ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

АНАЛИЗ И ИССЛЕДОВАНИЕ СРЕДСТВ ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ В СТОМАТОЛОГИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Магистерская диссертация
обучающегося по направлению подготовки
09.04.02 Информационные системы и технологии
очной формы обучения
группы 07001635
Лихачева Владислава Сергеевича

Научный руководитель
к.т.н., доцент
Шамраева Е.О.

Рецензент
к.т.н., доцент
Путивцева Н.П.

БЕЛГОРОД 2018

РЕФЕРАТ

Анализ и исследование средств представления информации в стоматологических информационных системах. – Лихачев Владислав Сергеевич, магистерская диссертация, Белгород, Белгородский государственный национальный исследовательский университет (НИУ «БелГУ»), количество страниц 62, включая приложения 88, количество рисунков 45, количество таблиц 2, количество использованных источников 30.

ТОМОГРАММА, DICOM-ФАЙЛ, СТОМАТОЛОГИЧЕСКАЯ ИНФОРМАЦИОННАЯ СИСТЕМА, VATECH, PLANMECA, SIRONA, GENDEX.

Объектом исследования является унификация представления стоматологической информации.

Предметом исследования являются алгоритмы преобразования данных томографов различных производителей.

Цель исследования: унификация представления стоматологической информации путем разработки автоматизированной системы анализа и преобразования данных томографов различных производителей (АС АПТД).

Задачи исследования:

– проведение анализа существующих стоматологических информационных систем;

– проектирование автоматизированной системы анализа и преобразования томографических данных;

– разработка и тестирование автоматизированной системы анализа и преобразования томографических данных.

Научная новизна заключается в разработке алгоритмов преобразования данных, содержащихся в DICOM-файлах различных производителей.

Методы исследования: методы обработки DICOM-файлов.

Полученные результаты: была решена актуальная научно–практическая задача унификации представления стоматологической информации путем разработки автоматизированной системы анализа и преобразования данных томографов различных производителей.

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ 5

ВВЕДЕНИЕ 6

1 Анализ стоматологических информационных систем 9

1.1 Обзор томографических информационных систем 9

1.2 Обзор особенностей стандарта DICOM 20

1.3 Постановка задачи исследования 29

2 Проектирование автоматизированной системы анализа и преобразования томографических данных 31

2.1 Анализ бизнес-процессов работы отдела компьютерной томографии 31

2.2 Проектирование работы отдела компьютерной томографии с использованием АС АПТД 34

2.3 Проектирование модели автоматизированной системы анализа и преобразования томографических данных 36

3 Программная реализация и тестирование автоматизированной системы анализа и преобразования томографических данных 41

3.1 Программная реализация АС АПТД 41

3.2 Тестирование АС АПТД 52

ЗАКЛЮЧЕНИЕ 57

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 59

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ПО – программное обеспечение;

DICOM – Digital Imaging and Communications in Medicine – формирование, передача и хранение медицинских изображений;

КЛК – конусно-лучевой компьютерный;

ТИС – томографическая информационная система;

MIP – проекция минимальной/средней/максимальной интенсивности;

ОПТГ – ортопантомограмма;

ISO – International Organization for Standardization – Международная организация по стандартизации;

SOP – Service-Object Pair – пара операция - объект;

RLE – Run-Length Encoding – кодирование длин серий;

АС АПТД – автоматизированная система анализа и преобразования данных томографов;

МЦСИ – межрегиональный центр стоматологических инноваций.

ВВЕДЕНИЕ

Современное медицинское оборудование позволяет выполнять обследование человека с использованием различных методик диагностики. Качественный рост методов диагностики и разработка соответствующих медицинских приборов привели к необходимости разработки специального формата данных, который поддерживался бы различными разработчиками аппаратуры и программного обеспечения (ПО), что позволило бы выполнять комплексные обследования пациентов. При этом спектр разрабатываемой аппаратуры достаточно широк, и значительная ее часть позволяет получить двумерные или трехмерные изображения исследуемых органов. Зачастую трехмерные изображения основываются на дискретных срезах (двумерных изображениях), соответственно необходимы современные информационные технологии для обработки, визуализации и организации хранения медицинских данных [1].

Для решения указанной выше проблемы разработан стандарт данных Digital Imaging and Communications in Medicine (DICOM), позволяющий обрабатывать и хранить как одиночные изображения, так и их объединения [2].

В различных стоматологических медицинских учреждениях России, используются томографы и программное обеспечение различных производителей. Все они сохраняют данные в DICOM-файлах, но программное обеспечение, используемое в одном медицинском учреждении, не способно работать с файлами, полученными из других учреждений из-за того, что в стандарте не указан единый вариант хранения данных.

В связи с этим, некоторые производители томографов используют свои собственные варианты хранения данных, которые хоть и основаны на стандарте DICOM, но не используют его. Кроме различия в варианте хранения данных, отличаются и алгоритмы сжатия пиксельных изображений, которые

используются для хранения. Данные пикселей могут сжиматься с использованием множества стандартов, включая JPEG, JPEG Lossless (без потерь), JPEG 2000 и RLE.

Таким образом, программное обеспечение, используемое в учреждениях для просмотра снимков с томографа, не является универсальным. Данное ПО будет работать только с файлами, полученными с томографа в конкретном учреждении, так как ПО и томограф разработаны одним производителем. В результате программное обеспечение, используемое в учреждении, не сможет работать с файлами, которые принесет пациент из другого учреждения. Таким образом возникает проблема совместимости между ПО, используемыми в разных медицинских учреждениях.

Идеальным вариантом для предотвращения такой ситуации является разработка нового универсального ПО, которое могло бы работать со всеми видами томографов, но, к сожалению, такой вариант будет очень дорогостоящим. Также производители томографов будет терять значительную часть прибыли, ведь при существовании универсального ПО, разработанное производителем не будет продаваться.

Еще одним выходом из данной ситуации является разработка отдельной системы, которая будет работать с DICOM-файлами разных производителей.

Объектом исследования является унификация представления стоматологической информации.

Предмет исследования – алгоритмы преобразования данных томографов различных производителей.

Целью магистерской диссертации является унификация представления стоматологической информации путем разработки автоматизированной системы анализа и преобразования данных томографов различных производителей.

Для достижения поставленной цели необходимо решить следующие задачи:

– провести анализ существующих стоматологических информационных систем;

– спроектировать автоматизированную систему анализа и преобразования томографических данных;

– разработать и протестировать автоматизированную систему анализа и преобразования томографических данных.

Научная новизна данной диссертационной работы заключается в разработке алгоритмов преобразования данных, содержащихся в DICOM-файлах различных производителей.

В первом разделе описан сравнительный анализ существующих томографических информационных систем, а также обзор стандарта DICOM и структуры DICOM-файла. В конце первого раздела поставлена задача о необходимости разработки автоматизированной системы анализа и преобразования данных томографов различных производителей с целью унификации структуры DICOM-файла.

Во втором разделе проводится анализ бизнес-процессов работы отдела компьютерной томографии. При помощи CASE-средств была спроектирована функциональная модель автоматизированной системы анализа и преобразования томографических данных.

В третьем разделе магистерской диссертации представлена программная реализация автоматизированной системы анализа и преобразования данных томографов различных производителей, а также ее тестирование.

Магистерская диссертация включает 61 страницу, 45 рисунков, 2 таблицы и приложения.

1 Анализ стоматологических информационных систем

1.1 Обзор томографических информационных систем

Развитие компьютерной томографии в современной стоматологии, челюстно-лицевой хирургии и оториноларингологии непрерывно идет вперед. Ведущие компании – производители рентгеновского оборудования на данный момент создали специализированные томографы для зубочелюстной системы, способные выделять слои тканей до 0,1 мм за счет высокой разрешающей способности плоскопанельного детектора. Практически все конусно-лучевые компьютерные (КЛК) томографы позволяют получить высококачественные исходные DICOM-файлы, из которых реконструируется изображение. Это говорит о том, что принципы получения и обработки изображения, конструкция и технические характеристики сенсоров КЛК-томографов достигли пика своего развития и ждать существенных изменений качества снимков на ближайшие 5-10 лет не придется [3]. Поэтому многие производители вектором развития своего оборудования выбрали совершенствование программного обеспечения аппаратов. Такое решение позволило расширить границы использования компьютерной томографии в стоматологической практике. На современном этапе стоматолог на основании данных компьютерной томографии не только оценивает состояние тканей зубочелюстной системы, но и получает с помощью дополнительных функций более узкую информацию, которая необходима для детального планирования лечения [4].

Производителей компьютерных томографов на рынке стоматологического оборудования на данный момент огромное количество. С одной стороны, это хорошо для развития технологий, так как высокая конкуренция среди компаний ведет к появлению новых технических средств и улучшению уже имеющихся. С другой стороны, каждая компания к своему

томографу обязательно выпускает оригинальное программное обеспечение, чтобы в дальнейшем активно его продвигать и улучшать. Следовательно, врач-стоматолог, желая постоянно учиться и развиваться, сталкивается с со следующими проблемами:

– выбор и поддержка конкретного производителя томографов;

– сложность в выборе программного обеспечения, наиболее подходящего для нужд конкретного врача;

– необходимость изучения различного ПО для выявления особенностей каждого из них.

В связи с этим необходимо рассмотреть наиболее популярные в России модели томографов и соответствующее им программное обеспечение, которые уже получили признание на территории России и с которыми работают многие стоматологи.

Сейчас на стоматологическом рынке в России чаще всего встречаются следующие производители оборудования для конусно-лучевой компьютерной томографии (в скобках указаны названия томографических информационных систем для просмотра и редактирования данных соответствующего томографа) [5]:

– Vatech (Ez3D2009);

– Sirona (Galaxis);

– Planmeca (Romexis);

– Gendex (I-Cat Vision).

Чтобы определить, каким функционалом и какими отличиями обладают томографические информационные системы (ТИС) каждого из приведенных выше производителей, необходимо сравнить их по следующим критериям:

– системные требования к компьютеру;

– система координат;

– 3D-модель;

– панорамная реконструкция зубочелюстной системы;
– дополнительные функции.

1.1.1 Системные требования к компьютеру

Каждая из программ способна запуститься практически на любом компьютере, но при этом скорость работы ТИС, а, следовательно, и скорость диагностики существенно отличаются у каждого производителя.

В компании Sirona смогли адаптировать работу ТИС до такой степени, что при запуске программы даже на слабом компьютере с объемом оперативной памяти 2 Гб нет никаких проблем при обработке исследования.

Ez3D2009 от Vatech также работает без проблем на достаточно слабых компьютерах с объемом оперативной памяти 2 Гб, но при переходе с одного активного окна на другое иногда возникает задержка изображения [6].

Совершенно другая ситуация с компаниями Gendex и Planmeca. I-Cat Vision, также как Vatech и Sirona, может запуститься на компьютере с небольшим объемом оперативной памяти и малоскоростной видеокартой. Однако при просмотре изображения будет происходить задержка пролистывания снимков от долей до нескольких секунд, что значительно тормозит скорость оценки КТ.

Что касается Planmeca, компания большое внимание уделила постобработке изображения, что привело к серьезному увеличению системных требований. В связи с этим минимальное требование по оперативной памяти составляет 4 Гб, иначе программа может работать некорректно [7].

1.1.2 Интерактивность системы координат

Одной из самых важных особенностей программного обеспечения КЛК-томографов стала возможность вращать оси координат реформатов относительно заданной точки, что позволило получать косые срезы, необходимые для достоверной оценки зуба во всех плоскостях. Известно, что зуб имеет свою собственную систему координат, которая редко совпадает с антропометрической системой тела. Поэтому оси координат мультипланарной реконструкции нужно выстраивать в соответствии с наклоном длинной оси зуба во всех проекциях. Возможность изменения наклона осей координат относительно точки вращения была названа интерактивностью [8].

По этому параметру первое место занимает Ez3D2009, которая обладает истинной интерактивностью осей координат, которая изображена на рисунке 1.1. При наведении курсора мыши на дистальное плечо любой оси координат появляется значок вращения, который позволяет развернуть оси под любым углом.

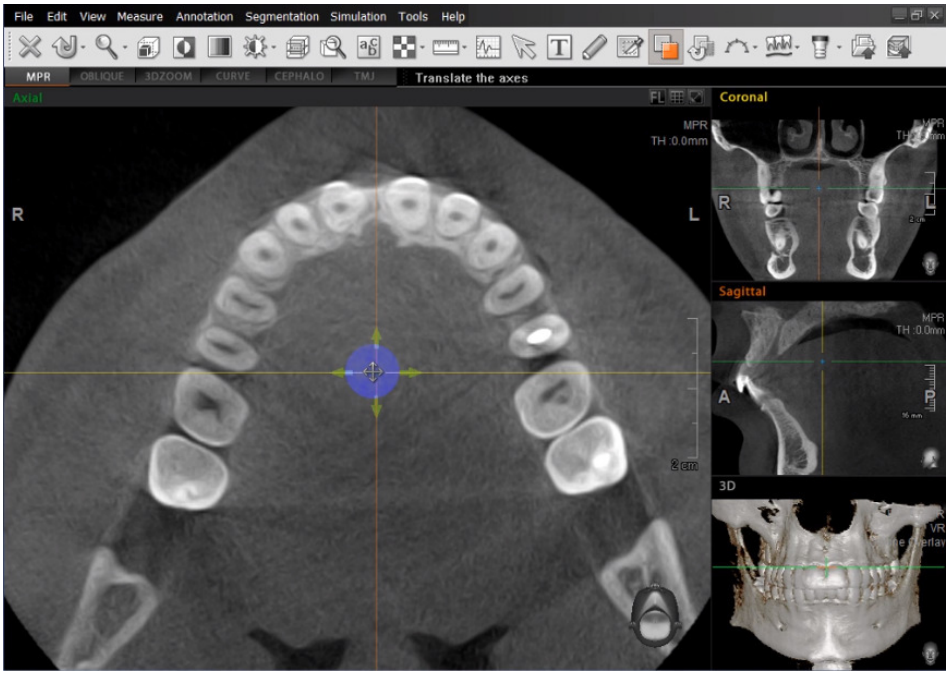

Рисунок 1.1 – Интерактивность осей координат

В остальных программах истинная интерактивность осей не реализована. Она была заменена на так называемую псевдоинтерактивность, изображенную на рисунке 1.2, когда оси координат остаются неподвижными, а функцией вращения обладает само изображение [8].

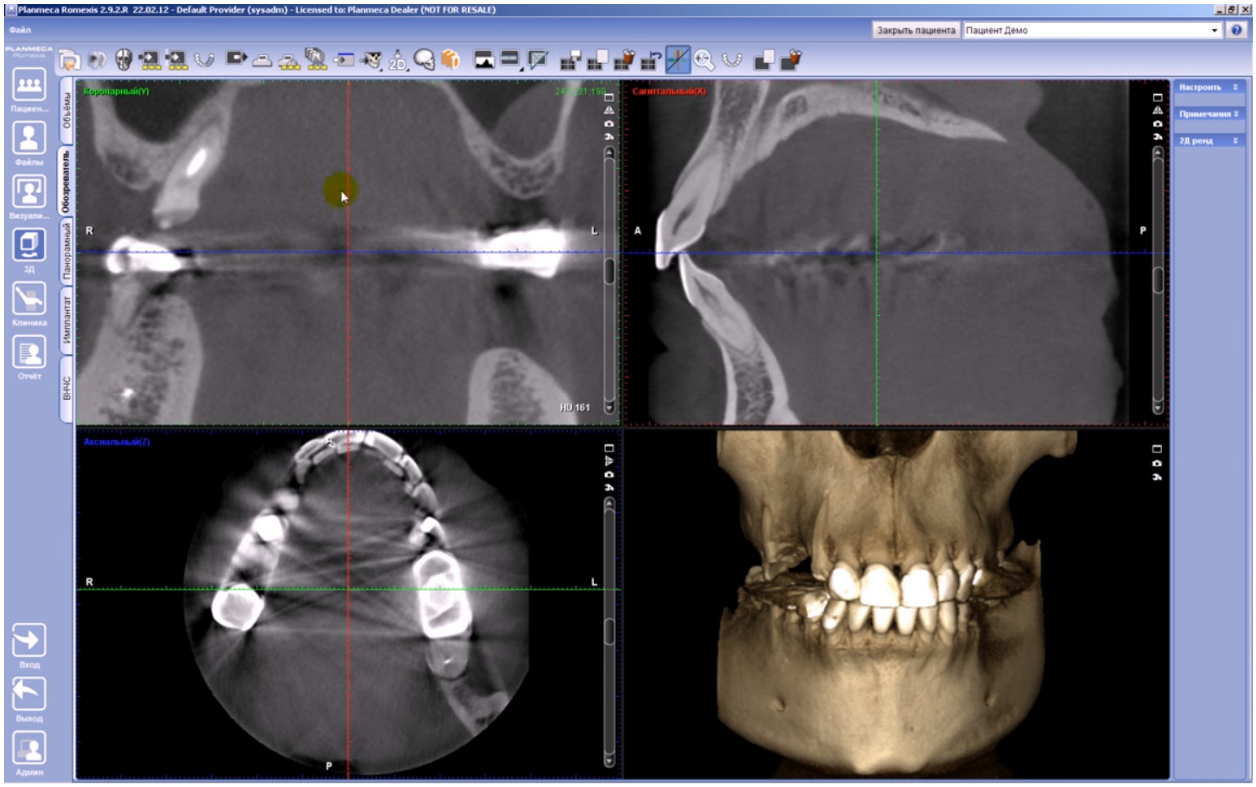

Рисунок 1.2 – Псевдоинтерактивность осей координат

В Romexis данная опция работает через правую кнопку мыши, зажав которую в любом месте изображения, можно вращать снимок относительно осей координат.

По такому же принципу функция вращения осей координат реализована в I-Cat Vision. Единственное отличие от Romexis заключается в том, что активируется данная функция только при наведении курсора мыши на правый нижний угол MPR-реформата, что менее удобно, чем в Romexis.

Отдельно от остальных программ стоит Galaxis. Здесь интерактивность была создана частично, то есть вращать оси координат возможно только в двух

из трех MPR-реформатах – в сагиттальном и коронарном [9]. Причем оси перемещаются за счет ползунка, который находится справа от каждого реформата, что не является лучшим решением. При этом смещая один из ползунков, вращение оси происходит на другом срезе.

1.1.3 Построение 3D-модели объекта

3D-модель дает преимущество для демонстрации патологических изменений пациенту. Пример 3D-модели представлен на рисунке 1.3. Особенно актуально для оториноларингологов и челюстно-лицевых хирургов.

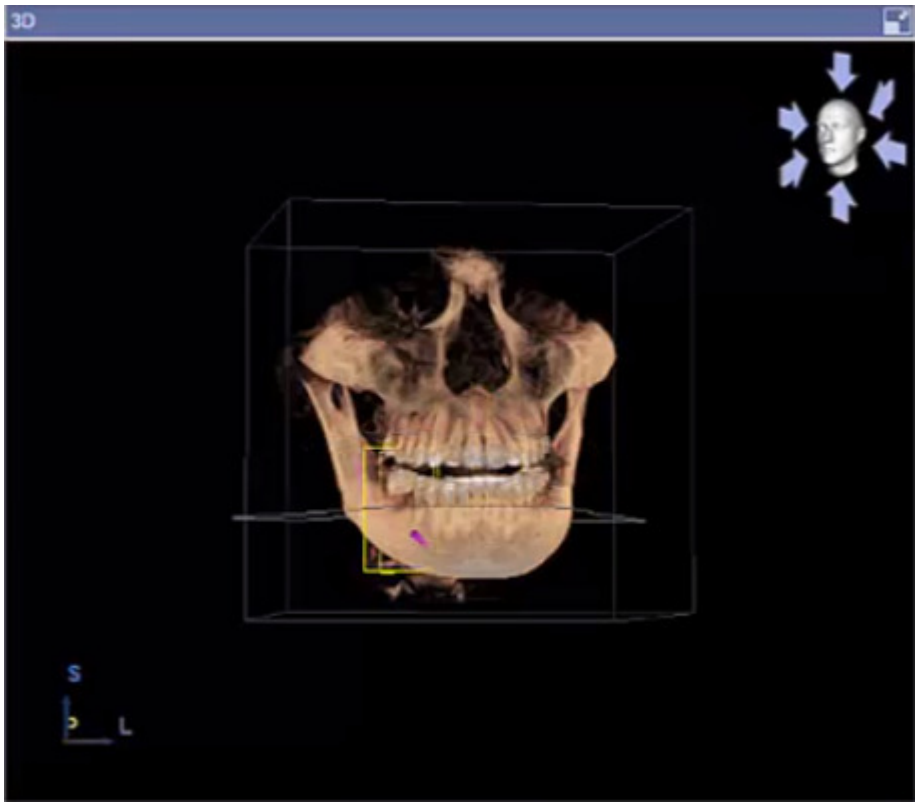

Рисунок 1.3 – 3D-модель

В Galaxis и I-Cat Vision 3D-модель используется только как объемное изображение костных структур пациента в режиме проекции минимальной/средней/максимальной интенсивности (MIP) без каких-либо

дополнительных возможностей [10]. Единственное отличие между ними – это более реалистичное изображение объемной модели в I-Cat Vision, чем в Galaxis.

Vatech и Planmeca уделили больше внимания 3D-модели, чем остальные производители. В Ez3D2009 и Romexis объемная модель выполняет функцию не только отображения костных структур, но и позволяет просматривать изображение в различных режимах, таких как MIP (как в Galaxis и I-Cat Vision), костный режим, представленный на рисунке 1.4, мягкотканный режим, представленный на рисунке 1.5, и некоторые другие.

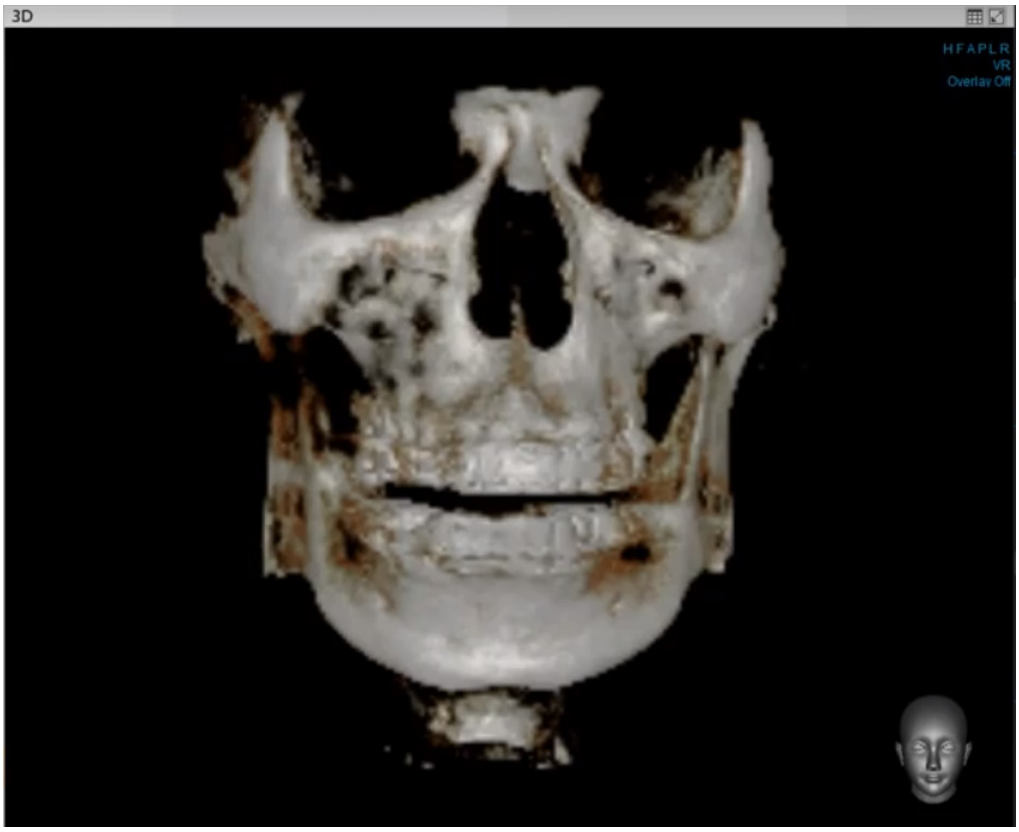

Рисунок 1.4 – Костный режим отображения 3D-модели

Также в данных программах имеется возможность на 3D-модели выделять интересующую область из массива тканей пациента, что позволяет более детально изучить зону интереса [11].

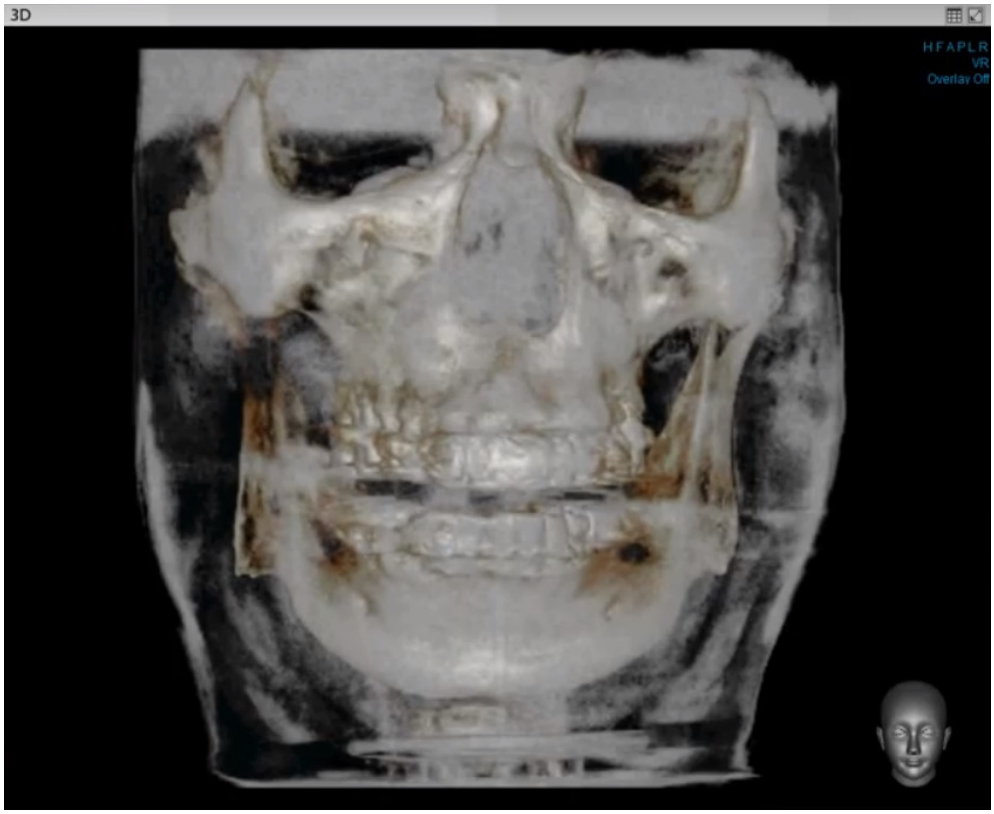

Рисунок 1.5 – Мягкотканный режим отображения 3D-модели

Несмотря на то, что функции 3D-модели в Ez3D2009 и Romexis реализованы практически на одном уровне, компания Planmeca создала наиболее реалистическую модель, чем все остальные известные программы, что выражено в высокой степени дифференциации мягких и костных тканей.

1.1.4 Панорамная реконструкция зубочелюстной системы

Так же, как и 3D-модель, функция панорамной реконструкции не является основной, но помогает врачам в диагностике. Эталоном диагностической стоматологии считается ортопантомограмма (ОПТГ). Это панорамный снимок обеих челюстей, который показывает состояние пародонта, костной ткани, зубных корней, гайморовых пазух и височно-нижнечелюстных суставов [12]. Здесь разработчики программного обеспечения пошли двумя путями – автоматическое и ручное построение.

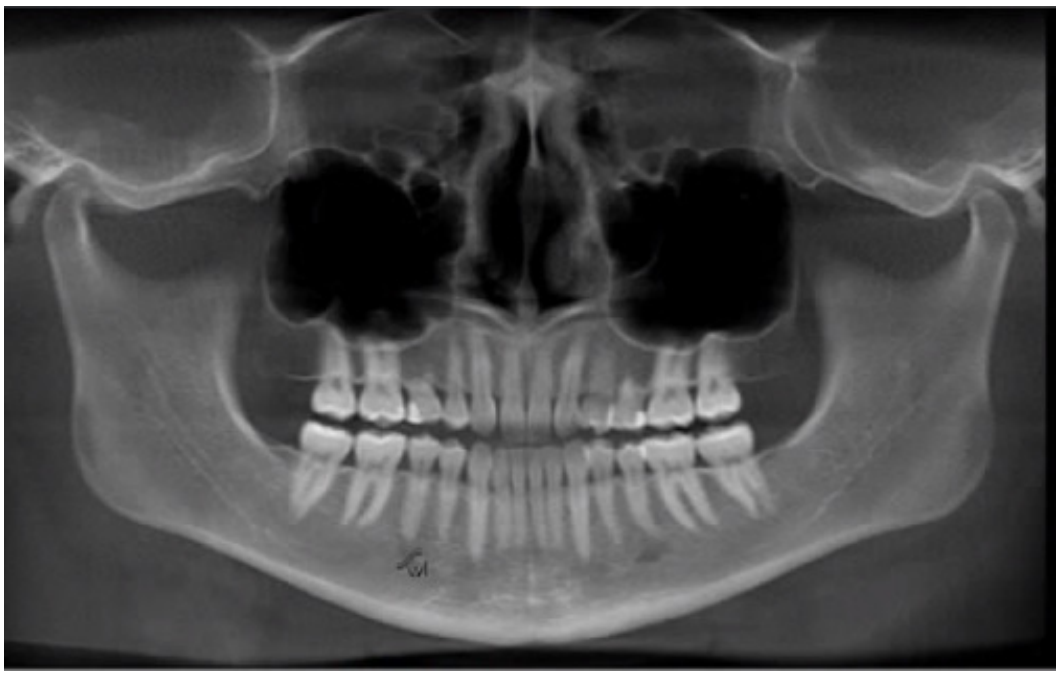

Рисунок 1.6 – отображение ОПТГ в Sirona Galaxis

Sirona и Gendex вынесли панорамную томограмму на начальный экран, и при загрузке программы пользователь сразу видит автоматически созданное изображение, что для многих является важным фактором удобства. В обеих программах панорамная томограмма реализована очень хорошо, ничуть не уступая стандартной ОПТГ, но в I-Cat Vision в некоторых случаях ее необходимо дополнительно подстраивать, в отличие от Galaxis.

Создатели Ez3D2009 и Romexis позволили пользователю, т.е. врачу, самостоятельно выстраивать панорамную томограмму в соответствии с индивидуальными анатомическими особенностями. Несмотря на то, что панорамная реконструкция в программе Romexis практически соответствует обыкновенной ОПТГ, Ez3D2009 способна быстро и удобно создавать панорамные реформаты отдельно верхней и нижней челюстей, сегмента и менее протяженного участка челюсти, между которыми можно переключаться. Панорамная томограмма, построенная в Ez3D2009 отображена на рисунке 1.7.

Рисунок 1.7 – Панорамная томограмма, построенная в Ez3D2009

1.1.5 Дополнительные функции

Одной из главных особенностей каждой ТИС является наличие дополнительных функций, которые, во-первых, отражают индивидуальность любого программного обеспечения, во-вторых, облегчают и улучшают диагностику.

Здесь среди программ существует большая вариабельность, хотя есть и определенные закономерности. Обособленно от всех в данном случае стоит Galaxis, в которой, несмотря на реализацию некоторых функций, возможность их использования закрыта, так как фирма Sirona заблокировала дополнительные возможности для пользователей, которые не имеют своего томографа [13].

В остальных программах существуют общие для всех дополнительные функции: окно кросс-секции для планирования имплантации, изменения толщины выделенного слоя для дифференциации патологических изменений, фильтры жесткости для улучшения качества изображения, возможность сохранять исследования для создания базы пациентов.

При этом Ez3D2009 и Romexis имеют более развитую систему планирования имплантации, позволяющую не только измерить параметры костной ткани в зоне будущей имплантации, но и выделить нижнечелюстной канал, виртуально установить имплантаты и учитывать возможные риски осложнения операции. Если сравнить количество доступных пользователю дополнительных опций, то здесь выигрывает Ez3D2009, в которой помимо названных уже функций существует несколько уникальных: 3D-Zoom, Oblique, рисование, использование различных режимов для каждого окна, консультант и т. д. [14].

Анализируя наиболее распространенные ТИС в России, можно прийти к следующим выводам:

– практически все рассмотренные ТИС находятся примерно на одном функциональном уровне;

– Ez3D2009 выделяется на общем фоне за счет наличия большого количества дополнительных опций;

– Romexis и Galaxis имеют некоторый дисбаланс по функциям: по некоторым показателям каждая из программ является ведущей, но по другим занимает последнее место.

– I-Cat Vision уверено занимает промежуточное положение по всем показателям, демонстрируя тем самым, что потенциально программа может быть усовершенствована при большей реализации функций;

– каждая из ТИС работает только с данными, полученными от соответствующего ей томографа;

– каждая ТИС поставляется только в комплекте с соответствующим томографом и отдельно не продается.

1.2 Обзор особенностей стандарта DICOM

DICOM (англ. Digital Imaging and Communications in Medicine) – отраслевой стандарт создания, хранения, передачи и визуализации медицинских изображений и документов обследованных пациентов. DICOM опирается на ISO-стандарт OSI, поддерживается основными производителями медицинского оборудования и медицинского программного обеспечения [15].

Стандарт DICOM, разрабатываемый Национальной ассоциацией производителей электронного оборудования (National Electrical Manufacturers Association), позволяет создавать, хранить, передавать и печатать отдельные кадры изображения, серии кадров, информацию о пациенте, исследовании, оборудовании, учреждениях, медицинском персонале, производящем обследование, и т.д. [16]

1.2.1 Структура стандарта DICOM

Стандарт DICOM разделен на связанные, но отдельные части. Каждый DICOM-документ имеет свой заголовок и номер в форме: PS 3.X-YYYY, где X – номер части стандарта и YYYY – ее год публикации [17]. Например, PS 3.1-1996 обозначает первую часть стандарта DICOM 3.0, выпущенная в 1996 году.

Текущая версия стандарта – 2008, включает следующие части:

– PS 3.1: Introduction and Overview. Введение и краткий обзор – описываются история разработки стандарта, его назначение и структура;

– PS 3.2: Conformance. Соответствие стандарту – указываются структура сертификата соответствия стандарту и критерии совместимости диагностического оборудования со стандартом DICOM.

– PS 3.3: Information Object Definitions. Определение информационных объектов – предлагается информационная модель «реального мира», описывающая взаимоотношения между нормализованными объектами (пациент, устройство) и составными объектами (исследования, изображения и др.), наследующими некоторые атрибуты нормализованных объектов.

– PS 3.4: Service Class Specifications. Спецификации классов операций – специфицируются классы действий или операций, которые могут выполняться над информационными объектами. Вводится понятие операция-объект SOP (service-object pair). Разработчики стандарта исходили из того, что применение операции к объекту может быть ограничено его свойствами, поэтому есть необходимость в отдельном описании классов SOP.

– PS 3.5: Data Structure and Encoding. Структура и семантика данных – описываются типы данных и правила кодирования, используемые при передаче данных из одной системы в другую. Специфицируются форматы передачи изображений. Стандарт допускает передачу исходных и уплотненных изображений; особо описывается синтаксис передачи при использовании неискажающих и искажающих алгоритмов уплотнения JPEG. Допускаются другие, не специфицируемые в стандарте алгоритмы уплотнения.

– PS 3.6: Data Dictionary. Словарь данных – приводится полный список элементов данных, описанных в стандарте DICOM.

– PS 3.7: Message Exchange. Обмен сообщениями – описывается структура команд и протокола обмена сообщениями. Изложение этой части существенно опирается на соответствующие стандарты OSI (ISO 8222 и ISO 8649).

– PS 3.8: Network Communication Support for Message Exchange. Обеспечение обмена сообщениями в сетевых средах. определяются

необходимые компоненты системы обмена сообщениями в сетевых средах, использующих протокол TCP/IP.

– PS 3.9: Point-to-Point Communication Support for Message Exchange. Обеспечение обмена сообщениями при прямой связи – приводится подробное описание прямого взаимодействия двух устройств, в т.ч. назначение каждой ножки 50-контактного разъема, уровня передаваемых сигналов, их временные характеристики и т.д.

– PS 3.10: Media Storage and File Format for Data Interchange. Носители данных и форматы файлов для обмена информацией – описываются теоретические основы хранения медицинских изображений на внешних носителях данных.

– PS 3.11: Media Storage Application Profiles. Прикладные характеристики хранения данных на внешних носителях – описываются требования к данным, которые должны храниться на внешних носителях, описания имеют клиническую направленность.

– PS 3.12: Storage Functions and Media Formats for Data Interchange. Форматы носителей и физическая среда хранения данных – специфицируются различные носители данных, которые могут использоваться для хранения медицинских изображений.

– PS 3.13: Print Management Point-to-Point Communication Support. Управление выводом на печатающие устройства при прямом соединении – описываются протоколы и операции, необходимые для вывода изображения на печатающее устройство. Вывод осуществляется системой-исполнителем, напрямую соединенной с системой-инициатором вывода.

– PS 3.14: Grayscale Standard Display Function. Стандарт функций отображений полутоновых изображений – определяет стандарт для непротиворечивого отображения полутоновых изображений. Также включает функции калибровки отображающих систем (мониторов).

– PS 3.15: Security and System Management. Безопасность и система управления профилями – определяет профили безопасности, которые должны поддерживать реализации DICOM. Определяются профили безопасности для использования схем кодирования данных, открытых ключей и SMART-карт.

– PS 3.16: Content Mapping Resource. Отображение содержания ресурса – определяет шаблоны для структурирования документов как информационных объектов DICOM, устанавливает кодирование терминов, используемых информационными объектами, задает специализированный перевод закодированных терминов в зависимости от страны.

– PS 3.17: Explanatory Information. Объяснительная информация – содержит дополнительную информацию в виде одного нормативного приложения и 20 информативных приложений. Объясняются положение пациента, маммографические CAD, ультразвуковые шаблоны, отчеты о эхокардиографических процедурах, варианты использования офтальмологии и т.д.

– PS 3.18: Web Access to DICOM Persistent Objects (WADO). Интернет-доступ к постоянным объектам DICOM (WADO) – определяет интернет сервисы для доступа и представления постоянных объектов DICOM, таких как изображения и медицинские отчеты. Он предназначен для распределения результатов и изображений медицинским работникам. WADO обеспечивает простой механизм для доступа к постоянным объектам DICOM.

1.2.2 Структура DICOM-файла

Файл, хранящий одно изображение в стандарте DICOM, является сложной структурой данных, включающей в себя не только непосредственно изображение, но и сопутствующую информацию, такую как:

– данные об оборудовании, на котором проводилось исследование;

– описание проведенного исследования;

– параметры и описание серии исследования;

– данные о системе координат, связанной с изображением;

– атрибуты, определяющие само изображение;

– текстово-графические элементы, графики и комментарии, выполняемые медицинским персоналом;

– атрибуты, описывающие преобразования над полученными данными, и т.д.

Отдельный DICOM-файл содержит как заголовок, так и все данные об изображении, которые могут содержать информацию в трех измерениях, в одном файле. DICOM-изображения могут быть сжаты для уменьшения размера файла с помощью алгоритмов сжатия JPEG с потерей или без потерь данных, а также алгоритмом кодирования длин серий RLE (Run-Length Encoding) [18].

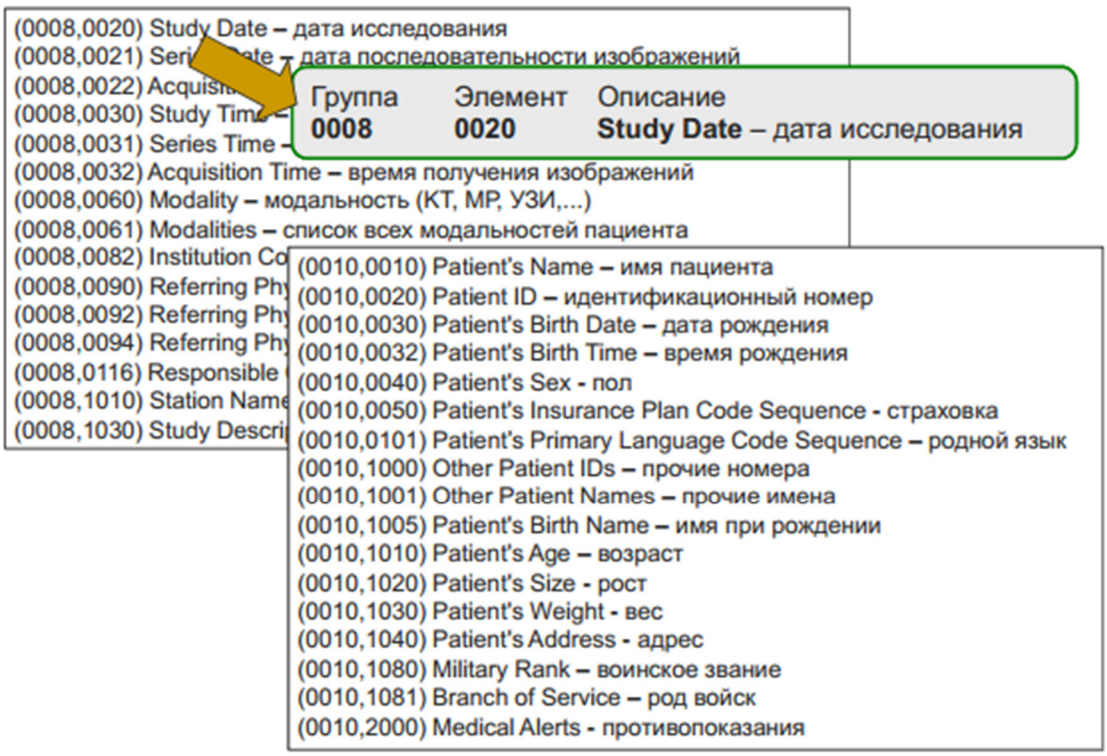

Рисунок 1.8 – Словарь тэгов

Любой DICOM-файл состоит из множества атрибутов, таких как имя пациента, его идентификатор, дата исследования и т.д. Также один, особенный, атрибут содержит данные изображения (pixel data). Таким образом, не существует какого-то отдельного заголовка у DICOM файла – только множество атрибутов, включающих и данные изображения.

Атрибуты в стандарте называются тэгами (tags), каждому тэгу присвоен свой номер, состоящий из двух полей – номера группы и номера элемента. Например, в тэге с номером (0010, 0010) (номера тэгов записываются в шестнадцатеричной нотации) всегда содержатся данные о ФИО пациента. У каждого тэга есть стандартное название, например, (0010, 0010) называется «Patient's Name» [19]. Список всех стандартизованных тэгов, который содержит более 3000 элементов, можно посмотреть в 6-м разделе стандарта PS 3.6.: Data Dictionary. Пример списка тэгов представлен на рисунке 1.8.

Тэг (7FFE, 0010) «Pixel Data» может содержать в себе одно или несколько изображений. В случае, когда Pixel Data содержит больше одного изображения, означает, что файл содержит мультифреймовое изображение (multi-frame image). В случае мультифреймового изображения в одном файле будет содержаться трехмерное изображение. Цифровые рентгеновские аппараты, цифровые считыватели кассет отдают информацию в виде однофреймового изображения. Аппараты УЗИ, ангиографы часто отдают мультифреймовые изображения. Старые рентгеновские и магнитно-резонансные томографы могли отдавать только однофреймовые изображения. Современные томографы могут отдавать как однофреймовые, так и мультифремовые изображения.

Данные изображения могут быть цветными и монохромными. Цветные могут быть в разной цветовой кодировке – RGB, YBR, Palette Color (цветная палитра). Монохромные могут быть разной глубины градации серого (1 – 16 бит). Данные изображения также могут быть упакованными. Стандартизованы следующие алгоритмы упаковки: RLE, JPEG, JPEG Lossless, JPEG 2000 [20].

Для всего файла может быть применена упаковка его с помощью алгоритма LZW, однако реализации такой упаковки в программах и оборудовании редки.

DICOM использует три различные схемы кодировки тэгов (transfer syntax) [21]:

– явное представление данных с длиной в 4 байта;

– явное представление данных с длиной в 2 байта;

– неявное представление данных.

Кодировка файла помечается соответствующим тэгом в этом же файле. Схемы получаются из комбинаций двух параметров – представления данных и кодировки порядка байт.

Представление данных может быть явным (Explicit) и неявным (Implicit). Нужно знать, как интерпретировать данные, содержащиеся в тэге, т.к. это простая последовательность байт. А какие именно данные там находятся – строка, число, либо последовательность тэгов (SQ-sequence) заранее не известно. Для определенности содержимое каждого тэга было стандартизовано. DICOM использует 27 стандартизированных представлений данных, именуемых в нем как VR (Value Representations). Форматы данных обозначаются 2-буквенными именами. Значение каждого из тэгов использует один из форматов данных, представленных в таблице 1.1 [22]. В зависимости от формата, значение может быть дополнено пробелами для четной длины данных. При явном представлении данных в тэгах явно записывается VR тэга. При неявном представлении VR не записывается, а берется из таблицы программы, которая работает с этим изображением.

Таблица 1.1 – Форматы данных

Формат данных	Описание
AE	Представление объекта в виде строки
AS	Строка, содержащая возраст

Продолжение таблицы 1.1

Формат данных	Описание
AT	Атрибут тэга
CS	Строка символов
DT	Дата и время
FL	Двоичное число с плавающей точкой одинарной точности
FD	Двоичное число с плавающей точкой двойной точности
IS	Строка, содержащая десятичные числа
LO	Символьная строка
LT	Символьная строка, содержащая один или несколько абзацев с максимальной длиной 10240 символов
OB	Поток байт, в котором кодирование содержимого представлено в 7-м разделе стандарта
OF	Поток 32-разрядных слов с плавающей точкой
OW	Поток 16-разрядных слов с плавающей точкой
PN	Символьная строка с описанием имени пациента
SH	Символьная строка длиной 16 символов
SL	Двоичное целое число длиной 32 бита
SQ	Последовательность элементов
SS	Двоичное целое число длиной 16 бит
ST	Символьная строка, содержащая один или несколько абзацев с максимальной длиной 1024 символа
TM	Время
UI	Строка символов, содержащая идентификатор
UL	Беззнаковое двоичное целое число длиной 32 бита
UN	Строка байт, где кодирование содержания неизвестно

Продолжение таблицы 1.1

Формат данных	Описание
US	Беззнаковое двоичное целое число длиной 16 бит
UT	Символьная строка, содержащая один или несколько абзацев с неограниченной максимальной длиной

Т.о., каждый тэг состоит из:

– номера группы (2 байта);

– номера элемента (2 байта);

– VR (2 байта, в явном представлении данных, в неявном не используется);

– размера блока данных в тэге (2 или 4 байта, в зависимости от VR);

– данных.

Структура тэга при явном и неявном представлениях данных приведены на рисунках 1.9-1.11.

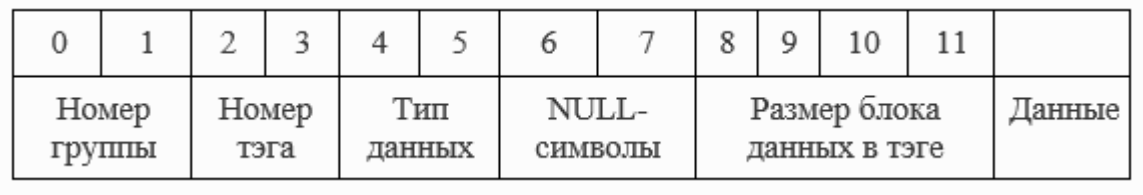

Рисунок 1.9 – Явное представление данных с длиной в 4 байта

Рисунок 1.10 – Явное представление данных с длиной в 4 байта

0	1	2	3	4	5	6	7	
Номер группы		Номер тэга		Размер блока данных в тэге				Данные

Рисунок 1.11 – Явное представление данных с длиной в 4 байта

На рисунках, представленных выше, количество байтов под данные не указано, т.к. их размер определяется значением, указанным в соответствующей части структуры тэга.

1.3 Постановка задачи исследования

В связи с тем, что DICOM-файл для различных томографов имеет одинаковый формат, но различную структуру, медицинские учреждения могут полноценно работать только с теми данными, которые получены с помощью томографа конкретного производителя, имеющегося в этом учреждении. Проблема возникает в неправильной интерпретации томографических данных томографами различных производителей. Существует несколько решений данной проблемы:

– использовать томографы и ТИС только одного конкретного производителя;

– закупить томографы всех наиболее распространенных производителей;

– отказывать в обслуживании пациентам, данные обследований которых не соответствуют имеющимся в наличии томографам;

– создать такую ТИС, которая позволила бы унифицировать представление стоматологической информации, полученной от томографов различных производителей.

Первые три пути решения проблемы являются неприемлемыми либо с финансовой точки зрения, либо с этической.

Поэтому актуальной задачей является разработка автоматизированной системы анализа и преобразования данных томографов (АС АПТД) различных производителей с целью унификации структуры DICOM-файла.

В качестве предприятия, для которого требуется разработать данную систему, выступает отдел компьютерной томографии Межрегионального центра стоматологических инноваций НИУ «БелГУ» (МЦСИ). Отдел компьютерной томографии МЦСИ предоставляет услуги по проведению компьютерной томографии зубов для детального обследования полости рта и выявления различных проблем с зубочелюстной системой. Для этого специалисты центра используют компьютерный томограф «Vatech» (Корея) [23].

Выводы по первому разделу: были проанализированы существующие стоматологические информационные системы различных производителей. В ходе анализа было проведено сравнение ТИС по различным критериям, в ходе которого были выявлены преимущества и недостатки каждой из них. Исходя из недостатков была установлена проблема неправильной интерпретации томографических данных томографами различных производителей, а также поставлена задача разработки АС АПТД.

2 Проектирование автоматизированной системы анализа и преобразования томографических данных

С учетом выбранной методики разрабатываются диаграммы, моделирующие систему и как набор взаимодействующих объектов, и как набор функций, преобразующий поступающий поток информации в выходной поток [24].

2.1 Анализ бизнес-процессов работы отдела компьютерной томографии

На рисунке 2.1 приведено описание работы отдела компьютерной томографии МЦСИ в виде контекстной диаграммы верхнего уровня методологии IDEF0.

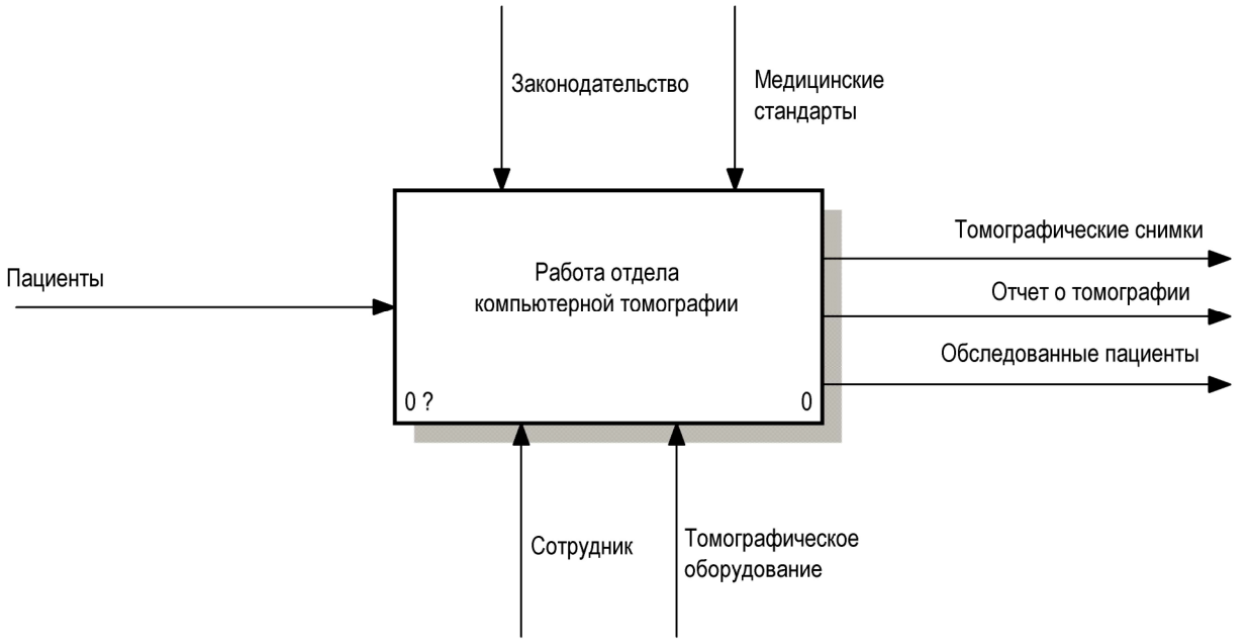

Рисунок 2.1 – Контекстная диаграмма работы отдела компьютерной томографии

Входными данными системы являются пациенты, которые приходят в отдел компьютерной томографии, а выходными данными – томографические снимки, отчет о томографии, а также обследованные пациенты. Работа отдела компьютерной томографии подчиняется законам и медицинским стандартам, а сотрудники с помощью томографического оборудования выполняют возложенные на них обязательства.

Детализация контекстной диаграммы A-0 представлена на рисунке 2.2 (диаграмма A0). Данная диаграмма описывает процесс работы отдела компьютерной томографии. На данной диаграмме присутствуют три функциональных блока: «Проверка возможности проведения томографии», «Проведение томографии», «Проведение консультации пациента». На диаграмме видно, что присутствует обратная связь по входу между блоками «Проведение консультации пациента» и «Проведение томографии». Она означает, что в случае отказа пациенту в консультации он сможет пройти томографию в отделе повторно, что может нанести вред здоровью.

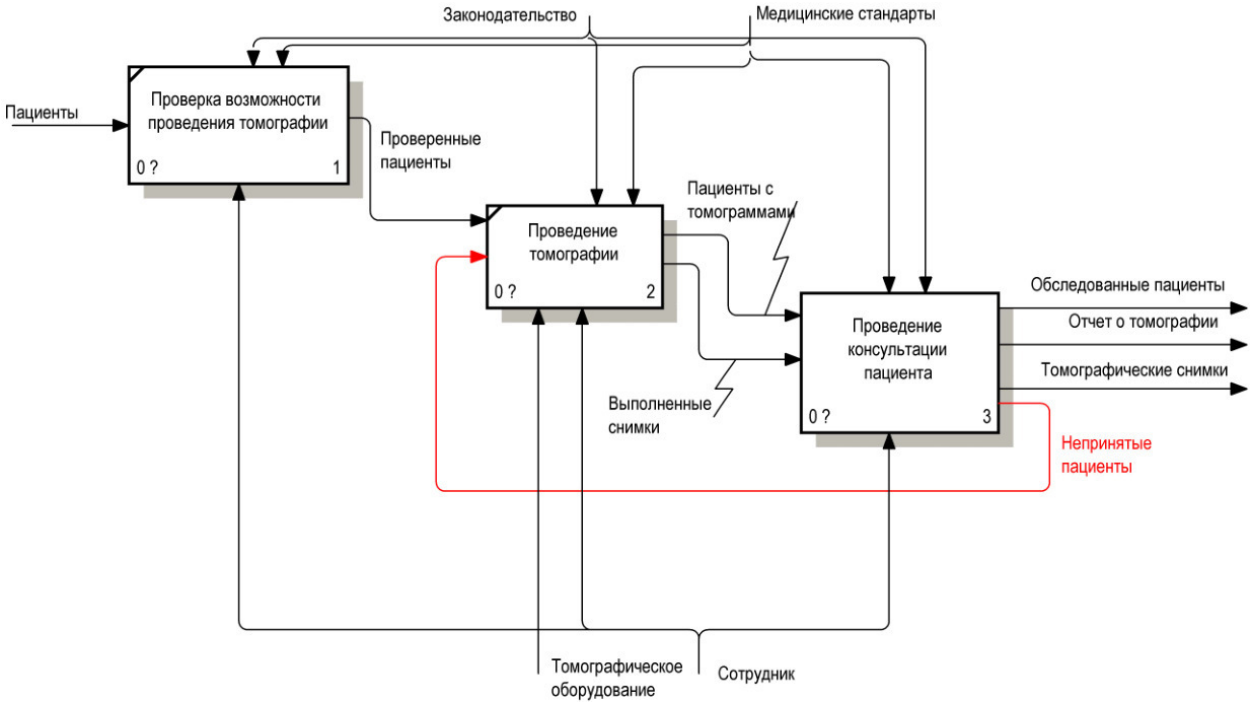

Рисунок 2.2 – Диаграмма декомпозиции первого уровня IDEF0

Для того, чтобы получше разобраться в причинах разработки АС АПТД необходимо произвести декомпозицию процесса «Проведение консультации пациента», используя методологию IDEF3. Диаграмма декомпозиции данного процесса отображена на рисунке 2.3.

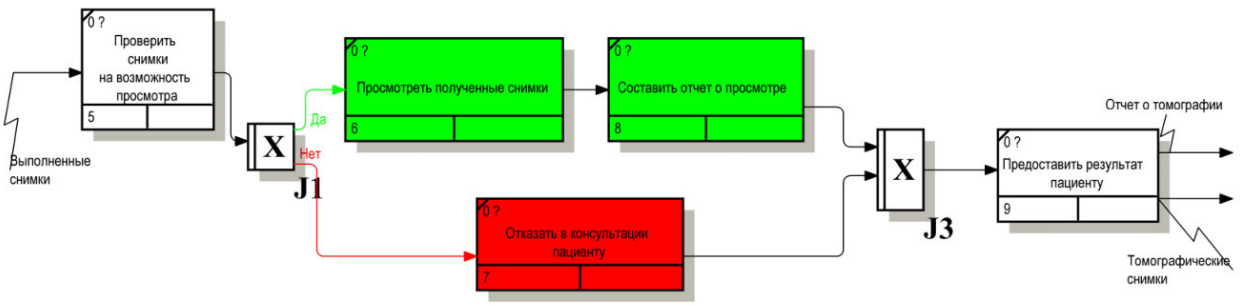

Рисунок 2.3 – Диаграмма декомпозиции процесса «Проведение консультации пациента»

Представленная выше диаграмма описывает процесс проведения консультации пациента, который обращается в отдел компьютерной томографии МЦСИ. Пациент может обратиться в отдел в двух случаях:

– Когда ему необходимо воспользоваться услугами компьютерного томографа для проведения томографии;

– Когда он уже сделал компьютерную томографию в другой клинике и хочет, чтобы его проконсультировали в отделе компьютерной томографии МЦСИ.

На диаграмме декомпозиции видно, что пациенту могут отказать в консультации тогда, когда томографические снимки, которые принес пациент не могут быть просмотрены в ТИС, которой пользуются сотрудники отдела компьютерной томографии МЦСИ. Для того, чтобы исправить ситуацию, была поставлена задача разработки АС АПТД.

Теперь необходимо спроектировать работу отдела компьютерной томографии МЦСИ с использованием АС АПТД.

2.2 Проектирование работы отдела компьютерной томографии с использованием АС АПТД

На рисунке 2.4 приведено описание работы отдела компьютерной томографии МЦСИ с использованием АС АПТД в виде контекстной диаграммы верхнего уровня методологии IDEF0.

Рисунок 2.4 – Контекстная диаграмма работы отдела компьютерной томографии с использованием АС АПТД

На контекстной диаграмме видно, что в качестве механизма, к уже имеющимся сотрудникам и томографическому оборудованию, была добавлена АС АПТД.

На рисунке 2.5 представлена диаграмма декомпозиции контекстной диаграммы, на которой в качестве механизма присутствует АС АПТД. Также можно увидеть, что отсутствует обратная связь между блоками «Проведение консультации пациента» и «Проведение томографии».

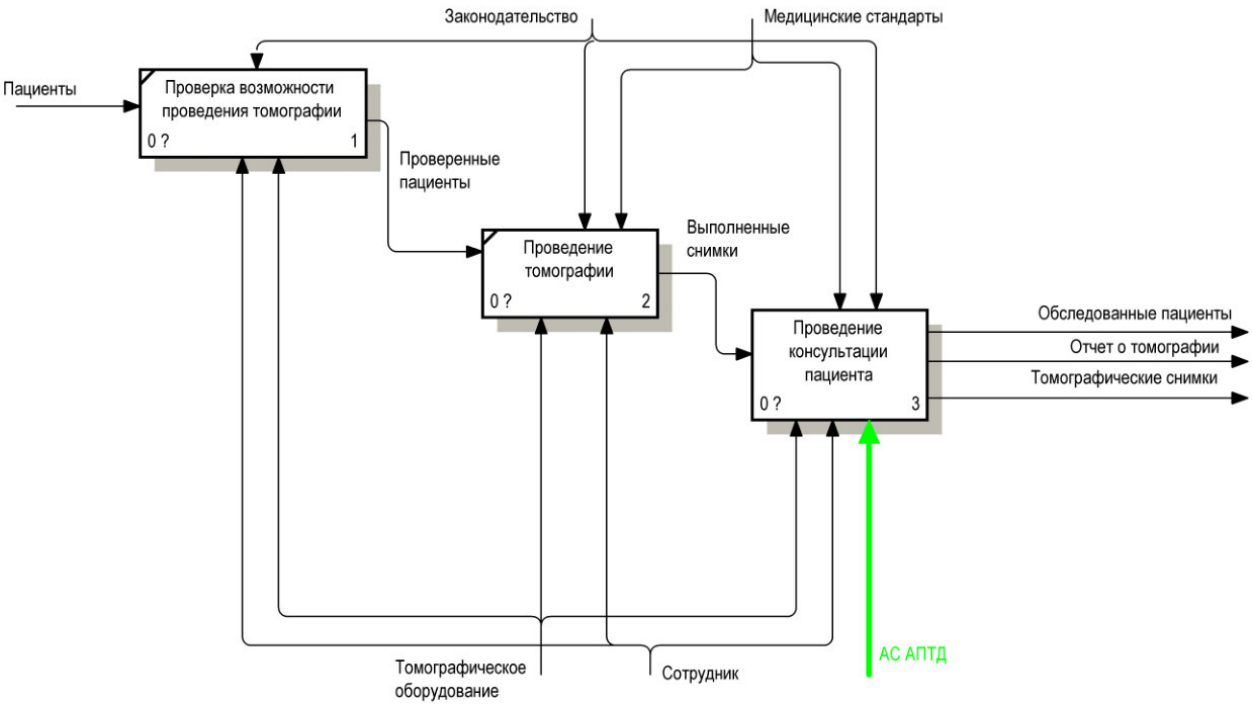

Рисунок 2.5 – Диаграмма декомпозиции первого уровня IDEF0 с использованием АС АПТД

Проводя декомпозицию блока «Проведение консультации пациента» можно увидеть, что АС АПТД используется для преобразования томографических снимков, которые ранее не могли быть просмотрены в имеющейся ТИС. Данная декомпозиция представлена на рисунке 2.6.

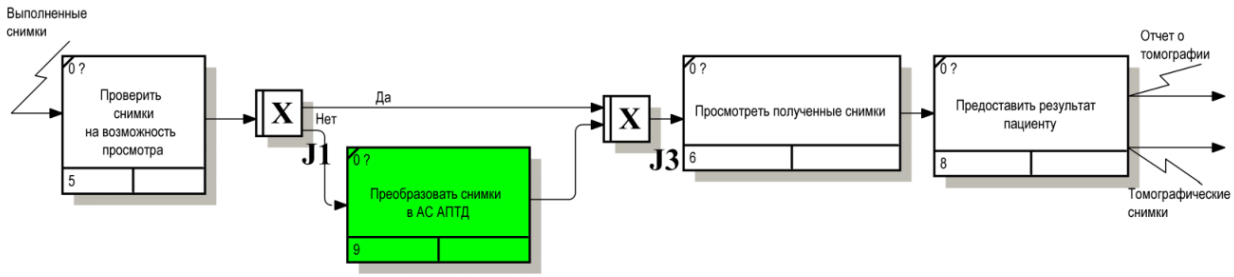

Рисунок 2.6 – Диаграмма декомпозиции процесса «Проведение консультации пациента» с использованием АС АПТД

2.3 Проектирование модели автоматизированной системы анализа и преобразования томографических данных

Проектирование функциональной модели АС АПТД имеет смысл начинать с разработки контекстной диаграммы верхнего уровня модели, которая представлена на рисунке 2.7.

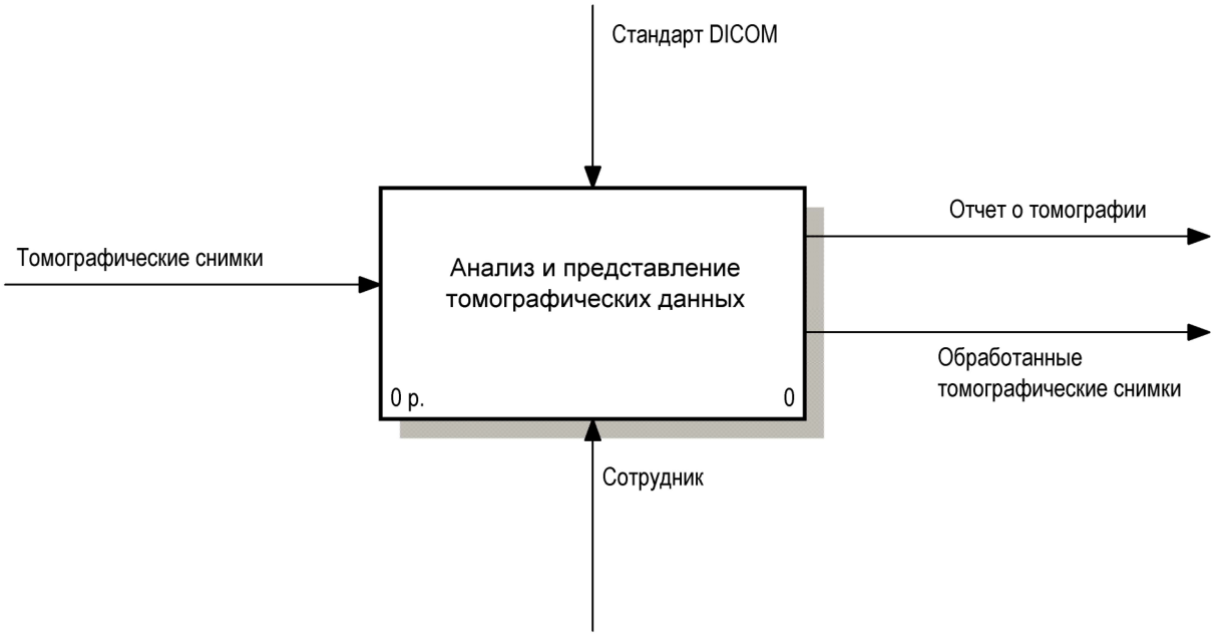

Рисунок 2.7 – Контекстная диаграмма IDEF0

Из диаграммы видно, что на вход системы поступают томографические снимки, результатами работы системы являются отчет о томографии и обработанные томографические снимки. Управляющее воздействие на систему оказывает стандарт DICOM, а в качестве механизма выступает сотрудник отдела компьютерной томографии МЦСИ.

Для более детального описания функций, выполняемых АС АПТД, необходима декомпозиция контекстной диаграммы, представленная на рисунке 2.8.

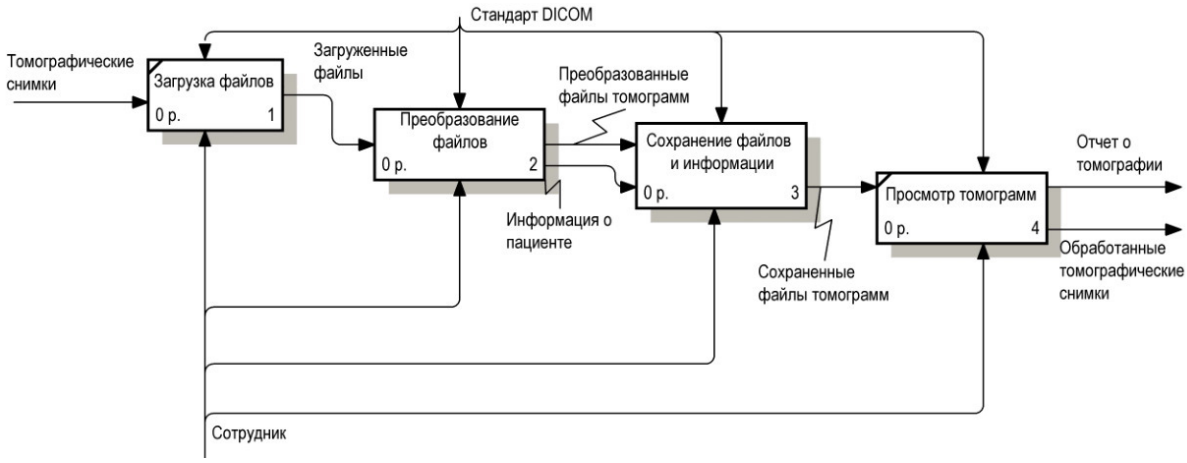

Рисунок 2.8 – Декомпозиция контекстной диаграммы

При декомпозиции получили следующие функциональные блоки: «Загрузка файлов», «Преобразование файлов», «Сохранение файлов и информации», «Просмотр томограмм». Управляющее воздействие и механизм при декомпозиции не изменились.

Данная декомпозиция определяет функции системы на более низком уровне. Основный функциаональным блоком полученой декомпозиции является блок «Преобразование файлов». Работа, выполняемая в данном блоке, содержит в себе предмет исследования магистерской диссертации. Для того, чтобы ознакомиться с работой блока «Преобразование файлов», необходимо провести его декомпозицию, при этом для создания диаграммы использовать методологию IDEF3. Диаграмма отображена на рисунке 2.9.

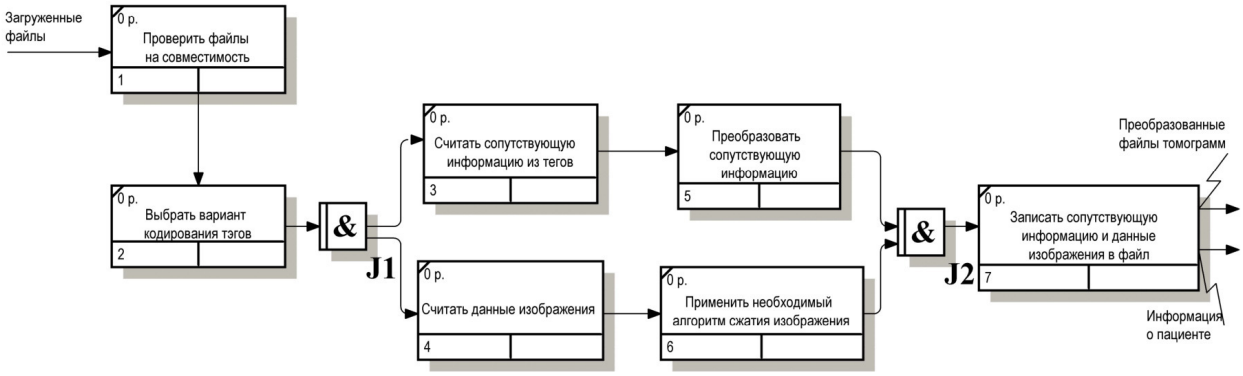

Рисунок 2.9 – Декомпозиция функционального блока «Преобразование файлов»

Диаграмма декомпозиции функционального блока «Преобразование файлов» содержит 7 функциональных блоков:

- «Проверить файлы на совместимость»;
- «Выбрать вариант кодирования тэгов»;
- «Считать сопутствующую информацию из тэгов»;
- «Преобразовать сопутствующую информацию»;
- «Считать данные изображения»;
- «Применить необходимый алгоритм сжатия изображения»;
- «Записать сопутствующую информацию и данные преобразования в файл».

В функциональном блоке «Проверить файлы на совместимость» DICOM-файлы проверяются на возможность преобразования. Так как в текущей реализации возможно преобразовать файлы четырех производителей, то файлы любых других фирм преобразовать невозможно.

В функциональном блоке «Выбрать вариант кодирования тэгов» происходит выбор одного из реализованных алгоритмов преобразования данных тэгов. По умолчанию DICOM-файлы преобразовываются к варианту хранения, реализованному производителем «Vatech». Вариант преобразования можно изменить, путем выбора, соответствующего в выпадающем списке.

После выбора варианта кодирования тэгов DICOM-файл делится на две составляющие: сопутствующую информацию и данные самого изображения. Действия над обеими частями выполняются параллельно.

В функциональных блоках «Считать сопутствующую информацию из тэгов» и «Считать данные изображения» происходит получение всех данных из тэгов, необходимых для преобразования. Так как состав DICOM-файла у разных производителей может отличаться друг от друга, то нет необходимости

получать информацию, которая должна отсутствовать в конечном варианте преобразования.

В функциональном блоке «Преобразовать сопутствующую информацию» происходит приведение полученной информации к необходимой кодировке, необходимому составу тэгов и т.д.

В блоке «Применить необходимый алгоритм сжатия изображения» к данным изображения применяется алгоритм сжатия изображений, используемый в выбранном варианте преобразования файла, например, RLE, JPEG, JPEG Lossless, JPEG 2000 и т.д.

После выполнения всех операций над данными исходного DICOM-файла происходит их сохранение в определенном порядке в новый файл под другим именем в блоке «Записать сопутствующую информацию и данные преобразования в файл».

Также необходимо построить диаграмму вариантов использования, представленную на рисунке 2.10, которая моделирует функциональные возможности АС АПТД для актера – сотрудника отдела компьютерной томографии МЦСИ.

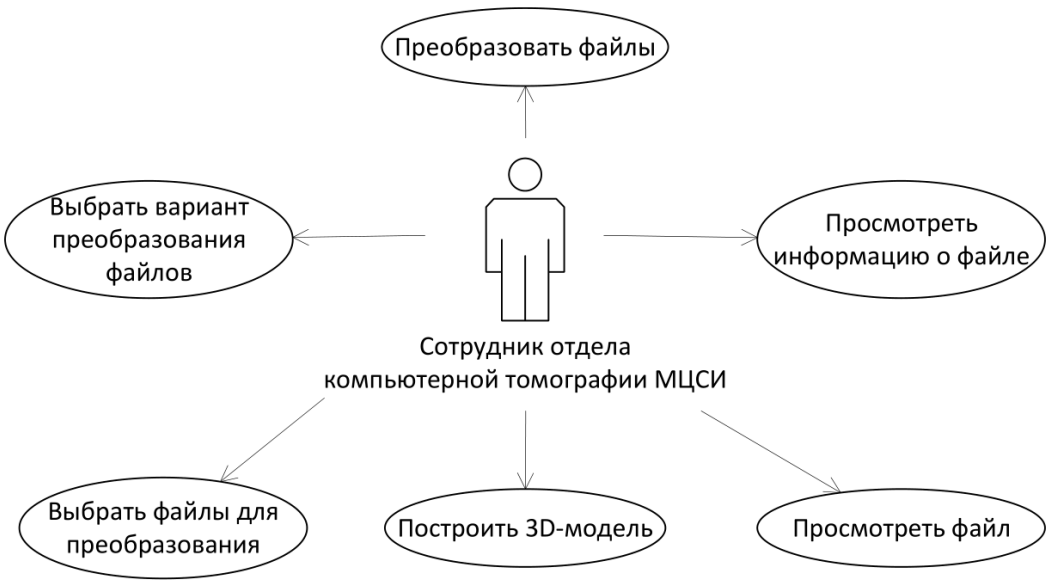

Рисунок 2.10 – Диаграмма вариантов использования

На приведенной диаграмме вариантов использования представлены следующие действия сотрудника отдела компьютерной томографии МЦСИ, которые он может выполнить при помощи АС АПТД:

- выбрать файлы для преобразования
- выбрать вариант преобразования файлов
- преобразовать файлы
- просмотреть информацию о файле
- просмотреть файл
- построить 3D-модель

Выводы по второму разделу: был проведен анализ бизнес-процессов работы отдела компьютерной томографии без использования АС АПТД и спроектирована работа отдела компьютерной томографии с использованием АС АПТД. Также проведено проектирование модели АС АПТД.

Все вышеперечисленные этапы были выполнены при помощи методологий функционального проектирования IDEF0 и IDEF3. Они позволили наглядно отобразить и описать функционирование работы отдела компьютерной томографии и автоматизированной системы, входную и выходную информацию и т.д.

3 Программная реализация и тестирование автоматизированной системы анализа и преобразования томографических данных

3.1 Программная реализация АС АПТД

В магистерской диссертации в качестве языка программирования был использован язык C#, в качестве среды разработки Microsoft Visual Studio [25].

Главное окно АС АПТД представлено на рисунке 3.1.

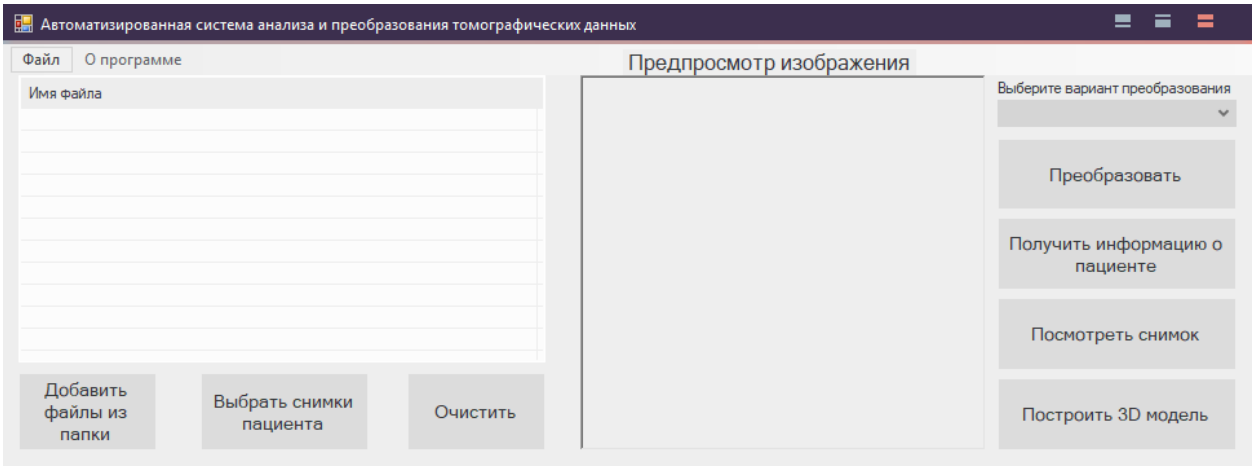

Рисунок 3.1 – Главное окно АС АПТД

Главное окно АС АПТД состоит из 5 областей, которые можно увидеть на рисунке 3.2.:

а) строка меню;

б) область отображения списка файлов;

в) панель инструментов для работы с областью отображения списка файлов;

г) область предпросмотра изображения;

д) панель инструментов для работы с DICOM-файлами.

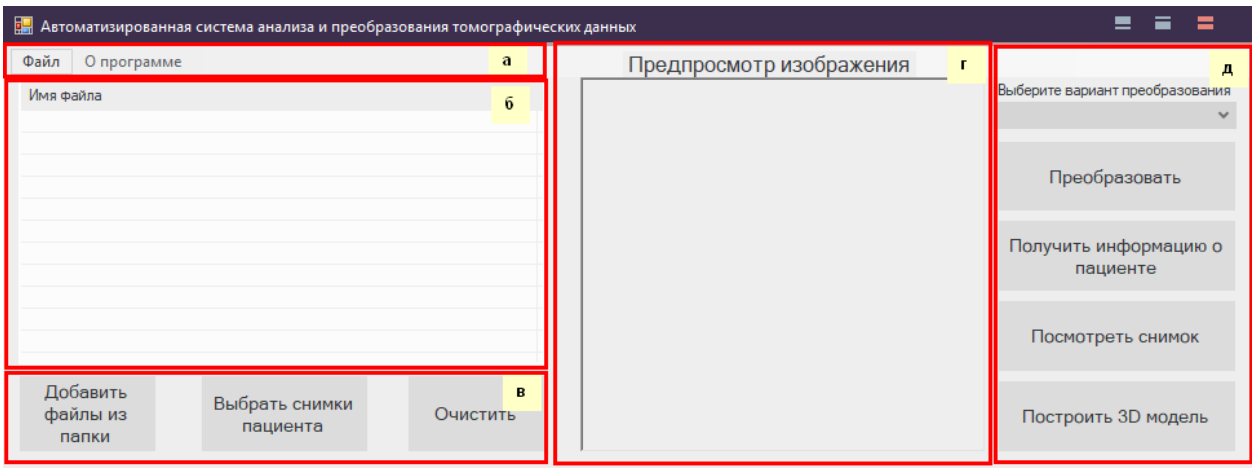

Рисунок 3.2 – Области главного окна АС АПТД

Элемент строки меню «Файл» состоит из 5 подпунктов, отображенных на рисунке 3.3:

– «Добавить»

– «Удалить»

– «Преобразовать в .png»

– «Сохранить в …»

– «Изменить рабочий каталог»

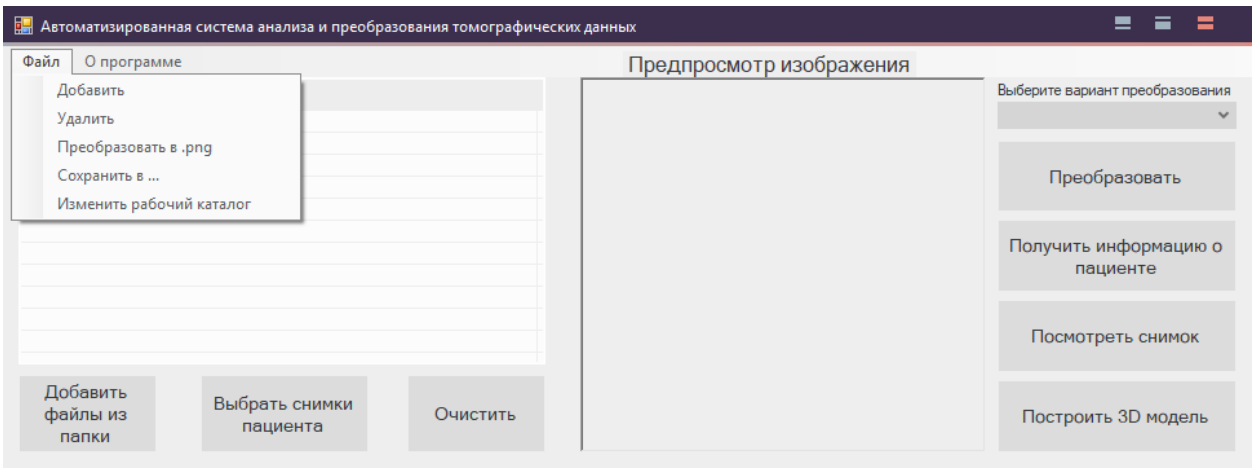

Рисунок 3.3 – Содержание элемента строки меню «Файл»

Функции «Добавить» и «Удалить» позволяют добавлять файлы в список по одному и удалять выбранные файлы соответственно.

Функция «Преобразовать в .png» позволяет экспортировать изображение из DICOM-файла в формат png, который может быть просмотрен как обычное изображение на любом компьютере [26]. Результат работы можно увидеть на рисунке 3.4.

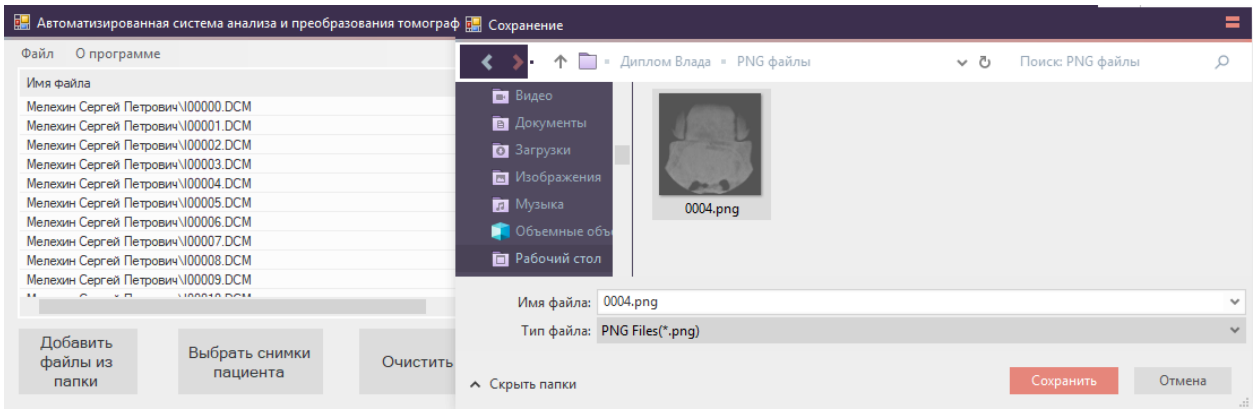

Рисунок 3.4 – Сохранение изображения в формате png

Функция «Сохранить в …» позволяет сохранить выбранный файл в указанное место.

Функция «Изменить рабочий каталог» позволяет изменить каталог, в который сохраняются преобразованные DICOM-файлы по умолчанию. На рисунках 3.5 - 3.6 информационные окна, которые содержат информацию о результате работы данной функции.

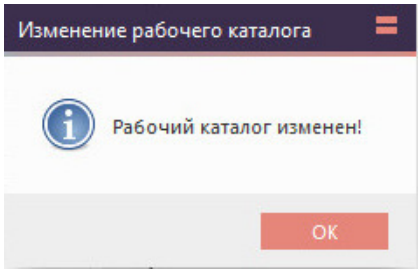

Рисунок 3.5 – Окно с информацией об успешном изменении рабочего каталога

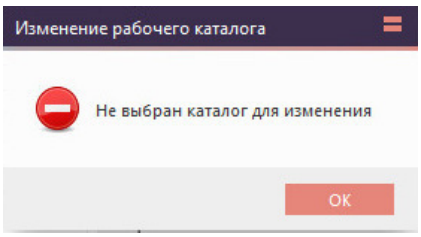

Рисунок 3.6 – Окно с информацией об ошибке при изменении рабочего каталога

При выборе элемента строки меню «О программе» появляется окно, изображенное на рисунке 3.7, которое содержит информацию о программе и ее разработчике.

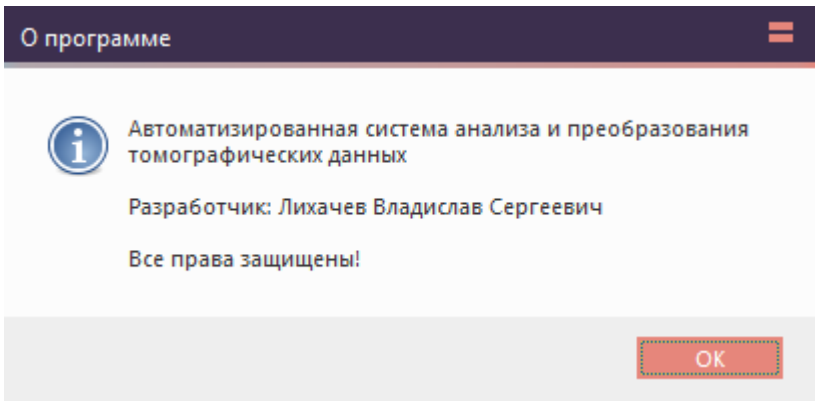

Рисунок 3.7 – Окно «О программе»

Панель инструментов для работы с областью отображения списка файлов содержит 3 кнопки:

– «Добавить файлы из папки»;

– «Выбрать снимки пациента»;

– «Очистить».

Кнопка «Добавить файлы из папки» позволяет выбрать все файлы из выбранного каталога. Так как после проведения томографии пациенту выдают диск, содержащий каталог с DICOM-файлами, то необходимо выбрать эту папку полностью, чтобы добавить, а потом и преобразовать все

томографические снимки. Окно для выбора папки с фалами представлено на рисунке 3.8.

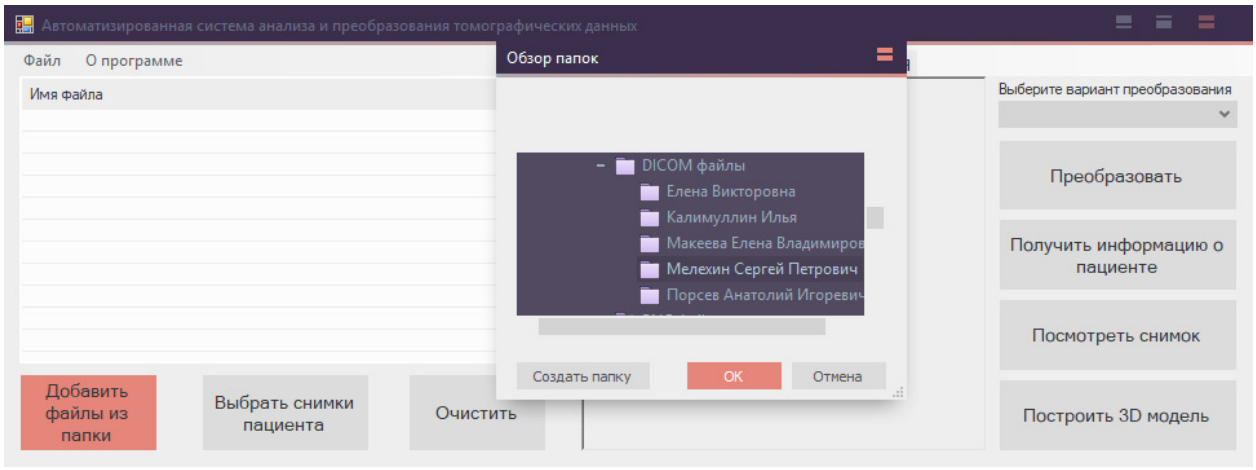

Рисунок 3.8 – Окно выбора папки с DICOM-файлами.

Кнопка «Выбрать снимки пациента» открывает окно со списком пациентов, представленное на рисунке 3.9. В этом списке содержатся пациенты, файлы которых уже были преобразованы в АС АПТД ранее. Пациент добавляется в этот список после преобразования его файлов. Для выбора пациента из списка необходимо выбрать его в списке и нажать кнопку «Выбрать»

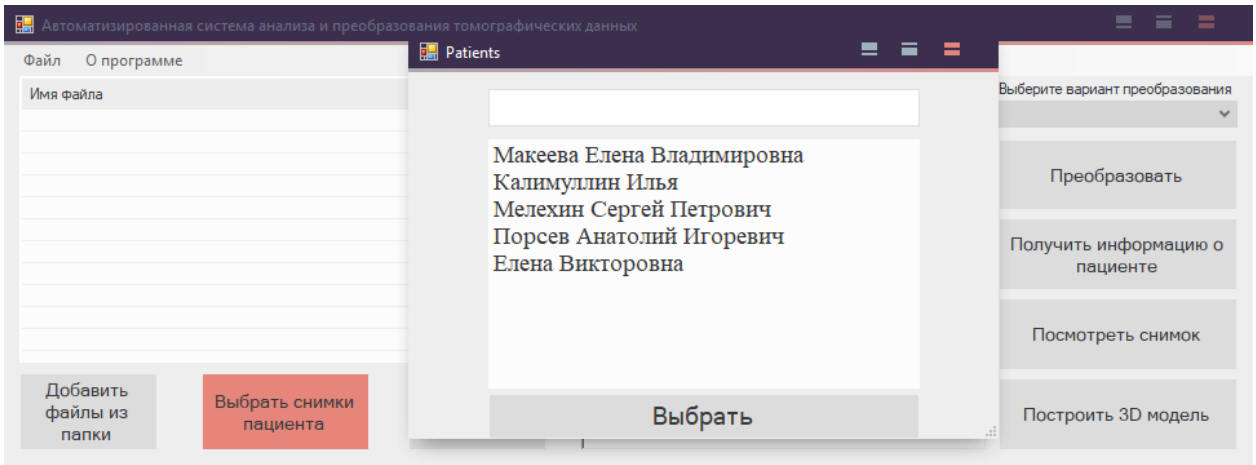

Рисунок 3.9 – Окно выбора пациента

Окно выбора пациента также содержит строку поиска, при вводе значения в которую, будет производиться поиск среди пациентов по заданной подстроке [27]. Поиск производится путем проверки каждого элемента списка на содержание введенной подстроки вне зависимости от регистра букв. Пример поиска можно увидеть на рисунке 3.10.

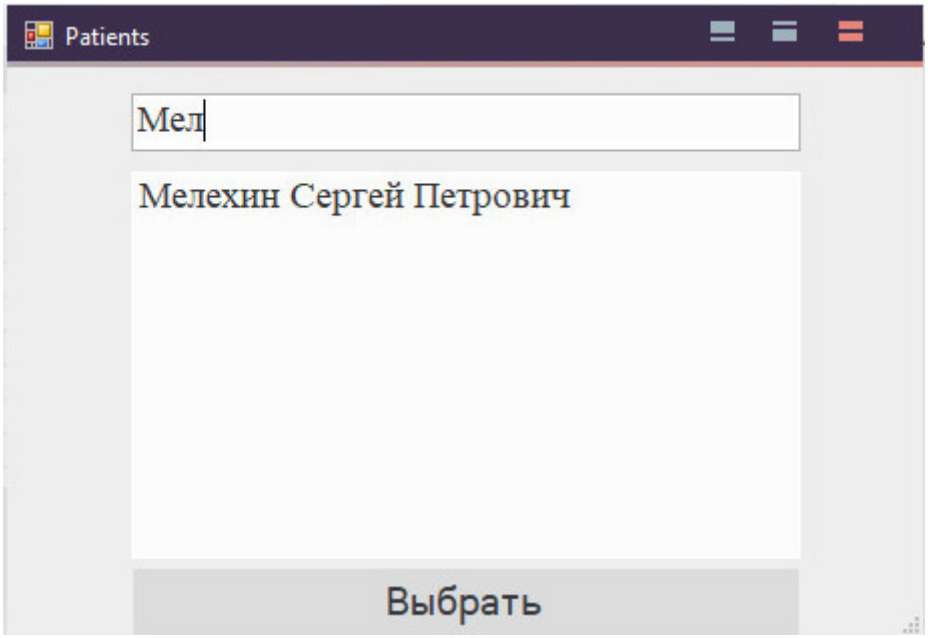

Рисунок 3.10 – Поиск пациента по подстроке

После выбора файлов из папки или при помощи окна выбора пациента в области отображения списка файлов появится список всех файлов из выбранной папки или из каталога выбранного пациента. Названия элементов имеют следующий формат – «Полный путь к папке\название файла», в случае выбора папки с файлами, либо «Наименование пациента\название файла», в случае выбора пациента из списка пациентов. Пример списка файлов для выбранного пациента можно увидеть на рисунке 3.11. Данный список можно очистить путем нажатия на кнопку «Очистить» на панели инструментов.

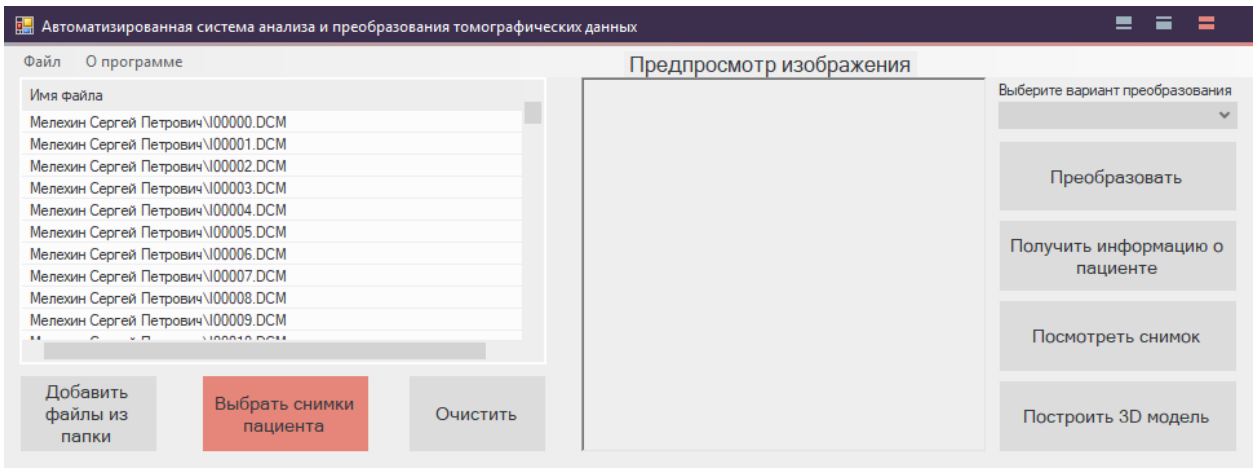

Рисунок 3.11 – Список файлов пациента Мелехин Сергей Петрович

При нажатии на один из файлов списка в области предпросмотра изображения будет отображено миниатюрное представление изображения, хранящегося в DICOM-файле. Пример предпросмотра изображения представлен на рисунке 3.12.

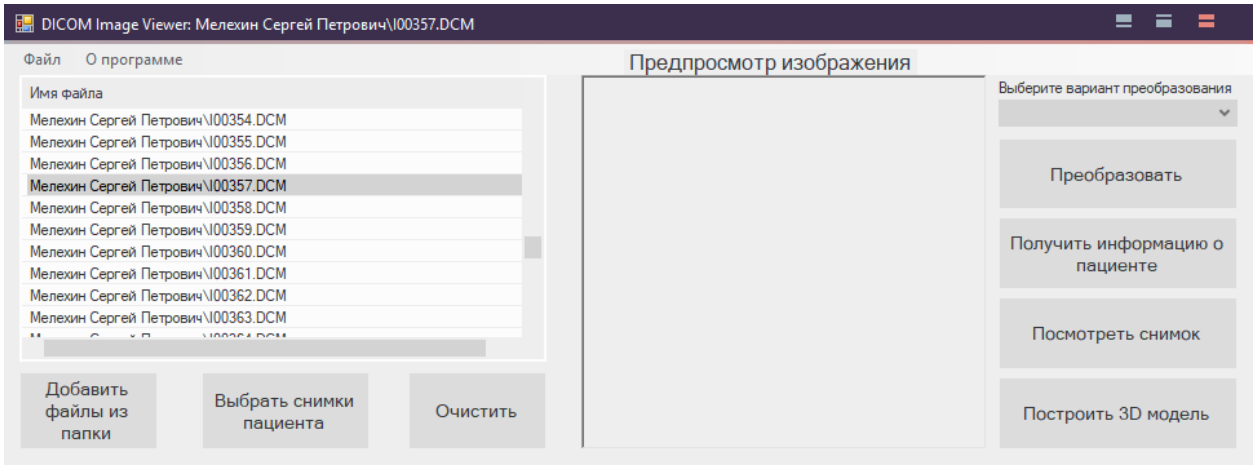

Рисунок 3.12 – Пример предпросмотра изображения

Основной функцией АС АПТД является преобразование DICOM-файлов из одного варианта хранения в другой. На данный момент программа может работать с наиболее популярными в России производителями томографов:

– Vatech (Ez3D2009);

– Sirona (Galaxis);

– Planmeca (Romexis);

– Gendex (I-Cat Vision).

Запуск преобразования файлов осуществляется нажатием кнопки «Преобразовать» на панели инструментов для работы с DICOM-файлами. Расположение данной кнопки можно увидеть на рисунке 3.13.

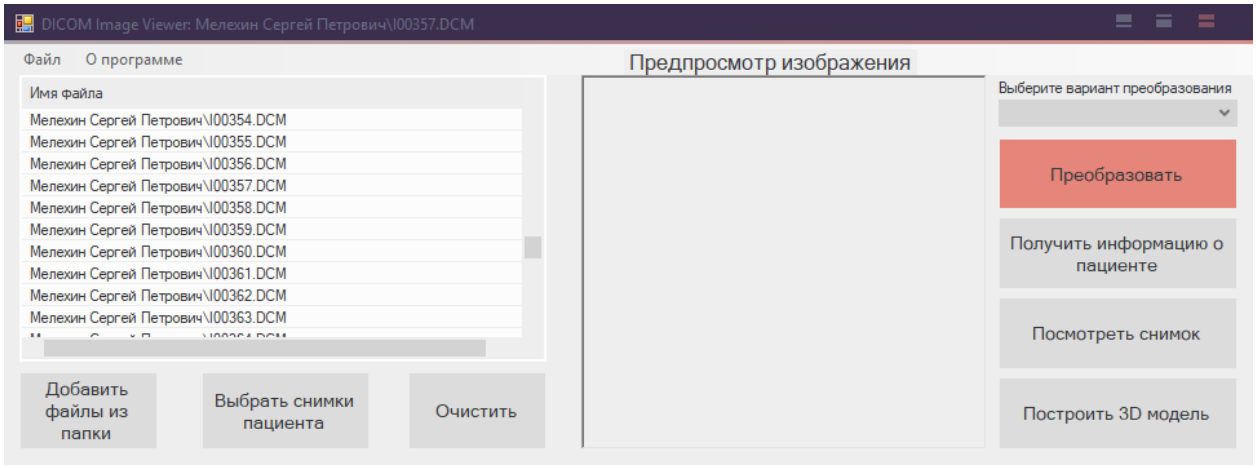

Рисунок 3.13 – Расположение кнопки «Преобразовать» на панели инструментов

После нажатия данной кнопки начинает работу следующий алгоритм:

а) производится анализ текущего варианта хранения данных файла;

б) выполняется считывание значения тэга «Patient’s Name», хранящего имя пациента;

в) в рабочем каталоге создается папка с названием «Имя пациента»;

г) в список пациентов добавляется запись с именем нового пациента;

д) каждый из файлов списка преобразуется в DICOM-файл необходимой структуры и сохраняется в созданную папку [28].

По умолчанию преобразование файлов одного из этих производителей осуществляется к формату, используемому производителем Vatech, т.к. в отделе компьютерной томографии используется оборудование данной фирмы,

но вариант преобразования всегда можно изменить. Для этого необходимо раскрыть выпадающий список «Выберите вариант преобразования» и выбрать необходимый вариант для сохранения данных DICOM-файла. Например, если необходимо преобразовать файлы для просмотра в ТИС Romexis, то в списке необходимо выбрать вариант Planmeca (Romexis). Текст кнопки «Преобразовать» изменится на «Преобразовать в Planmeca (Romexis)», при нажатии на которую файлы будут преобразованы к виду, используемому производителем Planmeca. Описанный выше вариант преобразования можно увидеть на рисунке 3.14.

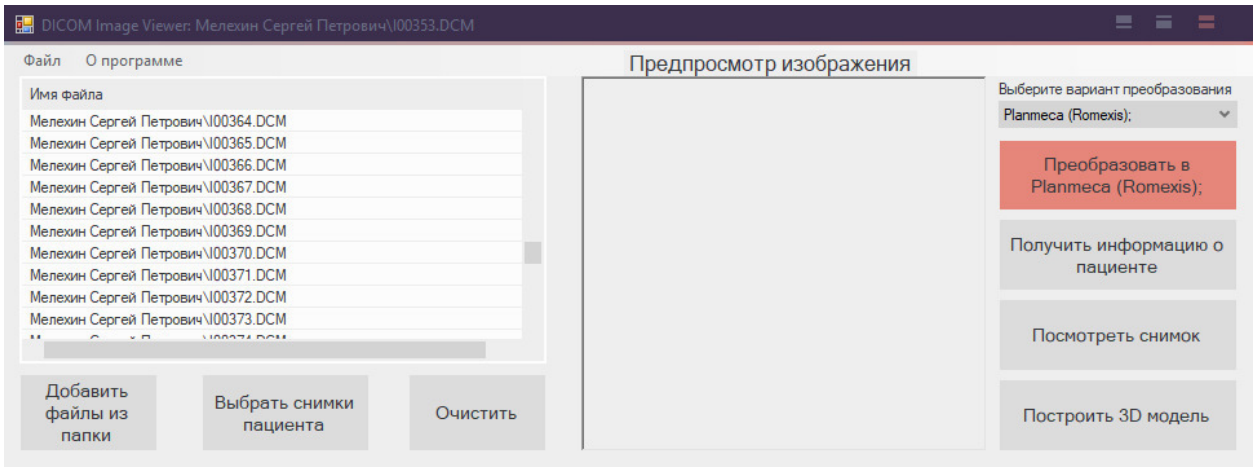

Рисунок 3.14 – Выбор варианта преобразования файлов Planmeca (Romexis)

Для любого из файлов в списке можно просмотреть сопутствующую информацию из тэгов. Для этого необходимо выбрать необходимый файл и нажать кнопку «Получить информацию о пациенте». После нажатия этой кнопки открывается диалоговое окно, на которой находится таблица, содержащая всю сопутствующую изображению информацию: различные параметры изображения, информацию об оборудовании, информацию о пациенте и т.д. Таблица состоит из четырех столбцов, в каждом из которых содержится информация, соответствующая названию столбца:

– «Номер группы»

– «Номер тэга»

– «Описание тэга»

– «Значение тэга»

Диалоговое окно с таблицей представлено на рисунке 3.15.

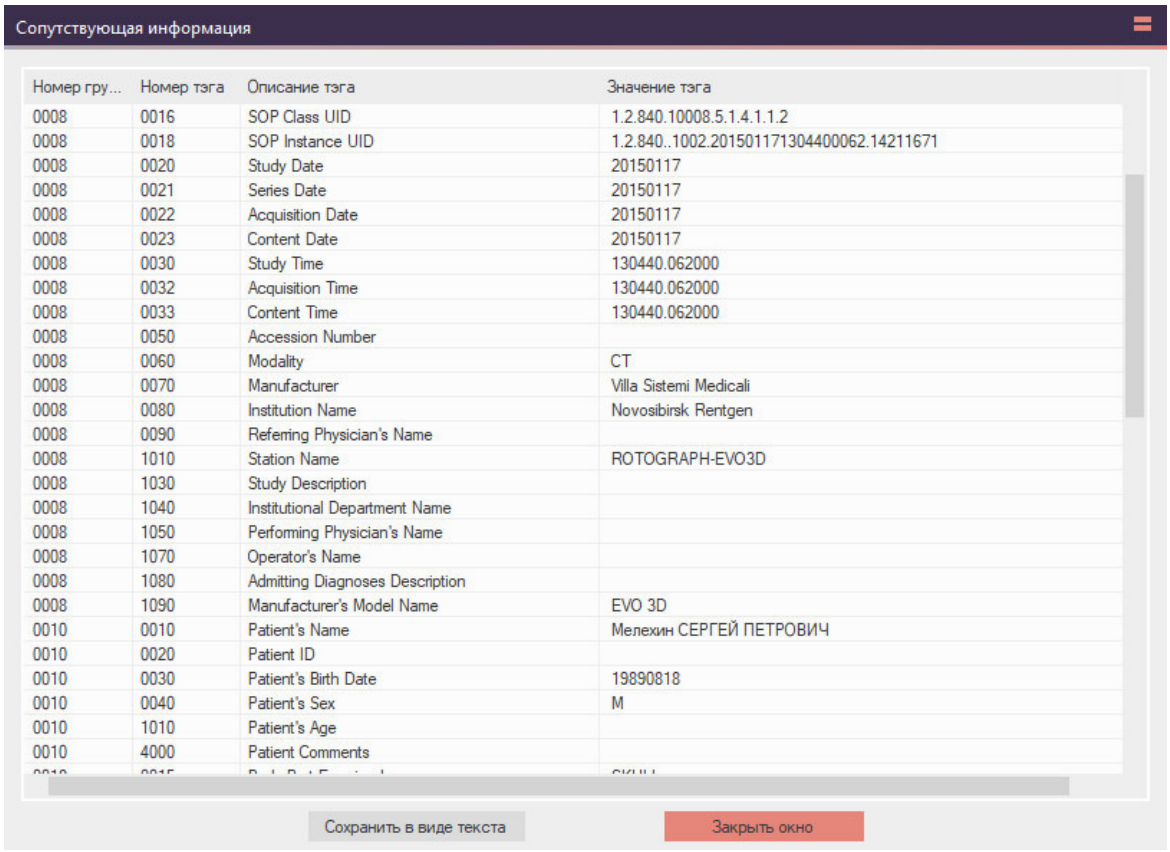

Рисунок 3.15 – Окно с сопутствующей информацией

Кроме просмотра информации на диалоговом окне существует функция выгрузки всей информации в текстовый файл. Для этого необходимо нажать кнопку «Сохранить в виде текста» и затем ввести имя и выбрать расположение сохранения файла. Содержимое получившегося текстового файла изображено на рисунке 3.16.

```
(0008,0023) Content Date          20150117
(0008,0030) Study Time       130440.062000
(0008,0032) Acquisition Time          130440.062000
(0008,0033) Content Time          130440.062000
(0008,0050) Accession Number
(0008,0060) Modality         CT
(0008,0070) Manufacturer          Villa Sistemi Medicali
(0008,0080) Institution Name          Novosibirsk Rentgen
(0008,0090) Referring Physician's Name
(0008,1010) Station Name          ROTOGRAPH-EVO3D
(0008,1030) Study Description
(0008,1040) Institutional Department Name
(0008,1050) Performing Physician's Name
(0008,1070) Operator's Name
(0008,1080) Admitting Diagnoses Description
(0008,1090) Manufacturer's Model Name         EVO 3D
(0010,0010) Patient's Name        Мелехин СЕРГЕЙ ПЕТРОВИЧ
(0010,0020) Patient ID
(0010,0030) Patient's Birth Date          19890818
(0010,0040) Patient's Sex         M
(0010,1010) Patient's Age
(0010,4000) Patient Comments
```

Рисунок 3.16 – Содержимое текстового файла с сопутствующей информацией

Оставшиеся на панели инструментов кнопки «Посмотреть снимок» и «Построить 3D модель» позволяют открыть на просмотр любой из выбранных преобразованных снимков [29] и построить 3D-модель по всей серии снимков соответственно [30]. Примеры просмотра снимка и построения примеры просмотра снимка и построения 3D-модели представлены на рисунках 3.17 – 3.18.

Рисунок 3.17 – Просмотр выбранного изображения

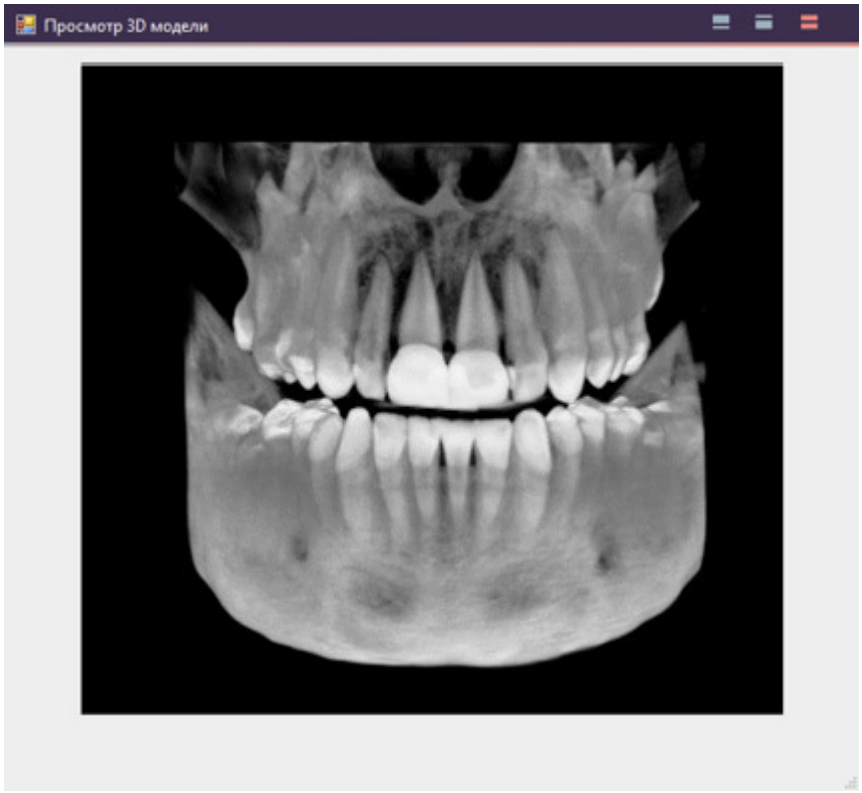

Рисунок 3.18 – Пример построения 3D-модели

3.2 Тестирование АС АПТД

Для тестирования разработанной АС АПТД были использованы DICOM-файлы, сделанные при помощи томографов четырех производителей, описанных выше.

Проверка достоверности работы, разработанной АС АПТД осуществлялась следующими способами:

Сначала в каждой из ТИС были сделаны попытки открыть файлы, сделанные на оборудовании других производителей. Ни один из файлов не был открыт в «чужой» программе. Результаты попыток открытия можно увидеть на рисунках 3.19-3.22.

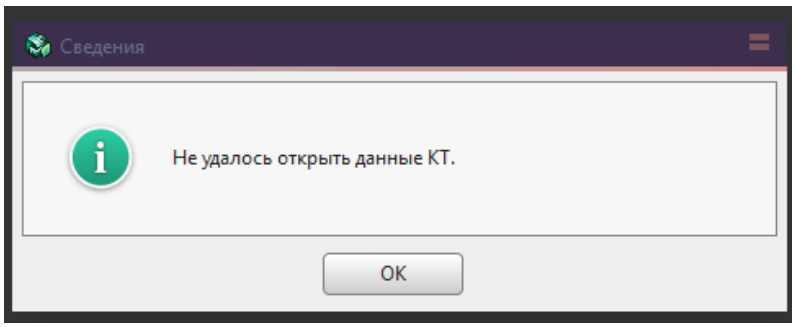

Рисунок 3.19 – Открытие файла в Planmeca (Romexis)

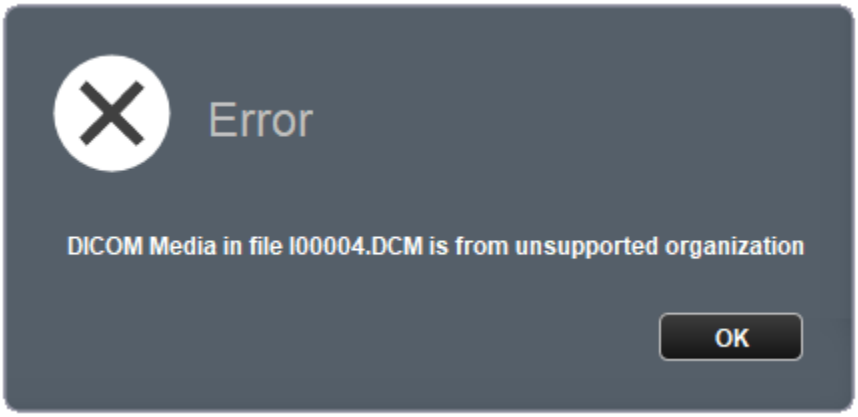

Рисунок 3.20 – Открытие файла в Vatech (Ez3D2009)

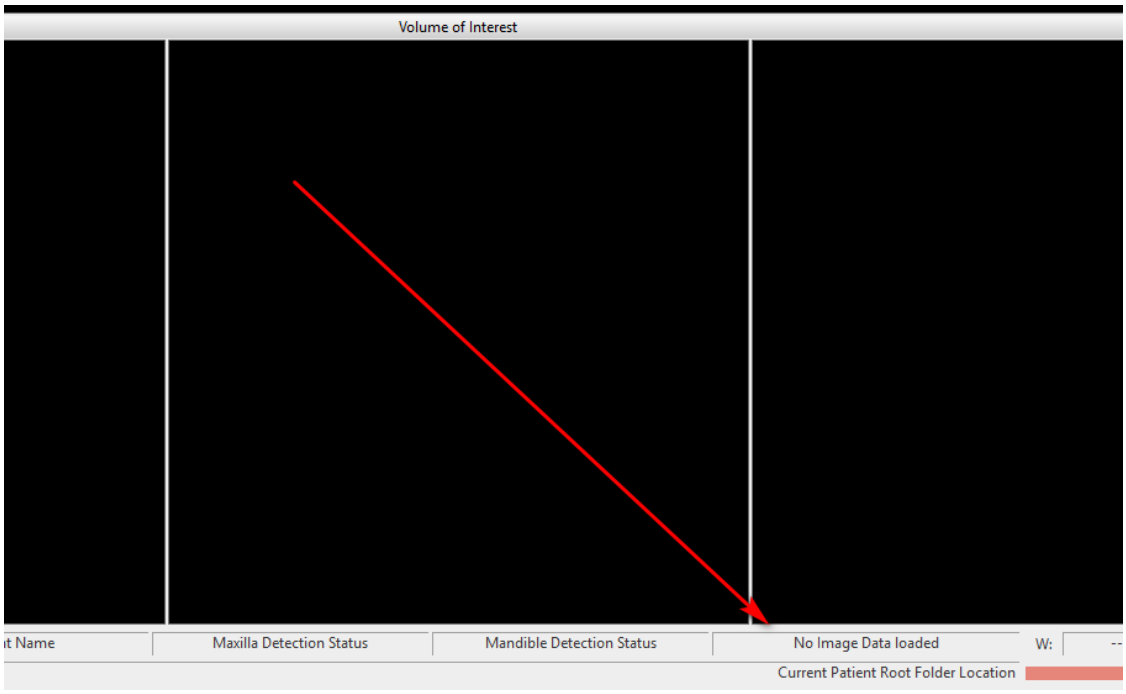

Рисунок 3.21 – Открытие файла в Gendex (I-Cat Vision)

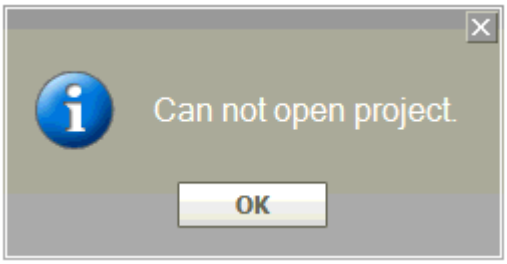

Рисунок 3.22 – Открытие файла в Sirona (Galaxis)

Следующим шагом являлось преобразование всех тестируемых файлов в формат, используемый производителем Vatech для ТИС Ez3D2009. На рисунке 3.23 представлен результат работы АС АПТД.

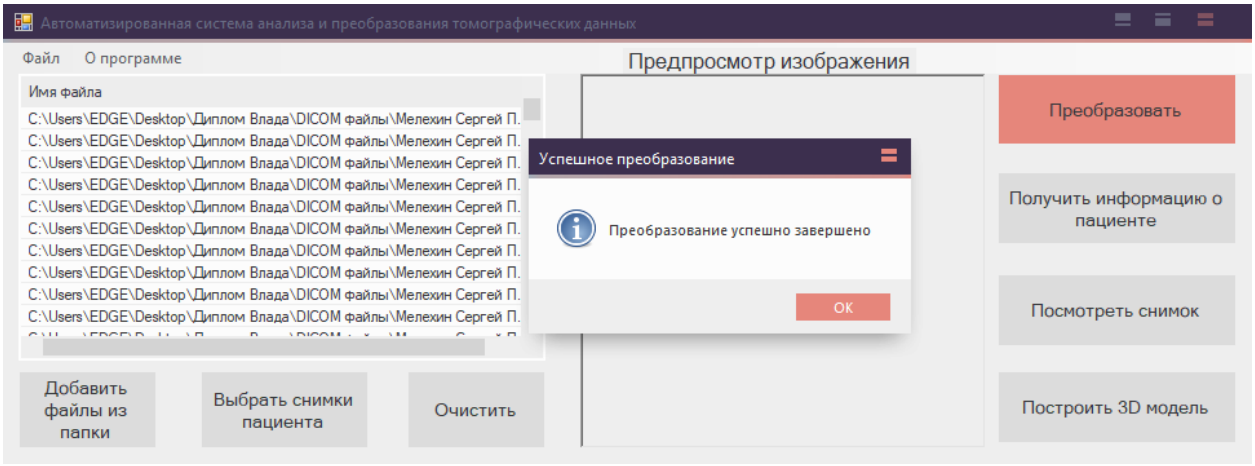

Рисунок 3.23 – Результат преобразования DICOM-файлов

Далее в ТИС Ez3D2009 были открыты полученные после преобразования файлы. Один из результатов открытия файлов можно увидеть на рисунке 3.24. Видно, что файлы были успешно открыты – по серии снимков была построена ортопантомограмма в левом нижнем углу, открыт один из срезов в левом верхнем углу, а также в правой части экрана отображена область для просмотра дополнительной информации по текущему срезу.

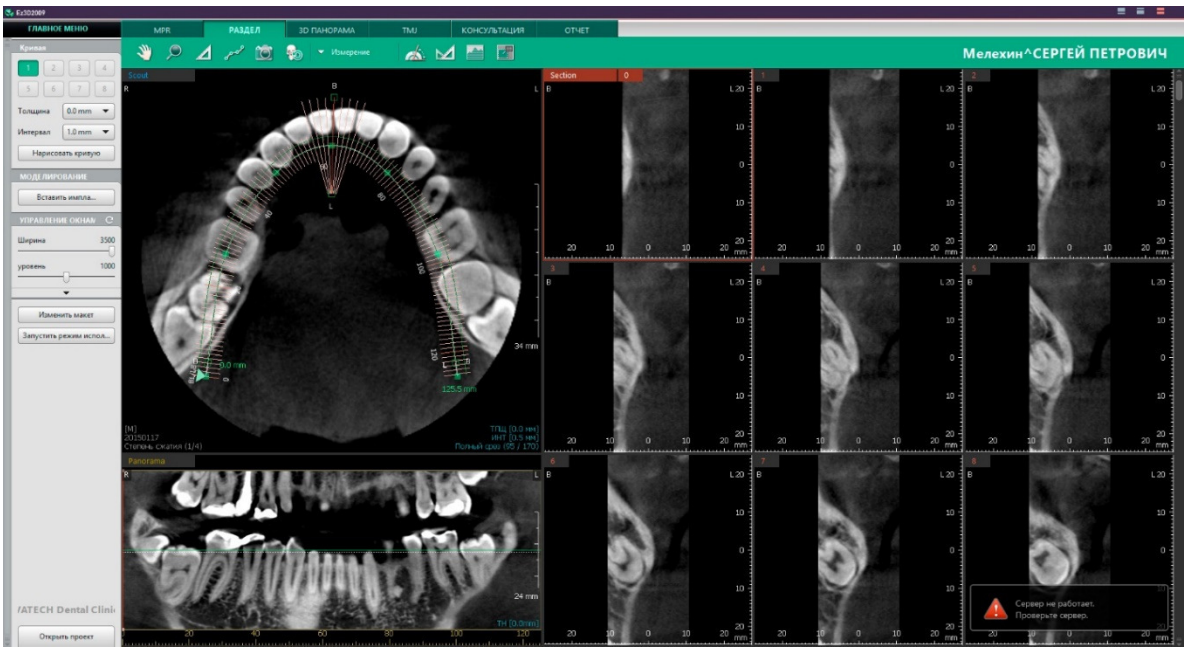

Рисунок 3.24 – Открытие преобразованных файлов в Ez3D2009

Т.о. результаты открытия файлов различных производителей в различных ТИС можно представить в виде таблицы 3.1. В строках таблицы перечислены названия производителей и соответствующих им ТИС, а в столбцах под значениями 1-4 подразумеваются те же самые ТИС в том же порядке, что и в строках. Если программа смогла открыть файлы другого производителя, то на пересечении ставится «галочка», если нет – «крестик».

Таблица 3.1 – Сравнительная таблица результатов открытия файлов

	Открытие файлов до преобразования				Открытие в АС АПТД	Открытие файлов после преобразования			
	1	2	3	4		1	2	3	4
Vatech (Ez3D2009)	✓	✗	✗	✗	✓	✓	✓	✓	✓
Sirona (Galaxis)	✗	✓	✗	✗	✓	✓	✓	✓	✓
Planmeca (Romexis)	✗	✗	✓	✗	✓	✓	✓	✓	✓
Gendex (I-Cat Vision)	✗	✗	✗	✓	✓	✓	✓	✓	✓

В таблице 3.1. видно, что изначально каждая из ТИС могла открыть только файлы, которые были сделаны с помощью соответствующего ей томографа. После преобразования файлов, стало возможным открыть абсолютно любые DICOM-файлы в любой из ТИС.

Выводы по третьему разделу: были разработаны экранные формы АС АПТД, реализован алгоритм для преобразования DICOM-файлов, а также сопутствующий функционал системы. Затем с помощью тествовых данных было проведено тестирование разработанной АС АПТД.

ЗАКЛЮЧЕНИЕ

В различных стоматологических медицинских учреждениях России, используются томографы и программное обеспечение различных производителей. Все томографы сохраняют данные в DICOM-файлах, но специализированное программное обеспечение, используемое в одном медицинском учреждении, не способно работать с файлами, полученными из других учреждений других производителей. Производители пользуются тем, что в стандарте не указан единый вариант хранения данных, т.к. если бы существовало ПО, способное работать со всеми существующими производителями, то каждый из них терял часть прибыли от продаж своего ПО и, главное, томографического оборудования.

В связи с чем в магистерской диссертации была разработана автоматизированная система, способная унифицировать представление стоматологической информации, полученной от томографов различных производителей.

В ходе выполнения магистерской диссертации были достигнуты все поставленные цели и задачи:

– проведен анализ существующих стоматологических информационных систем;

– спроектирована автоматизированная система анализа и преобразования томографических данных;

– разработана и протестирована автоматизированная система анализа и преобразования томографических данных.

Разработанная АС АПТД позволит сотрудникам отдела компьютерной томографии МЦСИ консультировать и обслуживать не только пациентов, которые проходили компьютерную томографию на оборудовании компании Vatech, но и на оборудовании других популярных производителей.

В текущем виде АС АПТД не является полноценной ТИС, так как она не обладает полноценным функционалом для работы с томографическими данными. В дальнейшем возможно усовершенствование АС АПТД до уровня ТИС, а также увеличение количества поддерживаемых производителей томографов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Календер, В.А. Компьютерная томография. Основы, техника, качество изображений и области клинического использования / Вилли А. Календер − М.: Техносфера, 2006. − 344 с.

2. Digital Imaging and Communications in Medicine (DICOM). Official site [Электронный ресурс]. – Режим доступа: https://www.dicomstandard.org/current/

3. Рогацкин, Д.В. Радиодиагностика челюстно-лицевой области. Конусно-лучевая компьютерная томография. Основы визуализации / Д.В.Рогацкин. – Львов: ГалДент, 2010. – 148 с.

4. Хофер, М.Д. Компьютерная томография / М.Д. Хофер. – М.: Медицинская литература, 2011. – 232 с.

5. Пронин, И.Н. Программное обеспечение для работы с данными в формате DICOM на IBM PC / И.Н. Пронин, П.В. Родионов, Л.М. Фадеева // Медицинская визуализация. – 2002. – №2. – С.138–142.

6. Руководство пользователя Ez3D2009 [Электронный ресурс] // 3D - diagnostics. – Режим доступа: http://www.3-d.com.ua/images/instrukcia /Ez3D2009_ru.pdf

7. Руководство пользователя i-CAT Vision [Электронный ресурс] // CT-dent Dental Imaging Centre. – Режим доступа: https://ct-dent.co.uk/images/i-CAT_Vision_Manual.pdf

8. Bishara, S. E. Textbook of Orthodontics / Samir E. Bishara - Saunders Company, 2001. – 599 с.

9. Сердобинцев, Е.В. Особенности при работе с программным обеспечением компьютерных томографов в практической деятельности врача-стоматолога / Е.В. Сердобинцев // X-RAY ART. – 2013. – №2. – С. 51-55.

10. Рогацкин, Д.В. Искусство рентгенографии зубов/ Д.В. Рогацкин, Н.В. Гинали. – STBOOK, 2007. – 206 с.

11. Якубенко, А.А. Алгоритмы построения трехмерных моделей объектов с регулярной структурой по фотографиям при взаимодействии с пользователем для виртуальных сред специальность: автореф. дис. на соиск. учен. степ. канд. тех. наук (05.13.11) / Якубенко Антон Анатольевич; Московский государственный университет имени М. В. Ломоносова. – М.: МГУ, 2013. – 23 с.

12. Марусина, М.Я. Современные виды томографии / М. Я. Марусина, А. О. Казначеева. – СПб.: СПбГУ ИТМО, 2006. – 132 с.

13. Сердобинцев, Е.В. Краткий анализ программного обеспечения конусно-лучевых компьютерных томографов, применяемых в России / Е.В. Сердобинцев // X-RAY ART. – 2013. – №3. – С. 43-48.

14. Моисеева, И.Л. Сравнительная характеристика программ-просмотрщиков конусно-лучевой компьютерной томографии / И. Л. Моисеева // X-RAY ART. – 2012. – №1. – С. 46-51.

15. Стандарт DICOM 3.0 [Электронный ресурс] // Адаптивные Медицинские Системы Обработки Реального Времени. – Режим доступа: http://www.course-as.ru/dicomdoc.html

16. Плотников, А.В. Стандарт DICOM в компьютерных медицинских технологиях. / А.В. Плотников, Д.А. Прилуцкий, С.В. Селищев // Медицинская техника. – 1997. – № 2. – С. 18-20.

17. Clunie, D. DICOM Structured Reporting / D. Clunie. – PixelMed Publishing, Bangor, 2001. – 394 с.

18. Структура DICOM-файла [Электронный ресурс] // Компью́терная томогра́фия. – Режим доступа: http://www.kievoncology.com/struktura-dicom-fayla.html

19. DICOM (Digital Imaging and COmmunications in Medicine) [Электронный ресурс] // Makhaon Software. – Режим доступа: http://www.makhaon.com/index.php?p=dicom

20. Миронина, И.А. Анализ графических форматов с целью эффективного хранения медицинских снимков / И.А. Миронина // Инновационные медицинские технологии. – 2013. – С. 47-55.

21. ArtiSynth Manuals and Guides [Электронный ресурс] // ArtiSynth A 3D Biomechanical Modeling Toolkit. – Режим доступа: https://www.artisynth.org/manuals/index.jsp

22. Pianykh, O.S. Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide / Oleg S. Pianykh. – Springer Science & Business Media, 2008. – 384 с.

23. Comparison of CBCT Machines [Электронный ресурс] // SEDENTEXCT Guidelines. – Режим доступа: http://www.sedentexct.eu/content/comparison-cbct-machines

24. Цуканова, О.А. Методология и инструментарий моделирования бизнес-процессов / О. А. Цуканова. – СПб.: Университет ИТМО, 2015. – 100 с.

25. Троелсен, Э. Язык программирования C# 6.0 и платформа .NET 4.6 / Э. Троелсен. – Вильямс, 2016. – 1440 с.

26. Красильников, Н. Н. Цифровая обработка 2D- и 3D-изображений / Н. Н. Красильников. – СПб.: БХВ-Петербург, 2011. – 608 с.

27. Уоррен, Г. С. Алгоритмические трюки для программистов / Г. С. Уоррен. – Вильямс, 2014. – 512 с.

28. Introduction to DICOM [Электронный ресурс] // Access a cacophony of neuro-imaging file formats. – Режим доступа: http://nipy.org/nibabel/dicom/dicom_intro.html

29. И. М. Журавель «Краткий курс теории обработки изображений» [Электронный ресурс] // Материалы по продуктам MATLAB & Toolboxes. – Режим доступа: http://matlab.exponenta.ru/imageprocess/book2/index.php

30. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. - М.: Техносфера, 2006. – 1072 с.

ПРИЛОЖЕНИЕ А

Текст программы

Class Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;


namespace test1
{
   public partial class DICOM : Form
   {
      private const string path = "C:\\Users\\EDGE\\Desktop\\Диплом Влада\\test1 v2\\test1";
      DicomDecoder dd;
      List<byte> pixels8;
      List<ushort> pixels16;
      List<byte> pixels24;
      int imageWidth;
      int imageHeight;
      int bitDepth;
      int samplesPerPixel;
      bool imageOpened;
      double winCentre;
      double winWidth;
      bool signedImage;
      int maxPixelValue;
      int minPixelValue;

      public DICOM()
      {
         InitializeComponent();
         comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
         dd = new DicomDecoder();
         pixels8 = new List<byte>();
         pixels16 = new List<ushort>();
         pixels24 = new List<byte>();
         imageOpened = false;
         signedImage = false;
         maxPixelValue = 0;
         minPixelValue = 65535;
      }

      private void button1_Click(object sender, EventArgs e)
      {
         Form2 form2 = new Form2();
         if (listView1.Items.Count > 0)
         {
            form2.richTextBox1.Text = "";
            TypeReader tagReader = new TypeReader(listView1.SelectedItems[0].Text);
            foreach (TypeValue value in tagReader.Data)
            {
```

```
            form2.richTextBox1.AppendText(value.NameType + " - " + value.Text.TrimEnd('\0') +
                                Environment.NewLine);
            if (value.NameType.Contains("Compression Description"))
            {
               break;
            }
         }
         form2.ShowDialog();
         MessageBox.Show("Преобразование успешно завершено", "Успешное преобразование",
           MessageBoxButtons.OK, MessageBoxIcon.Information);
      }
      else
      {
         MessageBox.Show("Не выбран файл для преобразования", "Ошибка преобразования",
           MessageBoxButtons.OK, MessageBoxIcon.Error);
      }

   }
   private void ReadAndDisplayDicomFile(string fileName, string fileNameOnly)
   {
      dd.DicomFileName = fileName;

      TypeOfDicomFile typeOfDicomFile = dd.typeofDicomFile;

      if (typeOfDicomFile == TypeOfDicomFile.Dicom3File ||
         typeOfDicomFile == TypeOfDicomFile.DicomOldTypeFile)
      {
         imageWidth = dd.width;
         imageHeight = dd.height;
         bitDepth = dd.bitsAllocated;
         winCentre = dd.windowCentre;
         winWidth = dd.windowWidth;
         samplesPerPixel = dd.samplesPerPixel;
         signedImage = dd.signedImage;


         imagePanelControl.NewImage = true;
         Text = "DICOM Image Viewer: " + fileNameOnly;

         if (samplesPerPixel == 1 && bitDepth == 8)
         {
            pixels8.Clear();
            pixels16.Clear();
            pixels24.Clear();
            dd.GetPixels8(ref pixels8);

            minPixelValue = pixels8.Min();
            maxPixelValue = pixels8.Max();

            if (dd.signedImage)
            {
               winCentre -= char.MinValue;
            }

            if (Math.Abs(winWidth) < 0.001)
            {
               winWidth = maxPixelValue - minPixelValue;
            }

            if ((winCentre == 0) ||
```

```
            (minPixelValue > winCentre) || (maxPixelValue < winCentre))
        {
            winCentre = (maxPixelValue + minPixelValue) / 2;
        }

        imagePanelControl.SetParameters(ref pixels8, imageWidth, imageHeight,
            winWidth, winCentre, samplesPerPixel, true, this);
    }

    if (samplesPerPixel == 1 && bitDepth == 16)
    {
        pixels16.Clear();
        pixels8.Clear();
        pixels24.Clear();
        dd.GetPixels16(ref pixels16);

        minPixelValue = pixels16.Min();
        maxPixelValue = pixels16.Max();

        if (dd.signedImage)
        {
            winCentre -= short.MinValue;
        }

        if (Math.Abs(winWidth) < 0.001)
        {
            winWidth = maxPixelValue - minPixelValue;
        }

        if ((winCentre == 0) ||
            (minPixelValue > winCentre) || (maxPixelValue < winCentre))
        {
            winCentre = (maxPixelValue + minPixelValue) / 2;
        }

        imagePanelControl.Signed16Image = dd.signedImage;

        imagePanelControl.SetParameters(ref pixels16, imageWidth, imageHeight,
            winWidth, winCentre, true, this);
    }

    if (samplesPerPixel == 3 && bitDepth == 8)
    {
        pixels8.Clear();
        pixels16.Clear();
        pixels24.Clear();
        dd.GetPixels24(ref pixels24);


        imagePanelControl.SetParameters(ref pixels24, imageWidth, imageHeight,
            winWidth, winCentre, samplesPerPixel, true, this);
    }
}
else
{
    if (typeOfDicomFile == TypeOfDicomFile.DicomUnknownTransferSyntax)
    {
        MessageBox.Show("Sorry, I can't read a DICOM file with this Transfer Syntax.",
            "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
```

```
        else
        {
            MessageBox.Show("Sorry, I can't open this file. " +
                "This file does not appear to contain a DICOM image.",
                "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        Text = "DICOM Image Viewer: ";
        pixels8.Clear();
        pixels16.Clear();
        pixels24.Clear();
        samplesPerPixel = 1;

        imageWidth = imagePanelControl.Width - 25;
        imageHeight = imagePanelControl.Height - 25;
        int iNoPix = imageWidth * imageHeight;

        for (int i = 0; i < iNoPix; ++i)
        {
            pixels8.Add(240);
        }
        winWidth = 256;
        winCentre = 127;
        imagePanelControl.SetParameters(ref pixels8, imageWidth, imageHeight,
            winWidth, winCentre, samplesPerPixel, true, this);
        imagePanelControl.Invalidate();
    }
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    button1.Text = "Преобразовать в " + comboBox1.SelectedItem.ToString();
}

private void button2_Click(object sender, EventArgs e)
{

    Patients form4 = new Patients();
    form4.Owner = this;
    form4.ShowDialog();
}

private void button4_Click(object sender, EventArgs e)
{
    dd = new DicomDecoder();
    imagePanelControl.Refresh();
    listView1.Items.Clear();
}

private void button3_Click(object sender, EventArgs e)
{
    listView1.Items.Clear();
    FolderBrowserDialog fbd = new FolderBrowserDialog();
    fbd.SelectedPath = path;

    DialogResult result = fbd.ShowDialog();

    if (!string.IsNullOrWhiteSpace(fbd.SelectedPath))
    {
        string[] files = Directory.GetFiles(fbd.SelectedPath);
```

```
            foreach (string file in files)
            {
               listView1.Items.Add(file);
            }

         }
      }
      public void UpdateWindowLevel(int winWidth, int winCentre, ImageBitsPerPixel bpp)
      {
         int winMin = Convert.ToInt32(winCentre - 0.5 * winWidth);
         int winMax = winMin + winWidth;

      }

      private void button5_Click(object sender, EventArgs e)
      {

         if (imageOpened == true)
         {
            List<string> str = dd.dicomInfo;

            DicomTagsForm dtg = new DicomTagsForm();
            dtg.SetString(ref str);
            dtg.ShowDialog();

            imagePanelControl.Invalidate();
         }
         else
            MessageBox.Show("Не выбран файл для просмотра информации", "Ошибка просмотра
информации",
               MessageBoxButtons.OK, MessageBoxIcon.Error);

      }

      private void закрытьToolStripMenuItem_Click(object sender, EventArgs e)
      {
         this.Close();
      }

      private void button6_Click(object sender, EventArgs e)
      {
         if (listView1.SelectedItems.Count > 0)
         {
            MainForm form4 = new MainForm(listView1.SelectedItems[0].Text);
            form4.ShowDialog();
         } else
         {
            MessageBox.Show("Не выбран файл для просмотра", "Ошибка просмотра",  MessageBoxButtons.OK,
MessageBoxIcon.Error);
         }
      }

      private void listView1_ItemSelectionChanged(object sender, ListViewItemSelectionChangedEventArgs e)
      {
         Cursor = Cursors.WaitCursor;
         dd = new DicomDecoder();
         ReadAndDisplayDicomFile(e.Item.Text, e.Item.Text);
         imageOpened = true;
         Cursor = Cursors.Default;
```

```
    }

    private void button7_Click(object sender, EventArgs e)
    {
      if (listView1.Items.Count > 0)
      {
        _3Dmodel form4 = new _3Dmodel();
        form4.ShowDialog();
      }
      else
      {
        MessageBox.Show("Отсутствуют файлы для построения 3D-модели", "Ошибка построения 3D-
модели", MessageBoxButtons.OK, MessageBoxIcon.Error);
      }
    }

    private void оПрограммеToolStripMenuItem1_Click(object sender, EventArgs e)
    {
      MessageBox.Show("Автоматизированная система анализа и преобразования томографических
данных\r\n\r\nРазработчик: Лихачев Владислав Сергеевич\r\n\r\nВсе права защищены!", "О программе",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    private void добавитьToolStripMenuItem1_Click(object sender, EventArgs e)
    {
      if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)
      {
        listView1.Items.Add(openFileDialog1.FileName);
      }
    }

    private void преобразоватьBbmpToolStripMenuItem_Click(object sender, EventArgs e)
    {
      if (listView1.SelectedItems.Count > 0)
      {
        SaveFileDialog sfd = new SaveFileDialog();
        sfd.Filter = "PNG Files(*.png)|*.png";

        if (sfd.ShowDialog() == DialogResult.OK)
          imagePanelControl.SaveImage(sfd.FileName);
      }
      else
      {
        MessageBox.Show("Выберите DICOM-файл для сохранения!", "Информация",
          MessageBoxButtons.OK, MessageBoxIcon.Information);
      }


      imagePanelControl.Invalidate();
    }

    private void удалитьToolStripMenuItem_Click(object sender, EventArgs e)
    {
      if (listView1.SelectedItems.Count > 0)
      {
        listView1.Items.Remove(listView1.SelectedItems[0]);
      }
      else
      {
```

```
            MessageBox.Show("Не выбран файл для удаления", "Ошибка удаления файла",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void изменитьРабочийКаталогToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
        {
            MessageBox.Show("Рабочий каталог изменен!", "Изменение рабочего каталога",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        } else
        {
            MessageBox.Show("Не выбран каталог для изменения", "Изменение рабочего каталога",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
  }
}
```

Class MainForm.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Linq;


namespace test1
{
  public enum ImageBitsPerPixel { Eight, Sixteen, TwentyFour };
  public enum ViewSettings { Zoom1_1, ZoomToFit };

  public partial class MainForm : Form
  {
    DicomDecoder dd;
    List<byte> pixels8;
    List<ushort> pixels16;
    List<byte> pixels24;
    int imageWidth;
    int imageHeight;
    int bitDepth;
    int samplesPerPixel;
    bool imageOpened;
    double winCentre;
    double winWidth;
    bool signedImage;
    int maxPixelValue;
    int minPixelValue;

    public MainForm(String filename)
    {
      InitializeComponent();
      dd = new DicomDecoder();
      pixels8 = new List<byte>();
      pixels16 = new List<ushort>();
      pixels24 = new List<byte>();
      imageOpened = false;
      signedImage = false;
```

```
    maxPixelValue = 0;
    minPixelValue = 65535;
    Cursor = Cursors.WaitCursor;
    dd = new DicomDecoder();
    ReadAndDisplayDicomFile(filename, filename);
    imageOpened = true;
    Cursor = Cursors.Default;
}

private void bnOpen_Click(object sender, EventArgs e)
{
    OpenFileDialog ofd = new OpenFileDialog();
    ofd.Filter = "All DICOM Files(*.*)|*.*";
    if (ofd.ShowDialog() == DialogResult.OK)
    {
        if (ofd.FileName.Length > 0)
        {
            Cursor = Cursors.WaitCursor;

            imageOpened = true;
            Cursor = Cursors.Default;
        }
        ofd.Dispose();
    }
}

private void ReadAndDisplayDicomFile(string fileName, string fileNameOnly)
{
    dd.DicomFileName = fileName;

    TypeOfDicomFile typeOfDicomFile = dd.typeofDicomFile;

    if (typeOfDicomFile == TypeOfDicomFile.Dicom3File ||
        typeOfDicomFile == TypeOfDicomFile.DicomOldTypeFile)
    {
        imageWidth = dd.width;
        imageHeight = dd.height;
        bitDepth = dd.bitsAllocated;
        winCentre = dd.windowCentre;
        winWidth = dd.windowWidth;
        samplesPerPixel = dd.samplesPerPixel;
        signedImage = dd.signedImage;


        imagePanelControl.NewImage = true;

        if (samplesPerPixel == 1 && bitDepth == 8)
        {
            pixels8.Clear();
            pixels16.Clear();
            pixels24.Clear();
            dd.GetPixels8(ref pixels8);

            minPixelValue = pixels8.Min();
            maxPixelValue = pixels8.Max();

            if (dd.signedImage)
            {
                winCentre -= char.MinValue;
            }
```

```
            if (Math.Abs(winWidth) < 0.001)
            {
               winWidth = maxPixelValue - minPixelValue;
            }

            if ((winCentre == 0) ||
               (minPixelValue > winCentre) || (maxPixelValue < winCentre))
            {
               winCentre = (maxPixelValue + minPixelValue) / 2;
            }

            imagePanelControl.SetParameters(ref pixels8, imageWidth, imageHeight,
               winWidth, winCentre, samplesPerPixel, true, this);
         }

         if (samplesPerPixel == 1 && bitDepth == 16)
         {
            pixels16.Clear();
            pixels8.Clear();
            pixels24.Clear();
            dd.GetPixels16(ref pixels16);

            minPixelValue = pixels16.Min();
            maxPixelValue = pixels16.Max();

            if (dd.signedImage)
            {
               winCentre -= short.MinValue;
            }

            if (Math.Abs(winWidth) < 0.001)
            {
               winWidth = maxPixelValue - minPixelValue;
            }

            if ((winCentre == 0) ||
               (minPixelValue > winCentre) || (maxPixelValue < winCentre))
            {
               winCentre = (maxPixelValue + minPixelValue) / 2;
            }

            imagePanelControl.Signed16Image = dd.signedImage;

            imagePanelControl.SetParameters(ref pixels16, imageWidth, imageHeight,
               winWidth, winCentre, true, this);
         }

         if (samplesPerPixel == 3 && bitDepth == 8)
         {
            pixels8.Clear();
            pixels16.Clear();
            pixels24.Clear();
            dd.GetPixels24(ref pixels24);


            imagePanelControl.SetParameters(ref pixels24, imageWidth, imageHeight,
               winWidth, winCentre, samplesPerPixel, true, this);
         }
      }
```

```
    else
    {
        if (typeOfDicomFile == TypeOfDicomFile.DicomUnknownTransferSyntax)
        {
            MessageBox.Show("Sorry, I can't read a DICOM file with this Transfer Syntax.",
                "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
        else
        {
            MessageBox.Show("Sorry, I can't open this file. " +
                "This file does not appear to contain a DICOM image.",
                "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        pixels8.Clear();
        pixels16.Clear();
        pixels24.Clear();
        samplesPerPixel = 1;

        imageWidth = imagePanelControl.Width - 25;
        imageHeight = imagePanelControl.Height - 25;
        int iNoPix = imageWidth * imageHeight;

        for (int i = 0; i < iNoPix; ++i)
        {
            pixels8.Add(240);
        }
        winWidth = 256;
        winCentre = 127;
        imagePanelControl.SetParameters(ref pixels8, imageWidth, imageHeight,
            winWidth, winCentre, samplesPerPixel, true, this);
        imagePanelControl.Invalidate();
    }
}

private void bnTags_Click(object sender, EventArgs e)
{
    if (imageOpened == true)
    {
        List<string> str = dd.dicomInfo;

        DicomTagsForm dtg = new DicomTagsForm();
        dtg.SetString(ref str);
        dtg.ShowDialog();

        imagePanelControl.Invalidate();
    }
    else
        MessageBox.Show("Load a DICOM file before viewing tags!", "Information",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
}

private void bnSave_Click(object sender, EventArgs e)
{
    if (imageOpened == true)
    {
        SaveFileDialog sfd = new SaveFileDialog();
        sfd.Filter = "PNG Files(*.png)|*.png";

        if (sfd.ShowDialog() == DialogResult.OK)
```

```
            imagePanelControl.SaveImage(sfd.FileName);
        }
        else
            MessageBox.Show("Load a DICOM file before saving!", "Information",
                MessageBoxButtons.OK, MessageBoxIcon.Information);

        imagePanelControl.Invalidate();
    }

    private void MainForm_Load(object sender, EventArgs e)
    {
        label1.Visible = false;
        label2.Visible = false;
        label3.Visible = false;
        label4.Visible = false;
    }

    private void MainForm_FormClosing(object sender, FormClosingEventArgs e)
    {
        pixels8.Clear();
        pixels16.Clear();
        if (imagePanelControl != null) imagePanelControl.Dispose();
    }

    private void bnResetWL_Click(object sender, EventArgs e)
    {

    }

    public void UpdateWindowLevel(int winWidth, int winCentre, ImageBitsPerPixel bpp)
    {
        int winMin = Convert.ToInt32(winCentre - 0.5 * winWidth);
        int winMax = winMin + winWidth;
        this.windowLevelControl.SetWindowWidthCentre(winMin, winMax, winWidth, winCentre, bpp,
signedImage);
    }

    private void viewSettingsCheckedChanged(object sender, EventArgs e)
    {
        if (rbZoom1_1.Checked)
        {
            imagePanelControl.viewSettings = ViewSettings.Zoom1_1;
        }
        else
        {
            imagePanelControl.viewSettings = ViewSettings.ZoomToFit;
        }

        imagePanelControl.viewSettingsChanged = true;
        imagePanelControl.Invalidate();
    }

    private void imagePanelControl_Load(object sender, EventArgs e)
    {

    }
        }

    }
```

Class DicomDecoder.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Globalization;
using System.Linq;
using System.Text;

namespace test1
{
   public enum TypeOfDicomFile
   {
      NotDicom,
      Dicom3File,
      DicomOldTypeFile,
      DicomUnknownTransferSyntax
   };

   class DicomDecoder
   {
      const uint PIXEL_REPRESENTATION      = 0x00280103;
      const uint TRANSFER_SYNTAX_UID       = 0x00020010;
      const uint MODALITY                  = 0x00080060;
      const uint SLICE_THICKNESS           = 0x00180050;
      const uint SLICE_SPACING             = 0x00180088;
      const uint SAMPLES_PER_PIXEL         = 0x00280002;
      const uint PHOTOMETRIC_INTERPRETATION = 0x00280004;
      const uint PLANAR_CONFIGURATION      = 0x00280006;
      const uint NUMBER_OF_FRAMES          = 0x00280008;
      const uint ROWS                      = 0x00280010;
      const uint COLUMNS                   = 0x00280011;
      const uint PIXEL_SPACING             = 0x00280030;
      const uint BITS_ALLOCATED            = 0x00280100;
      const uint WINDOW_CENTER             = 0x00281050;
      const uint WINDOW_WIDTH              = 0x00281051;
      const uint RESCALE_INTERCEPT         = 0x00281052;
      const uint RESCALE_SLOPE             = 0x00281053;
      const uint RED_PALETTE               = 0x00281201;
      const uint GREEN_PALETTE             = 0x00281202;
      const uint BLUE_PALETTE              = 0x00281203;
      const uint ICON_IMAGE_SEQUENCE       = 0x00880200;
      const uint PIXEL_DATA                = 0x7FE00010;

      const string ITEM                    = "FFFEE000";
      const string ITEM_DELIMITATION       = "FFFEE00D";
      const string SEQUENCE_DELIMITATION   = "FFFEE0DD";

      const int
         AE = 0x4145,
         AS = 0x4153,
         AT = 0x4154,
         CS = 0x4353,
         DA = 0x4441,
         DS = 0x4453,
         DT = 0x4454,
         FD = 0x4644,
         FL = 0x464C,
         IS = 0x4953,
         LO = 0x4C4F,
         LT = 0x4C54,
```

```
    PN = 0x504E,
    SH = 0x5348,
    SL = 0x534C,
    SS = 0x5353,
    ST = 0x5354,
    TM = 0x544D,
    UI = 0x5549,
    UL = 0x554C,
    US = 0x5553,
    UT = 0x5554,
    OB = 0x4F42,
    OW = 0x4F57,
    SQ = 0x5351,
    UN = 0x554E,
    QQ = 0x3F3F,
    RT = 0x5254;
const int ID_OFFSET = 128;
const int IMPLICIT_VR = 0x2D2D;
const String DICM   = "DICM";

public int bitsAllocated;
public int width;
public int height;
public int offset;
public int nImages;
public int samplesPerPixel;
public double pixelDepth = 1.0;
public double pixelWidth = 1.0;
public double pixelHeight = 1.0;
public string unit;
public double windowCentre, windowWidth;
public bool signedImage;
public TypeOfDicomFile typeofDicomFile;
public List<string> dicomInfo;
public bool dicmFound;

DicomDictionary dic;
BinaryReader file;
String dicomFileName;
String photoInterpretation;
bool littleEndian = true;
bool oddLocations;
bool bigEndianTransferSyntax = false;
bool inSequence;
bool widthTagFound;
bool heightTagFound;
bool pixelDataTagFound;
int location = 0;
int elementLength;
int vr;
int min8 = Byte.MinValue;
int max8 = Byte.MaxValue;
int min16 = short.MinValue;
int max16 = ushort.MaxValue;
int pixelRepresentation;
double rescaleIntercept;
double rescaleSlope;
byte[] reds;
byte[] greens;
byte[] blues;
```

```
byte[] vrLetters = new byte[2];
List<byte> pixels8;
List<byte> pixels24;
List<ushort> pixels16;
List<int> pixels16Int;


public DicomDecoder()
{
   dic = new DicomDictionary();
   signedImage = false;
   dicomInfo = new List<string>();
   InitializeDicom();
}

void InitializeDicom()
{
   bitsAllocated = 0;
   width = 1;
   height = 1;
   offset = 1;
   nImages = 1;
   samplesPerPixel = 1;
   photoInterpretation = "";
   unit = "mm";
   windowCentre = 0;
   windowWidth = 0;
   signedImage = false;
   widthTagFound = false;
   heightTagFound = false;
   pixelDataTagFound = false;
   rescaleIntercept = 0.0;
   rescaleSlope = 1.0;
   typeofDicomFile = TypeOfDicomFile.NotDicom;
}

public string DicomFileName
{
   set
   {
      dicomFileName = value;
      InitializeDicom();

      file = new BinaryReader(File.Open(dicomFileName, FileMode.Open, FileAccess.Read));
      location = 0;
      dicomInfo.Clear();
      try
      {
         bool readResult = ReadFileInfo();
         if (readResult && widthTagFound && heightTagFound && pixelDataTagFound)
         {
            ReadPixels();
            if (dicmFound == true)
               typeofDicomFile = TypeOfDicomFile.Dicom3File;
            else
               typeofDicomFile = TypeOfDicomFile.DicomOldTypeFile;
         }
      }
      catch
      {
```

```
            }
            finally
            {
               file.Close();
            }
         }
      }

      public void GetPixels8(ref List<byte> pixels)
      {
         pixels = pixels8;
      }

      public void GetPixels16(ref List<ushort> pixels)
      {
         pixels = pixels16;
      }

      public void GetPixels24(ref List<byte> pixels)
      {
         pixels = pixels24;
      }

      String GetString(int length)
      {
         byte[] buf = new byte[length];
         file.BaseStream.Position = location;
         int count = file.Read(buf, 0, length);
         location += length;
         string s = System.Text.ASCIIEncoding.ASCII.GetString(buf);
         if (s.Contains("??^"))
         {
            Encoding iso_8859_1 = System.Text.Encoding.GetEncoding("iso-8859-5");
            Encoding windows = System.Text.Encoding.GetEncoding("windows-1251");
            s = windows.GetString(buf);
            if (s.Contains("Њ") || s.Contains("ѓ") || s.Contains("\u0098") || s.Contains("°") || s.Contains("ї"))
            {
               s = iso_8859_1.GetString(buf);
               int i = 0;
               bool utf8 = false;
               while (i < s.Length - 1)
               {
                  if (buf[i] <= 0x7F) { i += 1; continue; }
                  if (buf[i] >= 0xC2 && buf[i] <= 0xDF && buf[i + 1] >= 0x80 && buf[i + 1] < 0xC0) { i += 2; utf8
= true; continue; }
                  if (buf[i] >= 0xE0 && buf[i] <= 0xF0 && buf[i + 1] >= 0x80 && buf[i + 1] < 0xC0 && buf[i + 2]
>= 0x80 && buf[i + 2] < 0xC0) { i += 3; utf8 = true; continue; }
                  if (buf[i] >= 0xF0 && buf[i] <= 0xF4 && buf[i + 1] >= 0x80 && buf[i + 1] < 0xC0 && buf[i + 2]
>= 0x80 && buf[i + 2] < 0xC0 && buf[i + 3] >= 0x80 && buf[i + 3] < 0xC0) { i += 4; utf8 = true; continue; }
                  utf8 = false; break;
               }
               if (utf8 == true)
               {
                  s = Encoding.UTF8.GetString(buf);
               }
            }
            s = s.Replace('^', ' ');
         }
         return s;
```

```
}

byte GetByte()
{
   file.BaseStream.Position = location;
   byte b = file.ReadByte();
   ++location;
   return b;
}

ushort GetShort()
{
   byte b0 = GetByte();
   byte b1 = GetByte();
   ushort s;
   if (littleEndian)
      s = Convert.ToUInt16((b1 << 8) + b0);
   else
      s = Convert.ToUInt16((b0 << 8) + b1);
   return s;
}

int GetInt()
{
   byte b0 = GetByte();
   byte b1 = GetByte();
   byte b2 = GetByte();
   byte b3 = GetByte();
   int i;
   if (littleEndian)
      i = (b3 << 24) + (b2 << 16) + (b1 << 8) + b0;
   else
      i = (b0 << 24) + (b1 << 16) + (b2 << 8) + b3;
   return i;
}

double GetDouble()
{
   byte b0 = GetByte();
   byte b1 = GetByte();
   byte b2 = GetByte();
   byte b3 = GetByte();
   byte b4 = GetByte();
   byte b5 = GetByte();
   byte b6 = GetByte();
   byte b7 = GetByte();

   long res = 0;
   if (littleEndian)
   {
      res += b0;
      res += (((long)b1) << 8);
      res += (((long)b2) << 16);
      res += (((long)b3) << 24);
      res += (((long)b4) << 32);
      res += (((long)b5) << 40);
      res += (((long)b6) << 48);
      res += (((long)b7) << 56);
   }
   else
```

```
    {
        res += b7;
        res += (((long)b6) << 8);
        res += (((long)b5) << 16);
        res += (((long)b4) << 24);
        res += (((long)b3) << 32);
        res += (((long)b2) << 40);
        res += (((long)b1) << 48);
        res += (((long)b0) << 56);
    }

    double d = Convert.ToDouble(res, new CultureInfo("en-US"));
    return d;
}

float GetFloat()
{
    byte b0 = GetByte();
    byte b1 = GetByte();
    byte b2 = GetByte();
    byte b3 = GetByte();

    int res = 0;

    if (littleEndian)
    {
        res += b0;
        res += (((int)b1) << 8);
        res += (((int)b2) << 16);
        res += (((int)b3) << 24);
    }
    else
    {
        res += b3;
        res += (((int)b2) << 8);
        res += (((int)b1) << 16);
        res += (((int)b0) << 24);
    }

    float f1;
    f1 = Convert.ToSingle(res, new CultureInfo("en-US"));
    return f1;
}

byte[] GetLut(int length)
{
    if ((length & 1) != 0)
    {
        String dummy = GetString(length);
        return null;
    }

    length /= 2;
    byte[] lut = new byte[length];
    for (int i = 0; i < length; ++i)
        lut[i] = Convert.ToByte(GetShort() >> 8);
    return lut;
}

int GetLength()
```

```
{
   byte b0 = GetByte();
   byte b1 = GetByte();
   byte b2 = GetByte();
   byte b3 = GetByte();

   vr = (b0 << 8) + b1;

   switch (vr)
   {
      case OB:
      case OW:
      case SQ:
      case UN:
      case UT:

         if ((b2 == 0) || (b3 == 0)) return GetInt();

         vr = IMPLICIT_VR;
         if (littleEndian)
            return ((b3 << 24) + (b2 << 16) + (b1 << 8) + b0);
         else
            return ((b0 << 24) + (b1 << 16) + (b2 << 8) + b3);

      case AE:
      case AS:
      case AT:
      case CS:
      case DA:
      case DS:
      case DT:
      case FD:
      case FL:
      case IS:
      case LO:
      case LT:
      case PN:
      case SH:
      case SL:
      case SS:
      case ST:
      case TM:
      case UI:
      case UL:
      case US:
      case QQ:
      case RT:

         if (littleEndian)
            return ((b3 << 8) + b2);
         else
            return ((b2 << 8) + b3);
      default:

         vr = IMPLICIT_VR;
         if (littleEndian)
            return ((b3 << 24) + (b2 << 16) + (b1 << 8) + b0);
         else
            return ((b0 << 24) + (b1 << 16) + (b2 << 8) + b3);
   }
```

```
}

int GetNextTag()
{
   int groupWord = GetShort();
   if (groupWord == 0x0800 && bigEndianTransferSyntax)
   {
      littleEndian = false;
      groupWord = 0x0008;
   }
   int elementWord = GetShort();
   int tag = groupWord << 16 | elementWord;

   elementLength = GetLength();


   if (elementLength == 13 && !oddLocations)
      elementLength = 10;


   if (elementLength == -1)
   {
      elementLength = 0;
      inSequence = true;
   }
   return tag;
}

String GetHeaderInfo(int tag, String value)
{
   string str = tag.ToString("X8");
   if (str == ITEM_DELIMITATION || str == SEQUENCE_DELIMITATION)
   {
      inSequence = false;
      return null;
   }

   string id = null;

   if (dic.dict.ContainsKey(str))
   {
      id = dic.dict[str];
      if (id != null)
      {
         if (vr == IMPLICIT_VR)
            vr = (id[0] << 8) + id[1];
         id = id.Substring(2);
      }
   }

   if (str == ITEM)
      return (id != null ? id : ":null");
   if (value != null)
      return id + ": " + value;

   switch (vr)
   {
      case FD:
         for (int i = 0; i < elementLength; ++i)
            GetByte();
```

```
            break;
        case FL:
            for (int i = 0; i < elementLength; i++)
                GetByte();
            break;
        case AE:
        case AS:
        case AT:
        case CS:
        case DA:
        case DS:
        case DT:
        case IS:
        case LO:
        case LT:
        case PN:
        case SH:
        case ST:
        case TM:
        case UI:
            value = GetString(elementLength);
            break;
        case US:
            if (elementLength == 2)
                value = Convert.ToString(GetShort());
            else
            {
                value = "";
                int n = elementLength / 2;
                for (int i = 0; i < n; i++)
                    value += Convert.ToString(GetShort()) + " ";
            }
            break;
        case IMPLICIT_VR:
            value = GetString(elementLength);
            if (elementLength > 44)
                value = null;
            break;
        case SQ:
            value = "";
            bool privateTag = ((tag >> 16) & 1) != 0;
            if (tag != ICON_IMAGE_SEQUENCE && !privateTag)
                break;
            goto default;
        default:
            location += elementLength;
            value = "";
            break;
    }

    if (value != null && id == null && value != "")
        return "---: " + value;
    else if (id == null)
        return null;
    else
        return id + ": " + value;
}

void AddInfo(int tag, string value)
{
```

```
    string info = GetHeaderInfo(tag, value);

    string str = tag.ToString("X");
    string strPadded = str.PadLeft(8, '0');
    string strInfo;
    if (inSequence && info != null && vr != SQ)
        info = ">" + info;
    if (info != null && str != ITEM)
    {
        if (info.Contains("---"))
            strInfo = info.Replace("---", "Private Tag");
        else
            strInfo = info;

        dicomInfo.Add(strPadded + "//" + strInfo);
    }
}

void AddInfo(int tag, int value)
{
    AddInfo(tag, Convert.ToString(value));
}

void GetSpatialScale(String scale)
{
    double xscale = 0, yscale = 0;
    int i = scale.IndexOf('\\');
    if (i == 1)
    {
        yscale = Convert.ToDouble(scale.Substring(0, i), new CultureInfo("en-US"));
        xscale = Convert.ToDouble(scale.Substring(i + 1), new CultureInfo("en-US"));
    }
    if (xscale != 0.0 && yscale != 0.0)
    {
        pixelWidth = xscale;
        pixelHeight = yscale;
        unit = "mm";
    }
}

public bool ReadFileInfo()
{
    long skipCount = Convert.ToInt32(ID_OFFSET);
    bitsAllocated = 16;

    file.BaseStream.Seek(skipCount, SeekOrigin.Begin);
    location += ID_OFFSET;

    if (GetString(4) != DICM)
    {

        file.BaseStream.Seek(0, SeekOrigin.Begin);
        location = 0;


        dicmFound = false;

    }
    else
    {
```

```
    dicmFound = true;
}

bool decodingTags = true;
samplesPerPixel = 1;
int planarConfiguration = 0;
photoInterpretation = "";
string modality;

while (decodingTags)
{
    int tag = GetNextTag();
    if ((location & 1) != 0)
        oddLocations = true;

    if (inSequence)
    {
        AddInfo(tag, null);
        continue;
    }

    string s;
    switch (tag)
    {
        case (int)(TRANSFER_SYNTAX_UID):
            s = GetString(elementLength);
            AddInfo(tag, s);

            if (s.IndexOf("1.2.840.10008.1.2.2") >= 0)
                bigEndianTransferSyntax = true;
            break;
        case (int)MODALITY:
            modality = GetString(elementLength);
            AddInfo(tag, modality);
            break;
        case (int)(NUMBER_OF_FRAMES):
            s = GetString(elementLength);
            AddInfo(tag, s);
            double frames = Convert.ToDouble(s, new CultureInfo("en-US"));
            if (frames > 1.0)
                nImages = (int)frames;
            break;
        case (int)(SAMPLES_PER_PIXEL):
            samplesPerPixel = GetShort();
            AddInfo(tag, samplesPerPixel);
            break;
        case (int)(PHOTOMETRIC_INTERPRETATION):
            photoInterpretation = GetString(elementLength);
            photoInterpretation = photoInterpretation.Trim();
            AddInfo(tag, photoInterpretation);
            break;
        case (int)(PLANAR_CONFIGURATION):
            planarConfiguration = GetShort();
            AddInfo(tag, planarConfiguration);
            break;
        case (int)(ROWS):
            height = GetShort();
            AddInfo(tag, height);
            heightTagFound = true;
            break;
```

```
case (int)(COLUMNS):
   width = GetShort();
   AddInfo(tag, width);
   widthTagFound = true;
   break;
case (int)(PIXEL_SPACING):
   String scale = GetString(elementLength);
   GetSpatialScale(scale);
   AddInfo(tag, scale);
   break;
case (int)(SLICE_THICKNESS):
case (int)(SLICE_SPACING):
   String spacing = GetString(elementLength);
   pixelDepth = Convert.ToDouble(spacing, new CultureInfo("en-US"));
   AddInfo(tag, spacing);
   break;
case (int)(BITS_ALLOCATED):
   bitsAllocated = GetShort();
   AddInfo(tag, bitsAllocated);
   break;
case (int)(PIXEL_REPRESENTATION):
   pixelRepresentation = GetShort();
   AddInfo(tag, pixelRepresentation);
   break;
case (int)(WINDOW_CENTER):
   String center = GetString(elementLength);
   int index = center.IndexOf('\\');
   if (index != -1) center = center.Substring(index + 1);
   windowCentre = Convert.ToDouble(center, new CultureInfo("en-US"));
   AddInfo(tag, center);
   break;
case (int)(WINDOW_WIDTH):
   String widthS = GetString(elementLength);
   index = widthS.IndexOf('\\');
   if (index != -1) widthS = widthS.Substring(index + 1);
   windowWidth = Convert.ToDouble(widthS, new CultureInfo("en-US"));
   AddInfo(tag, widthS);
   break;
case (int)(RESCALE_INTERCEPT):
   String intercept = GetString(elementLength);
   rescaleIntercept = Convert.ToDouble(intercept, new CultureInfo("en-US"));
   AddInfo(tag, intercept);
   break;
case (int)(RESCALE_SLOPE):
   String slop = GetString(elementLength);
   if (slop.Equals(""))
   {
      rescaleSlope = 0;
   } else
   {
      rescaleSlope = Convert.ToDouble(slop, new CultureInfo("en-US"));
   }
   AddInfo(tag, slop);
   break;
case (int)(RED_PALETTE):
   reds = GetLut(elementLength);
   AddInfo(tag, elementLength / 2);
   break;
case (int)(GREEN_PALETTE):
   greens = GetLut(elementLength);
```

```
            AddInfo(tag, elementLength / 2);
            break;
         case (int)(BLUE_PALETTE):
            blues = GetLut(elementLength);
            AddInfo(tag, elementLength / 2);
            break;
         case (int)(PIXEL_DATA):
            if (elementLength != 0)
            {
               offset = location;
               AddInfo(tag, location);
               decodingTags = false;
            }
            else
               AddInfo(tag, null);
            pixelDataTagFound = true;
            break;
         default:
            AddInfo(tag, null);
            break;
      }
   }
   return true;
}

void ReadPixels()
{
   if (samplesPerPixel == 1 &&  bitsAllocated == 8)
   {
      if (pixels8 != null)
         pixels8.Clear();
      pixels8 = new List<byte>();
      int numPixels = width * height;
      byte[] buf = new byte[numPixels];
      file.BaseStream.Position = offset;
      file.Read(buf, 0, numPixels);
      for (int i = 0; i < numPixels; ++i)
      {
         int pixVal = (int)(buf[i] * rescaleSlope + rescaleIntercept);

         if (photoInterpretation == "MONOCHROME1")
            pixVal = max8 - pixVal;
         pixels8.Add((byte)(pixelRepresentation == 1 ? pixVal : (pixVal - min8)));
      }
   }

   if (samplesPerPixel == 1 &&  bitsAllocated == 16)
   {
      if (pixels16 != null)                  pixels16.Clear();
      if (pixels16Int != null)                  pixels16Int.Clear();

      pixels16 = new List<ushort>();
      pixels16Int = new List<int>();
      int numPixels = width * height;
      byte[] bufByte = new byte[numPixels * 2];
      byte[] signedData = new byte[2];
      file.BaseStream.Position = offset;
      file.Read(bufByte, 0, numPixels * 2);
      ushort unsignedS;
      int i, i1, pixVal;
```

```
        byte b0, b1;

        for (i = 0; i < numPixels; ++i)
        {
          i1 = i * 2;
          b0 = bufByte[i1];
          b1 = bufByte[i1 + 1];
          unsignedS = Convert.ToUInt16((b1 << 8) + b0);
          if (pixelRepresentation == 0)
          {
            pixVal = (int)(unsignedS * rescaleSlope + rescaleIntercept);
            if (photoInterpretation == "MONOCHROME1")
              pixVal = max16 - pixVal;
          }
          else
          {
            signedData[0] = b0;
            signedData[1] = b1;
            short sVal = System.BitConverter.ToInt16(signedData, 0);

            pixVal = (int)(sVal * rescaleSlope + rescaleIntercept);
            if (photoInterpretation == "MONOCHROME1")
              pixVal = max16 - pixVal;
          }
          pixels16Int.Add(pixVal);
        }

        int minPixVal = pixels16Int.Min();
        signedImage = false;
        if (minPixVal < 0) signedImage = true;

        foreach (int pixel in pixels16Int)
        {
          if (signedImage)
            pixels16.Add((ushort)(pixel - min16));
          else
            pixels16.Add((ushort)(pixel));
        }

        pixels16Int.Clear();
      }

      if (samplesPerPixel == 3 && bitsAllocated == 8)
      {
        signedImage = false;
        if (pixels24 != null)
          pixels24.Clear();
        pixels24 = new List<byte>();
        int numPixels = width * height;
        int numBytes = numPixels * samplesPerPixel;
        byte[] buf = new byte[numBytes];
        file.BaseStream.Position = offset;
        file.Read(buf, 0, numBytes);

        for (int i = 0; i < numBytes; ++i)
        {
          pixels24.Add(buf[i]);
        }
      }
    }
```

Магистерская диссертация выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) (Ф.И.О.)