The Generation of a Digital Phantom for Testing of Digitally Reconstructed Radiographs

by

Nicholas Andrew Mason

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Engineering Science Department of Electrical Engineering College of Engineering University of South Florida

Co-Major Professor: Stanley R. Deans, Ph.D. Co-Major Professor: Kent Larsen, Ph.D. Luis Garcia-Rubio, Ph.D. Kenneth Buckle, Ph.D. James Pearlman, M.D.

Date of Approval: October 11, 2004

Keywords: Quality assurance, 3D treatment planning, CT, CT Simulation, Virtual simulation.

© Copyright 2004, Nicholas A. Mason
Dedication

I dedicate this work to my family for enduring with me through the years. I originally started on my Ph.D. before I met my wife. She has endured through a change of Universities, a change of topic, and still has the patience to provide all the love and support a husband needs.
Acknowledgments

I would like to acknowledge the help and drive from all of the people who have assisted me throughout this work. This includes Scot Hogan, who developed the main product on which this test platform was based, and the many physicists who I have had discussions with regarding this project.

I would like to thank my mentor, Jack Cunningham, who helped me get a start in this field. His patience, guidance and continued interest in my career has been inspiring. He truly defines what a mentor is.
# Table of Contents

List of Tables................................................................................................................. .iv

List of Figures ................................................................................................................ vi

Abstract........................................................................................................................viii

Introduction................................................................................................................... ..1

Radiation Oncology .............................................................................................1

The Cancer Treatment Process .............................................................................2

Port Films...................................................................................................................3

CT Simulation..............................................................................................................3

Digitally Reconstructed Radiographs ........................................................................4

CT Slices .....................................................................................................................5

Pixel .............................................................................................................................6

Electron Density .........................................................................................................6

Hounsfield Units ........................................................................................................7

Voxel ............................................................................................................................8

3D Virtual Patient ......................................................................................................8

Volume of Data ..........................................................................................................9

DRR’s .........................................................................................................................9

Physics of a DRR ..........................................................................................................9

Digital Phantom .........................................................................................................10

Materials and Methods ..................................................................................................13

Introduction ...............................................................................................................13

Generating Test Images ..........................................................................................14

Image Boundary .......................................................................................................14

Phantom Limits ..........................................................................................................14

Electron Density/Pixel Value ...................................................................................14

Inserting Objects into the Images ............................................................................15

Export Images ............................................................................................................16

DICOM Format ........................................................................................................17

Development Platform ..........................................................................................18

QwikSIM ....................................................................................................................18

Geometry ...................................................................................................................18

Perform Test .............................................................................................................20

Evaluate Results .......................................................................................................20
QwikSIM 1.2f ........................................................................................61
Systems Re-Tested .................................................................................64
QwikSIM 2.0 .........................................................................................65
Divergence Test ..............................................................................65
Position Test ................................................................................66
Density Test ................................................................................66
Theraplan Plus ................................................................................67
Divergence Test ..............................................................................67
Position Test ................................................................................68
Density Test ................................................................................69
Eclipse ..............................................................................................69
Divergence Test ..............................................................................70
Position Test ................................................................................72
Density Test ................................................................................72

Conclusions .............................................................................................75
Comparison of the Digital and Van Dyk Phantoms .........................76
Other Uses ........................................................................................77
Future Work ......................................................................................77

Reference .............................................................................................79

Bibliography ........................................................................................81

Appendices ..........................................................................................82
Appendix A: Test Phantom Coordinates ...........................................83
Appendix B: Program Source Code ....................................................95
Appendix C: Formulation of Scalar Values .........................................110

About the Author ....................................................................................End Page
List of Tables

Table 1. Test Geometry ........................................................................................20
Table 2. Standard Line Start and End Coordinates ........................................45
Table 3. Initial Test Geometries ........................................................................45
Table 4. Additional Tests ..................................................................................46
Table 5. Beam Entry Coordinates ...................................................................46
Table 6. Test Geometry .....................................................................................50
Table 7. Minimum Detectable Angle, Observer Results ....................................55
Table 8. QwikSIM 1.2f Geometry Results .......................................................62
Table 9. Systems to be Tested ..........................................................................65
Table 10. QwikSIM 2.0 Geometry Results .......................................................65
Table 11. QwikSIM 2.0 Position Results ...........................................................66
Table 12. QwikSIM 2.0 Density Results ............................................................66
Table 13. Theraplan Plus Geometry Results ....................................................67
Table 14. Theraplan Plus Position Results .......................................................68
Table 15. Theraplan Plus Density Results .......................................................69
Table 16. Eclipse Geometry Results ..................................................................70
Table 17. Eclipse Position Results ....................................................................72
Table 18. Eclipse Density Results .....................................................................72
Table 19. Test 1, Gantry 0, Couch 0 .................................................................83
Table 20. Test 2, Gantry 45, Couch 0 .................................................................84
Table 21. Test 3, Gantry 90, Couch 0 .................................................................85
Table 22. Test 4, Gantry 0, Couch 90 .................................................................86
Table 23. Test 5, Gantry 0, Couch 45 .................................................................87
Table 24. Test 6, Gantry 45, Couch 45 .................................................................88
Table 25. Test 7, Gantry 90, Couch 90 .................................................................89
Table 26. Test 8, Gantry 88, Couch 0 .................................................................90
Table 27. Test 9, Gantry 0, Couch 88 .................................................................91
Table 28. Test 10, Gantry 88, Couch 88 ..............................................................92
Table 29. Test 11, Van Dyk Phantom, Gantry 0, Couch 0 .................................93
Table 30. CT Number for the Van Dyk Density Objects ......................................94
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.</td>
<td>Divergent Line Definition</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2.</td>
<td>Sample CT Image of a Head</td>
<td>6</td>
</tr>
<tr>
<td>Figure 3.</td>
<td>Example of a Virtual Patient</td>
<td>8</td>
</tr>
<tr>
<td>Figure 4.</td>
<td>DRR Geometry</td>
<td>10</td>
</tr>
<tr>
<td>Figure 5.</td>
<td>Sample Multi Planar Reconstructed Images</td>
<td>12</td>
</tr>
<tr>
<td>Figure 6.</td>
<td>Image Components</td>
<td>16</td>
</tr>
<tr>
<td>Figure 7.</td>
<td>Simulated Rotating Gantry with Coordinate System</td>
<td>19</td>
</tr>
<tr>
<td>Figure 8.</td>
<td>Axes Definition</td>
<td>19</td>
</tr>
<tr>
<td>Figure 9.</td>
<td>Test Generation Procedure</td>
<td>21</td>
</tr>
<tr>
<td>Figure 10.</td>
<td>Sample Image Volume</td>
<td>22</td>
</tr>
<tr>
<td>Figure 11.</td>
<td>Image Parameter Dialog Box</td>
<td>23</td>
</tr>
<tr>
<td>Figure 12.</td>
<td>Dialog Box to Add Objects</td>
<td>24</td>
</tr>
<tr>
<td>Figure 13.</td>
<td>Internal Object Geometry</td>
<td>25</td>
</tr>
<tr>
<td>Figure 14.</td>
<td>Internal Coordinate System</td>
<td>29</td>
</tr>
<tr>
<td>Figure 15.</td>
<td>Line Entry Dialog Box</td>
<td>30</td>
</tr>
<tr>
<td>Figure 16.</td>
<td>Vector Definition</td>
<td>31</td>
</tr>
<tr>
<td>Figure 17.</td>
<td>Map 3D Vector to 2D Planes</td>
<td>32</td>
</tr>
<tr>
<td>Figure 18.</td>
<td>Line Square Intersection Tests</td>
<td>33</td>
</tr>
<tr>
<td>Figure 19.</td>
<td>Line Line Intersection Tests</td>
<td>34</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Figure 20.</td>
<td>Different Line Square Intersection Scenario’s</td>
<td>35</td>
</tr>
<tr>
<td>Figure 21.</td>
<td>Intersection of Vectors</td>
<td>37</td>
</tr>
<tr>
<td>Figure 22.</td>
<td>Scalar Calculation</td>
<td>38</td>
</tr>
<tr>
<td>Figure 23.</td>
<td>Special Case Intersections</td>
<td>40</td>
</tr>
<tr>
<td>Figure 24.</td>
<td>2D Conceptual Layout</td>
<td>42</td>
</tr>
<tr>
<td>Figure 25.</td>
<td>Beam Outline Layout</td>
<td>43</td>
</tr>
<tr>
<td>Figure 26.</td>
<td>Standard Test Geometry</td>
<td>44</td>
</tr>
<tr>
<td>Figure 27.</td>
<td>Van Dyk Phantom Photograph</td>
<td>48</td>
</tr>
<tr>
<td>Figure 28.</td>
<td>Van Dyk Phantom Diagram</td>
<td>48</td>
</tr>
<tr>
<td>Figure 29.</td>
<td>Expected DRR Image</td>
<td>52</td>
</tr>
<tr>
<td>Figure 30.</td>
<td>Expected Van Dyk Phantom DRR Image</td>
<td>53</td>
</tr>
<tr>
<td>Figure 31.</td>
<td>Expected DRR Image</td>
<td>59</td>
</tr>
<tr>
<td>Figure 32.</td>
<td>DRR with an Introduced 1.0 Degree Rotational Error</td>
<td>60</td>
</tr>
<tr>
<td>Figure 33.</td>
<td>DRR with an Introduced 1.0 Degree Rotational Error with No Axes</td>
<td>61</td>
</tr>
<tr>
<td>Figure 34.</td>
<td>QwikSIM 1.2f, DRR Gantry 88 Deg, Couch 88 Deg.</td>
<td>63</td>
</tr>
<tr>
<td>Figure 35.</td>
<td>Theraplan Plus DRR, Gantry 88, Couch 88</td>
<td>68</td>
</tr>
<tr>
<td>Figure 36.</td>
<td>Eclipse DRR, Gantry 88, Couch 88</td>
<td>71</td>
</tr>
<tr>
<td>Figure 37.</td>
<td>Eclipse DRR, Van Dyk Phantom, Gantry 88, Couch 88</td>
<td>73</td>
</tr>
<tr>
<td>Figure 38.</td>
<td>QwikSIM 2.0, Gantry 88, Couch 88</td>
<td>74</td>
</tr>
<tr>
<td>Figure 39.</td>
<td>Scalar Definitions</td>
<td>110</td>
</tr>
<tr>
<td>Figure 40.</td>
<td>Equality of Scalars</td>
<td>110</td>
</tr>
<tr>
<td>Figure 41.</td>
<td>Intersection of Two Vectors</td>
<td>111</td>
</tr>
</tbody>
</table>
The Generation of a Digital Phantom for Testing of Digitally Reconstructed Radiographs

Nicholas Andrew Mason

ABSTRACT

The construction of phantoms for testing imaging parameters has been well documented in the literature. As computers have been introduced into the different areas of medicine, they have become more and more relied upon to replace conventional technologies. One specific example is that of plane film X-rays. Digitally Reconstructed Radiographs (DRR’s) are computer generated images that are generated from a 3-D volume of data, such as CT or MRI axial scans, and can be used in place of conventional X-rays. The computer can generate a DRR image for any position, orientation and magnification, and geometries not physically possible in the real world.

In this work a technique is developed to generate phantoms that can be used for testing the accuracy of DRR’s. A computer generated phantom can produce multiple test cases that can be used to test specific variables of the DRR’s.

A series of 12 different standard phantoms were used to test the ability of three different commercially available treatment planning or virtual simulation systems to generate DRR’s. A virtual simulation system under development by the author and collaborators and seeking approval from the Food and Drug Administration (FDA), was used as a development platform for this work.
Initial evaluation of the usefulness of the digital phantoms for testing showed immediate results. The first virtual simulation system tested with the phantoms revealed a major error in its ability to generate accurate DRR’s. Subsequently tests of the three commercially available systems further demonstrated the usefulness of the work. The tests revealed errors in two of the three systems evaluated but it was determined that they were not clinically significant.

In conclusion, the digital phantoms developed in this work are a fast, accurate method for testing digitally reconstructed radiographs. It is an extremely versatile testing method, as the phantoms can be generated with ease for any geometry without needing access to a CT scanner. This method of testing can be used to test a number of different DRR image parameters. Should an error be found, it can be used to isolate errors that might exist in the imaging device.
Introduction

In last decade computers have infiltrated all aspects of medical care. One of the most technologically advanced areas of medicine is that of Radiation Oncology. The complexity of the treatment delivery process and the potential for simple errors to lead to serious patient complications has facilitated the proliferation of computers into this specialty.

Radiation Oncology

Radiation Oncology is the treatment of cancer utilizing ionizing radiation. X-rays were first used in a medical application in 1896. In 1903 George Perthes discovered that X-rays could inhibit growth in tumors and proposed the use of X-rays in the treatment of cancer. Naturally occurring radioactivity was discovered by Henri Becquerel in 1896. This discovery ultimately led to the development of a Cobalt 60 treatment unit, by Harold Johns in Saskatoon, Saskatchewan, used to treat its first patient in 1951. In 1953 the first linear accelerator, developed by the Varian brothers at Stanford University, California, was used to treat a patient at Hammersmith Hospital in London, England. Today, most treatment machines containing radioactive sources are no longer used. The flexibility of linear accelerators have proven them to be more versatile and efficient for cancer treatment.
The geometry of these two types of treatment units are slightly different. The amount of radiation that can be delivered from a radioactive source is limited by the specific activity (the amount of radioactivity per gram)\(^4\) which is limited by the physical size of the source. Linear accelerators do not have this limitation, and hence the source can be placed at distances farther from the patient, delivering higher amounts of radiation to larger areas.

**The Cancer Treatment Process**

Once a diagnosis of cancer is made, the patient is consulted by the Radiation Oncology Physician to determine if radiation therapy is appropriate, and if so, what type. The physician uses all of the diagnostic information to determine where the cancer is and then positions beams of radiation from various locations to hit the tumor while sparing as much healthy tissue as possible.

Once the appropriate treatment has been determined, the treatment is simulated. Historically, specially manufactured “simulators” have been used to perform this task. Simulators are conventional X-ray units that simulate the same geometries of the treatment units. They have had to be mechanically advanced as the treatment unit allowed movement in five degrees of freedom (three translations and two rotations).

Next a computer simulation of the treatment is generated. A series of Computerized Tomography (CT, formally known as Computerized Axial Tomography, CAT) images are imported into the treatment planning or virtual simulation system. The treatment beams (or fields) are then “placed” on the CT images and the computer calculates the
amount of radiation deposited to the tissue, also known as radiation dose. This “plan” is then reviewed by the Radiation Oncologist to ensure it is what he/she intended.

The patient’s treatment then commences. Prior to receiving the first treatment, the patient’s fields are verified by taking a port film (an X-ray taken with the high energy treatment unit) of each field. Once the Radiation Oncologist compares the port films with the simulation films and determines that they agree, the patient receives the treatment.

**Port Films**

Port films are difficult to read, as unlike diagnostic radiology which uses low energy X-rays (in the range of 30 – 150KV) port films are usually taken at 6MV. At the lower energies of diagnostic radiology, bone absorbs as much as 6 times as much radiation as does tissue, mainly due to the photoelectric effect. At the higher energies that are used for treatment (6MV) the Compton effect takes precedence and there is only a 10-20% difference in attenuation of the radiation between bone and tissue. The contrast is much less making it difficult to distinguish internal objects, such as bone.

**CT Simulation**

With the advance of technology, computers have become capable of storing large amounts of data and process data quicker. In Radiation Oncology one of the techniques that have come out of the advancement of technology is Virtual Simulation. This method was proposed by Sherouse et al. in 1990 and by using a series of CT slices, the patient need not be present while the computer forms a “virtual patient”, that is, a three-dimensional reconstruction of the patient.
Two things made this possible; the first is the ability to store and process large amounts of CT data in a reasonable amount of time. The second is the ability to replace the conventional simulation film with a digital version of it generated from the CT data. The films are called Digitally Reconstructed Radiographs (DRR’s). DRR’s are computer generated (digitally reconstructed) radiographs (or X-films).

**Digitally Reconstructed Radiographs**

DRR’s are computer generated films calculated from the 3D imaging data set, normally CT scans, but can be generated from any imaging modality such as MRI and PET scans. Like conventional X-rays, these DRR’s are usually calculated to be divergent as if taken using a point source.

DRR’s have also been called DRDR’s (Digitally Reconstructed Divergent Radiographs). The divergence refers to the ray line tracing that occurs to produce the computer generated film or X-ray. The ray line is traced from an imaginary point source to the film plane through the CT images illustrated in Figure 1. A different type of image is generated on a CT scanner referred to as a scout view. The scout view uses the non-divergent ray lines to generate the X-ray images.
Figure 1. Divergent Line Definition

In order to understand the DRR’s we need to understand the basic imaging concepts that make up the Virtual Patient, which in itself is comprised from a series of CT slices.

CT Slices

A Computerized Tomography image is an axial “slice” of patient information taken with X-rays. A CT image is made up of an array of pixels. Each pixel contains a value that represents the attenuation coefficient associated with that pixel. A typical CT image (Figure 2) contains an array of 512 x 512 pixels in both the width and length. Depending on the body part being scanned, the length and width of the pixel can vary from 0.5mm to 2 mm.
Figure 2. Sample CT Image of a Head

Pixel

A pixel is the fundamental object for a picture or image. A pixel is a dot, usually of finite width and length that has a value associated with it that represents an intensity of that particular location in the image. In a photograph, a pixel intensity would contain a color. For a CT image, the pixel represents a grayscale value which is proportional to the attenuation coefficient of that particular location. Typically the grayscale value is a number from 0 to 4095 which represents 12 bits of information.

Electron Density

The electron density of a substance represents the number of electrons per gram. The electron density is an important radiological feature of an object. The number of electrons per gram determines how X-rays and electrons travel through that substance and how it will be attenuated by it. The larger the number of electrons per gram, the more
attenuation will occur as X-rays and electrons travel through it. However the number of particles attenuated through different materials will change as the energy of the radiation changes.  

**Hounsfield Units**

The electron density is related to the attenuation coefficient for each pixel. Each pixel is assigned a number, referred to as a CT number which range from -1000, for air, to 1000 for bone. By definition, 0 is assigned to water. When CT numbers are normalized in this manner they are referred to Hounsfield Units, named after Godfrey Hounsfield who was awarded a Nobel prize for the development of CT scanners. Hounsfield units are defined as follows:

\[ H = \frac{\mu_{\text{tissue}} - \mu_{\text{water}}}{\mu_{\text{water}}} \times 1000 \]

As technology has advanced, and computers memory has increased, CT numbers are now assigned 16 bits of data. However, the original Hounsfield definitions has been retained, with a slight modification. The numbers above 1000 apply to higher density objects such as dense (or hard) bone and metals.

When used for treatment planning, a special calibration phantom is used that has multiple objects of different known densities. This phantom is scanned on a CT scanner and transferred to the treatment planning system. There a CT number to electron density calibration curve is determined and used. As a result, each CT scanner has its own calibration that is correct for that particular system. For example, -999 rather than -1000 might refer to air of one particular system.
Voxel

When a CT scan is obtained, the operator also defines a slice thickness. Slice thickness is the thickness of the radiation beam as it traverses through the patient. A voxel is therefore a pixel with a thickness associated with it. It also contains a number that represents the intensity at that point. Typical slice thickness’ are from 1 mm to 10 mm.

3D Virtual Patient

Figure 3. Example of a Virtual Patient

Part of the treatment planning process is to CT scan a patient encompassing the intended treatment area. As computers have advanced in their ability to handle large amounts of data, patients are routinely scanned well above and below the treatment area. Since the patients are going to have radiation therapy, the amount of radiation from a CT scan is not of concern. By scanning large amounts of the patient, the computer can reconstruct a
virtual patient as illustrated in Figure 3. All anatomical information regarding the patient is available. This means it is possible to perform calculations on the patient without the patient even being there.

**Volume of Data**

Each CT image typically comprises of 512 x 512 pixels. Each pixel has (up to) 16 bits (2 bytes of data) in which to represent the intensity of that pixel. That is an intensity of $2^{16}$ or 65536. Each pixel intensity illustrates the CT numbers that correspond to the attenuation of that pixel. The pixel is graphically represented by a grayscale value. One axial CT image consists of 524288 (512 x 512 x 2) bytes. An average CT data set for Radiation Oncology has 100 axial images, therefore the average CT images series has approximately 52.5 Megabytes (MB) of data.

**DRR’s**

Once we have a virtual patient, we have all the information we need to generate a DRR. In reality, the DRR is a way to transform the more advance technology of CT simulation back to the older method of conventional X-rays. The DRR is then used to verify the treatment field by comparing it against a port film taken of the actual treatment area.

**Physics of a DRR**

The generation of a DRR is a relatively simple geometry problem, however a complex mathematical problem. An imaginary source of X-ray’s is placed at a desired location and an imaginary X-ray film is placed at a plane beyond the patient. The patient is represented by the 3D volume of voxels each of which contains an intensity. A line is
drawn from the source to the appropriate point (pixel) on the film plane. A ray tracing algorithm\textsuperscript{10} is used to calculate the “attenuation” as the ray goes from the source, through the patient to the film plane. The resulting relative attenuation of that pixel is then stored as an intensity. Figure 4 shows the corners of a beam diverging through a patient onto the imaginary film plane.

![Figure 4. DRR Geometry](image)

**Digital Phantom**

The purpose of this research is to come up with a method to test the accuracy of digitally reconstructed radiographs. A literature review turned up only some basic techniques for testing DRR’s. One of the methods published uses an acrylic block with different objects to determine the effect slice thickness has on resolution. Another method refer to using
clinical images (either real patient or anthropomorphic phantom images) to evaluate DRR’s. It is difficult to evaluate the accuracy of a reconstruction algorithm using these clinical images as they contain too much anatomic information of varying density.

The technique developed with this work allows the tester to generate any number of test cases with specific imbedded objects to test specific parameters of the DRR’s. Tests can be generated for any combination of geometries, geometries that cannot be produced clinically.

Whereas a physical phantom has to be imaged before being used for testing, the digital phantom, which can be generated using a computer, lacks any introduced errors from an imaging device. Errors in the digital phantom test images are directly related to the inherent errors determined by the definition of the voxel sizes and locations of object. These errors are described later in this work.

The testing process can be applied to other advanced imaging techniques, such as Multi Planar Reconstructed (MPR) images. MPR images are images through any plane of a three dimensional volume. The images in Figure 5 contain orthogonal MPR images from a three dimensional CT image set. The axial image is the top image, the sagittal image is in the lower right image and the coronal is the lower left image.
Figure 5. Sample Multi Planar Reconstructed Images
Materials and Methods

Introduction

There are three main steps to generate an image series to be used for testing DRR’s. First a series of empty CT images need to be created. Then a phantom (something to emulate a patient’s body outline) needs be created within the CT image series. Finally, objects are inserted into the images. These objects should be selected such that they can be used to check specific parameters of the DRR which are discussed later in this work.

Once a successful image series has been created, a method of transferring the images to the test system needs to be available. Each system being tested has a different method of importing images so a versatile method is required.

Once the images have been imported into the test system, the user must then perform the test and evaluate the results. The tests were designed in such a way that the evaluation would be easy, requiring simple visual evaluation and measurements. All of the systems being tested include basic tools for evaluation such as a ruler function for measurement. Although many tools exist to perform some of the above functions, they each exist on different systems. The author had access to a product being developed by MC2 Scientific Systems as a Virtual Simulation System. The main purpose of a virtual simulation system is to import DICOM CT images, allow the user to contour anatomic structures and place
beams on them. Once beams have been placed, DRR’s are generated and printed so they can be used to verify the patient treatment by comparing them with the port films.

**Generating Test Images**

There are three steps for developing the image series for testing.

- The image boundary.
- The “patient” or phantom extents.
- The embedded objects that are used to test the images characteristic.

**Image Boundary**

The first step in creating the phantom for testing is to generate the image volume that will contain all of the test objects. It should be large enough to fully encompass the objects with a margin around the outside.

**Phantom Limits**

All treatment planning and virtual simulation systems require the patient outline to be manually performed by the operator prior to positioning a beam. One of the reasons, and the most import for this work, is that the distance from the source to the entrance point on the patient must be known as it will affect the divergence of the image.

A square phantom, 40 cm in width and height would be representative of a fairly large patient.

**Electron Density/Pixel Value**

As mentioned earlier, most medical images contain an intensity value that represent the main imaging parameter. CT images have a pixel value in the range of 1 to 4095 for 12 bit images, or 0 to 255 for 8 bit images. Although 16 bits are available for 12 bit images
most systems continue to use the Hounsfield scale which ranges from -1000 for air to 3095 for very dense objects such as metals.

The default density (CT number) of the phantom was set to a low value (100). The default density used for the internal objects was 4000, representing a very dense object. Some of the system automatically scale the grayscale of the image based on the highest and lowest values. The large difference between to two values assures an image with a good contrast.

**Inserting Objects into the Images**

By inserting specific objects in each image, some of the variables that determine the resulting DRR image can be tested. By creating a series of phantom images with the specific objects inserted into it, different tests can be performed to test the gantry angle, beam divergence as well as the actual algorithm that generates the image.

The best object for evaluation of an image is a straight line. A straight line is inserted by “lighting up” the voxels that intersect an imaginary line from the source to the film plane. A straight line is the best object to be used because when the line is projected back to the film plane, only the lit voxels determine the pixel value displayed on the resulting DRR. If all occurs correctly the straight line in the phantom will result in a single point on the DRR.

A straight line in a phantom is positioned along a divergent ray line from the imaginary source point as illustrated in Figure 6. This will result in a dot on the DRR. A dot is effectively a delta function which is easy to subjectively evaluate.
In order to test different planning or virtual simulation systems, the images that have been created need to be sent to the system to be tested. The DICOM format was used to ensure compatibility with the systems being tested.
DICOM Format

Once the image series has been created, they need to be exported to the system being tested. The QwikSIM system contains a DICOM export module. DICOM\textsuperscript{11} is an acronym for Digital Imaging and COmmunications in Medicine. It is a standard for transferring medical images (CT, MRI, PET, etc) between computer systems. The DICOM standard addresses both the hardware and software issues facilitating the transfer of images between PC and Unix based systems which has posed problems historically as they handle floating point numbers differently.

The Digital Imaging and Communications in Medicine (DICOM) standard was created by the National Electrical Manufacturers Association (NEMA) to aid the distribution and viewing of medical images, such as CT scans, MRI’s, and ultrasound. Included in the standard is the description of a file format for the distribution of images. This format is an extension of an older NEMA standard.

A single DICOM file contains both a header (which stores information about the patient's name, the type of scan, image dimensions, etc), as well as all of the image data.

DICOM is the most common standard for receiving scans from a hospital or medical practice. All recent (since the late 1990’s) medical software packages that utilize medical images incorporate DICOM as a means of accepting and sending the images.
Development Platform

QwikSIM

The choice of platforms for development required many things be considered. The purpose of this research was to develop a phantom for testing DRR’s, not develop a major software project. The resulting image series had to be flexible enough to test multiple systems. The development platform needed to be selected to meet the following requirements:

- The system had to be able to create large volumes of data easily.
- Multiple tests series needed to be created.
- The tests needed to be flexible enough to test multiple parameters of the DRR’s.
- The system had to be able to transfer the test images to the different systems for testing.

The platform chosen for development was the QwikSIM, originally developed by MC2 Scientific Systems, Inc. The QwikSIM is a Virtual Simulation System developed with Microsoft Visual C++ and supported the export of DICOM images. The author had access to all of the source code necessary to rebuild the program to include this work. A significant addition to the source code was made to add the functionality as described in detail below. Only the pertinent source code added to the QwikSIM project is attached as Appendix B.

Geometry

The image in Figure 7 demonstrates a linear accelerator’s gantry rotation. The gantry rotates around the Z axis. The couch, as shown in Figure 7 with a patient lying on it, also rotates, however it rotates around the Y axis. The collimator rotates also around the same
axis as the couch but does not move the patient, or the beam relative to the patient and hence can be ignored for the purpose of these tests.

Figure 7. Simulated Rotating Gantry with Coordinate System

Figure 8. Axes Definition
Perform Test

Table 1 summarizes the gantry and couch rotations for each of the tests to be run on the systems to be tested.

<table>
<thead>
<tr>
<th>Table 1. Test Geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gantry Angle</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>45</td>
</tr>
<tr>
<td>90</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>45</td>
</tr>
<tr>
<td>45</td>
</tr>
<tr>
<td>88</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>88</td>
</tr>
</tbody>
</table>

Each test requires a unique set of images that have a line thought the phantom determined by the gantry and couch angles

Evaluate Results

These tests were designed such that successful completion of each test would result in the same DRR image being generated regardless of what combination of gantry and couch angles are used.

Creating Images

The flowchart in Figure 9 illustrates the main steps in creating the test series:
Figure 9. Test Generation Procedure

Step 1: Generate Image Volume

The first step in generating a test case is to generate the image volume. The Image volume is a series of plane images that encompass all internal objects.

Image Generation

Within the image a phantom must exist. In a normal diagnostic image, there is air outside a patient. The boundary of the two is frequently used for automatic determination of the patient outline. Therefore the image should be larger than the outer boundary of the phantom. Typical CT images are 512 x 512 pixels. Assuming a 1 mm pixel size in both horizontal and vertical dimensions would support up to a 51 cm patient/phantom, leaving at least one pixel outside the phantom. That is larger than most patients. Older CT scanners and imaging computer systems used a default image size of 256 x 256 images. For all of the tests used in this dissertation, the image parameters are as follows:
• Each test series has 201 images.
• The spacing and thickness between each slice is 2mm.
• Each slice consists of a 2D matrix of 201x201 pixels. Each pixel is 2mm by 2mm.

The number of pixels and slices was chosen to be 201 to ensure that the voxels were cubic in size. A number of the treatment planning systems did limit the number of slices they could import. One system limited the number of slices it could import to 255, therefore 201 seemed like a reasonable number.

A sample image volume is shown in Figure 10. The user is prompted to enter the image parameter, number of pixels, image thickness and size of the voxels in a dialog box as shown in Figure 11.

Figure 10. Sample Image Volume

Upon creation of the image volume, all voxels were set to a density of 0, which is a Hounsfield unit of -1000, which represents the density of air.
Figure 11. Image Parameter Dialog Box

The user has the option to pick a pixel depth of 8 bit or 12 bit. 8 bit support was placed for testing older systems that supported only 256 x 256 images with 8 bits defining the depth, but was not used for this work.

**Step 2: Generate Phantom**

Once the image volume has been generated, internal objects can be added. The objects should remain inside the image volume with at least one row of pixels between any objects and the outer limit of the image. The user is prompted using a dialog box, shown in Figure 12, to enter the object type to be added to the image. This dialog box continues to prompt the user to enter objects until they have no more object to add, then they can click the done button.

The first object should be a phantom. This is the encompassing object that would relate to the patient within the image. In order to reduce the possibility of this phantom interfering with the image generation process, the object is usually of extremely low density, just slightly more dense than the original image matrix.

Most systems that calculate DRR’s require an external contour be present, that is, the lateral extents of the phantom. For ease of calculation, a default phantom was developed
such that the extents of the phantom were set to 75% of the image width and length. Based on the image parameters described above, 75% of the image results in a 30cm square phantom. The phantom was assigned a CT number of 100 or -900 Hounsfield units. This is an insignificant number when it comes to the eye determining the outline but allows the computer segmentation algorithms to easily find the edge of the object.

**Figure 12. Dialog Box to Add Objects**

The figure above is the dialog box that allows the user to continually add objects to be inserted into the image volume.

Although the image volume is cubic, the phantom is extended to within two slices of the most superior and inferior portion of the image volume. This truly emulates a typical patient CT scan series, but still ensures that a beam can be placed on the end of the image volume (normal CT scans extend beyond the end of the CT series). Some treatment planning computer systems do not allow the placement of a beam if the phantom contour is not closed at the superior or inferior end.

**Step 3: Adding Internal Objects**

The next step is to create shapes within a “phantom”. By carefully placing objects within the boundaries of the phantom, the following parameters of DRR’s can be tested.
- Algorithm and Divergence - Blurring
- Spatial Resolution – Position of Dots
- Density – Brightness of Pixels.

A line was determined to be the best object to use as an internal object. A line, if its position is calculated correctly within the phantom, will result in a single dot on a calculated DRR image as shown in Figure 13. By specifying the start and end coordinates of a line appropriately, all three of the parameters above can be tested.

![Diagram of Phantom, Image Volume, Film Plane](image)

**Figure 13. Internal Object Geometry**

- Algorithm and Divergence:

These two factors can not be discerned from each another. Assuming the algorithm is correct, a blurring of the dot on a DRR will be caused by incorrect divergence of the line. This could be caused by errors in the source to beam entry point or a rotation of the beam. Calculating the correct divergence for each line requires knowledge of the distance from the imaginary source to the beam entrance point on the phantom.
• Spatial Resolution:
Placing the lines, with the correct divergence, at a known position in the phantom will result in a dot on the DRR. Evaluation of whether the dot occurs at the right position will determine if the spatial resolution of the resultant DRR is correct.

• Density:
Varying the intensity of the pixels in the line, will change the intensity of the resulting dot on the DRR. The DRR algorithm will average the pixels in the divergent line to determine the density of the resulting dot.

Before describing the details on adding internal objects, knowledge of the geometry of the DRR’s is required.

**Gantry Rotation**

To calculate the internal objects start and end points of an internal object, a correction needs to be applied.\(^{12}\) This correction is applied to all internal objects end points by taking the un-rotated coordinates and using matrix multiplication calculate the new end points.

\[
\begin{pmatrix} x', y', z' \end{pmatrix} = \begin{pmatrix} x, y, z \end{pmatrix} * \begin{pmatrix} \cos(G) & \sin(G) & 0 \\ -\sin(G) & \cos(G) & 0 \\ 0 & 0 & 1 \end{pmatrix}
\]

Where \(G\) represents the gantry angle.
x, y and z are the original, unrotated coordinates of the object.
x’, y’ and z’ are the coordinates of the object after it had been rotated.

The rotation matrix is calculated using the following formalisms;

\[
c_{ik} = a_{ij} \cdot b_{jk}
\]
where j is summed over all possible values of i and k.

\[
\begin{pmatrix} x', y', z' \end{pmatrix} = \begin{pmatrix} x, y, z \end{pmatrix} \cdot \begin{pmatrix} \cos(G) & \sin(G) & 0 \\ -\sin(G) & \cos(G) & 0 \\ 0 & 0 & 1 \end{pmatrix}
\]

\[
x' = x \cdot \cos(G) + y \cdot (-\sin(G)) \\
y' = x \cdot \sin(G) + y \cdot \cos(G) \\
z' = z
\]

**Couch Rotation**

The couch rotation is similarly performed using the following matrix multiplication:

\[
\begin{pmatrix} x', y', z' \end{pmatrix} = \begin{pmatrix} x, y, z \end{pmatrix} \cdot \begin{pmatrix} \cos(C) & 0 & -\sin(C) \\ 0 & 1 & 0 \\ \sin(C) & 0 & \cos(C) \end{pmatrix}
\]

\[
x' = x \cdot \cos(C) + z \cdot \sin(C) \\
y' = y \\
z' = x \cdot (-\sin(C)) + z \cdot \cos(C)
\]

where C represents the couch angle.

**Collimator Rotation**

Since the collimator rotation does not change the position of the patient or beam relative to the source, no rotation is required for the images.

**Combination of Rotations**

When both a gantry and couch angles are chosen, both the gantry and couch rotational matrices are multiplied together. Since only the rotation of the couch moves the image volume, it is necessary to perform the gantry rotation before the couch rotation.
Example of an Internal Object

In the simplest example of no rotation or divergence to account for, a straight line through the center of the phantom would be defined by the vector;

\[
\begin{pmatrix}
    0,150,0 \\
    0,150,0 \\
    0,150,0 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
    \cos(G) & \sin(G) & 0 \\
    -\sin(G) & \cos(G) & 0 \\
    0 & 0 & 1 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
    \cos(C) & 0 & -\sin(C) \\
    0 & 1 & 0 \\
    \sin(C) & 0 & \cos(C) \\
\end{pmatrix}
\]

Where the numbers represent the position relative to the center of the phantom in millimeters.

If vector v1 is rotated through a 45 degree gantry and 45 degree couch rotation, the new vector v1’ describing the rotated line would be:

\[
\begin{pmatrix}
    -75,106.1,75 \\
    75,106.1,-75 \\
\end{pmatrix}
\]

Internal Coordinate System

The image volume is made up of an array of voxels that use integers as opposed to a floating point coordinate system. The voxels are referenced by their offset from an initial position or initial voxel.

In the above described configuration of the image planes, used as a default image volume for all of the test, the first voxel is voxel (0,0,0), the last is (200,200,200). The phantom starts at (25,25,25) and ends at (175,175,175). An internal object, a simple non divergent
line (a line parallel and perpendicular to the edge of the image volume) through the center of the phantom, shown to demonstrate the coordinate system, goes from \((100,100,25)\) to \((100,100,175)\). Figure 14 illustrates the internal coordinate system.

\[\text{Figure 14. Internal Coordinate System}\]

**Calculating the Line End Points**

In order to draw a line in the image series, the program prompts the user for the start and end coordinates of the line to be drawn. The program then runs through the entire 3D
image series, one pixel at a time and evaluates whether an intersection has occurred between the line and the voxel.

In order to determine whether the line and any individual voxel intersect, three algorithms were developed. The first program maps the vectors defining the line and voxel into multiple 2D planes.

The second program breaks up the plane into a series of vectors describing the voxel edges in each plane, effectively a square.

The third program examines each of the lines, the lines that make up the square defining the voxel, to see if they intersect the line to be drawn. If an intersection occurs, the pixel is set to the user defined value, otherwise it is left at its initial value of zero.

**Entering Line Coordinates**

In order to define a line to be drawn in the phantom, the user is prompted for a start and end coordinates as well as the intensity of the line as shown in Figure 15.

![Figure 15. Line Entry Dialog Box](image)

**Figure 15.** Line Entry Dialog Box
Figure 16. Vector Definition

In Figure 16, V1 represents the line to be drawn in the phantom. V2 is a 3D vector that represents the start and end points of the voxel to be tested.

In order to perform that test, the algorithm first maps the 3D vectors into 2D planes and examines each plane.

Line Cube Intersection

The algorithm steps through the cube and compares the vector v1, as defined by the user, with a vector defining each individual voxel. The decision was made to pass in the vector v2 defined as from the initial corner to the opposite corner as shown in Figure 16. The algorithm then maps both vectors into the XY, XZ and YZ planes as illustrated in Figure 17 to determine if an intersection occurred in each plane. In order for an intersection to
occur in three dimensional space, an intersection must occur in each of the three planes.

If the all three planes contain an intersection the voxel will be lit.

Figure 17. Map 3D Vector to 2D Planes
**Line Plane Intersection**

Although called line/plane intersection, we actually need to calculate the intersection of a line with a square, where the square has known start and end points as shown in Figure 18. The square is defined by a series of four vectors, v1, v2, v3, v4. The line vline is the line to be tested to see if an intersection occurs with the four vectors defining the square.

![Figure 18. Line Square Intersection](image)

To test whether an intersection occurs, each individual vector is tested for an intersection with the vector vline. The first test performed is v1 vs. vline, then v2 vs. vline, etc as shown in Figure 19.
Figure 19. Line Line Intersection Tests

An intersection has been determined to have occurred if at least two of the lines defining the square intersect.

Initially it was felt that the algorithm could check for exactly two intersections to have occurred, however there are a number of different intersection scenario’s (Figure 19) that can result in multiple intersections occurring.
After evaluating the different scenario’s that could occur, it was decided that any two or more intersections between the line and the square would constitute an intersection in that plane.

**Line Line Intersection**

The Line Plane intersection algorithm passes the line to be drawn along with each line that constructs the square into the line line intersection algorithm.
Calculating Intersecting Lines

In order to calculate the intersection of two lines, equations from Mantyla’s\textsuperscript{13} book were used. By solving simultaneous equations for two dimensional lines, the intersection point can be found.

\[
\begin{align*}
y &= m_1x + b_1 \\
y &= m_2x + b_2
\end{align*}
\]

If the lines intersect, solving the simultaneous equations will yield values for \(x\) and \(y\) at the point of intersection.

\[
\begin{align*}
x &= \left(\frac{b_2 - b_1}{m_1 - m_2}\right) \\
y &= m_1\left(\frac{b_2 - b_1}{m_1 - m_2}\right) + b_1
\end{align*}
\]

Calculation of the intersection point using this technique does not actually tell you if the two lines physically intersect in the region of interest, in this case within the phantom extents, but rather if any point along the line defined by the line equation \(y=mx+b\) would intersect. In order to calculate whether the intersection occurs within the two lines within the phantom, they were treated as vectors, which allows the consideration of the start and end coordinates.

Using vectors to define the lines, the intersection of two lines can be determined by calculating the ratio of the vector lengths of the original vector \(v_1\) with the vector defined by the starting point of \(v_1\) with the end point being the intersection point.
Figure 21. Intersection of Vectors

The ratio of the lengths of v3/v1 will return a scalar value, Scalar1, which is the fraction of the length of v1 that the intersection occurs (i.e. the distance from x1,y1 to x3,y3 as defined in Figure 21). If that returned scalar value is between 0 and 1, the intersection falls within the line defined by v1. If the scalar is less than 0 or greater than 1 an intersection would occur along the line defined by the equation y=mx+b but beyond the start and end points of the line. Likewise if the scalar value is 0 or 1, the intersection occurred at the beginning or end of the vector v1 respectively.

A second scalar, Scalar2 is the ratio of the distance that the intersect point falls along the vector v2. Both scalar values must be between 0 and 1 in order for an intersection to occur.
Calculation of the Scalar

Figure 22. Scalar Calculation

It can easily be shown that the scalar, \( f_{s1} \) in Figure 22, is the ratio of delta \( x \) for \( v_3 \) to delta \( x \) for \( v_1 \).

Appendix C shows the proof of calculating the scalar values \( f_{s1} \) and \( f_{s2} \), the distance along each vector the intersection occurs.

\[
f_{s1} = \frac{v_3}{v_1} = \frac{x_3 - x_{11}}{x_{12} - x_{11}} = \frac{x_3 - x_{11}}{\Delta x_1}
\]

\[
f_{s1} = \frac{((\Delta x_1 \Delta x_2)(y_{21} - y_{11}) + (\Delta y_2 \Delta x_1)(x_{11} - x_{21}))}{(\Delta x_1 \Delta y_2 - \Delta y_1 \Delta x_2) \Delta x_1}
\]

Similarly

\[
f_{s2} = \frac{((\Delta x_1 \Delta x_2)(y_{21} - y_{11}) + (\Delta y_1 \Delta x_2)(x_{21} - x_{11}))}{(\Delta x_1 \Delta y_2 - \Delta y_1 \Delta x_2) \Delta x_2}
\]
The equations will only work if the lines are not vertical. If either of the lines are vertical, i.e. a delta x of 0, then we get a divide by 0, and the equation fails. Solving for x’s rather than y’s will yield similar equations that can be used if the lines are horizontal, i.e. delta y = 0.

\[
f_{s1} = \frac{(\Delta y_1 \Delta x_2)(y_{11} - y_{21}) + \Delta y_2 \Delta y_1(x_{21} - x_{11})}{(\Delta x_1 \Delta y_2 - \Delta y_1 \Delta x_2)\Delta y_1}
\]

Similarly

\[
f_{s2} = \frac{(\Delta x_1 \Delta y_2)(y_{11} - y_{21}) + \Delta y_2 \Delta y_1(x_{21} - x_{11})}{(\Delta x_1 \Delta y_2 - \Delta y_1 \Delta x_2)\Delta y_2}
\]

Having two different ways to calculate both $f_{s1}$ & $f_{s2}$ gives us flexibility when calculating intersections of lines. Frequently the calculation of the intersection involves lines that are either parallel or perpendicular to the image matrix, hence yield a delta x or delta y of zero.

**Special Cases**

When calculating the intersection of two lines, it is important to pre-screen the two vectors to see if they are going to be any problems within the algorithms. If either of the lines are vertical or horizontal, the appropriate calculation of the scalars is required.

Case 1: Delta X=0

If either of the lines are vertical lines, the slope of the line goes to infinity. This should be prescreened to catch it to ensure the computer does not hang up.

Case 2: Delta Y=0
If the lines are horizontal, the slope of the line goes to zero. This results in divide by zeros during the calculation of the scalars.

Case 3: Scalar<0

If the scale is less than zero the point of intersection falls before the first point on the vector.

Case 4: Scalar>1

If the scale is greater than one the point of intersection falls after the last point on the vector.

Figure 23. Special Case Intersections

By prescreening these special cases (Figure 23), the algorithm can save time by easily determining intersections in the cases of intersections at the start and end points of the
lines. This saves time by reducing the amount of mathematically computations that need to occur if they are all treated equally. More importantly the algorithms can account for all possible scenarios that would cause abnormal program termination.

**Step 4: Exporting Images**

The QwikSIM contains a utility to export the images in a DICOM format. DICOM provides both hardware and software compatibility for transferring images between medical imaging systems. In order to fulfill the DICOM requirements, the QwikSIM generates each slice as an individual file. The QwikSIM informs the user where the files are stored. This allows the user to use a DICOM send program to send the images to the test system or copy them to a CDROM and hand carry them to the system being tested.

**Phantom Examples**

**What the Tests Show**

By selectively picking the objects within the phantom, the tests can be designed to pick up any of the following errors:

- Divergence or Algorithm Errors.
- Spatial Resolution Errors.
- Density Errors.

The first test, the divergence test, will determine if errors exist in the DRR’s reconstruction algorithm. An error can also result from incorrect beam geometry. If an error exists, determining the beam geometry is correct will indicate an algorithm problem. The second test, the spatial resolution test, evaluates the positioning of objects
in the DRR image. The third test, the density test, will determine the ability of the DRR algorithm to differentiate different density objects.

**Simple 2D Test**

![2D Conceptual Layout](image)

**Figure 24. 2D Conceptual Layout**

To test the concept of this technique, a simple phantom (Figure 24) was designed which contains a non-divergent line through the center of the phantom plus two divergent lines all in the same plane. Figure 24 has four images. The top left is an axial image. The axial view is an image in the XY plane as defined in Figure 8. The top right image is a sagittal image which is taken in the YZ plane. The bottom left view is the coronal image which is
an image taken in the XZ plane. The bottom right image is a three dimensional reconstruction of the objects.

**Beam Outline**

The basic shape used to in all of the tests is a series of lines designed to make up the corners or a treatment beam. The lines are setup such that each line is drawn, only within the phantom, on a path from the imaginary source to a point 100.0 cm from the source at the entry point of the phantom. Four lines are created, one in each quadrant 5cm out and 5 cm up from the center as shown in Figure 25.

![Beam Outline Layout](image)

Each circle represents a dot in each quadrant of the beam.

**Figure 25. Beam Outline Layout**

Figure 26 shows clockwise from the upper left image, an axial, sagittal, 3D and coronal view of the standard test geometry used in this work.
Table 2 contains the 3-D coordinates of the default beam used in all of the tests. The coordinates are relative to an origin point at the center of the phantom.
Table 2. Standard Line Start and End Coordinates

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Axis Start</td>
<td>0.0</td>
<td>-15.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Coordinate.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Central Axis End</td>
<td>0.0</td>
<td>15.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Coordinate.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quadrant 1 Start</td>
<td>-5.0</td>
<td>-15.0</td>
<td>-5.0</td>
</tr>
<tr>
<td>Quadrant 1 End</td>
<td>-6.5</td>
<td>15.0</td>
<td>-6.5</td>
</tr>
<tr>
<td>Quadrant 2 Start</td>
<td>-5.0</td>
<td>-15.0</td>
<td>5.0</td>
</tr>
<tr>
<td>Quadrant 2 End</td>
<td>-6.5</td>
<td>15.0</td>
<td>6.5</td>
</tr>
<tr>
<td>Quadrant 3 Start</td>
<td>5.0</td>
<td>-15.0</td>
<td>5.0</td>
</tr>
<tr>
<td>Quadrant 3 End</td>
<td>6.5</td>
<td>15.0</td>
<td>6.5</td>
</tr>
<tr>
<td>Quadrant 4 Start</td>
<td>5.0</td>
<td>-15.0</td>
<td>-5.0</td>
</tr>
<tr>
<td>Quadrant 4 End</td>
<td>6.5</td>
<td>15.0</td>
<td>-6.5</td>
</tr>
</tbody>
</table>

Phantom Series Generated

Seven series of test cases were generated as listed in Table 3

Table 3. Initial Test Geometries

<table>
<thead>
<tr>
<th>Gantry Angle</th>
<th>Couch Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>0*</td>
<td>90</td>
</tr>
<tr>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>90</td>
<td>90</td>
</tr>
</tbody>
</table>

* This phantom is the geometric equivalent to the gantry=0, couch=0 phantom so was removed from the series.

Each image series contains 201 images of 201x201 pixels, or 81K worth of data per image or 16 Megabytes per case. The images are mostly homogeneous, therefore they were compressed by the DICOM algorithms allowing all cases to fit on a single CD-ROM for testing purposes.
Three additional phantoms were generated as part of the results of the initial testing. The QwikSIM version 1.2f demonstrated an error at angles close to, but not equal to, 90 degree couch rotation. The three additional tests are listed in Table 4.

Table 4. Additional Tests

<table>
<thead>
<tr>
<th>Gantry Angle</th>
<th>Couch Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>88</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>88</td>
</tr>
<tr>
<td>88</td>
<td>88</td>
</tr>
</tbody>
</table>

**Beam Entry Coordinates**

Since the lines are rotated within the phantom they keep their original length which is an important consideration for the density test. Therefore the beam must enter at the first point of the central axis line.

Table 5 lists the entry points for the central axis for each test.

Table 5. Beam Entry Coordinates

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Gantry</th>
<th>Couch</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15.0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>0</td>
<td>10.61</td>
<td>-10.61</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>88</td>
<td>0</td>
<td>14.99</td>
<td>-0.523</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
<td>0</td>
<td>15.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>45</td>
<td>7.5</td>
<td>-10.61</td>
<td>-7.5</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>-15.0</td>
</tr>
<tr>
<td>7</td>
<td>88</td>
<td>88</td>
<td>0.523</td>
<td>-0.523</td>
<td>-14.982</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>-15.0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>88</td>
<td>0</td>
<td>-15.0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>45</td>
<td>0</td>
<td>-15.0</td>
<td>0</td>
</tr>
<tr>
<td>Van Dyk - 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15.0</td>
<td>0</td>
</tr>
<tr>
<td>Van Dyk - 2</td>
<td>88</td>
<td>88</td>
<td>0.523</td>
<td>-0.523</td>
<td>-14.982</td>
</tr>
</tbody>
</table>
Other Work

Phantoms of various types have routinely been used in the testing of treatment planning system software. Constantinou\textsuperscript{14} developed an inhomogeneous phantom for testing electron density for imhomogeneity corrections. The first published article on the use of phantoms for validation of DRR’s was written by McGee.\textsuperscript{15} They built a phantom and generated 6 different CT data sets using four different test patterns to test the different geometric parameters of DRR’s. Although others have used phantoms for verification of dose\textsuperscript{16,17} and image parameters a literature search showed the development of a “physical” phantom by Van Dyk.\textsuperscript{18} The phantom consisted of blocks with regions of different density objects as shown in Figure 27. These objects are built with sloped sides to emulate the divergence of the radiation beam, Figure 28.

Van Dyk Phantom

The Van Dyk phantom has a mechanism that allows the phantom to be rotated around multiple axes to simulate the rotations of the gantry and couch. The user must select the appropriate rotations of the object and the image it using the CT scanner. Once the series has been reconstructed the images are then transferred to the system to be tested.
Reproducing the Van Dyk Phantom

The Van Dyk phantom was reproduced in the same manner as the conventional tests. Lines were placed representing the corners between the different density diverging shapes. The density of the line was selected to be representative of the density of the object at the inner boundary of the diverging shape.
The Van Dyk phantom reproduction can be performed for any combination of gantry and couch angles.
Discussion

Tests

An images series was created for each test case listed in Table 6.

Table 6. Test Geometry

<table>
<thead>
<tr>
<th></th>
<th>Gantry Angle</th>
<th>Couch Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>90</td>
</tr>
<tr>
<td>6</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>7</td>
<td>45</td>
<td>90</td>
</tr>
<tr>
<td>8</td>
<td>88</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>88</td>
</tr>
<tr>
<td>10</td>
<td>88</td>
<td>88</td>
</tr>
<tr>
<td>Van Dyk 1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Van Dyk 2</td>
<td>88</td>
<td>88</td>
</tr>
</tbody>
</table>

To run each test, the images are to be imported into the computer system being tested. The external contour or body outline needs to be performed. Then a beam should be placed on the external contour at the position specified in Table 5. Once the beam has been placed on the phantom, a DRR for that beam should be generated and compared with the expect image which is discussed in the next section.

The tests are designed such that each test will yield the same DRR image if no problems exist in the DRR algorithm of the system being tested.
Expected DRR Images

With the exception of the Van Dyk phantom images, the resulting DRR images should be identical. All phantoms are generated such that the resulting DRR’s have a dot at the center of the image plus four dots, one in each quadrant of the image as shown in Figure 29.

Tests 1 – 10

The evaluation of the resulting DRR image is somewhat subjective, however we are looking for relatively large errors. Clinical setups are judged to be accurate enough if they are within 0.5 degrees of their expected position. It was determined that we have the ability to resolve a 0.2 degree error using this testing method. This is not the case if patient images are used for testing, it is not possible to resolve a 0.2 degree error in a patient DRR as there is too much anatomic information with too many different density objects.
Although the dots will appear in the same location, the phantom surrounding the internal objects will take on different shapes, as only the lines in the phantom have been rotated.

**Van Dyk Phantom Tests**

The Van Dyk Phantom results in a DRR as shown in Figure 30. The outer and center dots are identical to the other tests, however there are three additional dots of decreasing intensity as they get closer to the center in each quadrant.
Figure 30. Expected Van Dyk Phantom DRR Image

Each quadrant contains four different intensity dots, each of which represent the boundaries (corner) of the different density objects and a representative density in the original phantom.

Sensitivity

Gantry Angle

A phantom was generated with a 0.5, 1.0, 1.5, 2.0 and 2.5 degree rotation of the objects.

Three observers were used to determine at what angle the rotation became “obvious”. All three observers easily identified the 0.5 degree rotation.
Limitation of Phantom

The phantom is limited to detecting an n degree rotation because of the finite number of pixels and the positioning of the objects within the phantom as well as the phantom size. For the standard phantom, which is 30x30cm and within 2 slices of the top and bottom of the image series, the pixels are 2mm square. The minimum detectible angle should therefore be the angle between two adjacent pixels at the opposite end of the phantom.

\[
\tan^{-1}\left(\frac{\text{Pixel Size}}{\text{Phantom Separation}}\right)
\]

\[
= \tan^{-1}\left(\frac{2\text{mm}}{300\text{mm}}\right)
\]

\[
= 0.382 \text{ Degrees}
\]

If a greater resolution or ability to detect smaller angles is required it can be achieved by either increasing the size of the phantom (Phantom_Separation) or by decreasing the pixel size (Pixel_Size).

\[
\tan^{-1}\left(\frac{\text{Pixel Size}}{\text{Phantom Separation}}\right)
\]

\[
= \tan^{-1}\left(\frac{1\text{mm}}{300\text{mm}}\right)
\]

\[
= 0.191 \text{ Degrees}
\]

Determining the Minimum Detectable Rotation

The minimum gantry rotation detectable was determined by generating multiple beams with differing gantry angles, 0.0 degrees (the reference field), 0.1, 0.2 and 0.3 degree gantry rotation. The resultant DRR’s were examined by three different observers to determine what level was noticeable. The observers were asked to rank the images in
order of how blurred they were, 1 being not blurred at all to 4 being the most blurred. The results are tabulated in Table 7.

Table 7. Minimum Detectable Angle, Observer Results

<table>
<thead>
<tr>
<th>Angle [degrees]</th>
<th>Observer 0.0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Observer 1 and 2 could not tell the difference between the reference image and the 0.1 degree rotation. All three observers noted that a rotation had occurred when the lines were rotated by 0.2 degrees.

**Round Off Error**

Since the pixels are represented by integers, the line coordinates need to be rounded off. The rounding generates an error which is dependent upon the pixel size. The phantoms developed for the purposes of these tests used a length of 150 pixels (which at 2mm pixel size, represents 30 cm). The maximum error due to round off is 0.5 a pixel.

\[
Max\ _Error = \left( \frac{Maximum\ _RoundOff}{Number\ _of\ _Pixels} \right)
\]

\[
= \tan^{-1}\left( \frac{0.5}{150} \right)
\]

\[
= 0.191°
\]

Using a smaller phantom, i.e. decreasing the line length, and increasing the pixel size will both result in larger errors.
Significance of the Round Off Error

This error happens to be the same magnitude as the rotational error the user can discern, so it would be difficult to distinguish between round off error and a rotational error, however the author feels this is such a small value in relation to reasonable clinical errors, that it would not be noticeable.

Evaluating the Results

As discussed earlier in this section, evaluating the resulting DRR images is subjective. A number of techniques were tried to attempt to come up with a more quantitative evaluation. Two methods were examined;

- Evaluate the dots for their eccentricity.
- Count the number of pixels that comprise the dots.

Evaluating the dots for their eccentricity did not work, as depending on the rotation being tested (gantry or couch), the dots could be eccentric in either direction. In addition, each computer system being has different tools available. Two of the systems being evaluated did not have the ability to measure the eccentricity of a dot from the DRR views.

Counting the pixels around the dot was a reasonable method, however, on one of the systems tests (Eclipse), the dots were significantly blurred by the algorithm. In addition, as the image was zoomed up, the pixels were continually smoothed, making it impossible to count the number of pixels affected.
Results

The DRR algorithm will have a lot of influence as to the results of the different tests. If the DRR algorithm considers the length of each ray as it traverses each voxel, the results should be better (meaning less blurring, more accurate density results) than if the algorithm uses a simple “does it intersect” algorithm.

The images were evaluated using three different criteria.

1. Divergence/Algorithm: Did the DRR match the expected DRR image, specifically, were the dots sharper or more blurred when compared to the expected DRR?

2. Spatial Resolution: Were the dots in the right location?

3. Density: Were the dots the correct density?

The first test, Divergence/Algorithm, is a subjective analysis of whether the resultant DRR matches the expected DRR as shown in Figure 31. In the resulting image, each line is represented by a single dot. Should an error occur, it would appear as a blurring of the dot on the DRR which could occur in either plane. Blurring in the x axis of the image would indicate an error in the gantry plane, whereas blurring in the y axis of the image would indicate an error in the couch plane. A blurred dot indicates either an error in the algorithm or that the beam is not positioned the correct distance from the source relative to the phantom outline.
The second test, the position of the dots must be measured to determine the distance from the central axis. The tests were designed such that the dots should occur at 5cm from both x and y planes in each image quadrant. For the Van Dyk phantoms, additional dots if decreasing intensity should appear at 4.0cm, 2.5cm and 1.5 cm from both the x and y planes in each image quadrant.

The final test examines the intensity of the resulting dot. The intensity of the dot should be directly proportional to the density of the line. By subjectively examining the intensity of the dot, the algorithm can be evaluated for its ability to differentiate the density of the internal objects.
Figure 31. Expected DRR Image
Figure 32. **DRR with an Introduced 1.0 Degree Rotational Error**

A 1.0 degree rotation is shown in Figure 32 and Figure 33 as they are a lot clearer in print. When viewing the DRR’s on a computer monitor it is possible to visualize a 0.2 degree rotation as discussed earlier in this work.
Figure 33. **DRR with an Introduced 1.0 Degree Rotational Error with No Axes**

Figure 32 shows the expected DRR with a 1 degree rotation in the gantry plane. Figure 33 is the same image without the axes projected on the image to make the central axis dot more visible which demonstrates how obvious a one degree error is to discern.

**Systems Tested**

**QwikSIM 1.2f**

The concept was originally developed to find a way to evaluate the DRR’s from the QwikSIM which was being prepared for the rigorous testing required for FDA approval. The DRR algorithm was comprised of multiple subroutines or function. Each function
contained a test that had been developed to ensure the individual function performed as expected, there was no test that tested the entire DRR algorithm. As mentioned earlier, there are many combinations of gantry and couch angles that can be set, testing all of them is just not feasible.

Table 8 summarizes the initial tests performed early in the research. The only test performed was to check the algorithm or divergence. The spatial resolution and density tests, although important, were of secondary concern.

Table 8. QwikSIM 1.2f Geometry Results

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Gantry</th>
<th>Couch</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>3</td>
<td>90</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>45</td>
<td>Pass</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>90</td>
<td>Pass</td>
</tr>
<tr>
<td>6</td>
<td>45</td>
<td>45</td>
<td>Fail</td>
</tr>
<tr>
<td>7</td>
<td>90</td>
<td>90</td>
<td>Fail</td>
</tr>
</tbody>
</table>

All tests involving a single rotation appeared to work well; the tests were all showing that any errors were not visible to the evaluator. While running test 7, an error occurred in the setup of the test resulting in a 2 degree offset of the couch angle. The results were spectacular, but completely puzzling. Rather than having a blurred line (the expected result for a failed test) a wavy line occurred. The test worked at 45 degree’s and at 90 degree’s but not at 88 degrees.
Figure 34. QwikSIM 1.2f, DRR Gantry 88 Deg, Couch 88 Deg

Figure 34 is that of the failed DRR Image, which was generated on the QwikSIM version 1.2f with a gantry angle of 88 degrees and a couch angle of 88 degrees.

The code was exhaustedly evaluated for errors that cause this type of error. Simple geometry would dictate that if the algorithm worked at 45 degree’s and still worked at 90 degree’s, then it would work at all angles in between. Upon evaluation, it was found that the programmer had used a special case scenario that if the angle of the couch and/or gantry was within a 1.0 degrees of 90.0 degrees, this special case code took over and calculated for a 90 degree rotation. This forced the author to re-evaluate the tests and add additional tests to test the angles close to 90 degrees but far enough off that the
assumption that they were close enough to the 90 degree offset. Phantoms were added to
test the 88 degree gantry and couch angles.

The error in the algorithm that produced this spiral error was not particularly evident at
the 45 degree gantry and 45 degree couch test, but was obvious at the larger angles.

A 90 degree rotation is a simple rotation as in the rotational matrix, both of the angle
variables become either 0 or 1 and the coordinates just swap locations.

\[
\begin{bmatrix}
\cos(G) & \sin(G) & 0 \\
-\sin(G) & \cos(G) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} =
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} \times \left(\begin{bmatrix}
\cos(90) & \sin(90) & 0 \\
-\sin(90) & \cos(90) & 0 \\
0 & 0 & 1
\end{bmatrix}\right)
\]

\[
x' = -y \\
y' = x \\
z' = z
\]

It was determined that the entire algorithm had to be rewritten. QwikSIM v2.0 was the
rewritten code, and the system used in the final testing of this work.

**Systems Re-Tested**

These tests were run on three different virtual simulation or treatment planning systems.

Two of these systems were commercially available; the third is the rewritten version of
the QwikSIM, the product under development that incited these tests. The three systems
tested with this technique are listed in Table 9.
Table 9. Systems to be Tested

<table>
<thead>
<tr>
<th>Name</th>
<th>Software Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC2 QwikSIM.</td>
<td>2.0d</td>
</tr>
<tr>
<td>Theratronics Theraplan Plus</td>
<td>3.5</td>
</tr>
<tr>
<td>Varian Eclipse</td>
<td>7.1.35</td>
</tr>
</tbody>
</table>

QwikSIM 2.0

QwikSIM 2.0d was the second generation code with the new, corrected, DRR algorithm.

As summarized in Table 10, all of the divergence tests passed.

Divergence Test

Table 10. QwikSIM 2.0 Geometry Results

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Gantry</th>
<th>Couch</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>3</td>
<td>88</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>45</td>
<td>Pass</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>90</td>
<td>Pass</td>
</tr>
<tr>
<td>7</td>
<td>88</td>
<td>88</td>
<td>Pass</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>90</td>
<td>Pass</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>88</td>
<td>Pass</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>45</td>
<td>Pass</td>
</tr>
<tr>
<td>Van Dyk 1</td>
<td>0</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>Van Dyk 2</td>
<td>88</td>
<td>88</td>
<td>Pass</td>
</tr>
</tbody>
</table>

All divergence tests performed on the QwikSIM passed. All dots appeared as sharp as expected with no noticeable blurring.
Position Test

Table 11. QwikSIM 2.0 Position Results

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Gantry</th>
<th>Couch</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>3</td>
<td>88</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>45</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>90</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>7</td>
<td>88</td>
<td>88</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>90</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>88</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>45</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>Van Dyk 1</td>
<td>0</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>Van Dyk 2</td>
<td>88</td>
<td>88</td>
<td>&lt;0.2 cm</td>
</tr>
</tbody>
</table>

Table 11 summarizes the position test results for the QwikSIM. The minimum positional error is listed as 2mm as that corresponds to one voxel, being more accurate than that is not possible with these tests.

Density Test

The intensity of each dot should decrease as the dots get closer to the central axis. The central axis dot should have the same intensity as the outermost object.

Table 12 shows all dots are visible on both of the Van Dyk phantom tests.

Table 12. QwikSIM 2.0 Density Results

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Gantry</th>
<th>Couch</th>
<th>Differential Density</th>
<th>All objects Visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Van Dyk 1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Van Dyk 2</td>
<td>88</td>
<td>88</td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>
Theraplan Plus

The Theraplan system is a treatment planning system developed originally by Atomic Energy of Canada Limited, Medical division (AECL Medical). It was a commercial spin off out of Princess Margaret Hospital, developed originally by Dr. Jack Cunningham in the late 1970’s.

Divergence Test

Table 13. Theraplan Plus Geometry Results

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Gantry</th>
<th>Couch</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>3</td>
<td>88</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>45</td>
<td>Pass</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>90</td>
<td>Pass</td>
</tr>
<tr>
<td>7</td>
<td>88</td>
<td>88</td>
<td>Pass</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>90</td>
<td>Pass</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>88</td>
<td>Pass</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>45</td>
<td>Pass</td>
</tr>
<tr>
<td>Van Dyk 1</td>
<td>0</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>Van Dyk 2</td>
<td>88</td>
<td>88</td>
<td>Pass</td>
</tr>
</tbody>
</table>

The DRR image for the Van Dyk 2 test is shown on the left side of Figure 35. All of the tests as listed in Table 13 passed. All dots appeared as sharp as expected with no noticeable blurring.
**Figure 35.** Theraplan Plus DRR, Gantry 88, Couch 88

**Position Test**

**Table 14.** Theraplan Plus Position Results

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Gantry</th>
<th>Couch</th>
<th>Angle Detectable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>3</td>
<td>88</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>45</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>90</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>7</td>
<td>88</td>
<td>88</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>90</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>88</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>45</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>Van Dyk 1</td>
<td>0</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>Van Dyk 2</td>
<td>88</td>
<td>88</td>
<td>&lt;0.2 cm</td>
</tr>
</tbody>
</table>
As with the QwikSIM, the position tests as listed in Table 14 have a maximum error of one voxel, less than 2mm.

**Density Test**

**Table 15. Theraplan Plus Density Results**

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Gantry</th>
<th>Couch</th>
<th>Differential Density</th>
<th>All objects Visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Van Dyk 1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Van Dyk 2</td>
<td>88</td>
<td>88</td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

With the Theraplan system, the user can specify different parameters that can vary the display of the DRR. In Table 15, the test were listed as failed because the DRR’s showed the highest density object as a black dot, but all other dots in the phantom as the same density. Varying the DRR calculation parameters did not result in the display of the relative densities of each of the dots as they should appear.

**Eclipse**

The Somavision/Eclipse system is a combination virtual simulation/treatment planning system originally developed by a physics group out of Switzerland, currently being sold by Varian Medical Systems.
Divergence Test

Table 16. Eclipse Geometry Results

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Gantry</th>
<th>Couch</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>3</td>
<td>88</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>45</td>
<td>Pass</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>90</td>
<td>Pass</td>
</tr>
<tr>
<td>7</td>
<td>88</td>
<td>88</td>
<td>Pass*</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>90</td>
<td>Pass</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>88</td>
<td>Pass</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>45</td>
<td>Pass</td>
</tr>
<tr>
<td>Van Dyk 1</td>
<td>0</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>Van Dyk 2</td>
<td>88</td>
<td>88</td>
<td>Pass*</td>
</tr>
</tbody>
</table>

The asterisk in Table 16 indicates that the image was not as expected. The displayed DRR image was rotated through 45 degrees on the screen. This was not judged to be an error because the dots appeared as expected, just not what expected as indicated in Figure 36. The Van Dyk 2 test exhibited the same rotation of the displayed image.

The dots were much larger than the dots appeared on the other system. The dots on the DRR from the Eclipse system are approximately 5mm in diameter, where on the other systems, they were a single pixel wide with some blurring of adjacent pixels. This is an assumption that the DRR algorithm was making.
Figure 36. Eclipse DRR, Gantry 88, Couch 88
Position Test

Table 17. Eclipse Position Results

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Gantry</th>
<th>Couch</th>
<th>Angle Detectable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>3</td>
<td>88</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>45</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>90</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>7</td>
<td>88</td>
<td>88</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>90</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>88</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>45</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>Van Dyk 1</td>
<td>0</td>
<td>0</td>
<td>&lt;0.2 cm</td>
</tr>
<tr>
<td>Van Dyk 2</td>
<td>88</td>
<td>88</td>
<td>&lt;0.2 cm</td>
</tr>
</tbody>
</table>

At the gantry of 88 degrees and couch angle of 88 degrees, the field was slightly offset to the right by approximately 2mm (as shown in Table 17) which is due to the round off of lighting up the pixels that constitute the lines.

Density Test

The density test is relevant only to the Van Dyk Phantoms. Can the multiple objects of different density be seen as dots of different intensities?

Table 18. Eclipse Density Results

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Gantry</th>
<th>Couch</th>
<th>Differential Density</th>
<th>All objects Visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Van Dyk 1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Van Dyk 2</td>
<td>88</td>
<td>88</td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

With the Eclipse system, the user can select the density of the objects that are considered in the calculation of the DRR’s. If the object is above a certain density, the object will be
included in the resulting DRR, if it is below, it is ignored. The result being that all objects in the Density Test are displayed at the same intensity level.

The results of the density test of the Van Dyk 2 series, as shown in Table 18, was abnormal in that some of the dots were inexplicably left out, see Figure 37.

Figure 37. Eclipse DRR, Van Dyk Phantom, Gantry 88, Couch 88

To ensure this was a computer specific anomaly, the QwikSIM 2.0 version is shown in Figure 38.
Figure 38. QwikSIM 2.0, Gantry 88, Couch 88
Conclusions

This work demonstrates the feasibility and usefulness of using these computer generated phantoms for testing digitally reconstructed radiographs (DRR’s).

Because physical phantoms, such as the Van Dyk phantom, require that they be CT’d prior to exporting the images to the testing platform, they can contain introduced errors from the CT reconstruction algorithms. Using the digital phantom generator removes any variability that a physical phantom would have containing errors introduced with image reconstruction techniques.

This technique can be used for generating phantoms at any combination of geometry that needs to be tested without having to have access to anything more than a computer. It is accurate enough that it has the ability to detect errors that are smaller than the minimum discernable angle and more accurate than necessary to verify a patient’s treatment.

Three commercially available virtual simulation or treatment planning systems were tested using this technique. The QwikSIM 2.0 performed flawlessly on all tests. The Theraplan performed well on all but the density tests. The Eclipse had some unusual results for the test at gantry 88 and couch 88 degrees. The image is acceptable from the test criteria perspective however the image is just displayed at an unexpected angle. The density errors were dramatic on the Eclipse, some of the dot were just not displayed. Where appropriate, the vendors were contacted to report these errors.
The digital phantom generator is an excellent method for testing DRR’s and has applications for other imaging modalities such as Multi Planar Reconstructions.

**Comparison of the Digital and Van Dyk Phantoms**

The Van Dyk phantom requires obtaining CT images of the phantom in the specific geometry required for each test. The divergence of the objects is fixed in the Van Dyk phantom limiting the testing to one specific source to surface distance, i.e. beam divergence. The CT has finite size detectors, this and the reconstruction algorithms can introduce errors in the generation of the images used for testing.

The digital phantom can be used to check any geometrically possible test cases. Any number of test cases can be generated in just minutes from a desktop computer. Since the images are generated on the computer, there is no reconstruction error associated with these images allowing true evaluation of the DRR algorithms.

Both systems can be used to test any system that can import DICOM images (which is practically all medical imaging systems).

The digital phantom can generate image series that could be used to test DRR’s to a much higher accuracy. By creating an image series with much smaller voxel sizes, the limiting factors of this technique, such as the minimum detectable angle, could be decreased allowing the DRR algorithms to be tested to finer resolutions.
Other Uses

The Digital Phantom has applications other than verifying DRR’s. There are other imaging modalities that this technology could be applied to such as Multi Planar Reconstructed (MPR) images.

Future Work

There are a number of areas that this work can be continued in.

- Developing a quantitative analysis method.
- Develop tests for other imaging modalities.
- Different Thickness Objects.

Since this work started, there has been an extension of the DICOM standard called RT-extensions (Radiation Therapy) which allows the electronic export of a number of radiation oncology specific objects. One of these is the RT-Image, which is used to transfer the DRR’s. A software program could be developed to import the raw DRR’s from the treatment planning or virtual simulation systems and perform a numerical analysis of the images to evaluate the “dots” generated by the DRR algorithms. This program could be used to electronically compare the spread or blurring to provide a more quantitative analysis of the DRR’s. A more accurate evaluation of the intensity of the dots could also be performed.

Additional test can easily be developed to evaluate other imaging techniques. The medical imaging industry is advancing at a rapid rate, new functionality is being added daily. This technique can be used to develop phantoms to test most of these new
techniques. For Multi Planar Reconstructed (MPR) images, simple lines can be placed at different places in the phantom to test the MPR generation.

One of the new software techniques developed is the ability to perform virtual endoscopy. This technique allows a computer to track through a 3D reconstructed object as if you are traveling through it as if you were passing a camera through it. This requires finding the center of an area and tracking it through any direction. A phantom could be developed for this by adding different thickness objects (lines) throughout the phantom and allowing the endoscopy algorithm to find the center. Connecting multiple lines of different thicknesses could be used to develop a comprehensive phantom.
Reference


Bibliography


Appendices
## Appendix A: Test Phantom Coordinates

Table 19. Test 1, Gantry 0, Couch 0

<table>
<thead>
<tr>
<th>Quadrant</th>
<th>CAX Start</th>
<th>CAX End</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Central Axis Ray</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>100</td>
<td>25</td>
</tr>
<tr>
<td>CAX End</td>
<td>100</td>
<td>175</td>
</tr>
<tr>
<td><strong>Upper Left Quadrant</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>75</td>
<td>25</td>
</tr>
<tr>
<td>CAX End</td>
<td>67.5</td>
<td>175</td>
</tr>
<tr>
<td><strong>Upper Right Quadrant</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>75</td>
<td>25</td>
</tr>
<tr>
<td>CAX End</td>
<td>67.5</td>
<td>175</td>
</tr>
<tr>
<td><strong>Lower Left Quadrant</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>125</td>
<td>25</td>
</tr>
<tr>
<td>CAX End</td>
<td>132.5</td>
<td>175</td>
</tr>
<tr>
<td><strong>Lower Right Quadrant</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>125</td>
<td>25</td>
</tr>
<tr>
<td>CAX End</td>
<td>132.5</td>
<td>175</td>
</tr>
</tbody>
</table>
Table 20. Test 2, Gantry 45, Couch 0

<table>
<thead>
<tr>
<th>Quadrant Description</th>
<th>CAX Start</th>
<th>CAX End</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Axis Ray</td>
<td>153.03</td>
<td>46.97</td>
<td>100</td>
</tr>
<tr>
<td>Upper Left Quadrant</td>
<td>135.36</td>
<td>29.29</td>
<td>75</td>
</tr>
<tr>
<td>Upper Right Quadrant</td>
<td>135.36</td>
<td>29.29</td>
<td>125</td>
</tr>
<tr>
<td>Lower Left Quadrant</td>
<td>170.71</td>
<td>64.65</td>
<td>125</td>
</tr>
<tr>
<td>Lower Right Quadrant</td>
<td>170.71</td>
<td>64.65</td>
<td>75</td>
</tr>
</tbody>
</table>
## Appendix A: (continued)

Table 21. Test 3, Gantry 90, Couch 0

<table>
<thead>
<tr>
<th>Quadrant</th>
<th>CAX Start</th>
<th>CAX End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Axis Ray</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>175</td>
<td>100</td>
</tr>
<tr>
<td>CAX End</td>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td>Upper Left Quadrant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>175</td>
<td>75</td>
</tr>
<tr>
<td>CAX End</td>
<td>25</td>
<td>67.5</td>
</tr>
<tr>
<td>Upper Right Quadrant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>175</td>
<td>75</td>
</tr>
<tr>
<td>CAX End</td>
<td>25</td>
<td>67.5</td>
</tr>
<tr>
<td>Lower Left Quadrant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>175</td>
<td>75</td>
</tr>
<tr>
<td>CAX End</td>
<td>25</td>
<td>67.5</td>
</tr>
<tr>
<td>Lower Right Quadrant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>175</td>
<td>125</td>
</tr>
<tr>
<td>CAX End</td>
<td>25</td>
<td>132.5</td>
</tr>
</tbody>
</table>
### Table 22. Test 4, Gantry 0, Couch 90

<table>
<thead>
<tr>
<th>Quadrant</th>
<th>CAX Start (mm)</th>
<th>CAX End (mm)</th>
<th>Start (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Central Axis Ray</strong></td>
<td>100</td>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td><strong>Couch 0</strong></td>
<td>100</td>
<td>175</td>
<td>100</td>
</tr>
<tr>
<td><strong>Upper Left Quadrant</strong></td>
<td>75</td>
<td>25</td>
<td>125</td>
</tr>
<tr>
<td><strong>Couch 0</strong></td>
<td>67.5</td>
<td>175</td>
<td>132.5</td>
</tr>
<tr>
<td><strong>Upper Right Quadrant</strong></td>
<td>125</td>
<td>25</td>
<td>125</td>
</tr>
<tr>
<td><strong>Couch 0</strong></td>
<td>132.5</td>
<td>175</td>
<td>132.5</td>
</tr>
<tr>
<td><strong>Lower Left Quadrant</strong></td>
<td>125</td>
<td>25</td>
<td>75</td>
</tr>
<tr>
<td><strong>Couch 0</strong></td>
<td>132.5</td>
<td>175</td>
<td>67.5</td>
</tr>
<tr>
<td><strong>Lower Right Quadrant</strong></td>
<td>75</td>
<td>25</td>
<td>75</td>
</tr>
<tr>
<td><strong>Couch 0</strong></td>
<td>67.5</td>
<td>175</td>
<td>67.5</td>
</tr>
</tbody>
</table>
## Appendix A: (continued)

Table 23. Test 5, Gantry 0, Couch 45

<table>
<thead>
<tr>
<th>Quadrant</th>
<th>CAX Start</th>
<th>CAX End</th>
<th>CAX End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Axis Ray</td>
<td>100</td>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td>Upper Left Quadrant</td>
<td>64.65</td>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td>Upper Right Quadrant</td>
<td>100</td>
<td>175</td>
<td>145.96</td>
</tr>
<tr>
<td>Lower Left Quadrant</td>
<td>135.36</td>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td>Lower Right Quadrant</td>
<td>100</td>
<td>175</td>
<td>54.04</td>
</tr>
</tbody>
</table>
Appendix A: (continued)

Table 24. Test 6, Gantry 45, Couch 45

<table>
<thead>
<tr>
<th>Quadrant</th>
<th>CAX Start</th>
<th>CAX End</th>
<th>CAX End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Axis Ray</td>
<td>137.5</td>
<td>46.97</td>
<td>62.5</td>
</tr>
<tr>
<td></td>
<td>62.5</td>
<td>153.03</td>
<td>137.5</td>
</tr>
<tr>
<td>Upper Left Quadrant</td>
<td>107.32</td>
<td>29.29</td>
<td>57.32</td>
</tr>
<tr>
<td></td>
<td>23.27</td>
<td>130.05</td>
<td>130.77</td>
</tr>
<tr>
<td>Upper Right Quadrant</td>
<td>142.68</td>
<td>29.29</td>
<td>92.68</td>
</tr>
<tr>
<td></td>
<td>69.23</td>
<td>130.05</td>
<td>176.73</td>
</tr>
<tr>
<td>Lower Left Quadrant</td>
<td>167.68</td>
<td>64.65</td>
<td>67.68</td>
</tr>
<tr>
<td></td>
<td>101.73</td>
<td>176.01</td>
<td>144.23</td>
</tr>
<tr>
<td>Lower Right Quadrant</td>
<td>132.32</td>
<td>64.65</td>
<td>32.32</td>
</tr>
<tr>
<td></td>
<td>55.77</td>
<td>176.01</td>
<td>98.27</td>
</tr>
</tbody>
</table>
Appendix A: (continued)

Table 25. Test 7, Gantry 90, Couch 90

<table>
<thead>
<tr>
<th></th>
<th>CAX Start</th>
<th>CAX End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Axis Ray</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Upper Left Quadrant</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>67.5</td>
<td>67.5</td>
</tr>
<tr>
<td>Upper Right Quadrant</td>
<td>125</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>132.5</td>
<td>67.5</td>
</tr>
<tr>
<td>Lower Left Quadrant</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>132.5</td>
<td>132.5</td>
</tr>
<tr>
<td>Lower Right Quadrant</td>
<td>75</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>67.5</td>
<td>132.5</td>
</tr>
</tbody>
</table>
Table 26. Test 8, Gantry 88, Couch 0

<table>
<thead>
<tr>
<th>Quadrant</th>
<th>CAX Start</th>
<th>CAX End</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Axis Ray</td>
<td>174.95</td>
<td>97.38</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>25.05</td>
<td>102.62</td>
<td>100</td>
</tr>
<tr>
<td>Upper Left Quadrant</td>
<td>174.08</td>
<td>72.40</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>23.91</td>
<td>70.14</td>
<td>67.5</td>
</tr>
<tr>
<td>Upper Right Quadrant</td>
<td>174.08</td>
<td>72.40</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>23.91</td>
<td>70.14</td>
<td>132.5</td>
</tr>
<tr>
<td>Lower Left Quadrant</td>
<td>175.83</td>
<td>122.37</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>26.18</td>
<td>135.10</td>
<td>132.5</td>
</tr>
<tr>
<td>Lower Right Quadrant</td>
<td>175.83</td>
<td>122.37</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>26.18</td>
<td>135.10</td>
<td>67.5</td>
</tr>
</tbody>
</table>
## Appendix A: (continued)

Table 27. Test 9, Gantry 0, Couch 88

<table>
<thead>
<tr>
<th>Quadrant</th>
<th>CAX Start</th>
<th>CAX End</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Central Axis Ray</strong></td>
<td>100</td>
<td>25</td>
</tr>
<tr>
<td><strong>Upper Left Quadrant</strong></td>
<td>74.14</td>
<td>25</td>
</tr>
<tr>
<td><strong>Upper Right Quadrant</strong></td>
<td>124.11</td>
<td>25</td>
</tr>
<tr>
<td><strong>Lower Left Quadrant</strong></td>
<td>125.86</td>
<td>25</td>
</tr>
<tr>
<td><strong>Lower Right Quadrant</strong></td>
<td>75.89</td>
<td>25</td>
</tr>
</tbody>
</table>
Appendix A: (continued)

Table 28. Test 10, Gantry 88, Couch 88

<table>
<thead>
<tr>
<th>Quadrant</th>
<th>CAX Start</th>
<th>CAX End</th>
<th>CAX Start</th>
<th>CAX End</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Central Axis Ray</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>102.62</td>
<td>97.38</td>
<td>102.62</td>
<td>97.38</td>
</tr>
<tr>
<td>CAX End</td>
<td>97.38</td>
<td>102.62</td>
<td>174.91</td>
<td>174.91</td>
</tr>
<tr>
<td><strong>Upper Left Quadrant</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>77.60</td>
<td>72.40</td>
<td>77.60</td>
<td>72.40</td>
</tr>
<tr>
<td>CAX End</td>
<td>64.86</td>
<td>70.14</td>
<td>64.86</td>
<td>70.14</td>
</tr>
<tr>
<td><strong>Upper Right Quadrant</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>127.57</td>
<td>72.40</td>
<td>127.57</td>
<td>72.40</td>
</tr>
<tr>
<td>CAX End</td>
<td>129.83</td>
<td>70.14</td>
<td>129.83</td>
<td>70.14</td>
</tr>
<tr>
<td><strong>Lower Left Quadrant</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>127.63</td>
<td>122.37</td>
<td>129.90</td>
<td>135.10</td>
</tr>
<tr>
<td>CAX End</td>
<td>129.90</td>
<td>135.10</td>
<td>129.90</td>
<td>135.10</td>
</tr>
<tr>
<td><strong>Lower Right Quadrant</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAX Start</td>
<td>77.66</td>
<td>122.37</td>
<td>77.66</td>
<td>122.37</td>
</tr>
<tr>
<td>CAX End</td>
<td>64.94</td>
<td>135.10</td>
<td>64.94</td>
<td>135.10</td>
</tr>
</tbody>
</table>
### Appendix A: (continued)

#### Table 29. Test 11, Van Dyk Phantom, Gantry 0, Couch 0

<table>
<thead>
<tr>
<th>Quadrant Type</th>
<th>CAX Start</th>
<th>CAX End</th>
<th>CAX Start</th>
<th>CAX End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Axis Ray</td>
<td>100</td>
<td>25</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Upper Left Quadrant – Outer Lucite</td>
<td>75</td>
<td>25</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>Upper Right Quadrant - Outer Lucite</td>
<td>75</td>
<td>25</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>Lower Left Quadrant - Outer Lucite</td>
<td>125</td>
<td>25</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>Lower Right Quadrant - Outer Lucite</td>
<td>125</td>
<td>25</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>Upper Left Quadrant – Middle Cedar</td>
<td>80</td>
<td>25</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>Upper Right Quadrant - Middle Cedar</td>
<td>80</td>
<td>25</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>Lower Left Quadrant - Middle Cedar</td>
<td>120</td>
<td>25</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>Lower Right Quadrant - Middle Cedar</td>
<td>120</td>
<td>25</td>
<td>80</td>
<td></td>
</tr>
</tbody>
</table>
Appendix A: (continued)

<table>
<thead>
<tr>
<th>Quadrant</th>
<th>CAX Start</th>
<th>CAX End</th>
<th>CAX Start</th>
<th>CAX End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper Left Quadrant – Middle Inner Polystyrene</td>
<td>87.5</td>
<td>25</td>
<td>87.5</td>
<td>83.75</td>
</tr>
<tr>
<td>Upper Right Quadrant - Middle Inner Polystyrene</td>
<td>87.5</td>
<td>25</td>
<td>87.5</td>
<td>112.5</td>
</tr>
<tr>
<td>Lower Left Quadrant - Middle Inner Polystyrene</td>
<td>112.5</td>
<td>25</td>
<td>87.5</td>
<td>116.25</td>
</tr>
<tr>
<td>Lower Right Quadrant - Middle Inner Polystyrene</td>
<td>112.5</td>
<td>25</td>
<td>87.5</td>
<td>116.25</td>
</tr>
<tr>
<td>Upper Left Quadrant – Inner Air</td>
<td>92.5</td>
<td>25</td>
<td>92.5</td>
<td>90.25</td>
</tr>
<tr>
<td>Upper Right Quadrant - Inner Air</td>
<td>92.5</td>
<td>25</td>
<td>107.5</td>
<td>109.75</td>
</tr>
<tr>
<td>Lower Left Quadrant - Inner Air</td>
<td>107.5</td>
<td>25</td>
<td>107.5</td>
<td>109.75</td>
</tr>
<tr>
<td>Lower Right Quadrant - Inner Air</td>
<td>107.5</td>
<td>25</td>
<td>92.5</td>
<td>90.25</td>
</tr>
</tbody>
</table>

Table 30. CT Number for the Van Dyk Density Objects

<table>
<thead>
<tr>
<th>Label</th>
<th>CT Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outer Lucite</td>
<td>4000</td>
</tr>
<tr>
<td>Middle Cedar</td>
<td>3000</td>
</tr>
<tr>
<td>Middle Inner Polystyrene</td>
<td>2000</td>
</tr>
<tr>
<td>Inner Air</td>
<td>1000</td>
</tr>
</tbody>
</table>

94
Appendix B: Program Source Code

Generate Image Function.

    {3
    %-------------------------------------------------------------------------------%
    % STATIC FUNCTION:                                                              %
    %                                                                               %
    CImageVolume* WINAPI CShapeGenerator::GenerateNewVolume( void )
    %                                                                               %
    % DESCRIPTION:                                                                  %
    %    This static function serves as the driving function for the Shape          %
    %    Generator class.  It brings up the Shape Properties dialog and             %
    %    processes the user defined attributes.  It then creates the Volume         %
    %    Generator class, accesses it's members to generate a CImageVolume          %
    %    object, then destroys the class.                                           %
    %                                                                               %
    % RETURN VALUES:                                                                %
    %    NULL            Error: Image Volume could not be generated                 %
    %    CImageVolume*    Pointer to newly generated Image Volume object            %
    %                                                                               %
    % NOTICES & WARNINGS:                                                           %
    %    If the function fails for any reason or if the user cancels the          %
    %    operation, the return pointer is NULL.                                     %
    %                                                                               %
    % FILE ACCESS:                                                                  %
    %    None                                                                       %
    %                                                                               %
    % MODIFICATION LOG:                                                             %
    %    22Oct96 SEH     Initial Revision Create Volume Class                       %
    %    28May99    NAM    Added phantom generation                                 %
    %                                                                               %
    %-------------------------------------------------------------------------------%
Appendix B: (continued)

```c
int    iOriginX;    // Object Point 1, X Coordinate
int    iOriginY;    // Object Point 1, Y Coordinate
int    iOriginZ;    // Object Point 1, Z Coordinate
int    iOuterX;     // Object Point 2, X Coordinate
int    iOuterY;     // Object Point 2, Y Coordinate
int    iOuterZ;     // Object Point 2, Z Coordinate
int    iObjectWidth; // Object Width
int    iPixelValue; // Objects Pixel Value

// Initialize
p_ImageVolume = NULL;

// Create and bring-up shape selection dialog
// Default image parameters are 201,201 by 201 2mm voxels.
//
// p_ShpImDlg = new CShpImDlg( 201, 201, 201, (float)2.0, (float)2.0, 1);
ASSERT_VALID( p_ShpImDlg );
if ( p_ShpImDlg->DoModal() == IDOK )
{
    // Determine bytes per pixel
    if ( p_ShpImDlg->GetPixelBitDepth() == 8 )
        sBytesPerPixel = 1;  // Set to one bytes per pixel
    else if ( p_ShpImDlg->GetPixelBitDepth() == 12 )
        sBytesPerPixel = 2;  // Set to two bytes per pixel
    // Create new image volume
    p_ImageVolume = new CImageVolume();
    ASSERT_VALID( p_ImageVolume );
    bGenOK = p_ImageVolume->Create( p_ShpImDlg->GetWidth(), p_ShpImDlg->GetHeight(),
                                    p_ShpImDlg->GetNumSlices(), sBytesPerPixel,
                                    p_ShpImDlg->GetPixelBitDepth() );
    // If volume was created ok
    if ( bGenOK )
    {
        // Set coordinate system origin to volume center
        OriginPt.x = ( p_ImageVolume->GetImageWidth() - 1 ) / 2;
        OriginPt.y = ( p_ImageVolume->GetImageHeight() - 1 ) / 2;

        p_ImageVolume->SetOriginPoint( OriginPt );
        sRefSlice = ( p_ImageVolume->GetNumSlices() - 1 ) / 2;
        p_ImageVolume->SetReferenceSlice( sRefSlice );
        // Set size attribute
        p_ImageVolume->SetPixelSizeMM( p_ShpImDlg->GetPixelSizeMM() );

        // Set slice positions
        for ( k = 0; k < p_ShpImDlg->GetNumSlices(); k++ )
            { fSlicePositionMM = (float) ( sRefSlice - k ) * p_ShpImDlg-
              >GetSliceSpacingMM();
                p_ImageVolume->SetSlicePositionAt( k, fSlicePositionMM );
            }
        // Fill in volume with 0 density
        p_ImageVolume->FillVolumeMemory( 0 );
```

96
Appendix B: (continued)

// Get shape to be added from dialog
// loop around until no more objects are to be added.
do {
    p_ShpObDlg = new CShpObDlg( SHAPE_LINE );
    ASSERT_VALID( p_ShpObDlg );
    p_ShpObDlg->DoModal( );
    sShape = p_ShpObDlg->GetShape( );
    bNewObject = p_ShpObDlg->GetNewObject( );
    delete p_ShpObDlg;

    if (bNewObject == TRUE){  // If this is a new object, lets get the
        p_ShpPrmDlg = new CShpPrmDlg( 100, 200, 100, 0, sBytesPerPixel,
            p_ShpImDlg->GetWidth(),
            p_ShpImDlg->GetHeight(),
            p_ShpImDlg->GetNumSlices( ));
        ASSERT_VALID( p_ShpPrmDlg );
        p_ShpPrmDlg->DoModal( );
        // Get the paramaters of this object
        lRadius = p_ShpPrmDlg->GetRadius ( );
        usCenterValue = (unsigned short) p_ShpPrmDlg->GetCenterValue ( );
        usSurfaceValue = (unsigned short) p_ShpPrmDlg->GetSurfaceValue ( );
        usOuterValue = (unsigned short) p_ShpPrmDlg->GetOuterValue ( );
        iOriginX = p_ShpPrmDlg->GetOriginX ( );    // Object Point 1, X
        iOriginY = p_ShpPrmDlg->GetOriginY ( );    // Object Point 1, Y
        iOriginZ = p_ShpPrmDlg->GetOriginZ ( );    // Object Point 1, Z
        iOuterX = p_ShpPrmDlg->GetOuterX ( );    // Object Point 2, X
        iOuterY = p_ShpPrmDlg->GetOuterY ( );    // Object Point 2, Y
        iOuterZ = p_ShpPrmDlg->GetOuterZ ( );    // Object Point 2, Z
        iObjectWidth = p_ShpPrmDlg->GetObjectWidth ( );        // Object
        iPixelValue = p_ShpPrmDlg->GetPixelValue ( );        // Objects Pixel
        delete p_ShpPrmDlg;
    }

    // Create Volume Generator class
    p_VolGen = new CShapeGenerator( p_ImageVolume );
    ASSERT_VALID( p_VolGen );
    bGenOK = p_VolGen->GenerateVolumeShape( (eShapeType) sShape, lRadius,
        usCenterValue, usSurfaceValue, usOuterValue,
        iOriginX, iOriginY, iOriginZ,
        iOuterX, iOuterY, iOuterZ,
        iObjectWidth, iPixelValue
        );
    if ( !bGenOK ) {
        // Error - reset return pointer
        delete p_ImageVolume;
        p_ImageVolume = NULL;
    }
}
Appendix B: (continued)

    // Delete the Volume Generator class
    delete p_VolGen;
    }                      // Else exit
    while (bNewObject != 0);
    // Else lets go away and put the user back in the usual mode.

    }                    // if ( bGenOK )
if ( !bGenOK )
{
    // Delete and reset the return pointer
    delete p_ImageVolume;
    p_ImageVolume = NULL;
}

}                     // if ( p_ShapeDlg->DoModal( ) == IDOK )

// Delete the New Shape dialog
delete p_ShpImDlg;
return p_ImageVolume;
}
Appendix B: (continued)

Generate Phantom.

```cpp
void CShapeGenerator::BuildPhantom(
    long              lRadius,    // Radius of generated shape
    unsigned short    usCenterValue,        // Pixel value at center of shape
    unsigned short    usSurfaceValue,        // Pixel value at surface of shape
    unsigned short    usOuterValue        // Pixel value outside of shape
) {
    TRACE( "CShapeGenerator::BuildPhantom( )\n" );
    ASSERT( lRadius > 0 );
    ASSERT( usCenterValue >= usSurfaceValue );
    ASSERT( usSurfaceValue >= usOuterValue );
    ASSERT( ( (2 * lRadius) < mp_ImageVolume->GetImageWidth( ) ) &&
        ( (2 * lRadius) < mp_ImageVolume->GetImageHeight( ) ) );
    ASSERT( mp_ImageVolume->GetNumSlices( ) >= 4 );
    ASSERT( mp_ImageVolume->GetPixelDataPtr( ) != NULL );
    // Local declarations
    CRect            InhomoRect;      // Rectangle encompassing inhomogeneity on each slice
    int              iRectDim;        // Rectangle dimension
    int              i, j, k;         // used to index thorough pixels
    unsigned char*   p_uyPixel;       // byte pointer to pixel
    unsigned short*  p_usPixel;       // word pointer to pixel
    CPoint           OriginPt;        // Image volume origin pixel

    // Fill the image volume with surface value
    mp_ImageVolume->FillVolumeMemory( usOuterValue );
```
Appendix B: (continued)

// Check for minimum # required slices
if ( mp_ImageVolume->GetNumSlices( ) > 3 )
{
    // Set inhomogeneity rectangle
    iRectDim = mp_ImageVolume->GetImageWidth( ) / 8;
    if ( iRectDim < 2 )
        iRectDim = 2;
    InhomoRect.SetRect( iRectDim , iRectDim , iRectDim * 7, iRectDim * 7 );

    // Get volume origin point
    OriginPt = mp_ImageVolume->GetOriginPoint( );

    // Loop through inhomogeneity pixels, ignore top and bottom 2
    for ( k = 2; k < mp_ImageVolume->GetNumSlices( )-2; k++ )
    {
        for ( i = InhomoRect.top; i < InhomoRect.bottom + 1; i++ )
        {
            for ( j = InhomoRect.left; j < InhomoRect.right + 1; j++ )
            {
                // Switch according to bytes per pixel
                switch ( mp_ImageVolume->GetBytesPerPixel() )
                {
                    case 1:
                        p_uyPixel = (unsigned char*)
                        mp_ImageVolume->GetPixelPtrAt( k, i, j );
                        (*p_uyPixel) = (unsigned char) usSurfaceValue;
                        break;
                    case 2:
                        p_usPixel = (unsigned short*)
                        mp_ImageVolume->GetPixelPtrAt( k, i, j );
                        (*p_usPixel) = (unsigned short) usSurfaceValue;
                        break;
                }
            }
        }
    }

    // Increment progress dialog
    IncrProgress( );
}

Build Line

// %-----------------------------------------------------------------------%
// % MEMBER FUNCTION:                                                     %
// %                                                                       %
void CShapeGenerator::BuildLine (  
  int iOriginX,            // Object Point 1, X Coordinate  
  int iOriginY,            // Object Point 1, Y Coordinate  
  int iOriginZ,            // Object Point 1, Z Coordinate  
  int iOuterX,             // Object Point 2, X Coordinate  
  int iOuterY,             // Object Point 2, Y Coordinate  
  int iOuterZ,             // Object Point 2, Z Coordinate  
  int iObjectWidth,        // Object Width  
  int iPixelValue          // Objects Pixel Value  
)
// %                                                                       %
// % DESCRIPTION:                                                          %
// %       This function adds a line within an existing image             %
// %       volume. The origin is the start of the vector in               %
// %       pixel coordinates, the outer value is the end point of the     %
// %       line also if pixel coordinates                                  %
// %                                                                       %
// % RETURN VALUES:                                                        %
// %       None                                                           %
// % NOTICES & WARNINGS:                                                   %
// %       In debug mode, this function will ASSERT if required           %
// %       conditions are not met (see below)                             %
// %                                                                       %
// % FILE ACCESS:                                                          %
// %       None                                                           %
// %                                                                       %
// % MODIFICATION LOG:                                                     %
// %       08Aug00 NAM     Initial Revision                               %
// %                                                                       %
// %-----------------------------------------------------------------------%
// {
TRACE( "CShapeGenerator::BuildLine( )\n" );
// Local declarations
short             i, j, k, m;            // used to index thorough pixels
unsigned char*    p_uyPixel;             // byte pointer to pixel
unsigned short*   p_usPixel;             // word pointer to pixel
CPoint            OriginPt;              // H/V index of volume origin
float             fValue;                // The calculated pixel value
CVector3D         vorigin, vend;         // The line to be drawn's vectors
CVector3D         vpixelstart, vpixelend;       // the pixel start and end points
CVector3D         vtmp;
short             stmp, num_planes;
// Increment the progress update box to show the user how much longer
//
  m_fProgIncr = (float) 100.0 / (mp_ImageVolume->GetImageHeight( )  
  * mp_ImageVolume->GetNumSlices( ));
// Calculate value increment
//
  OriginPt = mp_ImageVolume->GetOriginPoint( );

101
Appendix B: (continued)

// Get pointers to pixel data
p_uyPixel = (unsigned char*) mp_ImageVolume->GetPixelDataPtr( );
p_usPixel = (unsigned short*) mp_ImageVolume->GetPixelDataPtr( );

// Define the start and end coordinates, in pixel coordinates.
//
vorigin = CVector3D ((float) iOriginX, (float) iOriginY, (float) iOriginZ);
vend = CVector3D ((float) iOuterX, (float) iOuterY, (float) iOuterZ);

fValue = (float) iPixelValue;

// Loop over the number of slices (Z)
//
for ( k = 0; k < mp_ImageVolume->GetNumSlices( ); k++ )
{

// Loop over the image height (Y)
/
for ( j = 0; j < mp_ImageVolume->GetImageHeight( ); j++ )
{

// Loop over the image width (X)
/
for ( i = 0; i < mp_ImageVolume->GetImageWidth( ); i++ )
{

// Calculate distance to pixel from origin
// Only look for calculate within the pixel volume that is specified, to speed up calc.
//
if ((k >= upperlower(iOriginZ,iOuterZ, 0)
&& k <= upperlower(iOriginZ,iOuterZ, 1))
&& (j >= upperlower(iOriginY,iOuterY, 0)
&& j <= upperlower(iOriginY,iOuterY, 1))
&& (i >= upperlower(iOriginX,iOuterX, 0)
&& i <= upperlower(iOriginX,iOuterX, 1)))
{

// Initialize the number of planes an intersection occurred in
//
num_planes = 0;

// Loop over the three primary planes, 1st X,Y
// then Y,Z and finally X,Z
// We need an intersection in all three planes in order to light the pixel.
//
for (m=0; m<3; m++)
{
    switch (m)
    {

// Set vorigin to the beginning of the line in pixel coordinates.
// Set vend to the end of the line in pixel coordinates.
// Set PixelStart to the start of the pixel
// Set PixelEnd to the opposite corner of the pixel
//
case 0: // X & Y, ignore last point
    vorigin = CVector3D ((float) iOriginX,
    (float) iOriginY, (float) iOriginZ);
    vend = CVector3D ((float) iOuterX, (float) iOuterY, (float)
    iOuterZ);
Appendix B: (continued)

```cpp
vpixelstart = CVector3D ((float) i, (float) j, (float) k);
vpixelend = CVector3D ((float) i+1, (float) j+1, (float) k);
stmp = vtmp.LinePlanIntersection(
    vorigin,        // Begin of Line 1
    vend,           // end of Line 1
    vpixelstart,    // Begin of Cube
    vpixelend       //end of Cube
);

// stmp is returned, which contains the number of intersections that occurred.
// as long as there are at least two, light up the pixel.
//
if (stmp >= 2) num_planes++;
break;
case 1:    // Y & Z
if (num_planes ==0) break;
vorigin = CVector3D ((float) iOriginY,
    (float) iOriginZ, (float) iOriginX);
vend = CVector3D ((float) iOuterY, (float) iOuterZ, (float) iOuterX);
vpixelstart = CVector3D ((float) j, (float) k, (float) i);
vpixelend = CVector3D ((float) j+1, (float) k+1, (float) i);
stmp = vtmp.LinePlanIntersection(
    vorigin,        // Begin of Line 1
    vend,           // end of Line 1
    vpixelstart,    // Begin of Cube
    vpixelend       //end of Cube
);
if (stmp >= 2) num_planes++;
break;
case 2:    // X & Z
if (num_planes <=1) break;
vorigin = CVector3D ((float) iOriginX, (float) iOriginZ, (float) iOriginY);
vend = CVector3D ((float) iOuterX, (float) iOuterZ, (float) iOuterY);
vpixelstart = CVector3D ((float) i, (float) k, (float) j);
vpixelend = CVector3D ((float) i+1, (float) k+1, (float) j);
stmp = vtmp.LinePlanIntersection(
    vorigin,        // Begin of Line 1
    vend,           // end of Line 1
    vpixelstart,    // Begin of Cube
    vpixelend       //end of Cube
);
if (stmp >= 2) num_planes++;
break;
}
// After all three planes have been tested, check to see if all three had an intersection occur.
//
switch ( mp_ImageVolume->GetBytesPerPixel( ) )
    // Switch based on bytes per pixel
    { 
        case 1:            // One byte per pixel case
            (*p_uyPixel) = (unsigned char) fValue;
            break;
```
case 2: // Two byte per pixel case
    (*p_usPixel) = (unsigned short) fValue;
    break;
  }
  }

// Increment pixel pointers
p_uyPixel++;
p_usPixel++;

// Increment progress dialog
IncrProgress();
}
Appendix B: (continued)

Line Plane Intersection:

```c
// %-------------------------------------------------------------------------%
// % MEMBER FUNCTION (Public):
// %                                                                         
short    CVector3D::LinePlanIntersection(
    const    CVector3D& vBeg1,    // Begin of Line 1
    const    CVector3D& vEnd1,    // End of Line 1
    const    CVector3D& vBeg2,    // Begin of Cube
    const    CVector3D& vEnd2 ) // Opposite corner of cube
// %                                                                         
// % DESCRIPTION:
// % Calculates the intersection of the line [Beg1:End1] with the Line
// % [Beg2:End2] in 3-Space. If the lines do not intersect or are parallel
// % appropriate errors are returned. On success, (*this) is returned with
// % the computed intersection point.
// %                                                                         
// % % RETURN VALUES:
// %    short        Number of intersections if successful
// %            MC2_LINES_ARE_PARALLEL if such is the case, or
// %            MC2_NO_INTERSECTIONS if the lines do not intersect.
// %                                                                         
// % NOTICES & WARNINGS:
// %    This algorithm was split from the original lineline intersection
// %    If the vector is perpendicular to one of the planes this algorithm
// %    Fails
// % /
// %    / B2(x,y) 1 /
// %     o---------x----->o
// %     ^ /        ^
// %   4 /         2
// %     /         /
// %   o-x---------o
// %     /         3 E2(x,y)
// %     /        
// %     /
// %     
// %     
// %     
// %     
// %     
// %     
// %     
// %     
// %     
// % % FILE ACCESS:
// %    None
// %                                                                         
// % MODIFICATION LOG:
// %    09Sep00 NAM        Initial Version, mc^2 Inc.
// %    21Jul04 NAM        Fix for four line check, PhD work
// %-------------------------------------------------------------------------%

{ //    TRACE( "CVector3D::LinePlanIntersection( )
    %--------------------%
    % Local declarations %
    %--------------------%
short        sOK1, sOK2, sOK3, sOK4;       // return value
short        num_intersections = 0;        // number of line crosses
float        fScal1, fScal2;               // float fraction along line intersection
occurs
    CVector2D    vB1, vE1, vB2, vE2, vXY, vXZ, vYZ;    // 2D vectors used in algorithm
```
Appendix B: (continued)

// %-------------------------%
// % Initialize line vector %
// %-------------------------%
vB1.m_fx = vBeg1.m_fx;
vB1.m_fy = vBeg1.m_fy;
vE1.m_fx = vEnd1.m_fx;
vE1.m_fy = vEnd1.m_fy;

// %-------------------------%
// % Try the line 1 first %
// %-------------------------%
vB2.m_fx = vBeg2.m_fx;  // Beginning Coordinate of the cube - X
vB2.m_fy = vBeg2.m_fy;  // Beginning Coordinate of the cube - Y
vE2.m_fx = vEnd2.m_fx;  // End point of line 1 - X
vE2.m_fy = vEnd2.m_fy;  // End point of line 1 - Y

// %-------------------------%
// % Compute the intersection point %
// %-------------------------%
sOK1 = LineLineIntersectFP( &fScal1, &fScal2, vB1, vE1, vB2, vE2 );
if ( sOK1 == NO_ERROR ) num_intersections++;

// % The X:Y Intersection is OK; try the Y:Z Plane %
// %-----------------------------------------------%
// % Compute the intersection point %
// %-----------------------------------------------%
sOK2 = LineLineIntersectFP( &fScal1, &fScal2, vB1, vE1, vB2, vE2 );
if ( sOK2 == NO_ERROR ) num_intersections++;

// % The X:Y and X:Z Intersections are OK; try the X:Z Plane %
// %-----------------------------------------------%
// % Compute the intersection point %
// %-----------------------------------------------%
sOK3 = LineLineIntersectFP( &fScal1, &fScal2, vB1, vE1, vB2, vE2 );
if ( sOK3 == NO_ERROR ) num_intersections++;

// % The X:Y and X:Z Intersections are OK; try the X:Z Plane %
// %-----------------------------------------------%
// % Compute the intersection point %
// %-----------------------------------------------%
sOK4 = LineLineIntersectFP( &fScal1, &fScal2, vB1, vE1, vB2, vE2 );
if ( sOK4 == NO_ERROR ) num_intersections++;

// % Return the number of line crossings %
// %-----------------------------------%
if (num_intersections >= 2) {
    return (num_intersections);
}
else
    return (MC2_NO_INTERSECTIONS);
Appendix B: (continued)

Line Line Intersection

```
// %-----------------------------------------------------------------------------%
// % FUNCTION:                                                                   %
// %                                                                             %
short LineLineIntersectFP(
    float* fScalarL1,    // Fraction dist from Begin to Inter.
    float* fScalarL2,    // Fraction dist from BegPoly to Inter.
    CVector2D Begin1,       // Beginning Vertex Line 1
    CVector2D End1,         // Ending Vertex Line 1
    CVector2D Begin2,       // Beginning Vertex Line 2
    CVector2D End2 )        // Ending Vertex Line 2

// %-----------------------------------------------------------------------------%
// % DESCRIPTION:                                                               %
// %    This function calculates the intersection between two lines. The        %
// %    algorithm employed is a derivation of the simultaneous solution of two %
// %    linear equations in the form "y = mx + b". The algorithm was taken from %
// %    Mantyla, 3rd edition, pgs 221-222.                                         %

// %    The 'fScalarL1' and 'fScalarL2' arguments are scalar multipliers along %
// %    lines 1 & 2 respectively. If these scalar values fall between 0 and 1 %
// %    inclusive then the point of intersection falls between the end points on %
// %    the respective line. If the scalar value is negative then the point of %
// %    intersection lies on the line in the direction opposite to the sense of %
// %    the line, i.e., pointing from vertex 1 away from vertex 2. If the scalar %
// %    is greater than 1 then the converse is true. Thus if a VECTOR 'vS' is %
// %    constructed from VECTORs 'vQ' (vertex 1) and 'vR' (vertex 2) then Vector %
// %    Multiplication of 'vS' by scalar 'fScalarL1' and subsequently added to %
// %    VECTOR 'vQ' will yield a new VECTOR 'vT' the ending point of which lies %
// %    along the original line 'QR'.                                              %

// %    The coordinates are given by:                                           %
// %                                                                            %
// %        x_intersection = Begin1.x + fScalarL1*(End1.x - Begin1.x);          %
// %        y_intersection = Begin1.y + fScalarL1*(End1.y - Begin1.y);          %

// %    All coordinates are in 2 Space. If line 1 and line 2 are parallel then %
// %    both 'fScalarL1' and 'fScalarL2' are returned as FLT_MAX <float.h>.       %

// %    The functional prototype is located in <mc2_supp.h>.                    %

// % RETURN VALUES:                                                             %
// %    If no intersection is found, the function returns the value             %
// %    MC2_LINES_ARE_PARALLEL <mc2error.h>.                                    %
// %    If an intersection is found, the function returns NO_ERROR; the number %
// %    of intersections found and two ordered arrays of segment indices        %
// %    and scalar multipliers locating the intersection points are VALID       %

// % NOTICES & WARNINGS:                                                        %
// %    Notices & warnings here                                                 %

// % FILE ACCESS:                                                               %
// % File access here                                                           %

// % MODIFICATION LOG:                                                          %
// % 30Sep97 BFH    Initial revision      mc^2, Inc.                          %
// % Copied from the LineLineIntersect function and modified to                %
// % use 2D Vector endpoints rather than CPoints.                              %
// % 09Aug00 NAM    Corrected for points that fall on a start & end point      %
```
Appendix B: (continued)

```c
{ float    denom, del_x1, del_y1, del_x2, del_y2, del_x12, del_y12;
  // % Calculate differences and the denominator %
  // %-------------------------------------------%
  del_x1  = (End1.m_fx   - Begin1.m_fx);
  del_y1  = (End1.m_fy   - Begin1.m_fy);
  del_x2  = (End2.m_fx   - Begin2.m_fx);
  del_y2  = (End2.m_fy   - Begin2.m_fy);
  del_x12 = (Begin2.m_fx - Begin1.m_fx);
  del_y12 = (Begin1.m_fy - Begin2.m_fy);
  // % Need to account for the case of points exactly coinciding %
  // %-----------------------------------------------------------------%
  if (Begin1.m_fx == Begin2.m_fx && Begin1.m_fy == Begin2.m_fy)
  // Point 1 and 2 begin coincides
  {
    *fScalarL1 = 0;
    *fScalarL2 = 0;
    return ( NO_ERROR );
  }
  else if (Begin1.m_fx == End2.m_fx && Begin1.m_fy == End2.m_fy)
  // Point 1 and 2 end coincides
  {
    *fScalarL1 = 0;
    *fScalarL2 = 1;
    return ( NO_ERROR );
  }
  else if (End1.m_fx == Begin2.m_fx && End1.m_fy == Begin2.m_fy)
  // Point 1 end and 2 begin coincides
  {
    *fScalarL1 = 1;
    *fScalarL2 = 0;
    return ( NO_ERROR );
  }
  else if (End1.m_fx == End2.m_fx && End1.m_fy == End2.m_fy)
  // Point 1 end and 2 end coincides
  {
    *fScalarL1 = 1;
    *fScalarL2 = 1;
    return ( NO_ERROR );
  }
  // %-----------------------------------%
  // % Calc denominator                  
  // %-----------------------------------%
  denom = del_x1 * del_y2 - del_x2 * del_y1;
  // % Check for parallel orthogonal lines %
  // %-----------------------------------------------------------------%
  if ((del_x1 == 0 && del_x2 == 0) ||
      (del_y1 == 0 && del_y2 == 0))
  {
    *fScalarL1 = FLT_MAX;
    *fScalarL2 = FLT_MAX;
    return ( MC2_LINES_ARE_PARALLEL );
  }
  // % There are two ways to calculate both scalers %
  // % If X1 = 0, use method 2 %
  // %-----------------------------------------------------------------%
```
if (fabs( (double) del_y1) > MC2_EPSILON)
    *fScalarL1 = (del_x2 * del_y1 * del_y12 + del_y1 * del_y2 * del_x12) / (del_y1 * denom);     // C in notes
else if (fabs( (double) del_x1) > MC2_EPSILON)
    *fScalarL1 = (del_x2 * del_x1 * del_y12 + del_x1 * del_y2 * del_x12) / (del_x1 * denom);
else
    return ( MC2_NO_INTERSECTIONS );

if (fabs( (double) del_y2) > MC2_EPSILON)
    *fScalarL2 = (del_x1 * del_y2 * del_y12 + del_y1 * del_y2 * del_x12) / (del_y2 * denom);     // D in notes
else if (fabs( (double) del_x2) > MC2_EPSILON)
    *fScalarL2 = (del_x2 * del_x1 * del_y12 + del_x2 * del_y1 * del_x12) / (del_x2 * denom);
else
    return ( MC2_NO_INTERSECTIONS );

// %----------------------------------------------------------%
// % Perform normal calculations %
// %----------------------------------------------------------%
//
if ((*fScalarL1 < 0.0 || *fScalarL1 > 1.0) || (*fScalarL2 < 0.0 || *fScalarL2 > 1.0))
    return ( MC2_NO_INTERSECTIONS );
else
    return ( NO_ERROR );
}
Appendix C: Formulation of Scalar Values

Basic Line Equations.

\[ fs_1 = \frac{v_3}{v_1} \]

Figure 39. Scalar Definitions

Figure 40. Equality of Scalars
Figure 40 shows that the scalar $f_{s1}$ can be calculated by examining the ratio of the delta $x$’s or $y$’s.

$$r^2 = x^2 + y^2$$

$$\cos(\phi) = \frac{x}{r} = \frac{x'}{r'}$$

$$f_{s1} = \frac{r'}{r} = \frac{x'}{x} = \frac{y'}{y}$$
Appendix C: (continued)

Using the start and end coordinates of each vector;

\[ y_{l1} = m_1 \cdot x_{l1} + b_i \]
\[ y_{l2} = m_1 \cdot x_{l2} + b_i \]
\[ y_{21} = m_2 \cdot x_{21} + b_2 \]
\[ y_{22} = m_2 \cdot x_{22} + b_2 \]

solve for the intersection \( y = y_3, x = x_3 \)

\[ 0 = [(m_1 \cdot x_{l1}) + b_i] - [(m_2 \cdot x_{21}) + b_2] \]
\[ [(m_1 \cdot x_{l1}) + b_i] = [(m_2 \cdot x_{21}) + b_2] \]

LHS
\[ = m_1 \cdot x_{l1} + b_i \]
\[ = m_1 \cdot x_{l1} + y_{l2} - m_1 \cdot x_{l2} \]
\[ = m_1 \cdot (x_{l1} - x_{l2}) + y_{l2} \]

RHS
\[ = m_2 \cdot (x_{21} - x_{22}) + y_{22} \]

combining
\[ m_1 \cdot (x_{l1} - x_{l2}) + y_{l2} = m_2 \cdot (x_{21} - x_{22}) + y_{22} \]
\[ m_1 \cdot (-\Delta x_1) + y_{l2} = m_2 \cdot (-\Delta x_2) + y_{22} \]
Appendix C: (continued)

Solving simultaneous equations for the intersection of the two vectors, v1 and v2;

\[ y_1 - y_2 = m_1x_1 + b_1 - (m_2x_2 + b_2) \]

but there is a common point, \( x_3, y_3 \)

\[ 0 = m_1x_3 + b_1 - m_2x_3 - b_2 \]

\[ -\left( m_1 - m_2 \right)x_3 = b_1 - b_2 \]

\[ x_3 = \frac{b_1 - b_2}{-\left( m_1 - m_2 \right)} \]

but \( b_i = y_1 \cdot m_1x_1, b_2 = y_2 \cdot m_2x_2 \)

\[ x_3 = \frac{(y_1 \cdot m_1x_1) - (y_2 \cdot m_2x_2)}{-\left( m_1 - m_2 \right)} \]

\[ x_3 = \frac{y_1 \cdot y_2 - \Delta y_1}{\Delta x_1} \cdot x_1 + \frac{\Delta y_2}{\Delta x_2} \cdot x_2 \]

\[ x_3 = \frac{y_1 \cdot y_2 - \Delta y_1}{\Delta x_1} \cdot x_1 + \frac{\Delta y_2}{\Delta x_2} \cdot x_2 \]

\[ x_3 = \frac{y_1 \cdot y_2 - \Delta y_1}{\Delta x_1} \cdot x_1 + \frac{\Delta y_2}{\Delta x_2} \cdot x_2 \]

\[ x_3 = \frac{y_1 \cdot (\Delta x_1 \Delta x_2) - y_2 \cdot (\Delta x_1 \Delta x_2) - \frac{\Delta y_1}{\Delta x_1} \cdot x_1 \cdot (\Delta x_1 \Delta x_2) + \frac{\Delta y_2}{\Delta x_2} \cdot x_2 \cdot (\Delta x_1 \Delta x_2)}{(-1)(\Delta y_1 \Delta x_2 - \Delta x_1 \Delta y_2)} \]

\[ x_3 = \frac{(\Delta x_1 \Delta x_2)(y_2 - y_1) + \Delta y_1 \Delta x_2x_1 - \Delta y_2 \Delta x_1x_2}{(\Delta y_1 \Delta x_2 - \Delta x_1 \Delta y_2)} \]
Appendix C: (continued)

Given \( x = x_3 \), calculate \( y \)

\[
y_3 = \frac{\Delta y_1}{\Delta x_1} x_3 + b_1
\]

\[
y_3 = \frac{\Delta y_1}{\Delta x_1} x_3 + y_{12} - \frac{\Delta y_1}{\Delta x_1} x_{12}
\]

\[
y_3 = \frac{\Delta y_1}{\Delta x_1} (x_3 - x_{12}) + y_{12}
\]

scalers \( fs_1 \) & \( fs_2 \)

\[
fs_1 = \frac{v_3}{v_1} = \frac{x_3 - x_{11}}{x_{12} - x_{11}} = \frac{y_3 - y_{11}}{y_{12} - y_{11}}
\]

\[
fs_2 = \frac{v_3}{v_1} = \frac{x_3 - x_{21}}{x_{22} - x_{21}} = \frac{y_3 - y_{21}}{y_{22} - y_{21}}
\]

from above:

\[
fs_1 = \frac{v_3}{v_1} = \frac{x_3 - x_{11}}{x_{12} - x_{11}} = \frac{x_3 - x_{11}}{\Delta x_1}
\]

and

\[
x_3 = \left( \Delta x_1 \Delta x_2 \right) \left( y_{21} - y_{11} \right) + \Delta y_1 \Delta x_2 x_{11} - \Delta y_2 \Delta x_1 x_{12} + \frac{\Delta y_1 \Delta x_2}{\Delta y_1 \Delta x_2 - \Delta x_1 \Delta y_2}
\]

\[
\frac{\left( \Delta x_1 \Delta x_2 \right) \left( y_{21} - y_{11} \right) + \Delta y_1 \Delta x_2 x_{11} - \Delta y_2 \Delta x_1 x_{12} - x_{11}}{\Delta y_1 \Delta x_2 - \Delta x_1 \Delta y_2}
\]

\[
fs_1 = \frac{\left( \Delta x_1 \Delta x_2 \right) \left( y_{21} - y_{11} \right) + \Delta y_1 \Delta x_2 x_{11} - \Delta y_2 \Delta x_1 x_{12} - x_{11}}{\Delta y_1 \Delta x_2 - \Delta x_1 \Delta y_2}
\]

\[
fs_1 = \frac{\left( \Delta x_1 \Delta x_2 \right) \left( y_{21} - y_{11} \right) + \Delta y_1 \Delta x_2 \left( x_{11} - x_{21} \right)}{\Delta y_1 \Delta x_2 - \Delta y_1 \Delta x_2}
\]

\[
similarly
fs_2 = \frac{\left( \Delta x_1 \Delta x_2 \right) \left( y_{21} - y_{11} \right) + \Delta y_1 \Delta x_2 \left( x_{21} - x_{11} \right)}{\Delta y_1 \Delta x_2 - \Delta y_1 \Delta x_2}
\]
Appendix C: (continued)

Solving for x’s rather than y’s will yield similar equations that can be used when one of
the lines is vertical, i.e. delta x is 0.

\[
x_1 - x_2 = \frac{y_3 - b_1}{m_1} - \frac{y_3 - b_2}{m_2}
\]

\[
x_1 = x_2 = x_3 \quad \text{and} \quad y_1 = y_2 = y_3
\]

\[
0 = \frac{y_3 m_2 - b_1 m_1 - y_3 m_1 + b_2 m_2}{m_1 m_2}
\]

\[
y_3 = \frac{b_1 m_2 - b_2 m_1}{m_2 - m_1}
\]

\[
f_{s1} = \frac{((\Delta y_1 \Delta x_2)(y_1 - y_2) + \Delta y_2 \Delta y_1 (x_2 - x_1))}{(\Delta x_1 \Delta y_2 - \Delta y_1 \Delta x_2) \Delta y_1}
\]

similarly

\[
f_{s2} = \frac{((\Delta x_1 \Delta y_2)(y_1 - y_2) + \Delta y_2 \Delta y_1 (x_2 - x_1))}{(\Delta x_1 \Delta y_2 - \Delta y_1 \Delta x_2) \Delta y_2}
\]
About the Author

Nicholas Mason received a Bachelor’s Degree in Physics from the University of Waterloo in 1985 and a Master’s Degree from the University of Florida in 1995. He started as a Junior Medical Physicist at Princess Margaret Hospital in Toronto, Canada in 1981. After working for eight years as a Medical Physicist, he decided that he needed a graduate degree in order to progress in his chosen profession.

Mr. Mason entered the Nuclear Engineering program at the University of Florida in 1988. He was admitted into the Ph.D. program, and transferred to the University of South Florida in 1995. He has continued to work as a Consulting Medical Physicist as well as co-founding a corporation, MC2 Scientific Systems. In developing a software product for MC2, Mr. Mason saw the necessity for a better testing method for treatment planning and virtual simulation systems which was the main stimulus for his Ph.D. topic.