddElement(elS,0.002);
wholeHeart->AddElement(elCl,0.002);
wholeHeart->AddElement(elK,0.003);


G4Material* pancreas=new G4Material("Pancreas", density=1.040*g/cm3,
numberofElements=9);
pancreas->AddElement(elH,0.116);
pancreas->AddElement(elC,0.179);
pancreas->AddElement(elN,0.002);
pancreas->AddElement(elO,0.694);
pancreas->AddElement(elNa,0.002);
pancreas->AddElement(elP,0.002);
pancreas->AddElement(elS,0.001);
pancreas->AddElement(elCl,0.002);
pancreas->AddElement(elK,0.002);


G4Material* spleen=new G4Material("Spleen", density=1.060*g/cm3,
numberofElements=9);
spleen->AddElement(elH,0.103);
spleen->AddElement(elC,0.113);
spleen->AddElement(elN,0.032);
spleen->AddElement(elO,0.741);
spleen->AddElement(elNa,0.001);
spleen->AddElement(elP,0.003);
spleen->AddElement(elS,0.002);
spleen->AddElement(elCl,0.002);
spleen->AddElement(elK,0.003);


G4Material* thyroid=new G4Material("Thyroid", density=1.050*g/cm3,
numberofElements=10);
thyroid->AddElement(elH,0.104);
thyroid->AddElement(elC,0.119);
thyroid->AddElement(elN,0.024);
thyroid->AddElement(elO,0.745);
thyroid->AddElement(elNa,0.002);
thyroid->AddElement(elP,0.001);
thyroid->AddElement(elS,0.001);
thyroid->AddElement(elCl,0.002);
thyroid->AddElement(elK,0.001);
thyroid->AddElement(elI,0.001);
```

```
G4Material* bladder=new G4Material("Bladder", density=1.040*g/cm3,
numberofElements=9);
bladder->AddElement(elH,0.105);
bladder->AddElement(elC,0.096);
bladder->AddElement(elN,0.026);
bladder->AddElement(elO,0.761);
bladder->AddElement(elNa,0.002);
bladder->AddElement(elP,0.002);
bladder->AddElement(elS,0.002);
bladder->AddElement(elCl,0.003);
bladder->AddElement(elK,0.003);


G4Material* kidney=new G4Material("Kidney", density=1.050*g/cm3,
numberofElements=10);
kidney->AddElement(elH,0.103);
kidney->AddElement(elC,0.132);
kidney->AddElement(elN,0.03);
kidney->AddElement(elO,0.724);
kidney->AddElement(elNa,0.002);
kidney->AddElement(elP,0.002);
kidney->AddElement(elS,0.002);
kidney->AddElement(elC,0.002);
kidney->AddElement(elK,0.002);
kidney->AddElement(elCa,0.001);


G4Material* ovary=new G4Material("Ovary", density=1.050*g/cm3,
numberofElements=9);
ovary->AddElement(elH,0.105);
ovary->AddElement(elC,0.093);
ovary->AddElement(elN,0.024);
ovary->AddElement(elO,0.768);
ovary->AddElement(elNa,0.002);
ovary->AddElement(elP,0.002);
ovary->AddElement(elS,0.002);
ovary->AddElement(elCl,0.002);
ovary->AddElement(elK,0.002);

G4Material* testis=new G4Material("Testis", density=1.040*g/cm3,
numberofElements=9);
testis->AddElement(elH,0.106);
testis->AddElement(elC,0.099);
testis->AddElement(elN,0.02);
testis->AddElement(elO,0.766);
testis->AddElement(elNa,0.002);
testis->AddElement(elP,0.001);
testis->AddElement(elS,0.002);
testis->AddElement(elCl,0.002);
testis->AddElement(elK,0.002);

G4Material* intestine=new G4Material("Intestine", density=1.030*g/cm3,
numberofElements=9);
intestine->AddElement(elH,0.106);
intestine->AddElement(elC,0.115);
```

```
intestine->AddElement(elN,0.022);
intestine->AddElement(elO,0.751);
intestine->AddElement(elNa,0.001);
intestine->AddElement(elP,0.001);
intestine->AddElement(elS,0.001);
intestine->AddElement(elCl,0.002);
intestine->AddElement(elK,0.001);


G4Material* liver=new G4Material("Liver", density=1.060*g/cm3,
numberofElements=9);
liver->AddElement(elH,0.102);
liver->AddElement(elC,0.139);
liver->AddElement(elN,0.030);
liver->AddElement(elO,0.716);
liver->AddElement(elNa,0.002);
liver->AddElement(elP,0.003);
liver->AddElement(elS,0.003);
liver->AddElement(elCl,0.002);
liver->AddElement(elK,0.003);

G4Material* redMarrow=new G4Material("RedMarrow", density=1.030*g/cm3,
numberofElements=9);
redMarrow->AddElement(elH,0.105);
redMarrow->AddElement(elC,0.414);
redMarrow->AddElement(elN,0.034);
redMarrow->AddElement(elO,0.439);
redMarrow->AddElement(elP,0.001);
redMarrow->AddElement(elS,0.002);
redMarrow->AddElement(elCl,0.002);
redMarrow->AddElement(elK,0.002);
redMarrow->AddElement(elFe,0.001);


G4Material* aorta=new G4Material("Aorta", density=1.050*g/cm3,
numberofElements=9);
aorta->AddElement(elH,0.099);
aorta->AddElement(elC,0.147);
aorta->AddElement(elN,0.042);
aorta->AddElement(elO,0.698);
aorta->AddElement(elP,0.004);
aorta->AddElement(elS,0.003);
aorta->AddElement(elK,0.001);
aorta->AddElement(elNa,0.002);
aorta->AddElement(elCa,0.004);
  //----- Put the materials in a vector

//*******************************************************
//The following list is for manually segmented phantoms
//*******************************************************
///*

  fOriginalMaterials.push_back(air);
  fOriginalMaterials.push_back(lungTissue);
  fOriginalMaterials.push_back(adiposeTissue);
  fOriginalMaterials.push_back(muscleTissue);
```

```
  fOriginalMaterials.push_back(glandularBreast);
  fOriginalMaterials.push_back(blood);
  fOriginalMaterials.push_back(hardBone);
  fOriginalMaterials.push_back(water);
 // fOriginalMaterials.push_back(lead);
//*/
//*****************************************************
//The following lis is for Helmholtz phantom Laura
//*****************************************************
/*
  fOriginalMaterials.push_back(air);
  fOriginalMaterials.push_back(hardBone);
  fOriginalMaterials.push_back(cartilage);
  fOriginalMaterials.push_back(skin);
  fOriginalMaterials.push_back(blood);
  fOriginalMaterials.push_back(muscleTissue);
  fOriginalMaterials.push_back(softTissue);
  fOriginalMaterials.push_back(adiposeTissue);
  fOriginalMaterials.push_back(lungTissue);
  fOriginalMaterials.push_back(Bodyair);
  fOriginalMaterials.push_back(water);
  fOriginalMaterials.push_back(connectiveTissue);
  fOriginalMaterials.push_back(eyeLens);
  fOriginalMaterials.push_back(trachea);
  fOriginalMaterials.push_back(stomach);
  fOriginalMaterials.push_back(wholeBreast);//for ella
 // fOriginalMaterials.push_back(testis);//for theo
  fOriginalMaterials.push_back(adrenal);
  fOriginalMaterials.push_back(gallBladder);
  fOriginalMaterials.push_back(wholeHeart);
  fOriginalMaterials.push_back(pancreas);
  fOriginalMaterials.push_back(spleen);
  fOriginalMaterials.push_back(thyroid);
  fOriginalMaterials.push_back(bladder);
  fOriginalMaterials.push_back(kidney);
  fOriginalMaterials.push_back(intestine);
  fOriginalMaterials.push_back(ovary);//for ella and billie
  fOriginalMaterials.push_back(aorta);
  fOriginalMaterials.push_back(liver);
  fOriginalMaterials.push_back(redMarrow);
*/
}

//------------------------------------------------------------
void DicomGeometry::ReadPatientData()
{
  std::ifstream finDF("Data.dat");
  G4String fname;
  if(finDF.good() != 1 ) {
    G4Exception(" DicomGeometry::ReadPatientData.  Problem reading data file:
Data.dat");
  }

  G4int compression, startFile;
  finDF >> compression; // not used here
  finDF >> startFile;
```

```
  finDF >> fNoFiles;

  for(G4int t=0; t<startFile; t++){
  finDF >> fname;
  }

  for(G4int i = startFile; i < fNoFiles; i++ ) {
    finDF >> fname;
    //--- Read one data file
    fname += ".g4";
    ReadPatientDataFile(fname);
  }

  //----- Merge data headers
  MergeZSliceHeaders();

  finDF.close();

}

//-------------------------------------------------------------
void DicomGeometry::ReadPatientDataFile(const G4String& fname)
{
#ifdef G4VERBOSE
  G4cout << " DicomGeometry::ReadPatientDataFile opening file " << fname <<
G4endl;
#endif
  std::ifstream fin(fname.c_str(), std::ios_base::in);
  if( !fin.is_open() ) {
    G4Exception("DicomGeometry::ReadPatientDataFil. File not found " + fname );
  }
  //----- Define density differences (maximum density difference to create a new
material)
  G4double densityDiff = 1.0;
  std::map<G4int,G4double>  fDensityDiffs; // to be able to use a different
densityDiff for each material
  for( unsigned int ii = 0; ii < fOriginalMaterials.size(); ii++ ){
    fDensityDiffs[ii] = densityDiff; //currently all materials with same
difference
  }

  //----- Read data header
  DicomPatientZSliceHeader* sliceHeader = new DicomPatientZSliceHeader( fin );
  fZSliceHeaders.push_back( sliceHeader );

  //----- Read material indices
  G4int nVoxels = sliceHeader->GetNoVoxels();

  //--- If first slice, initiliaze fMateIDs
  if( fZSliceHeaders.size() == 1 ) {
    //fMateIDs = new unsigned int[fNoFiles*nVoxels];
    fMateIDs = new size_t[fNoFiles*nVoxels];

  }

 // unsigned int mateID;
```

```
  G4int voxelCopyNo = (fZSliceHeaders.size()-1)*nVoxels; // number of voxels from
previously read slices
/*
 for( G4int ii = 0; ii < nVoxels; ii++, voxelCopyNo++ ){
    fin >> mateID;
    fMateIDs[voxelCopyNo] = mateID;
  }

  //----- Read material densities and build new materials if two voxels have same
material but its density is in a different density interval (size of density
intervals defined by densityDiff)
  G4double density;
  //G4int density;
  voxelCopyNo = (fZSliceHeaders.size()-1)*nVoxels; // number of voxels from
previously read slices
*/
unsigned int dens;
  for( G4int ii = 0; ii < nVoxels; ii++, voxelCopyNo++ ){
    fin >> dens;
    //G4cout<<"matl number: "<<dens<<G4endl;
    //-- Get material from list of original materials
    int mateID = fMateIDs[voxelCopyNo];
    G4Material* mateOrig  = fOriginalMaterials[dens];

    //-- Get density bin: middle point of the bin in which the current density is
included
    float densityBin = fDensityDiffs[mateID] *
(G4int(dens/fDensityDiffs[mateID])+0.5);
    //-- Build the new material name
    G4String newMateName = mateOrig->GetName()+"__"+ftoa(densityBin);
   G4double myDens = mateOrig->GetDensity();
    //-- Look if a material with this name is already created (because a previous
voxel was already in this density bin)
    unsigned int im;
    for( im = 0; im < fMaterials.size(); im++ ){
      if( fMaterials[im]->GetName() == newMateName ) {
       break;
      }
    }
    //-- If material is already created use index of this material
    if( im != fMaterials.size() ) {
      fMateIDs[voxelCopyNo] = im;
//-- else, create the material
    } else {
      fMaterials.push_back( BuildMaterialWithChangingDensity( mateOrig, myDens,
newMateName ) );
      fMateIDs[voxelCopyNo] = fMaterials.size()-1;
      //G4cout<<density<<G4endl;//me
    }
  }

}
```

```
//-------------------------------------------------------------
void DicomGeometry::ConstructPatientContainer()
{
///*
  //---- Extract number of voxels and voxel dimensions
  nVoxelX = fZSliceHeaderMerged->GetNoVoxelX();
  nVoxelY = fZSliceHeaderMerged->GetNoVoxelY();
  nVoxelZ = fZSliceHeaderMerged->GetNoVoxelZ();

  voxelHalfDimX = fZSliceHeaderMerged->GetVoxelHalfX();
  voxelHalfDimY = fZSliceHeaderMerged->GetVoxelHalfY();
  voxelHalfDimZ = fZSliceHeaderMerged->GetVoxelHalfZ();
/*
#ifdef G4VERBOSE
  G4cout << " nVoxelX " << nVoxelX << " voxelHalfDimX " << voxelHalfDimX <<G4endl;
  G4cout << " nVoxelY " << nVoxelY << " voxelHalfDimY " << voxelHalfDimY <<G4endl;
  G4cout << " nVoxelZ " << nVoxelZ << " voxelHalfDimZ " << voxelHalfDimZ <<G4endl;
  G4cout << " totalPixels " << nVoxelX*nVoxelY*nVoxelZ <<  G4endl;
#endif
*/
  //----- Define the volume that contains all the voxels
  container_solid = new
G4Box("PhantomContainer",nVoxelX*voxelHalfDimX,nVoxelY*voxelHalfDimY,nVoxelZ*voxel
HalfDimZ);
  container_logic =
    new G4LogicalVolume( container_solid,
                         fMaterials[0],  //the material is not important, it will be
fully filled by the voxels
                         "PhantomContainer",
                         0, 0, 0 );
//my stuff
G4VisAttributes* visualAttribs = new G4VisAttributes();
visualAttribs->SetForceSolid(false);
visualAttribs->SetColour(0.,0.,0.,0.);
container_logic->SetVisAttributes(visualAttribs);

//my stuff
  //--- Place it on the world
  G4double offsetX = (fZSliceHeaderMerged->GetMaxX() + fZSliceHeaderMerged-
>GetMinX() ) /2.;
  G4double offsetY = (fZSliceHeaderMerged->GetMaxY() + fZSliceHeaderMerged-
>GetMinY() ) /2.;
  G4double offsetZ = (fZSliceHeaderMerged->GetMaxZ() + fZSliceHeaderMerged-
>GetMinZ() ) /2.;

 //G4ThreeVector posCentreVoxels(offsetX,offsetY,offsetZ);
 G4ThreeVector posCentreVoxels(0.,0.,-0.5*cm);
//  G4ThreeVector posCentreVoxels(0.,0.,-46.*cm);
#ifdef G4VERBOSE
  G4cout << " placing voxel container volume at " << posCentreVoxels << G4endl;
```

```
#endif
G4RotationMatrix containerRot;

//containerRot.rotateX(180.*deg+-1*GlobalPhi*deg);

containerRot.rotateX(180.*deg+GlobalPhi*deg);
containerRot.rotateZ(viewAngle);// - viewAngle
//containerRot.rotateZ(225.*deg);
  container_phys =
    new G4PVPlacement(G4Transform3D(containerRot,  // rotation
                    posCentreVoxels),
                    container_logic,     // The logic volume
                    "PhantomContainer",  // Name
                    logicWorld,  // Mother
                    false,           // No op. bool.
                    1);              // Copy number
//*/
/*
G4RotationMatrix srcRotate;
srcRotate.rotateX(90.*deg);

G4Material* matl = fOriginalMaterials[11];
G4Cons *my_src = new G4Cons("mySource",0.25*cm, 0.5*cm,0.5*cm,8.25*cm,
3*cm,0,6.4);
G4LogicalVolume *logicalSource = new G4LogicalVolume(my_src, matl,
"logicalSource",0,0,0);
G4VPhysicalVolume *physicalSource = new G4PVPlacement(G4Transform3D(srcRotate,
G4ThreeVector(0.0*cm,54.0*cm,40.0*cm)),logicalSource,"physicalSource",logicWorld,f
alse,0); //54.0
*/
/*
G4Tubs* cyl=new G4Tubs("cyl",0.*cm,7.5*cm,3.9*m,0.*deg,360.*deg);
G4LogicalVolume* cyl_log= new
G4LogicalVolume(cyl,fOriginalMaterials[1],"cyl_log");
G4VPhysicalVolume* cyl_phys=new
G4PVPlacement(0,G4ThreeVector(0.,0.,0.),cyl_log,"cyl_phys",logicWorld,false,0);
*/
/*
G4RotationMatrix fieldRot;
fieldRot.rotateX(-90.*deg);

G4Trd *irradField = new G4Trd("irradField", 50.0*cm, 0.1*cm, 8.0*cm, 0.1*cm,
95.0*cm);

G4LogicalVolume * logicalField = new G4LogicalVolume(irradField, fMaterials[2],
"logicalField");

G4ThreeVector fieldPos(0.,0.,0.);

G4VPhysicalVolume * physicalField = new G4PVPlacement(G4Transform3D(fieldRot,
fieldPos), logicalField,"logicalField",logicWorld,false,0);
*/

/*
G4Box *solidDetector = new G4Box("solidDetector", 40.0*cm, 1.0*cm, 1.0*cm);
```

```
G4LogicalVolume *logicalDetector = new G4LogicalVolume(solidDetector,
fMaterials[8], "logicalDetector");
G4ThreeVector posi(0.,-40.0*cm,0.);
G4VPhysicalVolume *physicalDetector=new
G4PVPlacement(0,posi,logicalDetector,"logicalDetector",logicWorld,false,0);
*/
G4cout<<"after detec"<<G4endl;
}
///*
#include "G4SDManager.hh"
#include "G4MultiFunctionalDetector.hh"
#include "G4PSDoseDeposit_RegNav.hh"

//-------------------------------------------------------------
void DicomGeometry::SetScorer(G4LogicalVolume* voxel_logic)
{

  G4SDManager* SDman = G4SDManager::GetSDMpointer();
  //
  // Sensitive Detector Name
  G4String concreteSDname = "PatientSD";

  //------------------------
  // MultiFunctionalDetector
  //------------------------
  //
  // Define MultiFunctionalDetector with name.
  G4MultiFunctionalDetector* MFDet = new
G4MultiFunctionalDetector(concreteSDname);
  SDman->AddNewDetector( MFDet );                 // Register SD to SDManager

  voxel_logic->SetSensitiveDetector(MFDet);

  G4PSDoseDeposit_RegNav*   scorer = new G4PSDoseDeposit_RegNav("DoseDeposit");
  MFDet->RegisterPrimitive(scorer);

}
//*/




//-------------------------------------------------------------
void DicomGeometry::MergeZSliceHeaders()
{
  //----- Images must have the same dimension ...
  fZSliceHeaderMerged = new DicomPatientZSliceHeader( *fZSliceHeaders[0] );
  for( unsigned int ii = 1; ii < fZSliceHeaders.size(); ii++ ) {
    *fZSliceHeaderMerged += *fZSliceHeaders[ii];
  };
}

//-------------------------------------------------------------
G4Material* DicomGeometry::BuildMaterialWithChangingDensity( const G4Material*
origMate, G4double dens, G4String newMateName )
{
```

```
  //----- Copy original material, but with new density
  G4int nelem = origMate->GetNumberOfElements();

G4cout<<(dens/(g/cm3))<<" g/cm3   "<<newMateName<<G4endl;

  G4Material* mate = new G4Material( newMateName, dens, nelem, kStateUndefined,
STP_Temperature );

  for( G4int ii = 0; ii < nelem; ii++ ){
    G4double frac = origMate->GetFractionVector()[ii];
    G4Element* elem = const_cast<G4Element*>(origMate->GetElement(ii));
    mate->AddElement( elem, frac );
  }

  return mate;
}

//--------------------------------------------------------------------------
G4String DicomGeometry::ftoa(float flo)
{
  char ctmp[100];
  gcvt( flo, 10, ctmp );
  return G4String(ctmp);
}


G4VPhysicalVolume* DicomGeometry::Construct()
{
  InitialisationOfMaterials();
//STAT START START
//
//G4RotationMatrix* rm= new G4RotationMatrix;
//rm-> rotateX(-90*deg

       G4cout<<"Geometry Angle = "<<viewAngle/deg <<G4endl;
       G4cout<<"Phi Angle = "<<GlobalPhi<<G4endl;
       //solids

  G4double worldXDimension = 2.*m;
//  G4double worldXDimension = ((sliceThickness2*totalNumberOfFile2)+0.2)*m;
  G4double worldYDimension = 2.*m;
  G4double worldZDimension = 2.*m;

  solidWorld = new G4Box( "WorldSolid",
                          worldXDimension,
                          worldYDimension,
                          worldZDimension );

  logicWorld = new G4LogicalVolume( solidWorld,
                                    air,
                                    "WorldLogical",
                                    0, 0, 0 );
//My ADD
  logicWorld->SetVisAttributes(G4VisAttributes::Invisible);
  physiWorld = new G4PVPlacement( 0,
                                  G4ThreeVector(0,0,0),
```

```
                                    "World",
                                    logicWorld,
                                    0,
                                    false,
                                    0 );

ReadPatientData();
ConstructPatientContainer();
ConstructPatient();

  return physiWorld;
}
```

Description of the source and beam:

```
#ifndef LeiTestPrimaryGeneratorAction_h
#define LeiTestPrimaryGeneratorAction_h 1
#include "globals.hh"
#include "G4VUserPrimaryGeneratorAction.hh"
//#include "LeiTestRunAction.hh"

//class LeiTestDetectorConstruction;
class G4ParticleGun;
//class G4GeneralParticleSource;
class G4Event;
class LeiTestRunAction;

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class LeiTestPrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
  public:
    LeiTestPrimaryGeneratorAction();
    ~LeiTestPrimaryGeneratorAction();

  public:
    void GeneratePrimaries(G4Event* anEvent);

  private:
    G4ParticleGun* particleGun;
//G4GeneralParticleSource* particleGun;
   LeiTestRunAction* runManager;
   G4double sigmaX;
   G4double sigmaZ;
//G4double dir;
   //G4double dirSigma;
};

#endif



#include "LeiTestPrimaryGeneratorAction.hh"
#include "RegularDicomDetectorConstruction.hh"
#include "G4Event.hh"
```

```
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "globals.hh"
#include <math.h>
#include "G4ios.hh"
#include "Randomize.hh"
#include "CLHEP/Random/RandFlat.h"
#include "G4UImanager.hh"
#include "G4UIterminal.hh"
#include "G4UItcsh.hh"
#include "LeiTestDataSet.hh"
#include "LeiTestRunAction.hh"
#include <iostream>
#include <fstream>
#define PI 3.1415965


extern double GlobalAngle;
extern double GlobalPhi;
extern double rayPixelsX=0.0;
LeiTestPrimaryGeneratorAction::LeiTestPrimaryGeneratorAction()
{
//G4cout<<"prim gen start"<<G4endl;

  G4int n_particle = 1;

  particleGun = new G4ParticleGun(n_particle);

  runManager = new LeiTestRunAction();
}

LeiTestPrimaryGeneratorAction::~LeiTestPrimaryGeneratorAction()
{
  delete particleGun;
}

void LeiTestPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
//Bin number is the
int  binNum=anEvent->GetEventID();


 G4double sigmaX = 100.0*cm;//for scanner sim 124cm
// G4double sigmaZ = 2.0*cm;//for scanner sim
 G4double detPixelsX = 1024.0;
 G4double rayPixelsX=4.0;//4 is 5, 2 would be 3. because middle ray.
// G4double sigmaX = 15.0*cm;//for 15cm dia cyl dose
// G4double sigmaZ = 10.0*mm;//for cyl dose

G4double del=sigmaX/detPixelsX;
//G4double detPixelsZ = floor(sigmaZ/del);

//G4double detExtentZ=-(detPixelsZ/2.0)*del;
//G4double zIncr=ceil(binNum/1024.0);
//G4cout<<"zIncr number "<<zIncr<<G4endl;
```

```
//G4double zdir=detExtentZ+zIncr*del;
G4double zdir=0.0;
//G4cout<<"z dir "<<zdir/cm<<G4endl;

G4double rayDel = del/rayPixelsX;
G4double detExtent=-(detPixelsX/2.0)*del;
//G4double rayExtent=-(rayPixelsX/2.0)*rayDel;
G4double xdir=detExtent+binNum*del;

//G4cout<<"xdir : "<<xdir/cm<<G4endl;
//G4cout<<"Generator bin Number : "<<binNum<<G4endl;
G4int endRay=(rayPixelsX/2)+1;
G4int startRay=-1*rayPixelsX/2;

  for (int p=startRay;p<endRay;p++)
  {
//G4cout<<"p value is "<<p<<G4endl;
  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  G4String particleName;
  particleGun->SetParticleDefinition(particleTable-
>FindParticle(particleName="geantino"));
  particleGun->SetParticlePosition(G4ThreeVector(0.0*cm,54.0*cm,0.0*cm));


G4double rayDir=xdir+p*rayDel;

//G4cout<<"raylet dir: "<<rayDir/cm<<G4endl;
  particleGun->SetParticleEnergy(60.0*keV);//particleEnergy
 //G4cout<<"particle energy "<<particleEnergy/keV<<G4endl;
  particleGun->SetParticleMomentumDirection(G4ThreeVector(rayDir, -95.0*cm,
zdir));//for scanner sim


//G4cout<<"dirX "<<xdir/cm<<" dirZ "<<zdir/cm<<G4endl;
//G4cout<<"sigX "<<sigmaX/cm<<" sigZ "<<sigmaZ/cm<<G4endl;
//G4cout<<"Before prim vertex"<<G4endl;
  particleGun->GeneratePrimaryVertex(anEvent);
//G4cout<<"Bin Number PG: "<<binNum<<G4endl;
  }
}
```

The ray tracing algorithm:

```
#ifndef LeiTestSteppingAction_h
#define LeiTestSteppingAction_h 1

#include "G4UserSteppingAction.hh"
#include "LeiTestTrackInformation.hh"
#include <vector>

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

class LeiTestSteppingAction : public G4UserSteppingAction
{
```

```
  public:
    LeiTestSteppingAction();
   ~LeiTestSteppingAction();
int tracker;
//std::vector<double> stepMap;

void UserSteppingAction(const G4Step*);
//std::vector<double> GetStepMap(){return stepMap;}
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif




#include "LeiTestSteppingAction.hh"
//#include "LeiTestTrackInformation.hh"
//#include "G4Track.hh"
#include "G4RunManager.hh"
#include "G4ios.hh"
#include "G4Event.hh"
#include "G4EventManager.hh"
#include "G4THitsMap.hh"
#include <vector>
extern double GlobalAngle;
//std::vector<double> stepMap(20480,0.0);
std::vector<double> stepMap(1024,0.0);
extern double rayPixelsX;
extern int matlNum;
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
//std::vector<double> LeiTestSteppingAction::stepMap;

LeiTestSteppingAction::LeiTestSteppingAction()
{tracker=0;
//     for(int y=0;y<1024;y++){
//     stepMap.push_back(0.0);
//     }
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

LeiTestSteppingAction::~LeiTestSteppingAction()
{ }

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void LeiTestSteppingAction::UserSteppingAction(const G4Step* aStep)
{
/*
if (aStep->GetTrack()->GetParentID()>1)
{
aStep->GetTrack()->SetTrackStatus(fStopAndKill);
}
```

```
*/
tracker=tracker+1;
//G4cout<<"Tracker number : "<<tracker<<G4endl;
int binNum = G4EventManager::GetEventManager()->GetConstCurrentEvent()-
>GetEventID();
//G4cout<<"start dose dep"<<G4endl;
//  G4double edep = aStep->GetTotalEnergyDeposit();
//  edep=(edep/eV);
//  if ( edep == 0. ) return FALSE;
//  G4double volume  = aStep->GetPreStepPoint()->GetPhysicalVolume()
//    ->GetLogicalVolume()->GetSolid()->GetCubicVolume();
//  G4double density = aStep->GetTrack()->GetMaterial()->GetDensity();
G4String matty = aStep->GetPreStepPoint()->GetMaterial()->GetName();

///*
       double mu=0.0;
       //Neglect Air
       if(matlNum==1 && matty=="Lung__1.5"){
       mu=0.0534;}

       if(matlNum==2 && matty=="Adipose__2.5"){
       mu=0.1875;}

       if(matlNum==3 && matty=="Muscle__3.5"){
       mu=0.2160;}

       if(matlNum==4 && matty=="glandularBreast__4.5"){
       mu=0.2046;}

       if(matlNum==5 && matty=="Blood__5.5"){
       mu=0.2180;}

       if(matlNum==6 && matty=="Bone__6.5"){
       mu=0.6044;}
       //Neglect Water Below
       //if(matty=="Water__7.5"){
       //mu=0.2059;}
//*/
if(mu>0.0){
       G4double dist=aStep->GetStepLength();
       dist=dist/cm;

       //G4cout<<"distance calculated :"<<distance<<G4endl;
       //G4cout<<"step length dist : "<<dist<<G4endl;

       //double signal=(dist*mu)/(rayPixelsX+1.0);

       double signal=dist/(rayPixelsX+1.0);
       stepMap[binNum]=stepMap[binNum]+signal;
       }
}
```

The condor submission scripts for distribution of the computation and their called scripts:

```
#condor submission script for ray tracing submissions
Universe = vanilla
Executable = example_script.sh
#Arguments: "total projections" "projection # (process)" "tilt angle" "# matls"
Arguments = 1000 $(Process) 0 6
Log = logs/dor.log
Output = logs/dor$(Process).out
Error = logs/dor$(Process).error
getenv = True
should_transfer_files = YES
when_to_transfer_output = ON_EXIT_OR_EVICT
#Input Files: "spectrum" "Colourmap for Vis" "Matl number map" "phantom files
info" "tarball of phantom slices"
transfer_input_files = Edist.dat, ColourMap.dat, CT2Density.dat, Data.dat,
geantino_files.tgz
on_exit_remove = (ExitCode == 0)
RequestMemory=3000
RequestCpus=1
#Distribute/run 1000 projections
Queue 1000
```

#example_script.sh for simulation submission

```
#!/bin/bash

# Where is our executable located?
EXECUTABLE=/home/MARQNET/3935hoppem/geant4/bin/Linux-g++/G4RegNav
PROCESS_NUMBER=$2

# Output information
echo "Running on Host: $(hostname)"
echo "Current Time: $(date)"
echo "Preparing to Run: '$EXECUTABLE $@'"

# Make sure this is running as a condor executable!
if [ ! -f .machine.ad ]
then
       echo "ERROR: This script must be run as a Condor job." >&2
       exit 99
fi

# Make sure all our files exist
if [ ! -x $EXECUTABLE ]
then
       echo "ERROR: Unable to find $EXECUTABLE!" >&2
       exit 1
fi
```

```
if [ ! -f ColourMap.dat ]
then
       echo "ERROR: Unable to find Colormap.dat" >&2
       exit 2
fi

if [ ! -f CT2Density.dat ]
then
       echo "ERROR: Unable to find CT2Density.dat" >&2
       exit 3
fi

if [ ! -f Data.dat ]
then
       echo "ERROR: Unable to find Data.dat" >&2
       exit 4
fi

if [ ! -f geantino_files.tgz ]
then
       echo "ERROR: Unable to find geantino_files.tgz" >&2
       exit 5
fi

# Untar our input file (geantino_files.tgz)
tar xzvf geantino_files.tgz >/dev/null
if [ $? -ne 0 ]
then
       echo "ERROR: Unable to untar geantino_files.tgz." >&2
       echo "Abrupt end of tar ball!" >&2
       exit 6
fi

# Run the executable with the passed arguments
$EXECUTABLE $@
RC=$?
# The script exits with the return code of EXECUTABLE

# Clean up
rm -rf *.g4
rm -rf *.tgz *.tar.gz
rm -rf *.dat

ls

#if [ ! -f Laura_${PROCESS_NUMBER}.jpeg ]
#then
#      echo "ERROR: Unable to find Laura_${PROCESS_NUMBER}.jpeg" >&2
#      exit 7
#fi


#if [ ! -f dose_${PROCESS_NUMBER}.out ]
#then
#      echo "ERROR: Unable to find dose output">&2
```

```
#      exit 8
#fi
exit $RC
```

#condor submit script for noise addition MATLAB scripts

```
Universe = vanilla
Executable = noise_script.sh
#Arguments: phantom number
Arguments = 117
Log = logs/dor.log
Output = logs_nrg/nrg$(Process).out
Error = logs_nrg/nrg$(Process).error
getenv = True
should_transfer_files = YES
when_to_transfer_output = ON_EXIT_OR_EVICT
#Input Files: matlab codes, attenuation coeffs, photon fluence,
#spectrum, spectrum weights, image files
transfer_input_files = make_noisy_files.m, make_polyE_data.m,
make_sq_poly.m,mus_six.mat, n_photons.mat, spec.mat, wts.mat, 117_var.tgz
#on_exit_remove = (ExitCode == 0)

Queue 1
```

#noise_script.sh

```
#!/bin/bash

# Where is our executable located?

# Output information
echo "Running on Host: $(hostname)"
echo "Current Time: $(date)"
echo "Phantom Number: $1"

tar -xzvf $1_var.tgz
#set up to handle images generated at multiple tilt angles
#for angle in 5 10 15 20 25
for angle in 0
do
tar -xzvf var_$1_$angle.tgz
octave make_noisy_files.m
# Make sure all our files exist
tar -czvf noisy_$1_$angle.tgz noisy_proj_*
# Clean up
rm var_$1_$angle.tgz
rm matl_*
rm noisy_proj_*
done
```

```
tar -czvf noisy_$1.tgz noisy_$1_*

rm noisy_$1_*
rm $1_var.tgz
rm *.m
exit
```

Octave scripts to add noise to ray tracing projections:

```
clear all;

mus=load('mus_six.mat');
phos=load('n_photons.mat');
spec=load('spec.mat');
wts=load('wts.mat');


    fid=fopen(['matl_',num2str(1),'_proj_',num2str(1),'.out'],"r");

[tmp,cnt]=fread(fid,"float32");
num_pixels=cnt;
sgnn=make_sq_poly(num_pixels,1000,6);
disp("made it");
sg=make_polyE_data(sgnn,spec.spec,wts.wts,phos.numb_phos,mus.mus_six);

    for c=1:size(sg,1)
        fid2=fopen(['noisy_proj_',num2str(c-1),'.out'],"w");
        fwrite(fid2,sg(c,:),"float32",0);
        fclose(fid2);
    endfor



function sq_poly=make_sq_poly(pixels,projections,matls)
disp("in sq");
sq_poly=zeros(projections,pixels,matls);
disp(projections);
disp(pixels);
disp(matls);

for b=1:matls
    for c=1:projections

    fid=fopen(['matl_',num2str(b),'_proj_',num2str(c-1),'.out'],"r");
        [tmp,cnt] = fread(fid,"float32");

        fclose(fid);
        if(cnt~=pixels)
        disp('ERROR: acquisition error, check detector size and
pixels');
            exit;
        endif
sq_poly(c,:,b)=tmp;
```

```
endfor
    endfor

endfunction



%takes line integrals through different materials and combines them
into noisy polyenergetic line integral projections
%INPUTS
sg_poly:    A 3D array containing the line integral sinograms from the
            different materials.
sg_poly(:,:,k) contains the sinogram from the kth material, with
            the detector pixels along the columns and the rows
representing
            the projection angles.
spectrum:   polyenergetic x-ray spectrum
weights:    for energy-integrating this vector should contain the
energies
            at which the spectrum is sampled.  For photon counting it
should
            contain a vector of ones with length equal to the number of
            energies in the spectrum.
num_photons: the number of photons incident on a single detector pixel
in one
            projection in the raw beam.
mus:        An M x E array (M=number of materials, E=number of
energies)
%           with the linear attenuation coefficient of the materialsi.
%           The ith row of mus contains the coefficients for the ith
material
%           in sg_poly (sg_poly(:,:,i)).  The columns correspond to the
%           energies at which the spectrum is sampled.
%
% OUTPUT
% sg:       A sinogram containing the combined polyenergetic sinogram
with
%           poisson noise and after log normalization.
%By T.G.S., Marquette University, Milwaukee, WI

function sg = make_polyE_data(sg_poly, spectrum, weights, num_photons,
mus)

num_proj = size(sg_poly, 1);
num_dets = size(sg_poly,2);
spectrum=spectrum/sum(spectrum);
num_photonsE=spectrum*num_photons;
rand('twister', sum(100*clock));
sg=zeros(num_proj, num_dets);
mu_array = ones(size(sg,1), size(sg,2), size(mus,1));
proj_ff = zeros(size(sg));
proj_nn = zeros(size(sg));


  for(e=1:length(spectrum))
```

```
    disp(e)
    for(m=1:size(mus,1))
       mu_array(:,:,m) = mus(m,e);
    end
    li_proj = sum(mu_array.*sg_poly, 3);
    proj=num_photonsE(e)*exp(-li_proj);
    proj_ff=proj_ff+weights(e)*num_photonsE(e);
    %proj_nn = proj_nn+weights(e)*(proj);
    proj_nn = proj_nn+weights(e)*poissrnd(proj);
  endfor
  sg=-log(proj_nn./proj_ff);
  %sg=proj_nn;

endfunction
```