Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки Программная инженерия

РАБОТА ПРОВЕРЕНА
Рецензент,

_____
«____»_____2017г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
доцент
_____А.А.Замышляева
«____»_____2017 г.

Разработка программного модуля определения антропометрических параметров человека по фотографии

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–09.03.04.2017.73.ПЗ ВКР

Руководитель работы, ст. преп.
_____/М.Ю. Сартасова
«____»_____2017 г.
Автор работы
Студент группы ЕТ-484
_____/ Д.С. Худжанов
«____»_____2017 г.
Нормоконтролер, доцент
_____/Т.Ю. Оленчикова
«____»_____2017 г.

Челябинск 2017

АННОТАЦИЯ

Худжанов Д. С. Разработка программного модуля определения антропометрических параметров человека по фотографии. – Челябинск: ЮУрГУ, ЕТ-484, 45 с., 38 ил., 1 табл., библиогр. список – 17 наим., 1 прил.

Данная работа посвящена разработке программного модуля определения антропометрических параметров человека по изображению.

В работе выполнен обзор существующих приложений подобного типа, наиболее востребованных графических библиотек, позволяющих осуществлять анализ и синтез изображений, а также методы аппроксимации поверхности.

Разработана математическая модель и алгоритмы работы приложения. Реализован графический интерфейс пользователя. Разработано и отлажено приложение для построения 3D модели и определения основных антропометрических показателей человека.

Программа реализована на языке программирования C++. В приложении приведён текст программы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....6
1 СУЩЕСТВУЮЩИЕ МЕТОДЫ И СПОСОБЫ РЕШЕНИЯ ЗАДАЧИ ОПРЕДЕЛЕНИЯ АНТРОПОМЕТРИЧЕСКИХ ПАРАМЕТРОВ ЧЕЛОВЕКА ПО ИЗОБРАЖЕНИЮ.....7
1.1 Методика антропометрических обследований.....7
1.2 Патенты на программное обеспечение, решающее задачи определения антропометрических параметров человека по изображению.....8
1.3 Графические библиотеки, позволяющие осуществлять анализ и синтез изображений.....10
1.4 Vulkan.....16
1.5 Выбор программного интерфейса (API) для разработки приложения с 3D графикой.....17
1.6 Кривые Безье и триангуляция поверхности.....17
1.7 Выводы по разделу.....20
2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ ЗАДАЧИ ОПРЕДЕЛЕНИЯ АНТРОПОМЕТРИЧЕСКИХ ПАРАМЕТРОВ ЧЕЛОВЕКА ПО ИЗОБРАЖЕНИЮ.....21
2.1 Модель определения антропометрических параметров.....21
2.2 Модель формирования изображения.....23
2.3 Выводы по разделу.....25
3 РАЗРАБОТКА ПРИЛОЖЕНИЯ.....26
3.1 Разработка алгоритмов.....26
3.2 Разработка интерфейса.....32
3.3 Выводы по разделу.....36
4 ПРОВЕРКА РАБОТЫ ПРИЛОЖЕНИЯ.....37
4.1 Проверка работы программы на экспериментальных данных.....37
4.2 Оценка погрешности результата.....41
4.3 Выводы по разделу.....42
ЗАКЛЮЧЕНИЕ.....43
БИБЛИОГРАФИЧЕСКИЙ СПИСОК.....44
ПРИЛОЖЕНИЕ 1 ТЕКСТ ПРОГРАММЫ.....46

ВВЕДЕНИЕ

Актуальность данной темы обусловлена тем, что каждый день всё больше людей заказывают одежду в интернете. Есть люди, которые боятся заказывать одежду в интернете, потому что может не подойти размер и придётся отправлять одежду обратно. Продавцы магазинов терпят убытки из-за того, что люди возвращают товар. Если в обычных магазинах можно прийти и просто померить одежду, то в интернете так сделать нельзя. Количество возвратов одежды колеблется в пределах от 20 до 40%. Для разработки одежды необходимо знать не только внешнюю форму тела человека, но и основные размеры. Правильно сшитая одежда в основном зависит от точности снятия измерений. Поэтому возникла необходимость разработать программу, которая бесконтактно измеряет антропометрические данные человека и тем самым сокращает количество возвратов в магазин.

Целью данной работы является разработка программного модуля, определения антропометрических параметров человека по трём фотографиям: вид спереди, вид сбоку, вид сзади.

Для достижения данной цели необходимо решить следующие задачи.

1. Выполнить обзор существующих методов и способов, которые решают задачи определения антропометрических параметров человека по изображению.
2. Провести анализ патентов на программное обеспечение, решающих задачи определения антропометрических параметров человека по изображению.
3. Изучить графические библиотеки, позволяющие осуществлять анализ и синтез изображений.
4. Выполнить обзор методов аппроксимации поверхности.
5. Разработать математическую модель задачи определения антропометрических параметров человека по изображению.
6. Разработать и реализовать алгоритмы работы приложения.
7. Реализовать графический интерфейс пользователя.
8. Выполнить проверку работы приложения.

1 СУЩЕСТВУЮЩИЕ МЕТОДЫ И СПОСОБЫ РЕШЕНИЯ ЗАДАЧИ ОПРЕДЕЛЕНИЯ АНТРОПОМЕТРИЧЕСКИХ ПАРАМЕТРОВ ЧЕЛОВЕКА ПО ИЗОБРАЖЕНИЮ

Антропометрия – один из основных методов антропологического исследования, состоящий в измерении тела человека и его частей. Результаты исследования размеров тела человека используются для различных целей, например, при конструировании школьной мебели, клинической и спортивной медицине.

Информация о форме и размерах человеческого тела важны для различных отраслей промышленности. Например, массовое производство одежды, соответствующей формам и размерам тела человека, возможно тогда, когда специалисты швейной промышленности будут иметь информацию о форме и размерах человеческого тела и их особенностей среди различных групп населения. Информация может быть получена на основе антропометрических исследований. Основная задача швейной промышленности – обеспечение правильной посадки изделий на теле человека [1].

При разработке конструкций изделий для массового производства используют таблицы измерений мужских, женских и детских типовых фигур, созданных на основе антропометрических исследований, а при изготовлении изделий по индивидуальным заказам проводят антропометрические измерения конкретной фигуры. Для точного измерения фигуры, необходимы ориентирные точки на его поверхности. В антропологии такие точки называют антропометрическими [2].

1.1 Методика антропометрических обследований

В классической антропометрии используется более 100 антропометрических точек, а для швейной промышленности достаточно использовать не более 20 из них (см. рисунок 1.1). При разработке стандартов используются как классические антропометрические точки, так и точки на мягких тканях, которые являются базовыми при измерении некоторых признаков [3].

Фигура определяется рядом отдельных измерений фигуры человека, называемых размерными признаками. Все измерения тела человека производятся в вертикальных и горизонтальных плоскостях. Вертикальная плоскость – это плоскость, которую можно мысленно провести от головы до ног вдоль позвоночника, а также параллельные ей плоскости. Горизонтальные плоскости, проходящие перпендикулярно вертикальной плоскости.

Для точности измерений требуется строгое соблюдение техники измерения. Измеряемый должен стоять прямо без напряжения, сохраняя привычную осанку и режим дыхания. Голова фиксируется в одной плоскости [4]. Руки опущены, пальцы вытянуты, ноги выпрямлены в коленях, пятки вместе, носки 15,0– 20,0 см

друг от друга. При антропометрических обследованиях все измерения производят в нижнем белье, без обуви. Точность линейных измерений – 0,1 см.

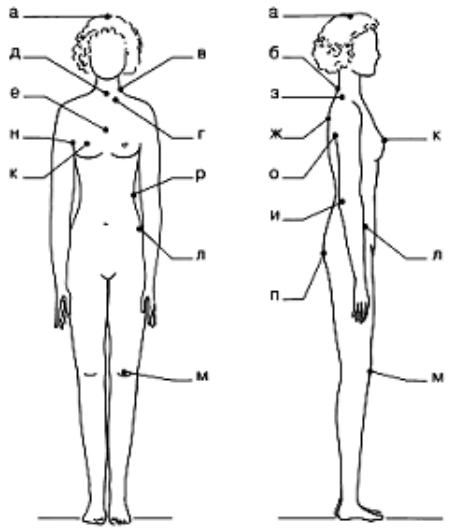


Рисунок 1.1 – Основные антропометрические точки

Для точного измерения размеров и связанных с ними других измерений необходимо установить горизонталь. Горизонталь устанавливается на талии, так как талию можно точно определить на фигуре.

Все измерения проводятся точно по фигуре без добавления, так как они добавляются при построении чертежей и зависят от вида изделия, силуэта и ткани [5].

Парные измерения снимаются по правой стороне тела человека, так как она является наиболее развитой.

Чертеж конструкции с учетом симметричности строят для одной половины фигуры, поэтому обхваты (кроме обхвата плеча), ширины (кроме ширины плеча) и расстояния между центрами груди сначала измеряются в полный размер, а потом записывают половину от каждого измерения. Все остальные измерения записывают полностью.

Измерения должны начинаться сверху и идти в строгой последовательности, чтобы избежать лишних движений измеряемого [6].

1.2 Патенты на программное обеспечение, решающее задачи определения антропометрических параметров человека по изображению

Патент DirkRutshmann: программа основана на том, что человек стоит в центре неподвижно, а вокруг него по орбите на фиксированном расстоянии вращается камера. Недостатками данного метода являются: тело должно быть зафиксировано в процессе съёмки; необходимо специальное оборудование, чтобы камера вращалась вокруг тела [7].

Патент ZiquanHong, Ryolnoshiri, Akira Yashida: человека фотографируют с заданного расстояния специальной камерой чувствительной к инфракрасным

лучам и построение трёхмерной модели тела человека. Недостаткам данного метода являются: использование дорогостоящей камеры; большие погрешности в расчётах [8].

Система «Image Twin» разработана Textile/Closing Technology Corporation, (TC2). Основана на использовании 6 зафиксированных камер (см. рисунок 1.2). Такое количество камер позволяет определить более 400000 точек, по которым строится фигура. Недостатком данного метода является использование большого количества камер [9].

Рисунок 1.2 – Рабочее окно приложения «Image Twin»

Система «Symcad» разработана Telmat Industries. Основана на сканировании человека, находящегося в кабинке с освещенной стеной, камерой и компьютером (см. рисунок 1.3). Когда человек сфотографировался на компьютере появляется трёхмерное изображение. Весь процесс занимает 30 секунд. Недостатком данного метода является дорогостоящее, специализированное оборудование [10].

Рисунок 1.3 – Кабинка для съёмки и построения трёхмерной модели человека

Система «Garment CAD System» в которую входит модуль AGSM-3D для 3D проектирования разработана Ashasi Karel AGSM Corporation. В данном модуле берётся манекен с типовыми размерами, а потом он корректируется по специальным алгоритмам. Недостатками данного метода являются сложность алгоритма; большая погрешность измерений [11].

1.3 Графические библиотеки, позволяющие осуществлять анализ и синтез изображений

OpenGL и Direct3D - две основные на сегодняшний день аппаратно-ускоряемые библиотеки для создания компьютерной трехмерной графики. Базовые функции реализованы аппаратно, а более сложные функции через программные модули, в основе которых лежат базовые команды. Когда функции, необходимые для работы приложения, не поддерживаются аппаратно видеокартой, тогда используются программные модули.

OpenGL

OpenGL создавалась как библиотека для работы с графикой. В основе интерфейса программирования приложений(API) лежит ядро, которое обрабатывает примитивные объекты. Сам интерфейс содержит несколько сотен процедур и функций, используемых для создания графических изображений в высоком качестве. Для аппаратной реализации многих функций в видеокарте должен быть кадровый буфер, причем некоторые операции выполняются только в нём (см. рисунок 1.4).

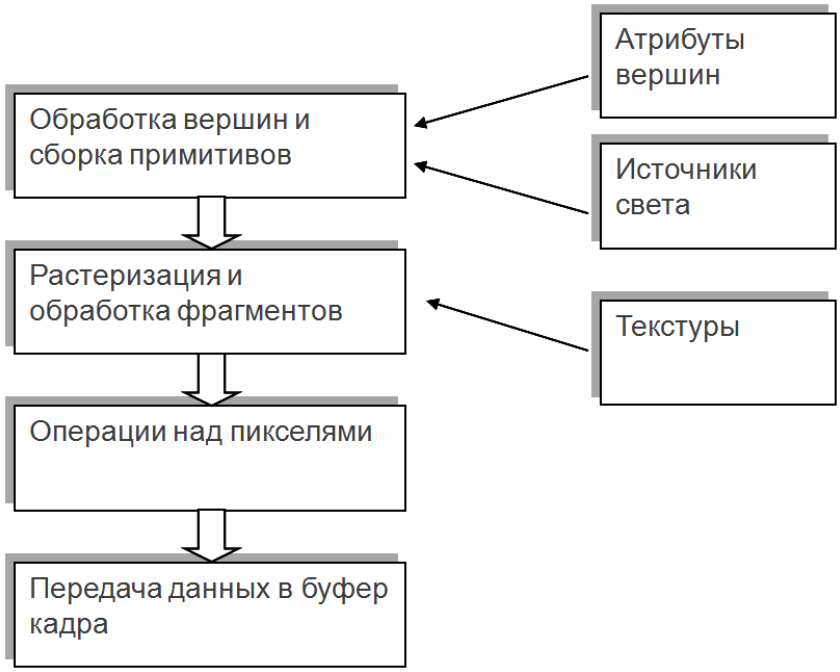


Рисунок 1.4 – Общая схема работы графической библиотеки OpenGL

Чтобы работать с библиотекой, программе необходимо открыть окно в кадровом буфере, а потом контекст библиотеки связывается с этим окном. Все эти

действия проделываются один раз, а далее все действия происходят через рисование простых геометрических объектов (линия, точка, полигон и т.д.).

OpenGL для разработчика процессора видеокарты это набор команд, виляющих на работу графического железа. Библиотека должна быть встроена в графический процессор, если есть только адресуемый кадровый буфер. Как правило помимо кадрового буфера, видеокарта содержит графический ускоритель. Процессор видеокарты способен лишь считать и трансформировать графические данные. Разработчик графической платы должен создать программный интерфейс графического процессора для разделения обработки команд библиотеки между графическим процессором и остальным графическим железом. Такое разделение необходимо оптимизировать под выполнение команд OpenGL.

Библиотека реализована в виде информации о структуре, определяющей как в кадровом буфере будут нарисованы объекты, часть структур доступны пользователю, который может вызывать различные процедуры для изменения параметров.

Рассмотрим базовые понятия, необходимые для рисования текстурированной пирамиды:

Vertex - рисуется вершина после указания координат в двух, трёх или четырёхмерном пространстве.

Vertex Arrays – массив для работы с множеством вершин, необходимый для снижения вычислительной нагрузки.

Buffer Objects (буферные объекты) – сохраняет данные в быстрой памяти видеоадаптера для повторного использования, есть буферные объекты, которые распределяют, инициализируют и извлекают информацию об объекте из памяти.

Evaluators – инструмент для множественных преобразований координат и цвета.

Coordinate Transformation – все объекты перед выводом в кадровый буфер претерпевают изменение своих координат. В этот момент обрезаются те части фигур, которые не помещаются на экране.

Rasterization – процесс, в котором вся картинка преобразовывается к 2D виду. Растеризатор обрабатывает объект, раскрашивает его, накладывает текстуру, и на выходе получается кадр, который далее поступает в кадровый буфер, а оттуда на экран.

Программист последовательно определяет только основные параметры для той или иной операции. Всё остальное происходит без его участия, средствами библиотеки, драйвера и, наконец, видеоадаптера.

FrameBuffer (кадровый буфер) - область памяти для временного хранения информации о точках, составляющих один кадр изображения на мониторе. Открыть окно в кадровом буфере — значит выделить видеопамять под изображение.

Программная поддержка:

Так как библиотека разрабатывалась группой компаний, она мультиплатформенная и поддерживается многими языками программирования и

операционными системами. Причем у каждого производителя видеоадаптеров есть возможность купить лицензию и выпустить свою версию OpenGL как для компьютера, так и для других устройств. Причем наблюдательный совет за архитектурой OpenGL просматривает большую часть расширений, создаваемых производителями железа, и могут включить их в спецификацию новой версии библиотеки [12].

1.3.1 DirectX

Данный программный интерфейс приложения является комплексной библиотекой, которая содержит в себе интерфейсы по обработке не только графики.

Аналогично OpenGL, DirectX на низком уровне взаимодействует с аппаратной частью компьютера и предоставляет программисту набор интерфейсов для обработки графики, однако большим отличием от конкурента является использование объектной модели компонентов. То есть для работы с изображением необходимо не просто вызвать некоторую функцию или процедуру, а сделать ряд манипуляций для получения доступа к объектам, а затем для всех объектов необходимо заполнить некий набор необходимых свойств, без которых он не определён. Всё это необходимо для программиста, так как после компиляции программы команды на обработку графики идут последовательно, не зависимо от того, как написана программа.

Для начала необходимо открыть окно программы, а потом создать устройство отрисовки и тем самым инициализировать Direct3D. Следом за этим идет процесс просчета и отображения сцены (см. рисунок 1.5).

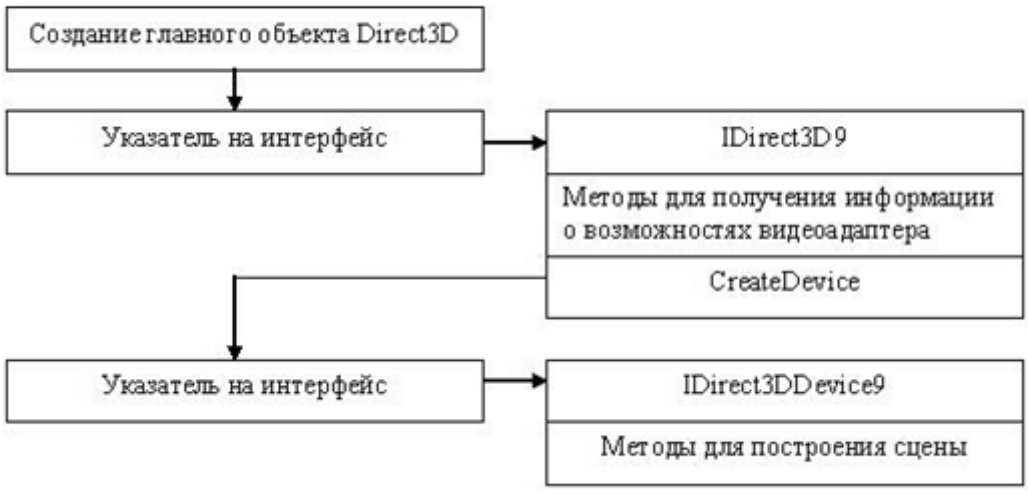


Рисунок 1.5 – Схема работы библиотеки Direct3D

На аппаратном уровне программный интерфейс приложения выполняется так: Direct3D обеспечивает независимость от аппаратного уровня абстракции. Аппаратный уровень абстракции – это железо-зависимый интерфейс, который разрабатывается создателем видеоадаптера и включает в себя поддержку

Direct3D. Microsoft описала, как должна быть реализована Direct3D аппаратно, а производители реализуют необходимые функции в чипах. Аппаратный уровень абстракции может реализовывать три разных типа модели обработки вершин - программный, аппаратный и смешанный. Кроме использования возможностей видеокарты, DirectX использует инструкции AMD 3DNow! и Intel Pentium SIMD, которые обрабатывают некоторые вершины шейдеров.

Для создания сцены требуется инициализировать Direct3D объект для рисования, включая создание объекта, указание параметров отображения, создание Direct3D устройства, притом в параметрах устройства существует возможность указать способ обработки вершин – аппаратный или программный. Затем происходят похожие с OpenGL преобразования, учетом того, что все операции работают над объектами (см. рисунок 1.6). При завершении программы необходимо закрыть устройство.

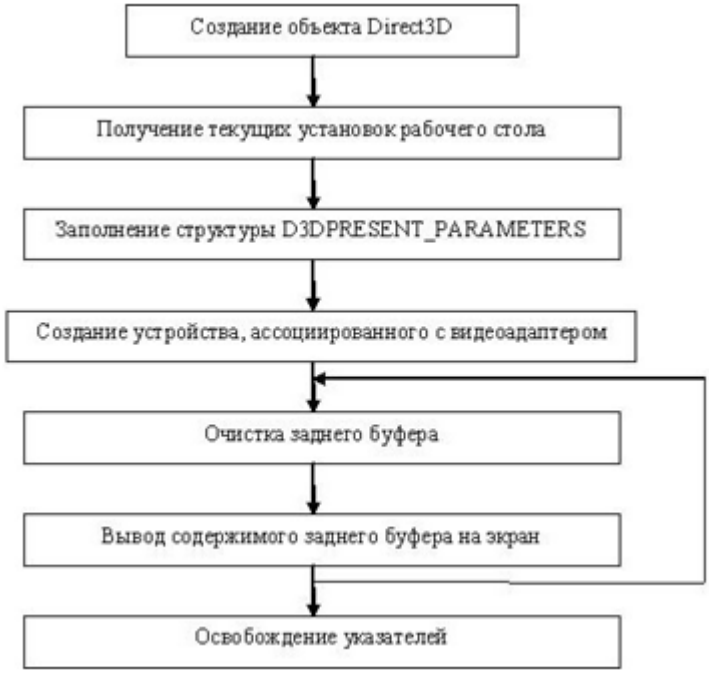


Рисунок 1.6 – Схема инициализации и процедуры рендеринга DirectX

Pixel Shaders - это небольшие подпрограммы, выполняющиеся на графическом процессоре, которые работают с каждым пикселем при создании текстуры и используются для просчета света, создания разнообразных эффектов. В OpenGL они отсутствуют так как когда создавалась спецификация, шейдеры еще не придумали, но сейчас их включили в виде расширений.

Библиотека XNA – надстройка над классическим DirectX для операционной системы Windows. А также XNA используется для других операционных систем. Она является кроссплатформенной библиотекой.

Библиотека разработана корпорацией Microsoft для разработки единого инструмента для создания игр на разных на платформах, принадлежащих одной и

той же компании, а именно: Microsoft Windows и Microsoft XBOX360. Поэтому XNA не является банальной оберткой DirectX.

Во-первых, математические классы XNA не являются обертками над DirectX, а разработаны с нуля. Это позволяет добиться производительности на уровне классического DirectX.

Во-вторых, используются специализированные классы библиотеки, с их помощью разработчик может не писать обработчики таких задач, как обработка потери устройства, загрузка/выгрузка контента, создания игрового цикла и многое другое, что усложняло жизнь программисту.

Однако библиотека XNA, несмотря на оптимизированность и удобство в использовании, нацелена главным образом для создания игр. Большая часть игр предполагает постоянную перерисовку кадра с большой затратой ресурсов. Данная библиотека имеет целью затратить как можно меньше вычислительных ресурсов процессора и видеокарты.

Программная поддержка: библиотека создана одной лишь Microsoft, и существует только официальная поддержка операционных систем семейства Windows и платформы XBox. Исходя из этого развитием и внедрением новых функций занимается только эта корпорация.

1.3.2 Сравнение OpenGL и Direct3D

Библиотеки отличаются друг от друга обработкой графической информации, если OpenGL – это процедурная система, то DirectX основан на объектной модели компонентов. Они обе настроены под взаимодействие с «железом». OpenGL не поддерживает шейдеры. DirectX может эмулировать некоторые функции, не поддерживаемые аппаратно, но на уровне пикселей это неосуществимо и программе необходимой самой проверять возможна ли такая эмуляции. Для внедрения новых технологий, у OpenGL есть механизм расширений, когда разнообразные функции разрабатываются производителем железа и поддерживаются спецификацией для определённой модели видеоадаптера. DirectX система отличается от OpenGL – определённая версия библиотеки может не содержать каких-то функций, даже если они есть в железе, для их поддержки необходимо ждать новой версии библиотеки, но Microsoft часто общается с производителями видеокарт, так что такая ситуация встречается довольно редко.

Microsoft включает в Windows библиотеки OpenGL, правда слегка устаревшую, которая не применяет аппаратные возможности ускорения, этот недостаток исправляют драйверы графических ускорителей [13].

Также компания Microsoft выпустила несколько версий библиотеки DirectX на платформу .NET. Для реализации алгоритмов настоящей работы будет использовать язык C# платформы .NET. OpenGL также адаптирована на C#, но, так как она является процедурной, она весьма плохо вписывается в концепцию объектно-ориентированного языка.

Хотя поддержка на уровне железа и реализована по-разному, выяснить какой механизм лучше трудно, так как новые функции OpenGL доступны только через расширения, а у DirectX необходимо ждать новой версии библиотеки.

Если обратить внимание на программную реализацию этих библиотек, то можно сделать вывод: программировать намного легче с использованием OpenGL, чем писать под DirectX. Однако разработчики Direct3D постоянно ищут способы облегчить работу программистам, объединяя часто используемые вызовы и операции в общие файлы (см. рисунок 1.7).

Обе библиотеки облегчают работу программиста и обеспечивают интерфейс программы с конкретным железом. Они не совместимы друг с другом и имеют свои особенности, поэтому производителям необходимо реализовывать в своих ядрах базовые функции обоих программных интерфейсов приложения, причем необходимо учитывать дополнительные функции новых версий и расширений.

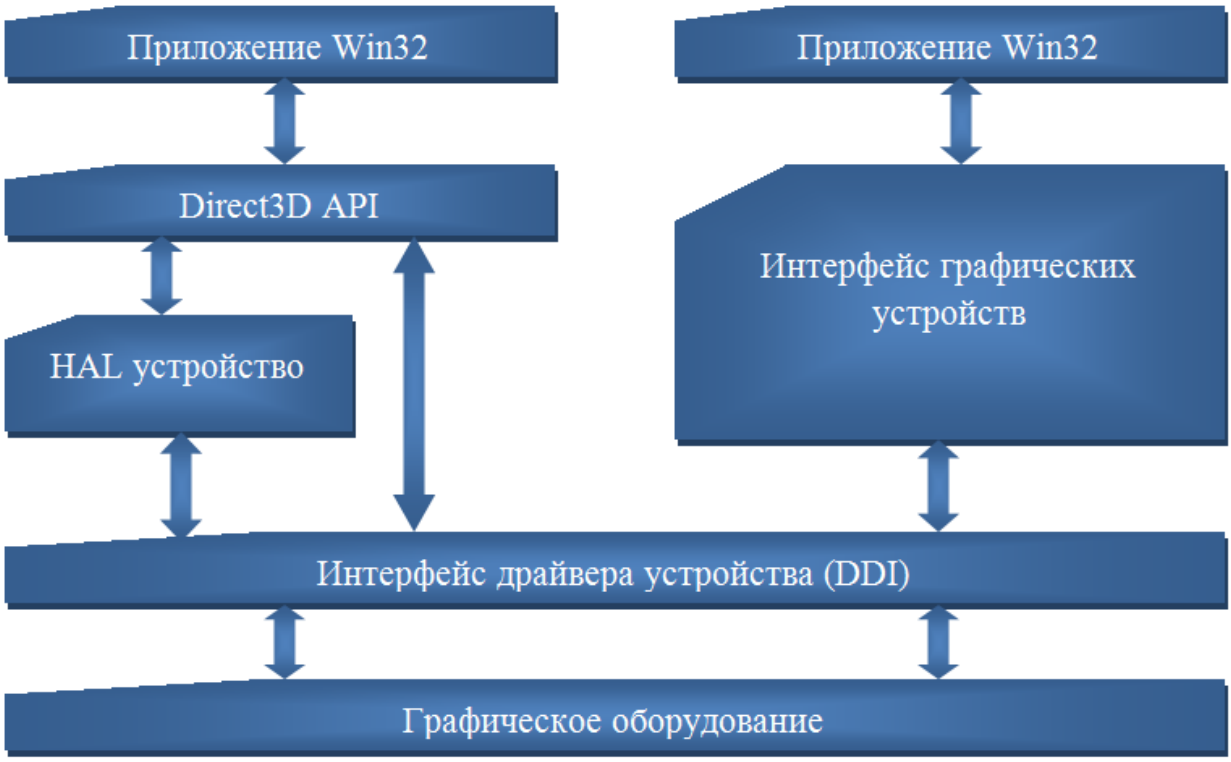


Рисунок 1.7 – Сравнительная схема программных реализаций DirectX (слева) и OpenGL (справа)

Итак, обе библиотеки достаточно мощные и производительные, но они имеют преимущества, проявляющиеся в разных областях. DirectX наилучшим образом подходит для создания игр и других графических приложений под операционной системой Windows, OpenGL же лучше всего подходит для мощных рабочих станций и там, где необходима совместимость приложений с различными платформами. Хотя и для игр эта библиотека так же достаточно востребована.

1.4 Vulkan

Vulkan – это низкоуровневый интерфейс программирования приложений, который даёт разработчикам прямой доступ к графическому процессору для полного контроля над его работой. Библиотека отличается более простыми и легкими драйверами, поэтому она демонстрирует меньшие задержки и меньшие накладные расходы при обработке графических команд в отличие от традиционных OpenGL и Direct3D. Vulkan отличается эффективной поддержкой многопоточности и даёт возможность многоядерным центральным процессорам наиболее эффективно загружать графический конвейер, увеличивая производительность железа.

Данная библиотека является кроссплатформенной. Программисты могут разрабатывать приложения для персональных компьютеров, мобильных и встроенных устройств, работающих под разнообразными операционными системами. Также как и OpenGL, Vulkan – это открытый бесплатный стандарт, доступный для разных платформ.

Преимущества Vulkan для пользователей

Данный интерфейс программирования приложений снижает затраты на адаптацию игр и открывает новые возможности для приложений на разных платформах [14].

В драйверах Vulkan и OpenGL используется бинарная архитектура, которая позволяет применять шейдеры в Vulkan. Программисты могут или остаться на OpenGL, или сменить OpenGL на Vulkan.

В ближайшем будущем библиотека будет доступна для Android и Linux. Vulkan будет поставляться вместе с OpenGL ES как ключевой интерфейс программирования приложений в новых версиях Android. Это значит, что у Android будет современный интерфейс программирования приложений с интегрированной графикой и вычислительной системой, что раскроет потенциал графического процессора для новейших визуальных и вычислительных приложений, а также для фантастической игровой графики.

Рассмотрев две наиболее популярных графических библиотеки – OpenGL и DirectX, одну из наиболее прогрессивных технологий для разработки трехмерных приложений – библиотеку XNA, а также одну из последних разработок библиотеку Vulkan можно сделать следующие выводы о выборе технологии для реализации системы согласно задаче настоящей работы. Для данной работы наиболее оптимальной библиотекой для написания программы является OpenGL так как код более простой в написании, в дальнейшем не составит труда перенести программу на другую платформу, функционала, который предоставляет библиотека достаточно для написания приложения, а также OpenGL обладает рядом преимуществ.

1.5 Выбор программного интерфейса (API) для разработки приложения с 3D графикой

Пять основных преимуществ, важных с точки зрения получаемых результатов.

Производительность. В OpenGL заложена возможность отрисовки динамических сцен. Для получения необходимых результатов в систему заложено большое количество параметров (режимов рисования). Если режим или их комбинация на данном устройстве не позволяет обеспечить интерактивного взаимодействия и достаточной частоты обновления сцены, то у пользователя или у программы должна быть возможность отключать дополнительные функции до тех пор, пока не получится «живой» картинки [12].

Независимость. Практически все функции OpenGL являются независимыми. То есть существует возможность использовать их в произвольной комбинации.

Полнота. OpenGL старается не использовать всего, что должно быть реализовано программно. Так гарантируется получение рабочей картинки. То есть, если программа работает на одной платформе, то этаже программа будет работать и на другой.

Интероперабельность. В сети важно передавать данные между компьютерами на разных платформах. Поэтому OpenGL изначально ориентирован на работу в режиме клиент-сервер, даже если и клиент, и сервер расположены на одном и том же компьютере.

Расширяемость. Так как OpenGL соответствует максимальным возможностям аппаратуры, то он содержит механизмы добавления новых функций. С другой стороны, плохой интерфейс затрудняет жизнь программистов, поэтому новые функции накапливаются некоторое время и выпускаются в новой версии.

1.6 Кривые Безье и триангуляция поверхности

1.6.1 Кривые Безье

Кривые Безье применяются в компьютерной графике для рисования плавных изгибов, в CSS-анимации и многом другом.

Виды кривых Безье

Кривая Безье задаётся опорными точками. Их может быть две, три, четыре или больше (см. рисунок 1.8).

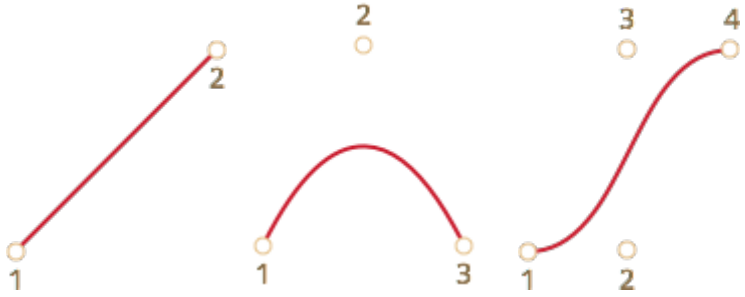


Рисунок 1.8 – Кривая Безье по двум, трём и четырём опорным точкам

Не всегда точки располагаются на кривой. Степень кривой равняется числу точек без одной. Для двух точек – это прямая, для трёх точек – парабола, для четырёх – кубическая.

Кривая всегда находится внутри выпуклой оболочки (см. рисунок 1.9), образованной опорными точками:

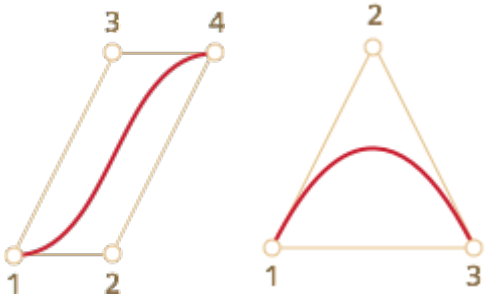


Рисунок 1.9 – Кривая Безье внутри выпуклой оболочки

Благодаря выпуклой оболочке в компьютерной графике существует возможность оптимизировать проверку пересечений двух кривых. Если они не пересекаются, то и кривые тоже не пересекаются [15].

Основная ценность кривых Безье для рисования – двигая точки, можно менять кривую, при этом кривая меняется предсказуемым образом (см. рисунок 1.10). После недолгой работы с кривыми становится понятно, как поместить точки, чтобы получить нужную форму. А, соединяя несколько кривых, существует возможность получить практически что угодно.

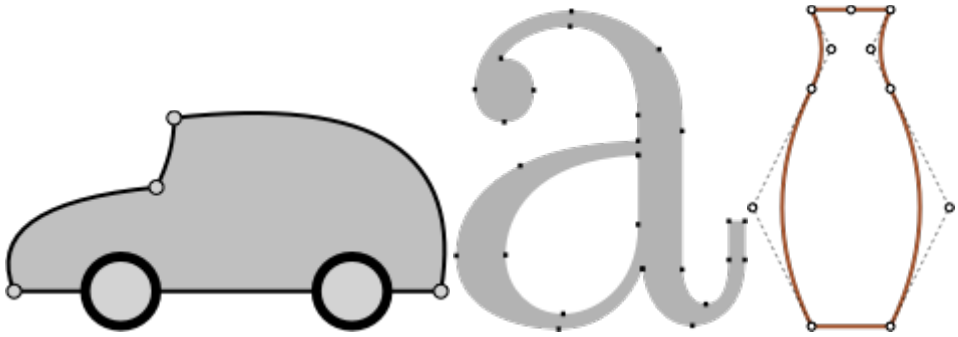


Рисунок 1.10 – Примеры соединения нескольких кривых

1.6.2 Триангуляция поверхностей

Триангуляция представляет собой аппроксимацию поверхности моделируемого объекта треугольными пластинами, отстоящими от нее на расстоянии, не превышающем некоторой заданной величины δ. Все треугольные пластины должны стыковаться между собой. Их вершины лежат на поверхности (см. рисунок 1.11). С набором треугольных пластин легче работать, чем с поверхностью общего вида. Треугольные пластины называются треугольниками. Для треугольника достаточно быстро вычисляются расстояние до заданной точки или точка пересечения с заданной прямой в пространстве. Триангуляция граней выполняется для визуального восприятия геометрической модели, поэтому стороны треугольников выбираются, такими, чтобы глаз не мог заметить изломы.

При отображении геометрических объектов по треугольникам на параметрических плоскостях поверхностей должна быть построена пространственная триангуляция граней тела путем вычисления массива точек в пространстве p_i (u_i, v_i) и массива нормалей $m_i(u_i, v_i)$ к граням тела в этих точках по массиву двухмерных точек $p_i = [u_i\ v_i]^T$. Для быстрого отображения тел их грани аппроксимируют треугольными пластинами, построенными на точках p_i. Нормали требуются для определения поведения световых лучей, взаимодействующих с гранями тела.

Результатом триангуляции поверхности мы хотим иметь массив двухмерных точек $p_i = [u_i\ v_i]^T$ на параметрической плоскости и массив троек целых чисел, являющихся номерами точек в первом упомянутом массиве. Таким образом, каждый треугольник будет представлен тремя номерами его вершин в массиве параметров. По каждой двухмерной точке параметрической области могут быть вычислены пространственная точка p_i (u_i, v_i) на поверхности и нормаль $m_i(u_i, v_i)$ поверхности в ней. Пространственные точки и нормали могут храниться в массивах, аналогичных массиву двухмерных точек [16].

Остановимся на некоторых способах триангуляции. Для плоских поверхностей существуют экономичные методы триангуляции, в которых треугольники строятся на граничных точках поверхности и не требуется искать точки внутри параметрической области.

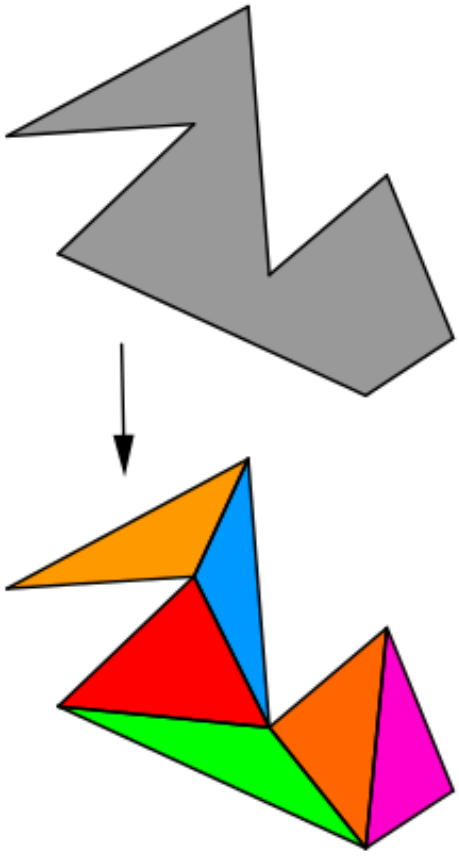


Рисунок 1.11 – Триангуляция многоугольника

Для работы решено использовать триангуляцию поверхности, так как она позволяет быстро и наглядно построить поверхность тела человека.

1.7 Выводы по разделу

В данной главе рассмотрены существующие программные продукты для бесконтактного определения антропометрических данных человека и построения 3D модели человека по изображениям или видео.

Рассмотрены 4 библиотеки для работы с графикой. Каждая обладает своими преимуществами и недостатками. Для данной программы выбрана библиотека OpenGL и перечислены её достоинства.

Также в работе используется триангуляция поверхности. С помощью них точки, отмеченные пользователем, соединяются треугольниками и получается объёмная модель тела человека.

2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ ЗАДАЧИ ОПРЕДЕЛЕНИЯ АНТРОПОМЕТРИЧЕСКИХ ПАРАМЕТРОВ ЧЕЛОВЕКА ПО ИЗОБРАЖЕНИЮ

2.1 Модель определения антропометрических параметров

Для построения 3D модели субъекта пользователь расставляет 32 точки на 3 фотографиях.

$$A(x, y, z)$$

Все точки заносятся в массив. Пользователь расставляет точки на правой половине тела человека, а левая часть строится симметрично относительно оси OZ (см. рисунок 2.1).

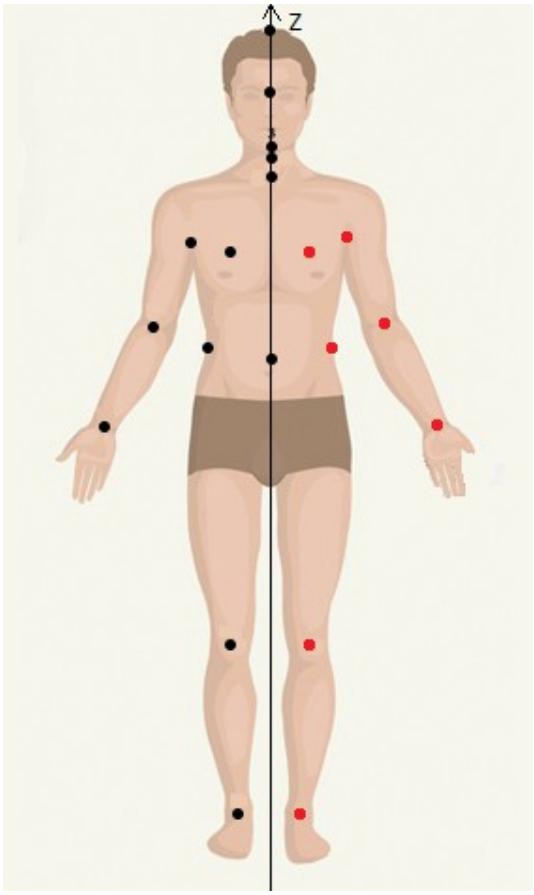


Рисунок 2.1 – Симметрия точек относительно оси OZ

Для расчёта прямых линейных размерных признаков, а именно кратчайшего расстояния между двумя точками A (x_1, y_2, z_3) и B (x_2, y_2, z_2) модели человека используется формула:

$$l=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2} \quad ,$$

где x_1, y_1, z_1 координаты первой точки, а x_2, y_2, z_2 координаты второй точки.

Для расчёта расстояния между двумя точками вдоль оси OX используется формула:

$$l_x=|x_1-x_2| \quad ,$$

где x1 – первая координата первой точки, x2 – первая координата второй точки.

Для расчёта расстояния между двумя точками вдоль оси OY используется формула:

$$l_y = |y_1 - y_2| \quad ,$$

где y1 – вторая координата первой точки, y2 – вторая координата второй точки.

Для расчёта расстояния между двумя точками вдоль оси OZ используется формула:

$$l_z = |z_1 - z_2| \quad ,$$

где z1 – третья координата первой точки, z2 – третья координата второй точки

Для подсчёта обхвата талии, бёдер и груди используется формула длины эллипса.

$$L = 2 * \pi * \sqrt{\frac{(a^2 + b^2)}{2}} \quad ,$$

где a и b полуоси эллипса (см. рисунок 2.2).

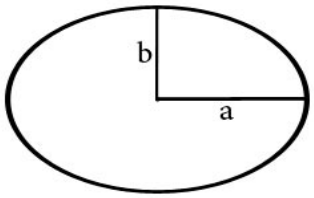


Рисунок 2.2 – Эллипс с полуосями

Для того чтобы размеры модели совместить с реальными размерами человека, при фоторгафии небходимо держать в руке купюру в 10, 50, 100 или 500 рублей. Купюра имеет следующий размер:

l=150 мм, h=65 мм,

где l длина купюры, h высота купюры.

На фотографии пользователь расставляет точки. На купюре пользователь ставит 3 точки в трёх углах (см. рисунок 2.3). По точкам определяется длина и высока купюры на фотографии. Для того чтобы соотнести размеры используются формулы:

$$k_l = \frac{l}{l_{фот}} \quad ,$$

где l реальная длина купюры, $l_{фот}$ длина купюры на фотографии, k_l коэффициент соотношения реальных размеров купюры размерам на фотографии.

$$k_h = \frac{h}{h_{фот}} \quad ,$$

где l реальная длина купюры, $l_{фот}$ длина купюры на фотографии, k_h коэффициент соотношения реальных размеров купюры размерам на фотографии.

Соотношение измеряется по всем осям координат.

Рисунок 2.3 – Изображение размера купюры

2.2 Модель формирования изображения

В работе используется перспективная проекция (см. рисунок 2.4). Охват перспективы - 45 градусов. Таким образом, пространство сцены определено усеченной пирамидой.

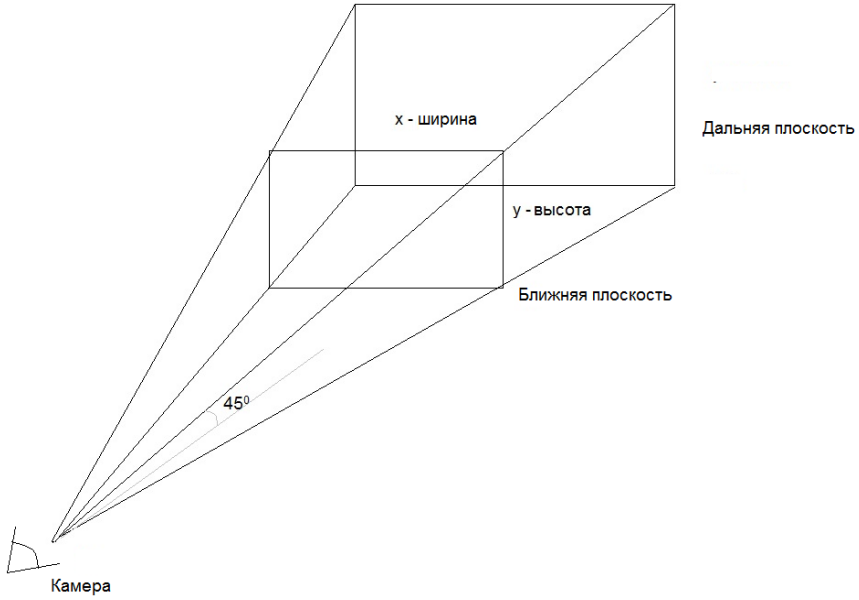


Рисунок 2.4 – Перспективная проекция

Пусть фигура находится внутри некоторой сферы. Тогда при изменении изображения на экране будем перемещать камеру по параллелям и меридианам сферы, а фигура будет в статичном положении (см. рисунок 2.5).

Обозначим: x0, y0, z0 – центр сфер, вокруг которой движется камера.

Для перемещения по сфере используются формулы (x1, y1, z1 – новые координаты):

$$\begin{cases} x_1 = x + R_1 \sin\theta \cos\varphi \\ y_1 = y + R_1 \sin\theta \sin\varphi \\ z_1 = z + R_1 \cos\theta \end{cases}$$

Для приближения и отдаления:

$$\begin{cases} x_1 = \frac{x}{n} \\ y_1 = \frac{y}{n} \\ z_1 = \frac{z}{n} \end{cases}$$

Для того чтобы фигура вращалась правильно необходимо определить центр, вокруг которого будет вращаться камера.

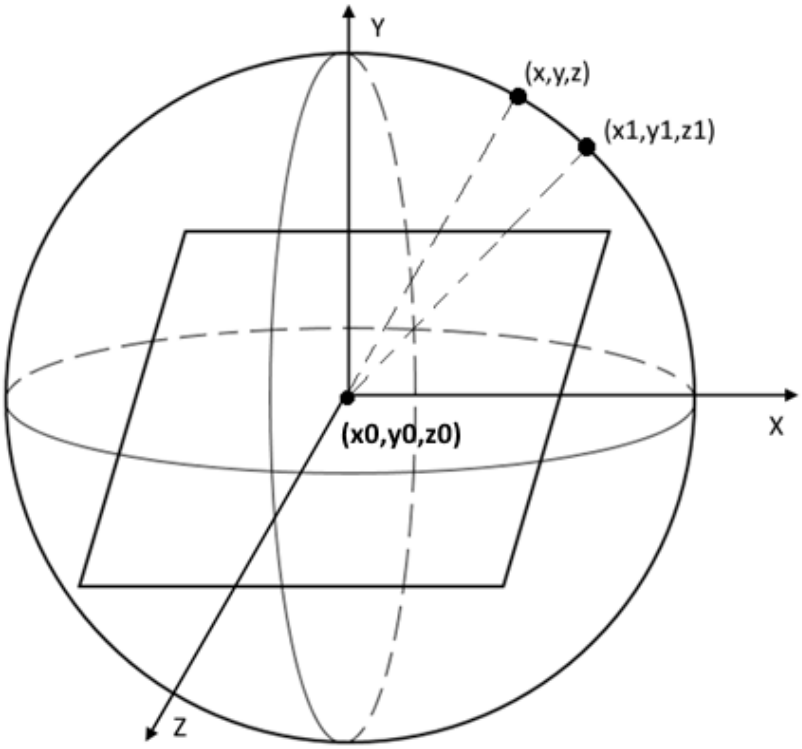


Рисунок 2.5 – Сфера вращения камеры

Центр вдоль оси OX находится по следующей формуле:

$$x_{центр} = \frac{x_{лев} + x_{прав}}{2},$$

где $x_{лев}$ первая координата самой левой точки по оси OX $x_{прав}$ первая координата самой правой точки по оси OX.

Центр вдоль оси OZ находится по следующей формуле:

$$z_{центр} = \frac{z_{верх} + z_{низ}}{2},$$

где $x_{верх}$ третья координата самой верхней точки по оси OZ $x_{низ}$ третья координата самой нижней точки по оси OZ.

Так как точки симметрично отражаются вдоль оси OY, то центр вычислим по формуле:

$$y_{центр} = \frac{y_1 + y_2 + y_3 + y_4 + y_5 + y_6}{6},$$

где y_1, y_2, y_3, y_4, y_5, y_6 вторая координата точек расположенных посередине тела человека (см. рисунок 2.6).

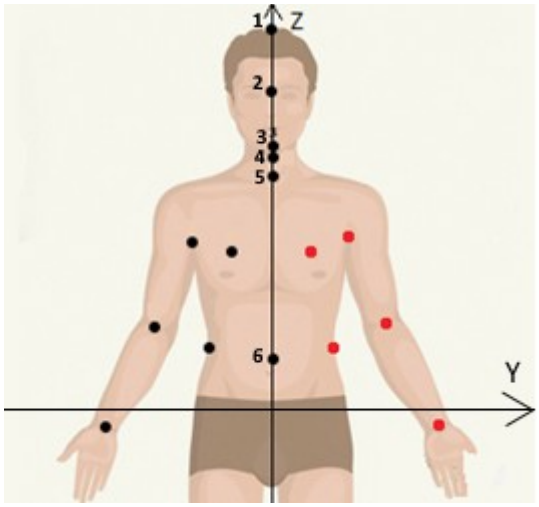


Рисунок 2.6 – Симметрия тела

Таким образом центр относительно которого будет вращаться камера находится в точке А.

$$A\ (x_{центр}, y_{центр}, z_{центр}),$$

где $x_{центр}$ первая координата центра вращения камеры, $y_{центр}$ вторая координата центра вращения камеры, $z_{центр}$ первая координата центра вращения камеры.

2.3 Выводы по разделу

В результате количество расставляемых точек удалось уменьшить в два раза, за счёт симметрии тела. Также по формулам используемым в данной главе вычисляются основный антропометрические характеристики теля человека (рост, длина рук, длина плеч, обхват груди и т.д.).

3 РАЗРАБОТКА ПРИЛОЖЕНИЯ

3.1 Разработка алгоритмов

Для решения задачи определения антропометрических параметров человека по изображению необходимо выделить отдельные модули, которые должны реализовывать определенную часть функций и выполнять следующие действия:

- загружать изображения клиента в систему;
- организовывать хранение данных;
- производить автоматическое построение трёхмерной модели антропометрических схем;
- организовывать представление данных в виде антропометрических схем;
- осуществлять управление приложением.

Приложение разрабатывается для операционной системы Windows и должно использовать принцип обработки сообщений. Основной алгоритм работы приложения изображен на рисунке 3.1.

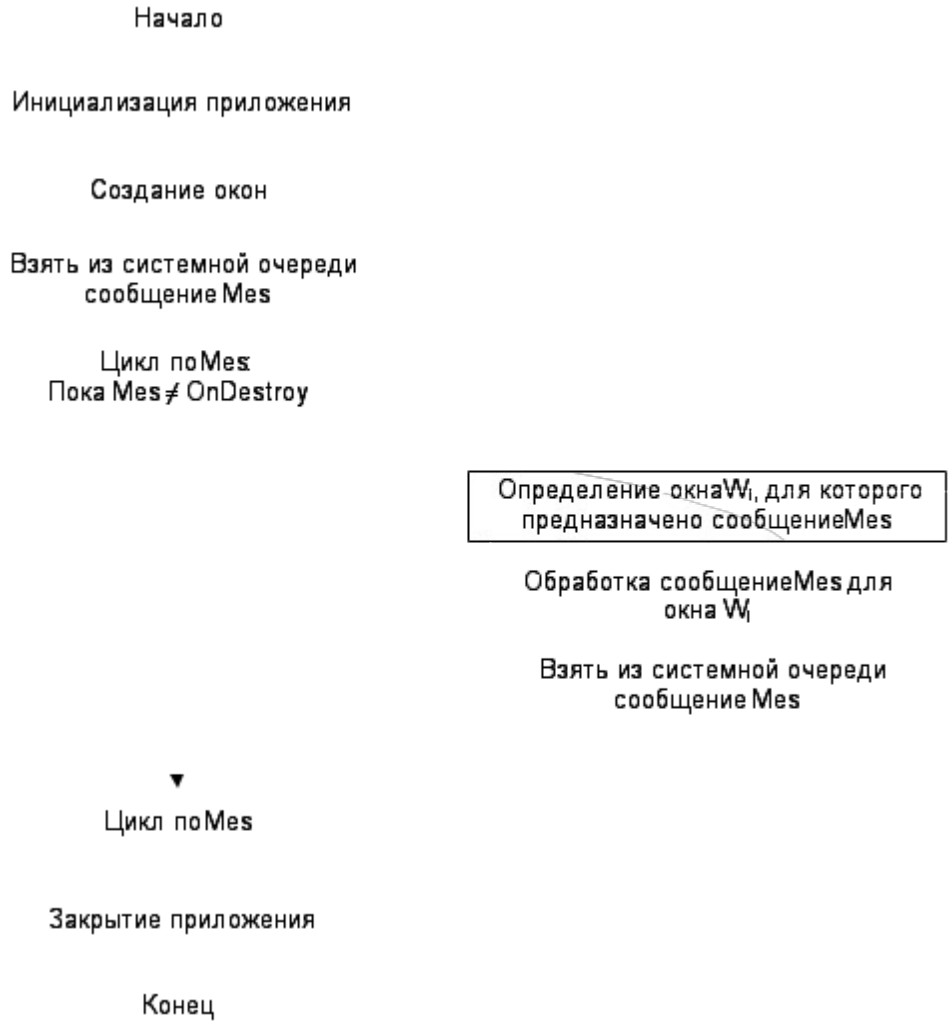


Рисунок 3.1 – Основной алгоритм работы приложения

3.1.1 Алгоритм обработки событий главного окна

После двойного щелчка по ярлыку приложения происходит запуск приложения. Затем система ожидает события от пользователя. Если выбран пункт меню «Загрузить манекен», то происходит вызов окна «3D модель». В случае если нажата кнопка «Создать 3D Модель» происходят последовательные действия:

- вызов окна «Добавление фотографий»;
- вызов окна «Расстановка точек» (вид спереди);
- вызов окна «Расстановка точек» (вид сбоку);
- вызов окна «Расстановка точек» (вид сзади);
- вызов окна «3D модель».

Если выбран пункт меню «О программе» происходит вызов окна «О программе». В случае если выбран пункт меню «Выход» происходит закрытие приложения. После любого действия происходит передача сообщения стандартному обработчику Windows. Алгоритм обработки событий главного окна приведён на рисунке 3.2.

3.1.2 Алгоритм обработки событий окна «Загрузка изображений»

После нажатия на кнопку «Создать 3D модель» открывается окно загрузки приложения. После этого система ожидает события от пользователя. Если нажата одна из кнопок «Обзор…», то происходит загрузка изображения. Если нажата кнопка «Расстановка точек» и загружены все фотографии, то последовательно происходит:

- вызов окна «Расстановка точек» (вид спереди);
- вызов окна «Расстановка точек» (вид сбоку);
- вызов окна «Расстановка точек» (вид сзади);
- вызов окна «3D модель».

В случае нажата кнопка «Расстановка точек», но загружены не все фотографии, то происходит вызов сообщения об ошибке. После любого действия происходит передача сообщения стандартному обработчику Windows. Алгоритм обработки событий окна «Загрузка изображений» приведён на рисунке 3.3.

3.1.3 Алгоритм обработки событий окна «Расстановка точек»

После того как пользователь добавил фотографии и нажал на кнопку «Расстановка точек» перед ним открывается окно расстановки точек (вид спереди). После этого система ожидает события от пользователя. Происходит инициализация массива точек mes и количества точек на фигуре length. Если выбран пункт меню «Инструкция», то происходит вызов окна «Инструкция». В случае если выбран пункт меню «О программе» происходит вызов окна «О программе». Если выбран пункт меню «Выход» происходит закрытие приложения. В случае если количество точек < length, нажата левая кнопка мыши

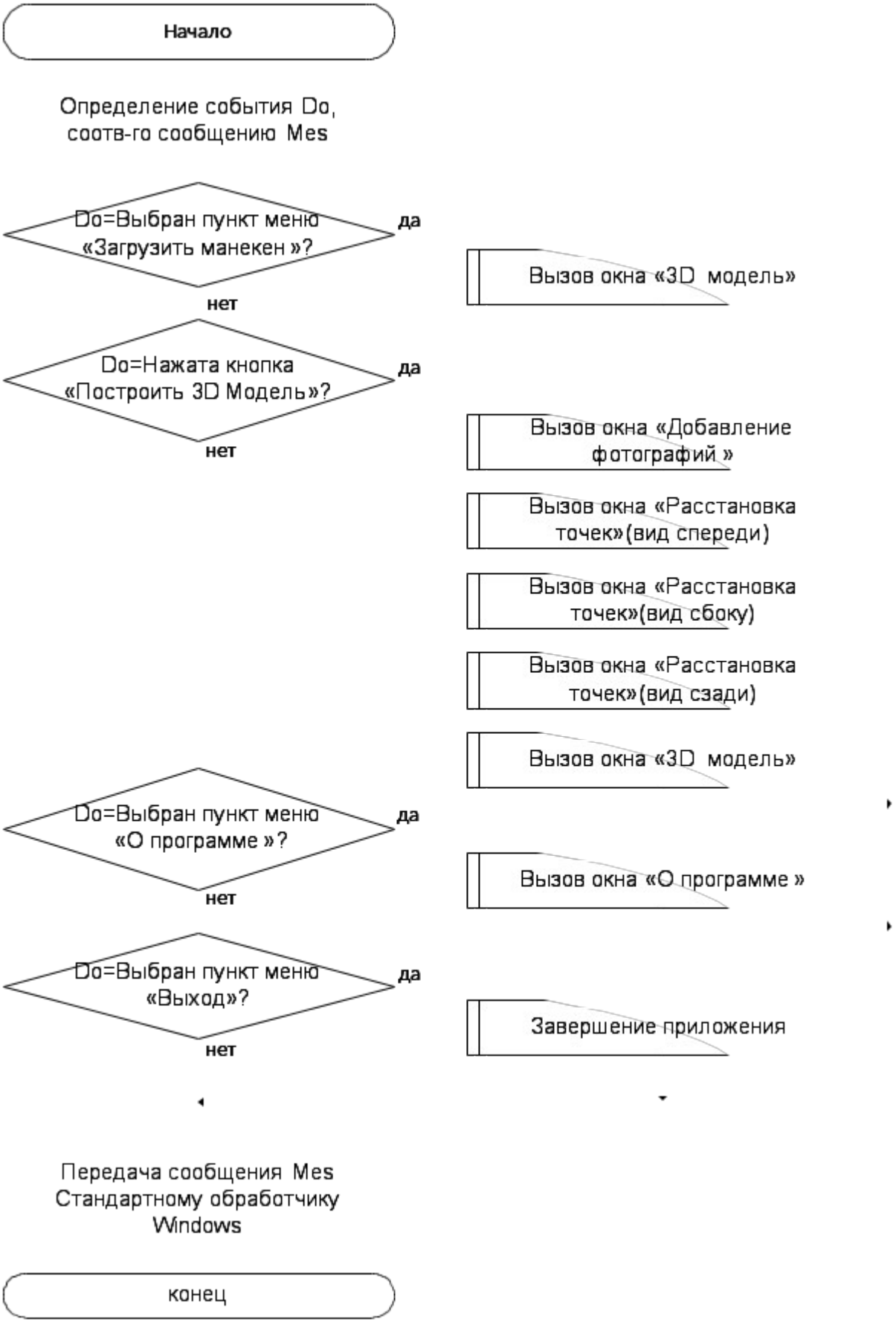


Рисунок 3.2 – Алгоритм обработки событий главного окна

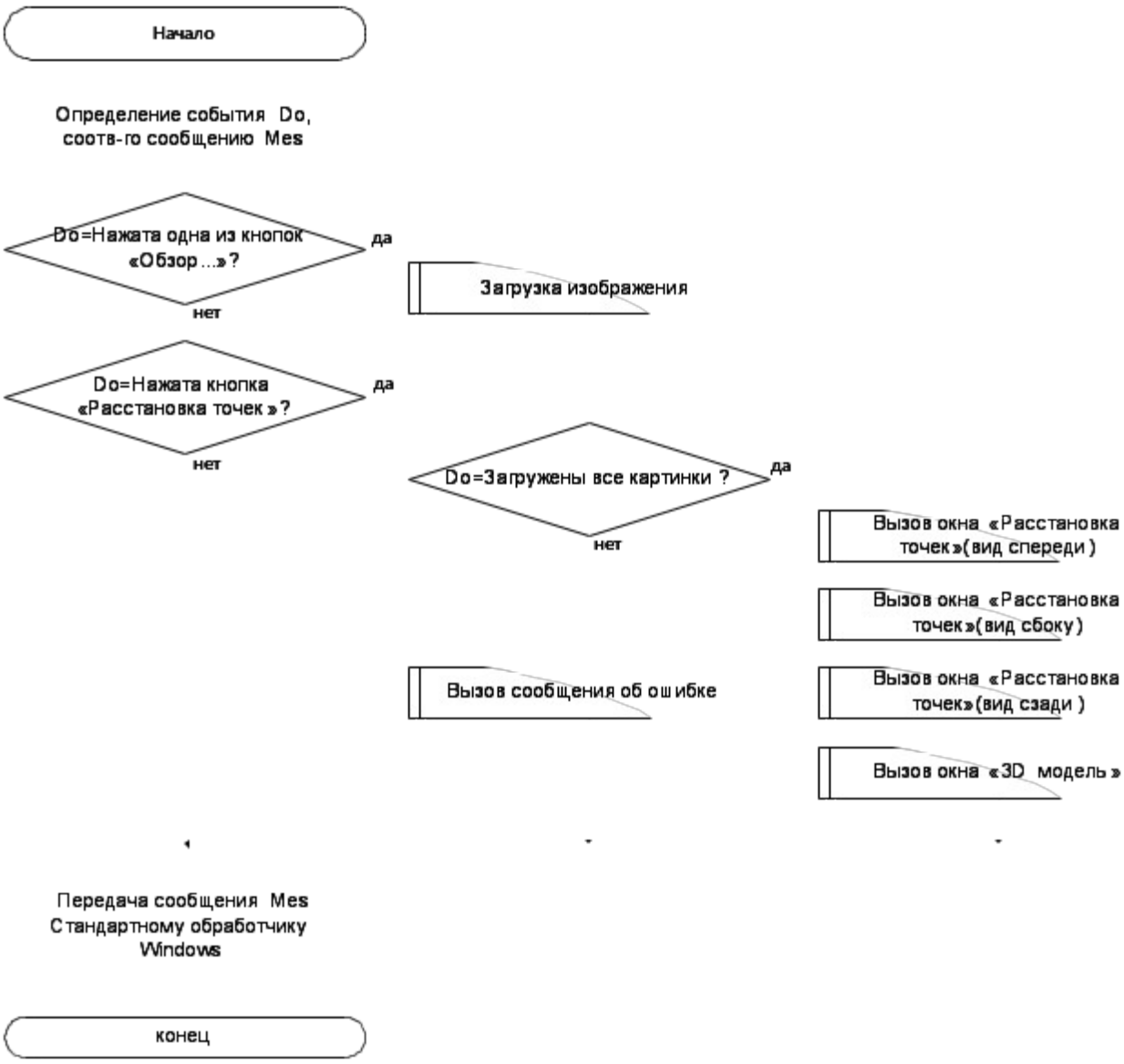


Рисунок 3.3 – Алгоритм обработки событий окна «Загрузка изображений»

и мышь перемещается, то происходит изменение координат точки в mes .Если количество точек < length, нажата левая кнопка мыши, но мышь не перемещается, то происходит добавление точки в mes. В случае если нажата кнопка «Далее», то последовательно происходит:

- вызов окна «Расстановка точек» (вид сбоку);
- вызов окна «Расстановка точек» (вид сзади);
- вызов окна «3D модель».

После этого система ожидает события от пользователя. После любого действия происходит передача сообщения стандартному обработчику Windows. Алгоритм обработки событий окна «Расстановка точек (вид спереди)» приведён на рисунке 3.4. Вид сбоку и вид сзади имею такой же алгоритм, как и вид спереди.

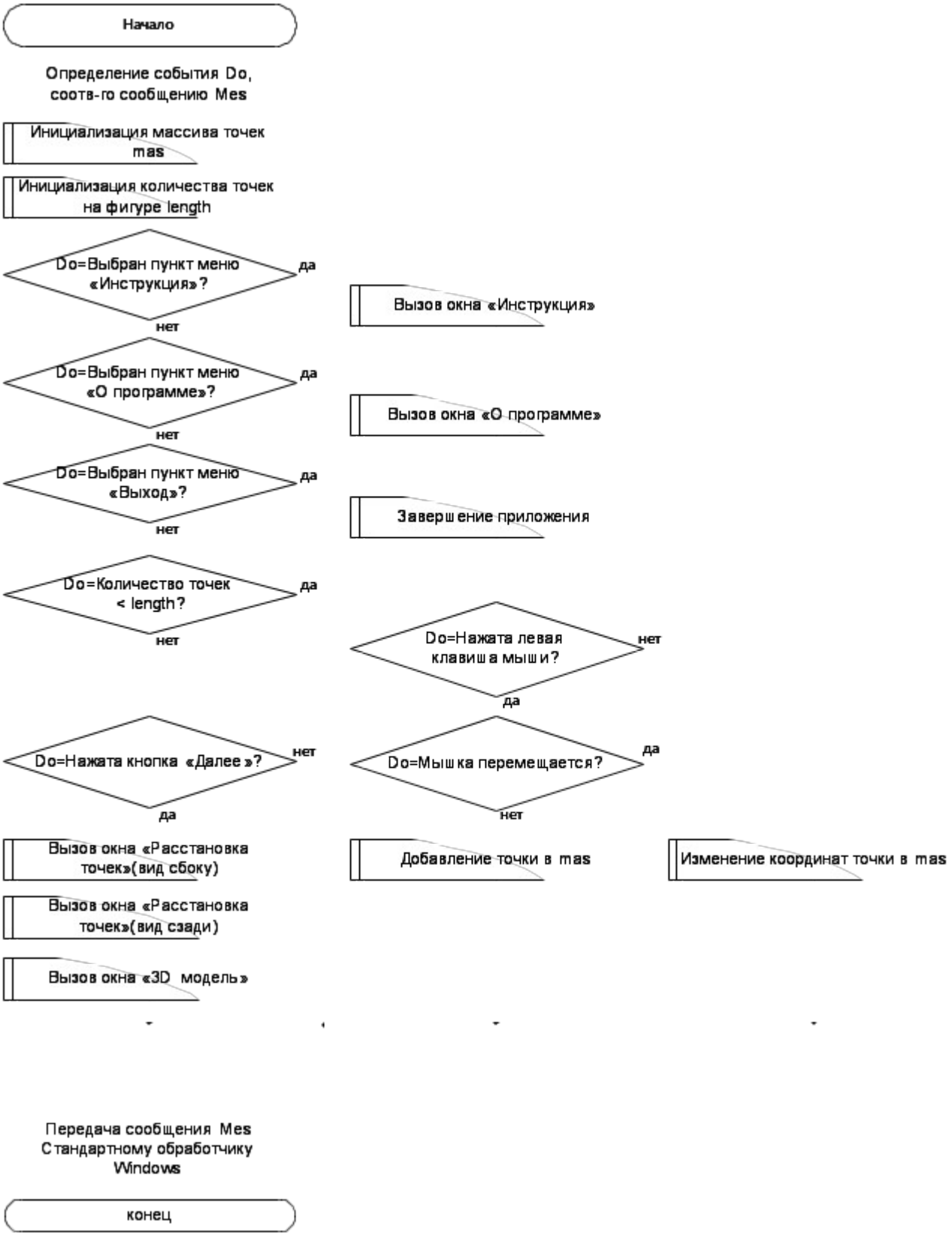


Рисунок 3.4 – Алгоритм обработки событий окна «Расстановка точек (вид спереди)»

3.1.4 Алгоритм обработки событий окна «3D Модель»

После расстановки точек на виде сзади и нажатии кнопки «Далее» открывается окно «3D Модель». После этого система ожидает события от пользователя. В случае если выбран пункт меню «Сохранить как…», то происходит сохранение. Если выбран пункт меню «Загрузить из…», то последовательно происходит:

- загрузка данных из файла;
- перерисовка окна.

В случае если выбран пункт меню «Справка», то происходит вызов окна «Справка». Если нажата одна из стрелок на клавиатуре, то последовательно происходит:

- изменение положения камеры;
- перерисовка окна.

В случае если нажата правая кнопка мыши и мышь перемещается, то последовательно происходит:

- изменение положения камеры;
- перерисовка окна.

Если выбран пункт меню «Выход» происходит закрытие приложения. После любого действия происходит передача сообщения стандартному обработчику Windows. Алгоритм обработки событий окна «3D Модель» приведён на рисунке 3.5.

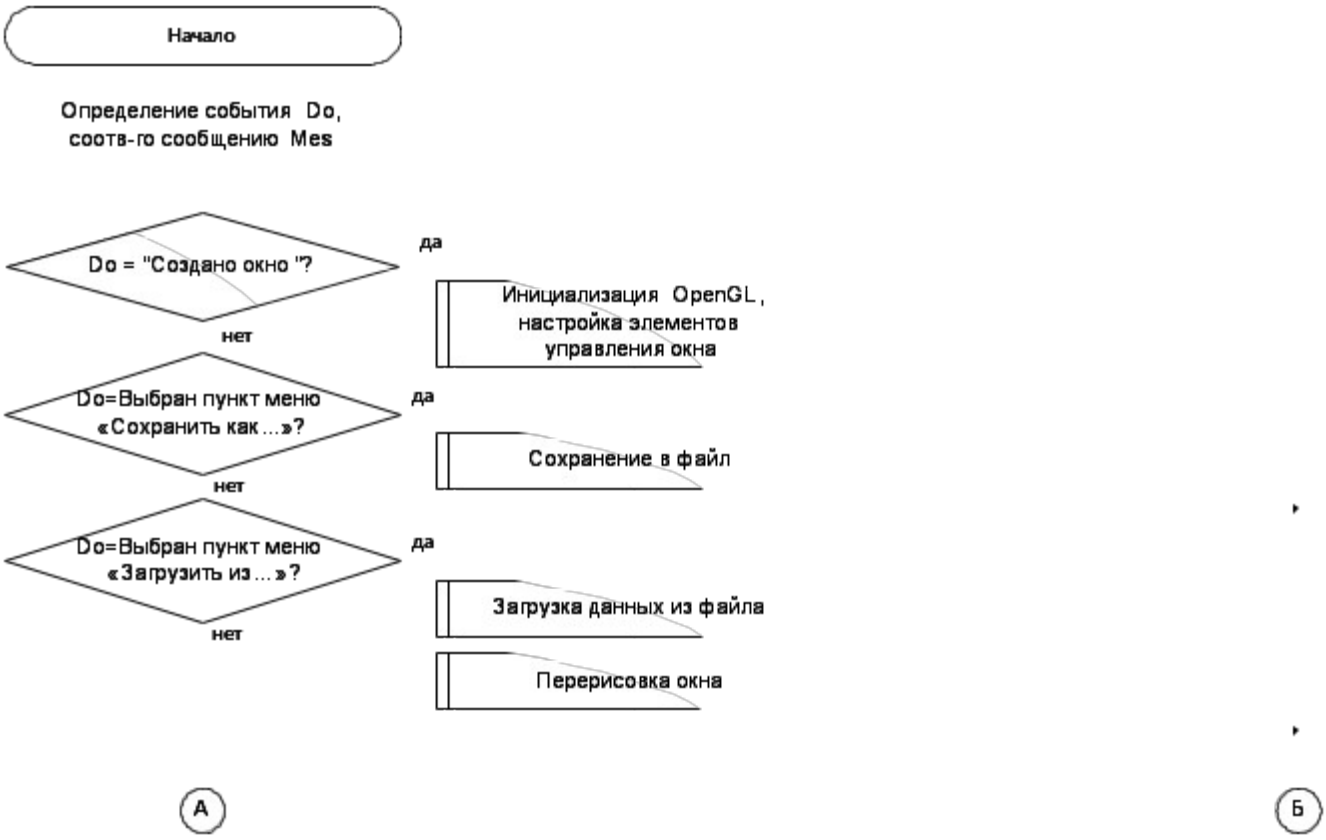
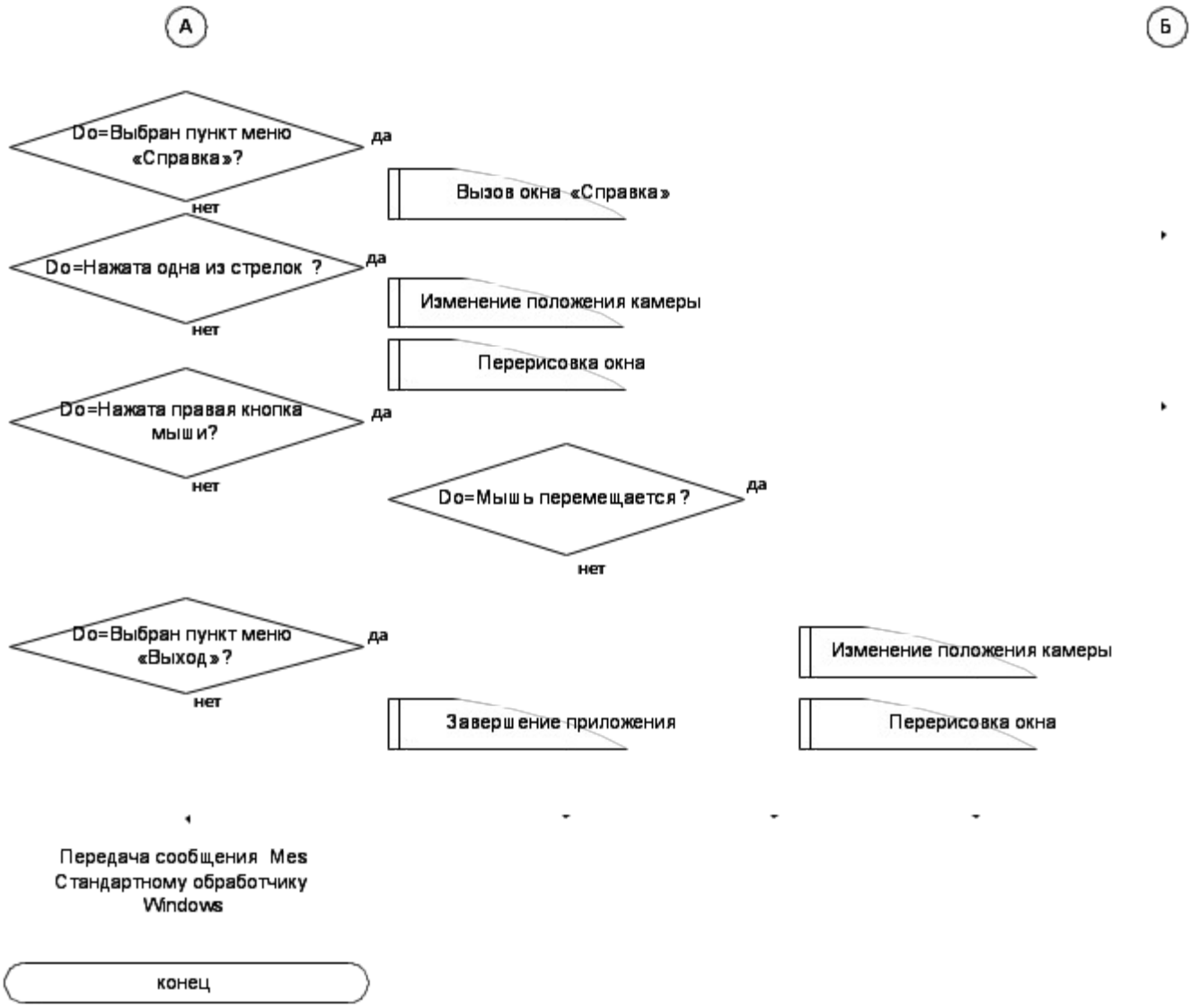


Рисунок 3.5 – Алгоритм обработки событий окна «3D Модель»



Продолжение рисунка 3.5

3.2 Разработка интерфейса

При запуске приложения открывается форма с описанием программы (см. рисунок 3.6). Здесь можно загрузить уже имеющуюся модель, прочитать описание программы, выйти из приложения или начать создавать 3D Модель нажав на кнопку «Создание 3D модели».

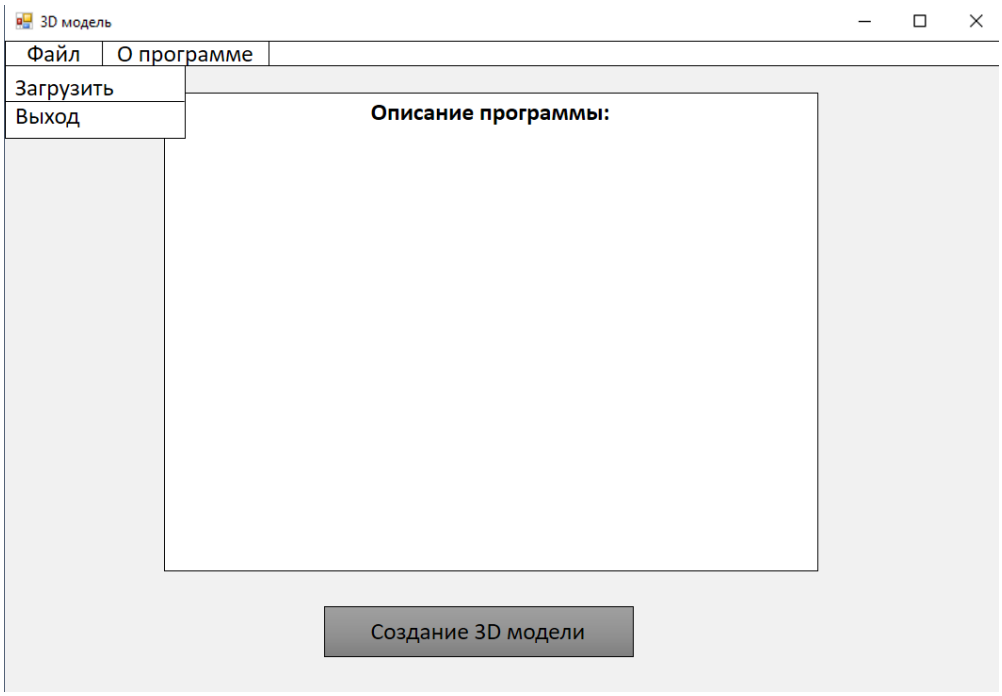


Рисунок 3.6 – Форма описания программы

При нажатии на кнопку «Создание 3D модели» форма с описанием закрывается, а открывается форма, где надо загрузить три фотографии (см. рисунок 3.7) . При нажатии на одну из кнопок «Пример» появляется новая форма где можно посмотреть образец фотографии, которую надо загрузить. При нажатии на кнопку «Обзор…» открывается диалоговое окно загрузки фотографии. После загрузки всех фотографий пользователь нажимает на кнопку «Построить 3D модель».

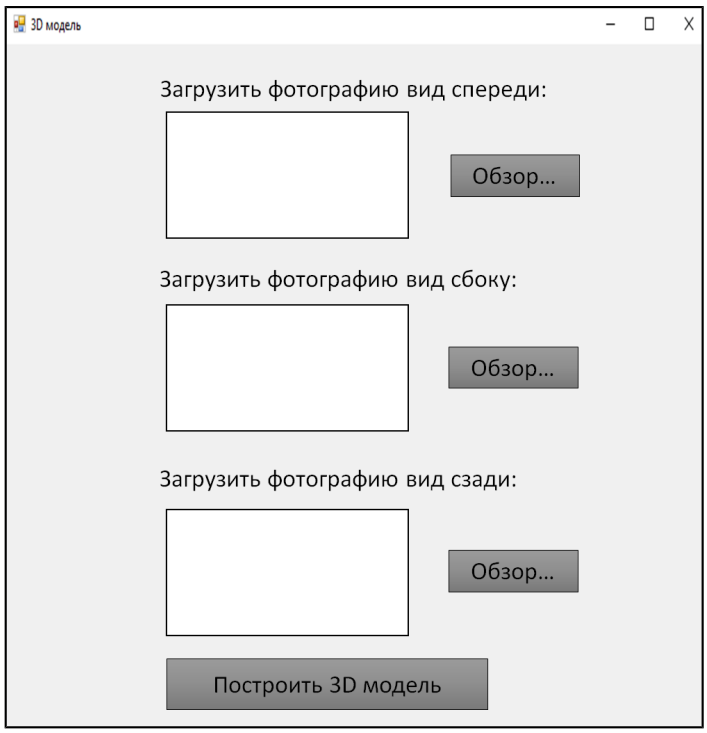


Рисунок 3.7 – Форма загрузки фотографий

После нажатия на кнопку «Построить 3D модель» форма с загрузкой фотографии закрывается и открывается форма с расстановкой точек, где пользователь по образцу расставляет точки (см. рисунки 3.8 - 3.10). Сначала точки расставляются на виде спереди, потом сбоку и затем на виде сзади. Нажимая кнопку «Назад» или «Вперёд» можно перемещаться с одного вида на другой. В меню «Справка» находится инструкция по расстановке точек.

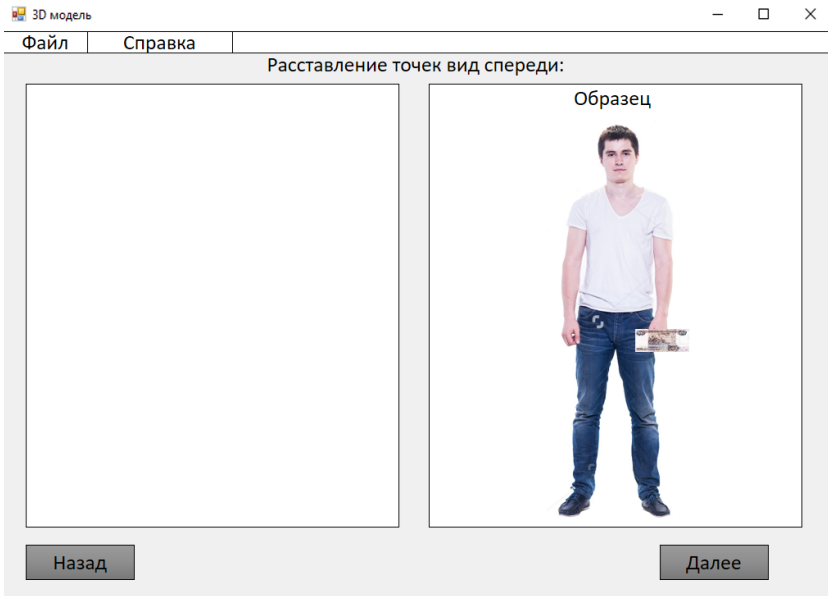


Рисунок 3.8 – Форма расстановки точек (вид спереди)

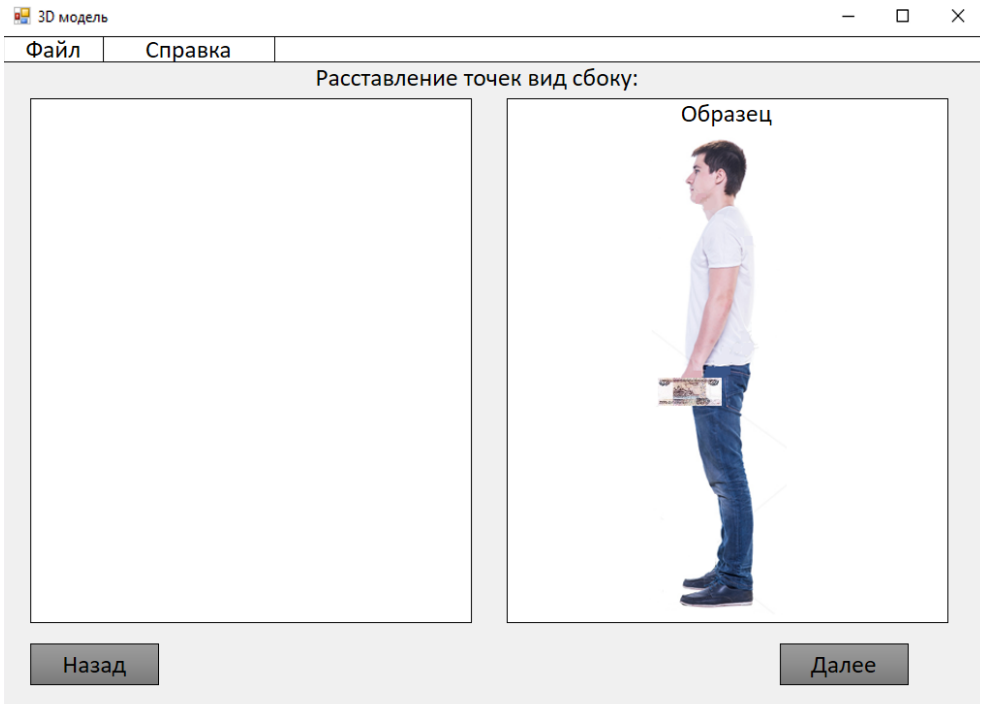


Рисунок 3.9 – Форма расстановки точек (вид сбоку)

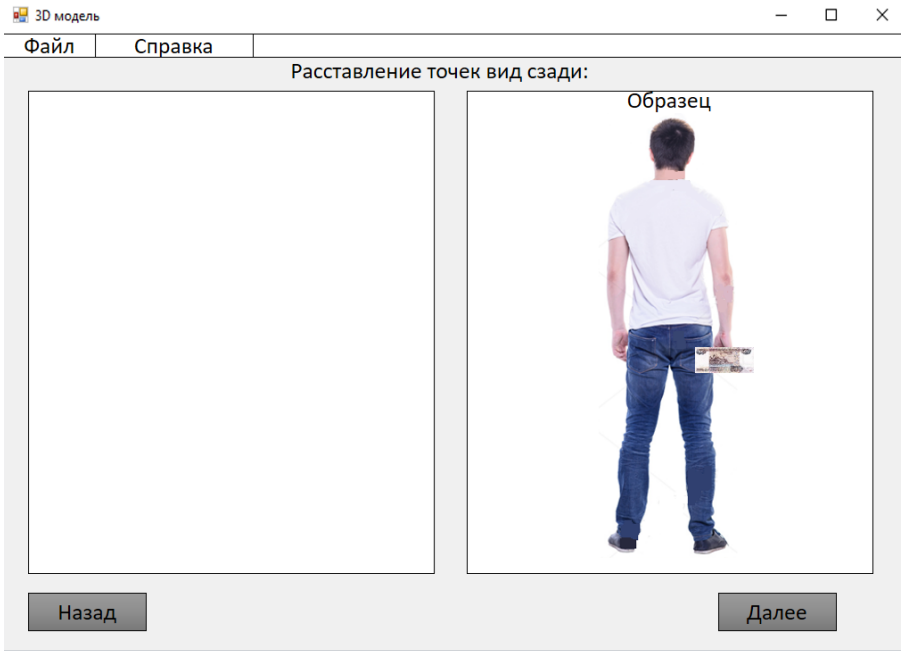


Рисунок 3.10 – Форма расстановки точек (вид сзади)

При нажатии на кнопку «Далее» на форме с видом сзади открывается форма с 3D моделью тела человека (см. рисунок 3.11). В правом верхнем углу отображаются его основные антропометрические показатели. В пункте меню «Файл» можно сохранить, загрузить фотографию или выйти из приложения. Фигуру можно поворачивать.

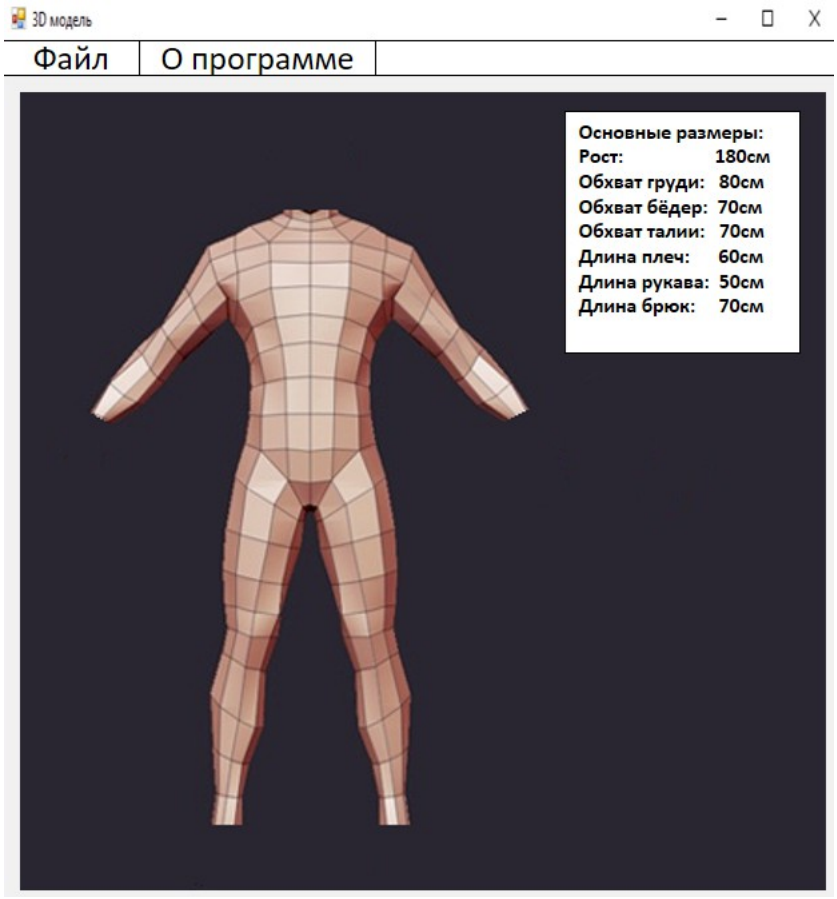


Рисунок 3.11 – Форма «3D Модель»

3.3 Выводы по разделу

В результате разработаны алгоритмы работы различных окон приложения, а также работы всего приложения целиком.

Также разработан интерфейс пользователя. Приложение состоит из множества различных окон. В одном окне находится описание приложения, в другом загрузка изображений, в третьем, четвёртом и пятом расстановка точек, а в шестом трёхмерная модель. Часть форм служат инструкциями и примерами к программе.

4 ПРОВЕРКА РАБОТЫ ПРИЛОЖЕНИЯ

4.1 Проверка работы программы на экспериментальных данных

В папке с приложением лежат файл с расширением .exe и папка с картинками. Запуск приложения производится двойным нажатием левой кнопки мыши по файлу с расширением .exe (см. рисунок 4.1).

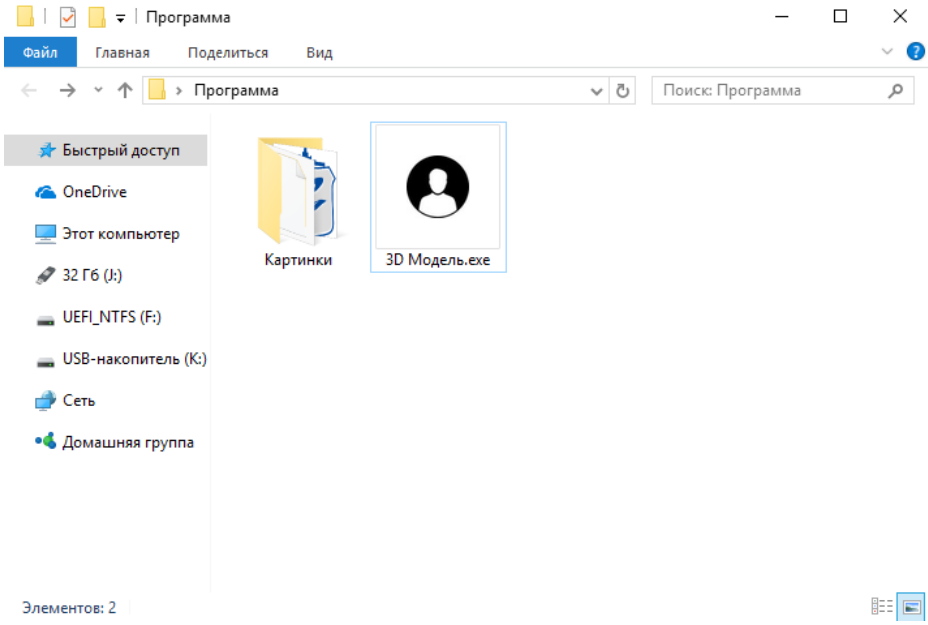


Рисунок 4.1 – Папка с приложением

При запуске приложения открывается окно «Описание программы» (см. рисунок 4.2). Здесь кратко описано, что должен сделать пользователь и что в итоге получается. В пункте меню «Файл» можно загрузить уже готовую 3D модель или выйти из приложения. В пункте меню «О программе» находится информация о создателе приложения.

После нажатия на кнопку «Создать 3D модель» форма с описание приложения закрывается, а форма с загрузкой фотографий открывается (см. рисунок 4.3). При нажатии на одну кнопок «Пример» отрывается форма с образцом изображения (см. рисунок 4.5). Она может оставаться открытой до тех пор, пока форму не закроет пользователь или пока не нажмёт на кнопку «Расстановка точек».

При нажатии на одну из кнопок «Обзор…» открывается диалоговое окно «Открыть» (см. рисунок 4.4), где пользователь выбирает изображение для загрузки. Загруженное изображение появляется в белой области.

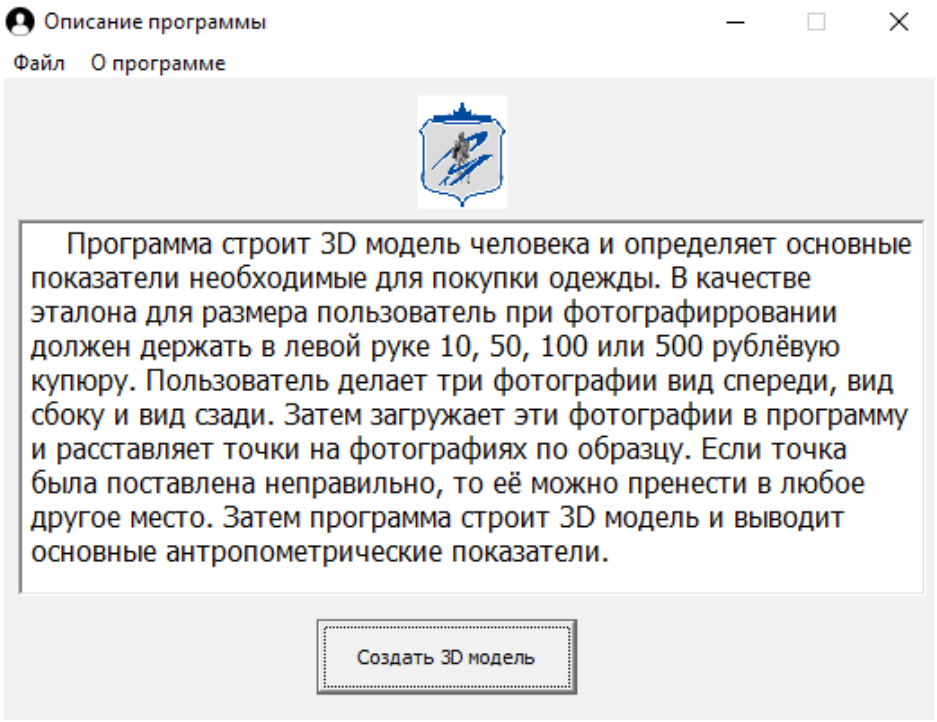


Рисунок 4.2 – Первое окно программы

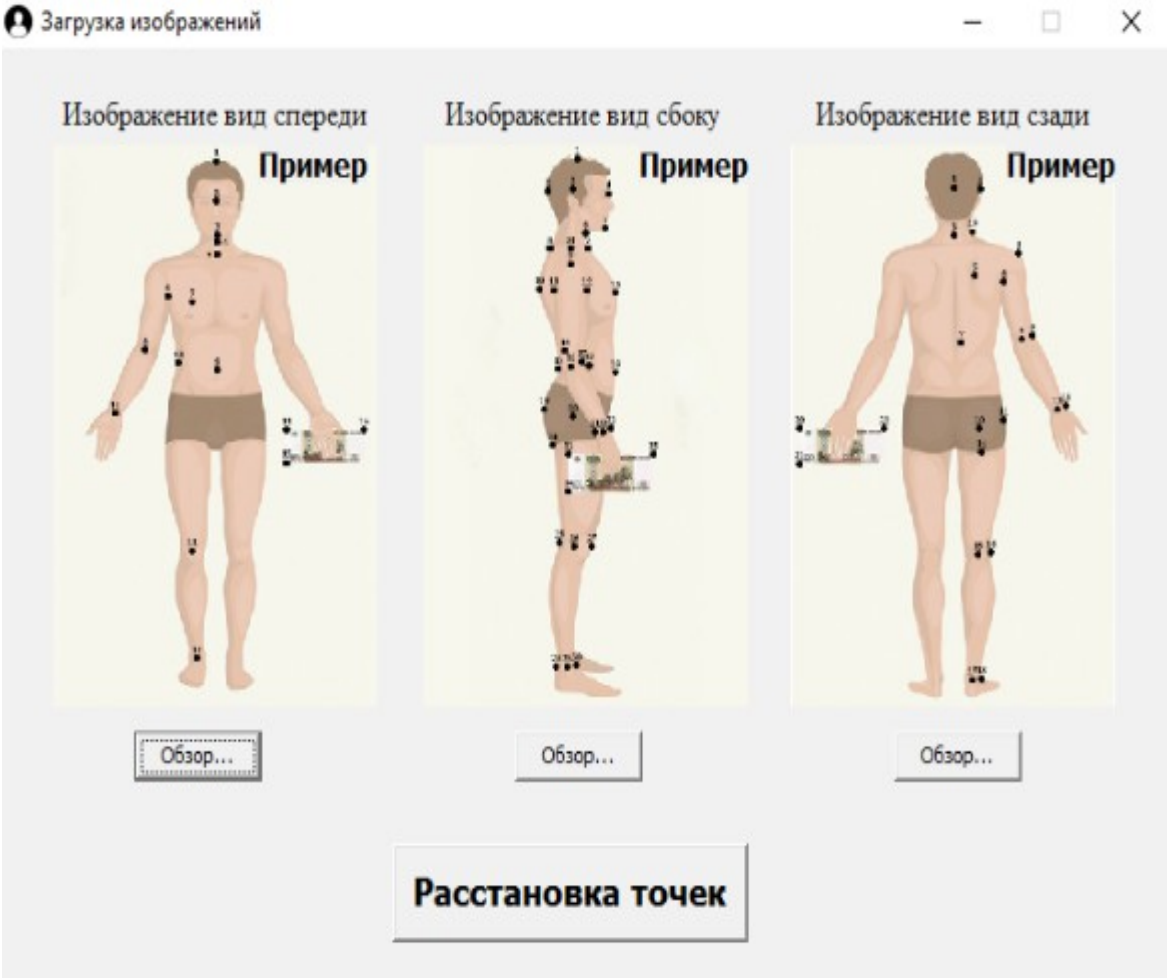


Рисунок 4.3 – Загрузка изображений

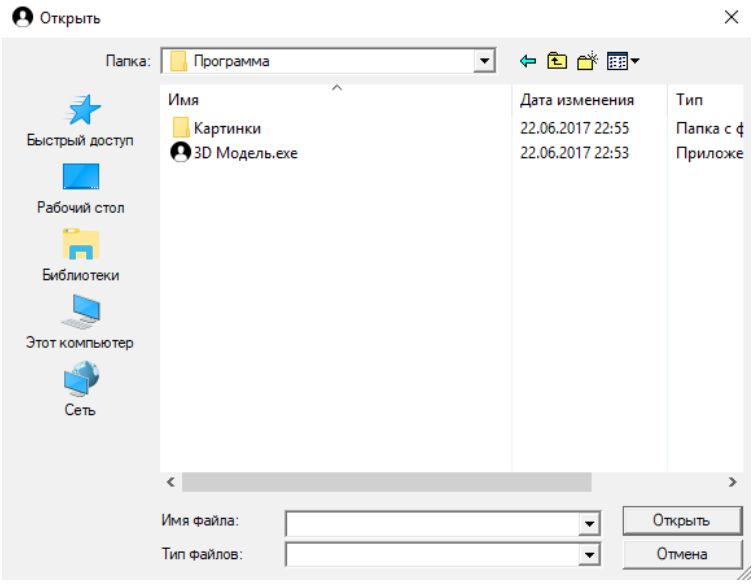


Рисунок 4.4 – Диалоговое окно «Загрузка изображения»

При нажатии на кнопку «Расстановка точек» закрывается форма «Загрузка изображений» и открывается форма «Расстановка точек» (вид спереди) на ней пользователь расставляет точки на левой картинке по образцу справа (см. рисунок 4.5). Если пользователь наверно поставил точку, он может переместить её в новое место. После расстановки всех точек пользователь нажимает кнопку «Далее» и аналогично расставляет точки на виде сбоку и виде сзади. При нажатии на пункт меню «Файл» можно выйти из приложения. При нажатии на пункт меню «Справка» можно посмотреть инструкцию к расстановке точек или информацию о создателе приложения (см. рисунок 4.6). При нажатии на пункт «Инструкция» открывается форма с инструкцией по расстановке точек (см. рисунок 4.7). Перемещаясь между формами вид спереди, вид сбоку и вид сзади инструкция меняется, а при завершении расстановки точек она закрывается.

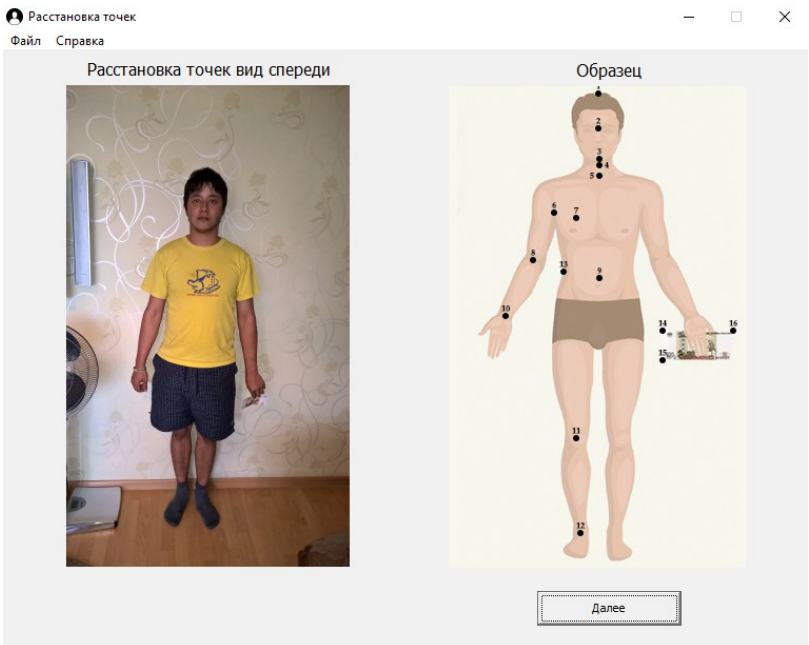


Рисунок 4.5 – Форма расстановки точек

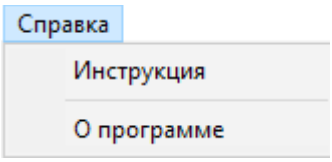


Рисунок 4.6 – Пункт меню «Справка»

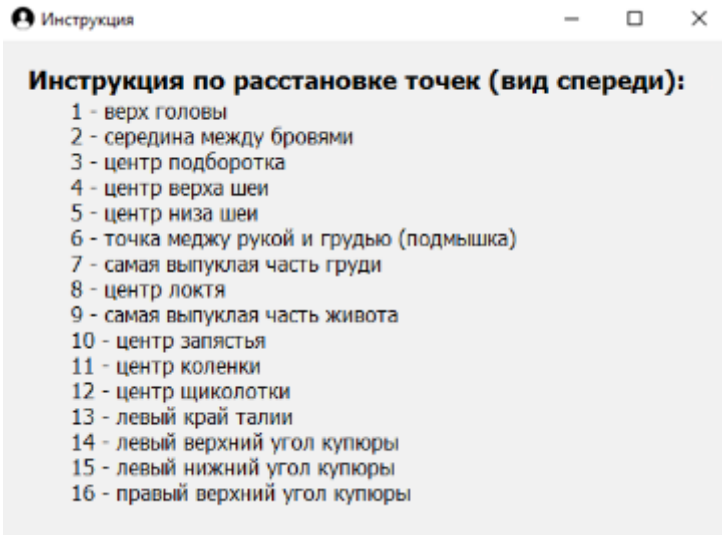


Рисунок 4.7 – Форма с инструкцией к расстановке точек

После нажатия на форме «Расстановка точек» (вид сзади) кнопки «Далее» закрывается форма расстановки точек и открывается форма с трёхмерной моделью (см. рисунок 4.8). На данной форме находится 3D модель тела человека, которую можно вращать. В правом верхнем углу показаны основные антропометрические показатели. В пункте меню «Файл» можно загрузить другую готовую 3D модель или сохранить текущую, а также выйти из приложения (см. рисунок 4.9). При нажатии на пункт меню «О программе» можно посмотреть информацию о создателе.

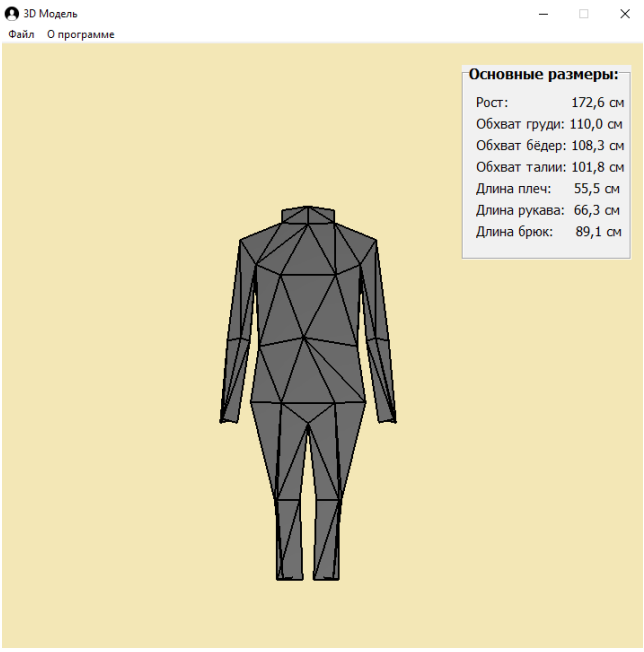


Рисунок 4.8 – Форма с 3D Моделью

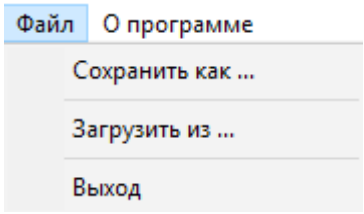


Рисунок 4.9 – Пункт меню «Файл»

4.2 Оценка погрешности результата

Программа посчитала основные антропометрические показатели. Для проверки правильности работы программы необходимо сравнить расчёты сделанные программой и реальные размеры человека. Найдём относительную и абсолютную погрешность

4.2.1 Абсолютная погрешность

Для вычисления абсолютной погрешности используется следующая формула:

$$\Delta_{аб.} = \left| x_{прогр.} - x_{действ.} \right| \quad ,$$

где $x_{прогр.}$ значение вычисленное в программе, $x_{действ}$ реальное значение.

4.2.2 Относительная погрешность

Для вычисления относительной погрешности используется следующая формула:

$$\delta = \frac{\Delta_{аб.}}{x_{действ.}} * 100\% \quad ,$$

где $\Delta_{аб.}$ абсолютная погрешность измерений, $x_{действ}$ реальное значение.

4.2.3 Оценка погрешности

Таблица 4.1

Оценка погрешности вычислений

Основные показатели	Значение в программе, см	Реальное значение, см	Абсолютная погрешность, см	Относительная погрешность, %
Рост	172,6	172,0	0,6	0,35
Длина брюк	89,1	88,2	0,9	1,02
Длина рукава	66,3	65,5	0,8	1,22
Длина плеч	55,5	54,7	0,8	1,46
Обхват груди	110,0	109,3	0,7	0,64
Обхват талии	101,8	100,6	1,2	1,19
Обхват бёдер	108,3	106,7	1,6	1,50

Относительная погрешность находится в пределах 1,5%, что говорит о хорошей точности измерений.

4.3 Выводы по разделу

В результате проверена работоспособность программы, построена 3D модель человека и выведены его основные антропометрические показатели. Оценка погрешности показала, что программа вычисляет основные показатели достаточно точно.

ЗАКЛЮЧЕНИЕ

Данная работа посвящена разработке программного модуля определения антропометрических параметров человека по изображению. В результате работы получена программа, которая строит 3D модель тела человека и определяет основные показатели необходимые для покупки одежды:

- рост;
- длина брюк;
- длина рукава;
- длина плеч;
- обхват груди;
- обхват талии;
- обхват бёдер.

Зная данные показатели можно заказывать одежду в интернете.

В ходе работы над проектом решены следующие задачи.

1. Проведён обзор существующих методов и способов, которые решают задачи определения антропометрических параметров человека по изображению.
2. Выполнен обзор графических библиотек, позволяющих осуществлять анализ и синтез изображений.
3. Проведён обзор методов аппроксимации поверхности.
4. Разработана математическая модель задачи определения антропометрических параметров человека по изображению.
5. Разработаны и реализованы алгоритмы работы приложения.
6. Реализован графический интерфейс пользователя.
7. Выполнена проверка работы приложения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Антропометрические данные. – URL: http://gardenweb.ru/antropometricheskie-dannye (дата обращения: 15.02.2017)
2. Болдырева, Л.М. Анализ бесконтактных способов получения информации о поверхности тела человека / Л.М. Болдырева, И.В. Лашина // Научно-технический вестник Поволжья. – 2014. – №1. – С. 73-77.
3. Дистанционное определение размеров тела – Дата обновления: 05.12.2014. URL: http://coolidea.ru/2014/12/05/distancionnoe-opredelenie-razmerov. (дата обращения: 01.02.2017).
4. Способ бесконтактного измерения прямых линейных размерных признаков фигуры человека : пат. 2264768 Рос. Федерация : МПК[7] A 41 H 1/00 / Раздомахин Н. Н., Басуев А. Г., Калитеевский Н. А. ; заявитель и патентообладатель Раздомахин Н. Н., Басуев А. Г. – № 2004120566/12 ; заявл. 05.07.04 ; опубл. 27.11.05, Бюл. № 33. – 17 с. (описание под заглавием)
5. Как сделать выкройку без «косяков» или 3d манекен Вам в помощь. – URL: http://kompkroy.ru/kak-sdelat-vykrojku-bez-kosyakov-ili-3d-maneken-vam-v-pomoshh.html (дата обращения: 15.02.2017)
6. Способ и система для перемещения виртуальной модели человека в виртуальной среде : пат. 2355030 Рос. Федерация : МПК[7] G 06 T 13/00, G 06 F 17/50 / Мэй Б., Рамстейн Э., Шедмэ П. ; заявитель и патентообладатель Снекма – № 2004112085/09 ; заявл. 20.10.05 ; опубл. 10.05.09, Бюл. № 13. – 26 с. (описание под заглавием)
7. Three-dimensional, digitized sensing of the spatial shape of bodies and body parts with mechanically positioned imaging sensors: пат. CA2518017 C : A61B5/107, A61B5/053 / Dirk Rutschmann ; заявитель и патентообладатель Corpus.E Ag, Dirk Rutschmann – № PCT/EP2004/002136; заявл. 03.03.04 ; опубл. 03.01.12 (описание под заглавием)
8. Clothing amount measuring apparatus and method using image processing:пат. US5805718 A : G01N25/18, G01N25/00, G06T7/60 / Ryo Inoshiri, Akira Yoshida, Ziquan Hon; заявитель и патентообладатель Sharp Kabushiki Kaisha – № US 08/539,761; заявл. 06.10.95 ; опубл. 08.09.98 (описание под заглавием)
9. Способ отображения трехмерного аватара и система, осуществляющая этот способ : пат. 2396599 Рос. Федерация : МПК[8] G 06 T 15/70, H 04 L 12/58 / Гу С., Ся Л., Цзя Я. ; заявитель и патентообладатель Тенсент Текнолоджи (Шэньчжэнь) компании лимитед – № 2008130658/09 ; заявл. 31.12.06 ; опубл. 10.08.10, Бюл. № 22. – 34 с. (описание под заглавием)
10. Способ обеспечения удалённой примерки и/или выбора одежды : пат. 2504009 Рос. Федерация : МПК[8] G 06 Q 50/10, A 41 H 1/02 / Гринблат А. О., Мартиросян В. И. ; заявитель и патентообладатель ООО «Дрессформер». – № 2012128687/08 ; заявл. 10.07.12 ; опубл. 10.01.14, Бюл. № 1. – 11 с. (описание под заглавием)

11. Способ проектирования одежды на основе бесконтактной антропометрии : пат. 2358628 Рос. Федерация : МПК7 A 41 H 3/00 / Сеницкий И. А., Корнилова Н. Л. ; заявитель и патентообладатель ООО «Центр наукоемких инновационных технологий для швейной промышленности». – № 2007111766/12 ; заявл. 02.04.07 ; опубл. 20.06.09, Бюл. № 17. – 17 с. (описание под заглавием)
12. Создание графических моделей с помощью Open Graphics Library . – URL: http://www.intuit.ru/studies/courses/2313/613/lecture/13296 (дата обращения: 07.02.2017).
13. Сравнение OpenGL и Direct3D – Дата обновления: 25.12.2009. URL: https://habrahabr.ru/post/79257 (дата обращения: 15.01.2017).
14. Vulkan API (glNext) от Khronos Group – Дата обновления: 13.05.2016. URL: https://habrahabr.ru/post/283490. (дата обращения: 01.02.2017).
15. Кривые Безье. – Дата обновления: 17.05.2016. URL: http://grafika.me/node/521 (дата обращения: 16.02.2017)
16. Скворцов, А.В. Алгоритмы построения и анализа триангуляции / А.В. Скворцов, Н.С. Мирза. – Томск: Изд-во Том. ун-т, 2006. – 168 с.
17. ГОСТ 19.201 – 78. Техническое задание. Требование к содержанию и оформлению. – М.: Изд-во стандартов, 2004. – 34 с.

ТЕКСТ ПРОГРАММЫ

Start.cpp – форма с описанием программы

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Start.h"
#include "AddFhoto.h"
#include "Image_front.h"
#include "Model.h"
#include "fstream.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
int mas[71][3]; // массив точек загруженных из файла
int open=0; //флаг загрузки фигуры
int  length_Y_front,length_Y_back,length_X_side,length_Z_side;  //размеры  купюры  в
программе
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
      Form1->Show();
      Form2->Visible=false;
}
//---------------------------------------------------------------------------


void __fastcall TForm2::N2Click(TObject *Sender)
{
//загрузка манекена из файла
if(OpenDialog1->Execute()){
      char buf[81];
      ifstream f(OpenDialog1->FileName.c_str());
      if(!f){
            ShowMessage("Файла не существует");
            return ;
      }
      try{
      while(!f.eof()){
            int j=0;
            for(int i=0;i<3;i++){
                  for(int j=0;j<70;j++){
                        f.getline(buf,81);
                        mas[j][i]=StrToInt(buf);
                  }
            }
            f.getline(buf,81);
            length_Y_front=StrToInt(buf);
            f.getline(buf,81);
            length_Y_back=StrToInt(buf);
            f.getline(buf,81);
```

```
            length_X_side=StrToInt(buf);
            f.getline(buf,81);
            length_Z_side=StrToInt(buf);
            open=1;
            Form3->Show();
            Form2->Visible=false;
      }
      }catch(...){
            ShowMessage("Файл повреждён или неверно записан");
      }
}
}
//---------------------------------------------------------------------------

void __fastcall TForm2::N5Click(TObject *Sender)
{
      if((Application->MessageBox("Вы действительно хотите выйти?\nВсе сохранённые
данные пропадут!",
                  "Сообщение",MB_YESNO | MB_ICONQUESTION))==IDYES){
            Close();
      }
}
//---------------------------------------------------------------------------

void __fastcall TForm2::FormCloseQuery(TObject *Sender, bool &CanClose)
{
      if((Application->MessageBox("Вы действительно хотите выйти?\nВсе сохранённые
данные пропадут!",
                  "Сообщение",MB_YESNO | MB_ICONQUESTION))==IDYES){
            CanClose=true;
      }else CanClose=false;
}
//---------------------------------------------------------------------------

void __fastcall TForm2::N4Click(TObject *Sender)
{
     MessageBox(Handle, "Приложение манекен.\nАвтор: Худжанов Дмитрий.\nГруппа ЕТ-
484\nЮУрГУ,2017",
                  "О программе",MB_ICONASTERISK);
}
//---------------------------------------------------------------------------




void __fastcall TForm2::FormActivate(TObject *Sender)
{
      Form2->Image1->Picture->LoadFromFile("Картинки/Susu.bmp");
}
//---------------------------------------------------------------------------
```

Start.h – заголовочный файл.

```
//---------------------------------------------------------------------------
#ifndef StartH
#define StartH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Mask.hpp>
```

```
#include <Menus.hpp>
#include <Dialogs.hpp>
#include <ComCtrls.hpp>
#include <jpeg.hpp>

//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:      // IDE-managed Components
      TButton *Button1;
      TMainMenu *MainMenu1;
      TMenuItem *N1;
      TMenuItem *N2;
      TMenuItem *N3;
      TMenuItem *N4;
      TMenuItem *N5;
      TOpenDialog *OpenDialog1;
      TMemo *Memo1;
      TImage *Image1;
      void __fastcall Button1Click(TObject *Sender);
      void __fastcall N2Click(TObject *Sender);
      void __fastcall N5Click(TObject *Sender);
      void __fastcall FormCloseQuery(TObject *Sender, bool &CanClose);
      void __fastcall N4Click(TObject *Sender);
      void __fastcall FormActivate(TObject *Sender);
private:    // User declarations
public:           // User declarations
      __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```

AddFhoto.cpp – форма с добавлением фотографий.

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "AddFhoto.h"
#include "Start.h"
#include "Image_front.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
String image_front,image_side,image_back;
int count=0;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
            Image1->Canvas->Pen->Color=clBlack;
            Canvas->Rectangle(0,0,Image1->Width,Image1->Height);
            Image2->Canvas->Pen->Color=clBlack;
            Canvas->Rectangle(0,0,Image2->Width,Image2->Height);
            Image3->Canvas->Pen->Color=clBlack;
            Canvas->Rectangle(0,0,Image3->Width,Image3->Height);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
```

```
{
      //добавление фотографии вид спереди
      if (OpenDialog1->Execute()) {
            image_front=OpenDialog1->FileName;
            Form1->Image1->Picture->LoadFromFile(image_front);
            Label4->Visible=false;
      }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
      //добавление фотографии вид сбоку
      if (OpenDialog1->Execute()) {
            image_side=OpenDialog1->FileName;
            Form1->Image2->Picture->LoadFromFile(image_side);
            Label5->Visible=false;
      }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
      //добавление фотографии вид сзади
     if (OpenDialog1->Execute()) {
            image_back=OpenDialog1->FileName;
            Form1->Image3->Picture->LoadFromFile(image_back);
            Label6->Visible=false;
      }
}
//---------------------------------------------------------------------------


void __fastcall TForm1::Button4Click(TObject *Sender)
{
      if (image_front=="") {
            MessageBox(NULL,"Не выбрано изображение вид спереди!","Ошибка",MB_OK);
            return;
      }
      if (image_side=="") {
            MessageBox(NULL,"Не выбрано изображение вид сбоку!","Ошибка",MB_OK);
            return;
      }
      if (image_back=="") {
            MessageBox(NULL,"Не выбрано изображение вид сзади!","Ошибка",MB_OK);
            return;
      }
      Form4->Show();
      Form1->Visible=false;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
      Form2->Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormActivate(TObject *Sender)
{
      Form1->Image1->Picture->LoadFromFile("Картинки/1.bmp");
      Form1->Image2->Picture->LoadFromFile("Картинки/2.bmp");
```

```
      Form1->Image3->Picture->LoadFromFile("Картинки/3.bmp");
}
//---------------------------------------------------------------------------
```

AddFhoto.h – заголовочный файл.

```
//---------------------------------------------------------------------------
#ifndef AddFhotoH
#define AddFhotoH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Dialogs.hpp>
extern String image_front,image_side,image_back;
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
      TButton *Button1;
      TButton *Button2;
      TButton *Button3;
      TImage *Image1;
      TImage *Image2;
      TImage *Image3;
      TOpenDialog *OpenDialog1;
      TOpenDialog *OpenDialog2;
      TOpenDialog *OpenDialog3;
      TButton *Button4;
      TLabel *Label1;
      TLabel *Label2;
      TLabel *Label3;
      TButton *Button5;
      TButton *Button6;
      TButton *Button7;
      void __fastcall Button1Click(TObject *Sender);
      void __fastcall Button2Click(TObject *Sender);
      void __fastcall Button3Click(TObject *Sender);
      void __fastcall Button4Click(TObject *Sender);
      void __fastcall Button5Click(TObject *Sender);
      void __fastcall Button6Click(TObject *Sender);
      void __fastcall Button7Click(TObject *Sender);
      void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
private:    // User declarations
public:           // User declarations
      __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
extern String image_front,image_side,image_back;
//---------------------------------------------------------------------------
#endif
```

Model.cpp – форма с 3D моделью и основными показателями.

```
//---------------------------------------------------------------------------
#include <vcl.h>
#include <math.h>
#include <stdlib.h>
#pragma hdrstop
#include <string>
```

```
#include "fstream.h"

#include "Model.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
int oldX=0;
int oldY=0;
//первоначальное расположения камеры
float camX = 0;
float camY = 0;
float camZ = 0;
float camR = 1000;
int tetta = 90;
int phi = 0;
extern const int length1=16,length2=35,length3=22;
extern int Xob,Yob,Zob,sizex1,sizey1,sizex2,sizey2,sizex3,sizey3;
extern int mas1[length1][2],mas2[length2][2],mas3[length3][2];
extern int mas[71][3];
int siz=372; //количество точек для триангуляции
extern int open;
int m_Placeholders[4][3];
extern  int  length_Y_front,length_Y_back,length_X_side,length_Z_side;   //размеры
купюры в программе
double relX,relY1,relY2,relZ; //соотношение реального расстояния и в прграмме
double growth,    //рост
       chest,     //обхват груди
       waist,     //обхват талии
       hip,       //обхват бедра
       sleeve,    //длина рукава
       leg,       //длина штанин
       shoulder; //длина плеч
int
MAScon1[]={44,59,48,48,59,60,49,44,48,44,49,45,16,45,22,16,22,59,59,22,60,45,49,22
,  //низ правой руки
57,24,58,58,24,27,27,24,28,24,25,28,25,37,28,37,38,28,37,57,58,38,37,58,//низ
левой ноги
24,23,19,19,25,24,25,19,64,64,37,25,63,37,64,63,57,37,63,23,57,23,24,57,//верх
левой ноги
51,53,52,52,53,54,52,54,26,26,54,29,26,29,61,61,29,62,61,62,51,51,62,53,//низ
правой нога
15,20,14,15,21,20,21,15,34,35,21,34,35,34,55,35,55,56,56,55,14,20,56,14,//низ
левой руки
50,51,47,47,51,52,47,52,65,65,52,26,63,65,26,63,26,61,63,61,50,50,61,51,//верх
правой ноги
63,64,65,23,18,19,18,23,63,47,46,50,50,46,63,46,18,63,//пах
46,13,18,13,36,18,18,36,19,19,36,17,19,17,64,17,65,64,65,17,31,65,31,47,31,46,47,4
6,66,13,31,66,46,  //живот
17,33,12,17,36,33,36,32,33,32,36,10,10,36,13,9,10,13,66,42,13,9,13,42,42,66,43,31,
43,66,43,31,11,31,12,11,31,17,12,  //от живота до груди
6,12,33,6,33,30,30,33,32,32,8,30,10,30,8,9,30,10,30,9,7,9,42,7,
7,42,40,42,43,40,41,40,43,40,41,11,6,40,11,12,6,11,
32,55,34,34,8,32,34,15,8,8,15,14,8,14,10,10,14,55,10,55,32,
16,59,11,11,41,16,16,41,45,41,44,45,41,43,44,44,43,59,43,11,59,
58,27,38,38,27,28,53,62,29,54,53,29,56,20,35,35,20,21,60,22,48,22,49,48,
30,5,6,5,30,69,30,7,69,7,68,69,7,67,68,67,7,40,67,40,5,40,6,5,5,68,67,69,68,5};
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
      : TForm(Owner)
{
}
```

```
//---------------------------------------------------------------------------
BOOL TForm3::bSetupPixelFormat(HDC hDC)
{

  PIXELFORMATDESCRIPTOR pfd; //Создаем структуру
  int pixelformat;
  pfd.nSize = sizeof (PIXELFORMATDESCRIPTOR); //Размер структуры
  pfd.nVersion = 1;  //Версия структуры
  pfd.dwFlags = PFD_DOUBLEBUFFER | PFD_SUPPORT_OPENGL | PFD_DRAW_TO_WINDOW;
  pfd.iLayerType =  PFD_MAIN_PLANE;  //Тип поверхности
  pfd.iPixelType = PFD_TYPE_RGBA;  //Формат указания цвета
  pfd.cColorBits = 16;  //Глубина цвета
  pfd.cDepthBits = 16;  //Размер буфера глубины
  pfd.cAccumBits = 0; //Общее число битовых плоскостей в буфере аккумулятора
  pfd.cStencilBits = 0;  //Размер буфера трафарета
  if (!(pixelformat=ChoosePixelFormat(hDC, &pfd))) {
      MessageBox(NULL, "Невозможно выбрать формат пикселей", "Error", MB_OK);
      return false;
  }
  if (!SetPixelFormat (hDC, pixelformat, &pfd)) {
      MessageBox(NULL, "Невозможно установить формат пикселей", "Error", MB_OK);
      return false;
  }
  return true;
}
//---------------------------------------------------------------------------

void __fastcall TForm3::FormCreate(TObject *Sender)
{
      hDC = GetDC(Handle); //Handle – дескриптор окна (hwnd в WinAPI)
      if (!bSetupPixelFormat(hDC)) //Устанавливаем формат пикселей
            return;
      ghRC = wglCreateContext(hDC); //Создаем контекст воспроизведения
      wglMakeCurrent(hDC, ghRC);  //Делаем его текущим

      glClearColor(0.95, 0.9,0.7, 1); //Цвет экрана при очищении
      glEnable(GL_COLOR_MATERIAL);  //Разрешаем задавать цвет объектам
      glEnable(GL_DEPTH_TEST);  //Тест глубины для объемности изображения
      glEnable(GL_LIGHTING);  //Разрешаем освещение
      glEnable(GL_LIGHT0);  //Включили освещение 0
      glEnable(GL_CULL_FACE);
      float p[4]={1000,1000,1000,1};
      glLightfv(GL_LIGHT0,GL_POSITION,p);  //Установка позиции освещения
      //glEnable(GL_TEXTURE_2D);
}
//---------------------------------------------------------------------------

void __fastcall TForm3::FormDestroy(TObject *Sender)
{
      if(ghRC)
      {
            wglMakeCurrent(hDC,0);
      wglDeleteContext(ghRC);
      }
      if(hDC) ReleaseDC(Handle, hDC);
}
//---------------------------------------------------------------------------

void __fastcall TForm3::FormResize(TObject *Sender)
{
      int x, y, dx, dy;
      x = 0;
```

```
      y = 0;
      dx = ClientHeight;
      dy = ClientHeight;
      glViewport(x, y, dx, dy);
      Draw();
}
//---------------------------------------------------------------------------
void TForm3::Draw(){
      int xsm=mas[9][0]+(mas[12][0]-mas[9][0])/2;
      int ysm=mas[3][1];
      int zsm=mas[28][2]+(mas[0][2]-mas[28][2])/2;
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluPerspective (35, 1,  0.1, 10000);
      float angle1 = tetta*M_PI/180;
      float angle2 = phi*M_PI/180;

      camX = camR * sin(angle1)*cos(angle2); //новое коодринаты камеры по оси OX
      camY = camR * cos(angle1); //новое коодринаты камеры по оси OY
      camZ = camR * sin(angle1)*sin(angle2); //новое коодринаты камеры по оси OZ
      gluLookAt(camX, camZ, camY, 0, 0, 0, 0,0,sin(angle1)); //положение камеры
      glMatrixMode(GL_MODELVIEW);
      glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
      glPolygonMode(GL_FRONT_AND_BACK, GL_LINE); // см. выше
      glLineWidth(2);   //размер линии
      glTranslatef(-xsm,-ysm,-zsm);
      glBegin(GL_TRIANGLES);
      glColor3d(0,0,0);
      glPointSize(10);
      //рисование линий треугольников
      for(int i=0;i<siz;i=i+3){
            glVertex3d(mas[MAScon1[i]][0],mas[MAScon1[i]][1],mas[MAScon1[i]][2]);
            glVertex3d(mas[MAScon1[i+1]][0],mas[MAScon1[i+1]][1],mas[MAScon1[i+1]]
[2]);
            glVertex3d(mas[MAScon1[i+2]][0],mas[MAScon1[i+2]][1],mas[MAScon1[i+2]]
[2]);
      }
      glEnd();
      //рисование внутренней области треугольников
      glPolygonMode(GL_FRONT_AND_BACK, GL_FILL); // см. выше
      glLineWidth(10);
      glBegin(GL_TRIANGLES);
      glColor3d(0.5,0.5,0.5);
  //for(int i=0;i<318;i=i+3){
      for(int i=0;i<siz;i=i+3){
            glVertex3d(mas[MAScon1[i]][0],mas[MAScon1[i]][1],mas[MAScon1[i]][2]);
            glVertex3d(mas[MAScon1[i+1]][0],mas[MAScon1[i+1]][1],mas[MAScon1[i+1]]
[2]);
            glVertex3d(mas[MAScon1[i+2]][0],mas[MAScon1[i+2]][1],mas[MAScon1[i+2]]
[2]);
      }
      glEnd();

      /*glPointSize(3);
      glBegin(GL_POINTS);
      glColor3d(0,0,0);
      for(int i=5;i<70;i++){
            glVertex3d(mas[i][0],mas[i][1],mas[i][2]); // первая точка
      }
      glEnd(); */
      glTranslatef(xsm,ysm,zsm);
      glDisable(GL_LINE_STIPPLE);
```

```
      SwapBuffers(hDC);
      //основные показатели
      TVarRec args1[] = {growth/10};
      TVarRec args2[] = {chest/10};
      TVarRec args3[] = {waist/10};
      TVarRec args4[] = {hip/10};
      TVarRec args5[] = {shoulder/10};
      TVarRec args6[] = {sleeve/10};
      TVarRec args7[] = {leg/10};
      Label1->Caption=Format("Рост:              %.1f",args1,1);
      Label1->Caption=Label1->Caption + " см";
      Label2->Caption=Format("Обхват груди: %.1f",args2,1);
      Label2->Caption=Label2->Caption + " см";
      Label3->Caption=Format("Обхват талии: %.1f",args3,1);
      Label3->Caption=Label3->Caption + " см";
      Label4->Caption=Format("Обхват бёдер: %.1f",args4,1);
      Label4->Caption=Label4->Caption + " см";
      Label5->Caption=Format("Длина плеч:     %.1f",args5,1);
      Label5->Caption=Label5->Caption + " см";
      Label6->Caption=Format("Длина рукава:  %.1f",args6,2);
      Label6->Caption=Label6->Caption + " см";
      Label7->Caption=Format("Длина штанов: %.1f",args7,1);
      Label7->Caption=Label7->Caption + " см";
}

void __fastcall TForm3::FormActivate(TObject *Sender)
{
if(open==1){
      //рисование фигуры при загрузке из формы с описанием
      Calculation();
      Draw();
}else{
// объединение трёх массивов с форм расстановок точек в один общий
      for (int i=0; i <= 31; i++) {
            mas[i][0]=mas2[i][0];
            mas[i][2]=sizey2-mas2[i][1];
      }
      int offset = mas1[0][0]-mas3[0][0];
      for (int i=0; i < length3-1; i++) {
            mas3[i][0]+=offset;
      }
      int       med=(mas1[0][0]+mas1[1][0]+mas1[2][0]+mas1[3][0]+mas1[4][0]+mas1[8]
[0])/6;
      mas[0][1]=mas1[0][0];
      mas[1][1]=mas3[0][0];
      mas[2][1]=mas3[1][0];
      mas[3][1]=mas1[1][0];
      mas[4][1]=mas1[2][0];
      mas[5][1]=mas1[3][0];
      mas[6][1]=mas1[4][0];
      mas[7][1]=mas3[2][0];
      mas[8][1]=mas3[3][0];
      mas[9][1]=mas3[4][0];
      mas[10][1]=mas3[5][0];
      mas[11][1]=mas1[5][0];
      mas[12][1]=mas1[6][0];
      mas[13][1]=mas3[6][0];
      mas[14][1]=mas3[7][0];
      mas[15][1]=mas3[8][0];
      mas[16][1]=mas1[7][0];
      mas[17][1]=mas1[8][0];
      mas[18][1]=mas3[9][0];
```

```
      mas[19][1]=mas3[10][0];
      mas[20][1]=mas3[11][0];
      mas[21][1]=mas3[12][0];
      mas[22][1]=mas1[9][0];
      mas[23][1]=mas3[13][0];
      mas[24][1]=mas3[14][0];
      mas[25][1]=mas3[15][0];
      mas[26][1]=mas1[10][0];
      mas[27][1]=mas3[16][0];
      mas[28][1]=mas3[17][0];
      mas[29][1]=mas1[11][0];
      mas[30][1]=mas3[18][0];
      mas[31][1]=mas1[12][0];
      mas[32][0]=mas2[11][0];mas[32][1]=med-(mas1[5][0]-med);mas[32][2]=sizey2-
mas2[11][1];
      mas[33][0]=mas2[12][0];mas[33][1]=med-(mas1[6][0]-med);mas[33][2]=sizey2-
mas2[12][1];
      mas[34][0]=mas2[16][0];mas[34][1]=med-(mas1[7][0]-med);mas[34][2]=sizey2-
mas2[16][1];
      mas[35][0]=mas2[22][0];mas[35][1]=med-(mas1[9][0]-med);mas[35][2]=sizey2-
mas2[22][1];
      mas[36][0]=mas2[31][0];mas[36][1]=med-(mas1[12][0]-med);mas[36][2]=sizey2-
mas2[31][1];
      mas[37][0]=mas2[26][0];mas[37][1]=med-(mas1[10][0]-med);mas[37][2]=sizey2-
mas2[26][1];
      mas[38][0]=mas2[29][0];mas[38][1]=med-(mas1[11][0]-med);mas[38][2]=sizey2-
mas2[29][1];
      mas[39][0]=mas2[2][0];mas[39][1]=med-(mas3[1][0]-med);mas[39][2]=sizey2-
mas2[2][1];
      mas[40][0]=mas2[30][0];mas[40][1]=med-(mas3[18][0]-med);mas[40][2]=sizey2-
mas2[30][1];
      mas[41][0]=mas2[8][0];mas[41][1]=med-(mas3[3][0]-med);mas[41][2]=sizey2-
mas2[8][1];
      mas[42][0]=mas2[9][0];mas[42][1]=med-(mas3[4][0]-med);mas[42][2]=sizey2-
mas2[9][1];
      mas[43][0]=mas2[10][0];mas[43][1]=med-(mas3[5][0]-med);mas[43][2]=sizey2-
mas2[10][1];
      mas[44][0]=mas2[14][0];mas[44][1]=med-(mas3[7][0]-med);mas[44][2]=sizey2-
mas2[14][1];
      mas[45][0]=mas2[15][0];mas[45][1]=med-(mas3[8][0]-med);mas[45][2]=sizey2-
mas2[15][1];
      mas[46][0]=mas2[18][0];mas[46][1]=med-(mas3[9][0]-med);mas[46][2]=sizey2-
mas2[18][1];
      mas[47][0]=mas2[19][0];mas[47][1]=med-(mas3[10][0]-med);mas[47][2]=sizey2-
mas2[19][1];
      mas[48][0]=mas2[20][0];mas[48][1]=med-(mas3[11][0]-med);mas[48][2]=sizey2-
mas2[20][1];
      mas[49][0]=mas2[21][0];mas[49][1]=med-(mas3[12][0]-med);mas[49][2]=sizey2-
mas2[21][1];
      mas[50][0]=mas2[23][0];mas[50][1]=med-(mas3[13][0]-med);mas[50][2]=sizey2-
mas2[23][1];
      mas[51][0]=mas2[24][0];mas[51][1]=med-(mas3[14][0]-med);mas[51][2]=sizey2-
mas2[24][1];
      mas[52][0]=mas2[25][0];mas[52][1]=med-(mas3[15][0]-med);mas[52][2]=sizey2-
mas2[25][1];
      mas[53][0]=mas2[27][0];mas[53][1]=med-(mas3[16][0]-med);mas[53][2]=sizey2-
mas2[27][1];
      mas[54][0]=mas2[28][0];mas[54][1]=med-(mas3[17][0]-med);mas[54][2]=sizey2-
mas2[28][1];
      mas[55][0]=mas2[15][0];mas[55][1]=mas3[7][0]-(mas3[8][0]-mas3[7][0]);mas[55]
[2]=sizey2-mas2[15][1];
```

```
      mas[56][0]=mas2[21][0];mas[56][1]=mas3[11][0]-(mas3[12][0]-mas3[11]
[0]);mas[56][2]=sizey2-mas2[21][1];
      mas[57][0]=mas2[25][0];mas[57][1]=mas3[14][0]-(mas3[15][0]-mas3[14]
[0]);mas[57][2]=sizey2-mas2[25][1];
      mas[58][0]=mas2[28][0];mas[58][1]=mas3[16][0]-(mas3[17][0]-mas3[16]
[0]);mas[58][2]=sizey2-mas2[28][1];
      mas[59][0]=mas2[15][0];mas[59][1]=med-((mas3[7][0]-(mas3[8][0]-mas3[7][0]))-
med);mas[59][2]=sizey2-mas2[15][1];
      mas[60][0]=mas2[21][0];mas[60][1]=med-((mas3[11][0]-(mas3[12][0]-mas3[11]
[0]))-med);mas[60][2]=sizey2-mas2[21][1];
      mas[61][0]=mas2[25][0];mas[61][1]=med-((mas3[14][0]-(mas3[15][0]-mas3[14]
[0]))-med);mas[61][2]=sizey2-mas2[25][1];
      mas[62][0]=mas2[28][0];mas[62][1]=med-((mas3[16][0]-(mas3[17][0]-mas3[16]
[0]))-med);mas[62][2]=sizey2-mas2[28][1];
      mas[63][0]=mas2[19][0];mas[63][1]=med;mas[63][2]=sizey2-mas2[23][1];
      mas[64][0]=mas2[26][0];mas[64][1]=mas3[9][0];mas[64][2]=sizey2-mas2[19][1];
      mas[65][0]=mas2[26][0];mas[65][1]=med-(mas3[9][0]-med);mas[65][2]=sizey2-
mas2[19][1];
      mas[66][0]=mas2[13][0];mas[66][1]=med-(mas3[6][0]-med);mas[66][2]=sizey2-
mas2[13][1];
      mas[67][0]=mas2[30][0];mas[67][1]=med-(mas3[18][0]-med);mas[67][2]=sizey2-
mas2[5][1];
      mas[68][0]=mas2[7][0];mas[68][1]=mas3[2][0];mas[68][2]=sizey2-mas2[5][1];
      mas[69][0]=mas2[30][0];mas[69][1]=mas3[18][0];mas[69][2]=sizey2-mas2[5][1];
      length_Y_front=pow(mas1[15][0]-mas1[13][0],2)+(pow(mas1[15][1]-mas1[13]
[1],2));
      length_Y_back=pow(mas3[21][0]-mas3[19][0],2)+(pow(mas3[21][1]-mas3[19]
[1],2));
      length_X_side=pow(mas2[34][0]-mas2[32][0],2)+(pow(mas2[34][1]-mas2[32]
[1],2));
      length_Z_side=pow(mas2[33][0]-mas2[32][0],2)+(pow(mas2[33][1]-mas2[32]
[1],2));
      Calculation();
      Draw();
}
}
//---------------------------------------------------------------------------

void __fastcall TForm3::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
//перемещение камеры при нажатии как стрелки мыши
 switch (Key){
            case 39:
                  phi+=2;
                  break;

            case 37:
                  phi-=2;
                  break;
            case 40:
                  tetta+=2;
                  break;
            case 38:
                  tetta-=2;
                  break;
 }
              Form3->Draw();
}
//---------------------------------------------------------------------------

void __fastcall TForm3::FormMouseMove(TObject *Sender, TShiftState Shift, int X,
```

```
      int Y)
{
// поворот камеры при зажатой правой кнопки мыши
            if (Shift==TShiftState()<<ssRight) {
            if (X > oldX) phi-=4;
            if (X < oldX) phi+=4;
            if (Y > oldY) tetta-=4;
            if (Y < oldY) tetta+=4;
            oldX = X;
            oldY = Y;
            Form3->Draw();
            }
}
//---------------------------------------------------------------------------

void __fastcall TForm3::N2Click(TObject *Sender)
{
//сохранение фигуры
    if(SaveDialog1->Execute()){
            ofstream f(SaveDialog1->FileName.c_str());
            for(int i=0;i<3;i++){
                  for(int j=0;j<70;j++){
                              f<<mas[j][i]<<"\n";
                  }

            }
            f<<length_Y_front<<"\n";
            f<<length_Y_back<<"\n";
            f<<length_X_side<<"\n";
            f<<length_Z_side;
      }
}
//---------------------------------------------------------------------------

void __fastcall TForm3::N4Click(TObject *Sender)
{
//загрузка фигуры
if(OpenDialog1->Execute()){
      char buf[81];
      ifstream f(OpenDialog1->FileName.c_str());
      if(!f){
            ShowMessage("Файла не существует");
            return ;
      }
      try{
      while(!f.eof()){
            int j=0;
            for(int i=0;i<3;i++){
                  for(int j=0;j<70;j++){
                        f.getline(buf,81);
                        mas[j][i]=StrToInt(buf);
                  }
            }
            f.getline(buf,81);
            length_Y_front=StrToInt(buf);
            f.getline(buf,81);
            length_Y_back=StrToInt(buf);
            f.getline(buf,81);
            length_X_side=StrToInt(buf);
            f.getline(buf,81);
            length_Z_side=StrToInt(buf);
            Calculation();
```

```
            Form3->Draw();
      }
      }catch(...){
            ShowMessage("Файл повреждён или неверно записан");
      }
}
}
void __fastcall TForm3::FormMouseWheel(TObject *Sender, TShiftState Shift,
        int WheelDelta, TPoint &MousePos, bool &Handled)
{
//приближение и отдаление камеры
if(camR<2000)
{
      if(WheelDelta<0)camR+=10;
      Form3->Draw();
}
if(camR>700)
{
      if(WheelDelta>0)camR-=10;
      Form3->Draw();
}
}
//---------------------------------------------------------------------------


void __fastcall TForm3::FormPaint(TObject *Sender)
{
      Draw();
}
//---------------------------------------------------------------------------
//вычисление основных показателей
void TForm3::Calculation(){
      if(length_X_side!=0    &&    length_Y_front!=0    &&    length_Y_back!=0    &&
length_Z_side!=0){
      relX=150/sqrt((float)length_X_side);
      relY1=150/sqrt((float)length_Y_front);
      relY2=150/sqrt((float)length_Y_back);
      relZ=65/sqrt((float)length_Z_side);
      growth=sqrt(pow((mas[0][2]-mas[28][2])*relZ,2)+pow((mas[0][0]-mas[28]
[0])*relX,2))+80; //рост
      chest=2*M_PI*sqrt((pow((Len(mas[12][0],mas[9][0],(mas[12][1]+mas[33][1])/2,
(mas[9][1]+mas[42][1])/2,mas[12][2],mas[9][2]))/2,2)+
                        pow((Len(mas[8][0],mas[41][0],mas[11][1],mas[32][1],mas[11]
[2],mas[32][2]))/2,2))/2);    //обхват груди
      waist=2*M_PI*sqrt((pow((Len(mas[17][0],mas[13][0],mas[17][1],(mas[13]
[1]+mas[66][1])/2,mas[17][2],mas[13][2]))/2,2)+
                        pow((Len(mas[31][0],mas[36][0],mas[31][1],mas[36][1],mas[31]
[2],mas[36][2]))/2,2))/2);    //обхват талии
      hip=2*M_PI*sqrt((pow((Len(mas[18][0],mas[64][0],mas[63][1],mas[63]
[1],mas[18][2],mas[64][2]))/2,2)+
                        pow((Len(mas[19][0],mas[47][0],mas[19][1],mas[47][1],mas[19]
[2],mas[47][2]))/2,2))/2);      //обхват бедра
      sleeve=Len(mas[8][0],mas[15][0],mas[8][1],mas[15][1],mas[8][2],mas[15][2])+
      Len(mas[15][0],mas[21][0],mas[15][1],mas[21][1],mas[15][2],mas[21]
[2]);   //длина рукава
      leg=Len(mas[19][0],mas[31][0],mas[19][1],mas[31][1],mas[19][2],mas[31]
[2])/2+
            Len(mas[19][0],mas[25][0],mas[19][1],mas[25][1],mas[19][2],mas[25][2])
+
            Len(mas[25][0],mas[28][0],mas[25][1],mas[28][1],mas[25][2],mas[28]
[2]);      //длина штанин
```

```
      shoulder=Len(mas[8][0],mas[41][0],mas[8][1],mas[41][1],mas[8][2],mas[41]
[2]); //длина плеч
      }
}
//---------------------------------------------------------------------------
//вычисление расстояния между точками
float TForm3::Len(int X1,int X2,int Y1,int Y2,int Z1,int Z2){
      float l;
      l=sqrt(pow((X1-X2)*relX,2)+pow((Y1-Y2)*relY2,2)+pow((Z1-Z2)*relZ,2));
      return l;
}

void __fastcall TForm3::N7Click(TObject *Sender)
{
  MessageBox(Handle, "Приложение манекен.\nАвтор: Худжанов Дмитрий.\nГруппа ЕТ-
484\nЮУрГУ,2017",
                   "О программе",MB_ICONASTERISK);
}
//---------------------------------------------------------------------------
```

Model.h – заголовочный файл.

```
//---------------------------------------------------------------------------
#ifndef ModelH
#define ModelH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
#include <Dialogs.hpp>

#include <GL/gl.h>
#include <GL/glu.h>

#include <GL/glaux.h>
#pragma comment (lib, "glaux.lib")
//---------------------------------------------------------------------------
class TForm3 : public TForm
{
__published:      // IDE-managed Components
      TMainMenu *MainMenu1;
      TMenuItem *N1;
      TMenuItem *N2;
      TMenuItem *N3;
      TOpenDialog *OpenDialog1;
      TSaveDialog *SaveDialog1;
      TMenuItem *N4;
      TMenuItem *N5;
      TGroupBox *GroupBox1;
      TLabel *Label1;
      TLabel *Label2;
      TLabel *Label3;
      TLabel *Label4;
      TLabel *Label5;
      TLabel *Label6;
      TLabel *Label7;
      TMenuItem *N6;
      TMenuItem *N7;
      void __fastcall FormCreate(TObject *Sender);
      void __fastcall FormDestroy(TObject *Sender);
      void __fastcall FormResize(TObject *Sender);
```

```
      void __fastcall FormActivate(TObject *Sender);
      void __fastcall FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift);
      void __fastcall FormMouseMove(TObject *Sender, TShiftState Shift, int X,
          int Y);
      void __fastcall N2Click(TObject *Sender);
      void __fastcall N4Click(TObject *Sender);
      void __fastcall FormMouseWheel(TObject *Sender, TShiftState Shift,
          int WheelDelta, TPoint &MousePos, bool &Handled);
      void __fastcall FormPaint(TObject *Sender);
      void __fastcall N7Click(TObject *Sender);
private:    // User declarations
HGLRC ghRC;  // указатель на контекст воспроизведения (Rendering Context)
HDC    hDC; // дескриптор (контекст) устройства
public:           // User declarations
      BOOL bSetupPixelFormat(HDC hDC);
      void Draw();
      void Calculation();
      float Len(int,int,int,int,int,int);
      float Len2(int,int,int,int,int,int);
      __fastcall TForm3(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm3 *Form3;
//---------------------------------------------------------------------------
#endif
```

Image_front.cpp – форма расстановки точек (вид спереди).

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Image_front.h"
#include "Instruction.h"
#include "AddFhoto.h"
#include "Image_side.h"
#include "Start.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
int perem=0;      //флаг перемещения точки
int j=0,    //текущее количество точек
      x1,y1,      //переменные для хранения координат точки
      schet=0, //флаг попадания в точку
      x2,y2; //переменные для хранения координат точки
int Per=-1; //переменная для переноса точки
const int length1=16;   //количество точек на фигуре
int Yob,    //середина картинки по оси OX
      sizex1,sizey1;    //размер image с загруженной картинкой
int mas1[length1][2];  //массив точек
Graphics::TBitmap *img = new Graphics::TBitmap;
int vis=0; //переменная хранящая расположение фигуры
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
      : TForm(Owner)
{
      Form4->Image2->Center= true;
      Form4->Image2->Picture->LoadFromFile("Картинки/1.bmp");
}
//---------------------------------------------------------------------------
void draw1(){
      //рисование точек
```

```
      for(int i=0;i<j;i++){
            Form4->Image1->Canvas->Brush->Style = bsClear;
            Form4->Image1->Canvas->Font->Color=bsClear;
            if(i<9){
                  Form4->Image1->Canvas->TextOutA(mas1[i][0]-2,mas1[i][1]-16,i+1);
            }else{
                  Form4->Image1->Canvas->TextOutA(mas1[i][0]-5,mas1[i][1]-16,i+1);
            }
            Form4->Image1->Canvas->Brush->Color=clBlack;
            Form4->Image1->Canvas->Ellipse(mas1[i][0]-3,mas1[i][1]-3,mas1[i]
[0]+3,mas1[i][1]+3);
            Form4->Image1->Canvas->FloodFill(mas1[i][0],mas1[i]
[1],clBlack,fsSurface);
            Form4->Image1->Canvas->Pen->Mode=pmCopy;
      }
}
void drawrec1(){
      //перерисовка точек при перемещениии
      Form4->Image1->Canvas->Brush->Style = bsClear;
      PatBlt(Form4->Image1->Canvas->Handle,  0,  0,  Form4->Image1->Width,  Form4-
>Image1->Height, WHITENESS);
      Form4->Image1->Canvas->Draw(0,0,img);
      for(int i=0;i<j;i++){
            Form4->Image1->Canvas->Brush->Style = bsClear;
            Form4->Image1->Canvas->Font->Color=bsClear;
            if(i<9){
                  Form4->Image1->Canvas->TextOutA(mas1[i][0]-2,mas1[i][1]-16,i+1);
            }else{
                  Form4->Image1->Canvas->TextOutA(mas1[i][0]-5,mas1[i][1]-16,i+1);
            }
            Form4->Image1->Canvas->Brush->Color=clBlack;
            Form4->Image1->Canvas->Ellipse(mas1[i][0]-3,mas1[i][1]-3,mas1[i]
[0]+3,mas1[i][1]+3);
            Form4->Image1->Canvas->FloodFill(mas1[i][0],mas1[i]
[1],clBlack,fsSurface);
            Form4->Image1->Canvas->Pen->Mode=pmCopy;
      }
}

void __fastcall TForm4::FormActivate(TObject *Sender)
{
      Form4->Image1->Canvas->Brush->Style = bsClear;
      Form4->Image1->AutoSize=true;
      img->LoadFromFile(image_front);
      Form4->Image1->Picture->LoadFromFile(image_front);
      Yob=(Form4->Image1->Width)/2;
      sizex1=Form4->Image1->Width;
      sizey1=Form4->Image1->Height;
      draw1();
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button1Click(TObject *Sender)
{
      vis=1;
      if(Form8->Visible==true){
            Form8->Show();
      }
      Form5->Show();
      Form4->Visible=false;
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm4::Image1MouseDown(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
     if (j<length1) {
          x1=X;
          y1=Y;
          int k=j;
          int est=0;
          //добавление новой точки
          for (int i = 0; i <= k; i++) {
               if                  (((abs(x1-(mas1[i][0]+5))+abs(x1-(mas1[i][0]-
5))==abs((mas1[i][0]+5)-(mas1[i][0]-5))))    &&    (abs(y1-(mas1[i][1]+5))+abs(y1-
(mas1[i][1]-5))==abs((mas1[i][1]+5)-(mas1[i][1]-5)))){
                        Per=i;
                        est=1;
               }
          }
          if(est==0){
               mas1[j][0]=x1;
               mas1[j][1]=y1;
               j++;
          }
          draw1();
     }else{
          x1=X;
          y1=Y;
          Per=-1;
          for (int i = 0; i <= j; i++) {
               if                  (((abs(x1-(mas1[i][0]+5))+abs(x1-(mas1[i][0]-
5))==abs((mas1[i][0]+5)-(mas1[i][0]-5))))    &&    (abs(y1-(mas1[i][1]+5))+abs(y1-
(mas1[i][1]-5))==abs((mas1[i][1]+5)-(mas1[i][1]-5)))){
                        Per=i;
               }
          }
     }
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Image1MouseMove(TObject *Sender, TShiftState Shift,
      int X, int Y)
{
      if(X<Form4->Image1->Width && Y<Form4->Image1->Height && X>0 && Y>0){
      if(Shift==TShiftState()<<ssLeft){
      x2=X;
      y2=Y
      //перемещение уже поставленной точки
      if(schet==0){
          for (int i = 0; i <= j; i++){
                          if      (((abs(x1-(mas1[i][0]+5))+abs(x1-(mas1[i][0]-
5))==abs((mas1[i][0]+5)-(mas1[i][0]-5))))    &&    (abs(y1-(mas1[i][1]+5))+abs(y1-
(mas1[i][1]-5))==abs((mas1[i][1]+5)-(mas1[i][1]-5)))){
                        Per=i;
                        schet++;
                  }
          }
      }
          PatBlt(Form4->Image1->Canvas->Handle,  mas1[Per][0]-3,  mas1[Per][1]-3,
6, 6, WHITENESS);
               mas1[Per][0]=X;
               mas1[Per][1]=Y;
          drawrec1();
```

```
            perem=1;
      }
      schet=0;
      }
}
//---------------------------------------------------------------------------


void __fastcall TForm4::Image1MouseUp(TObject *Sender, TMouseButton Button,
        TShiftState Shift, int X, int Y)
{
      if(perem==1){perem=0;draw1();}
}
//---------------------------------------------------------------------------


void __fastcall TForm4::N4Click(TObject *Sender)
{
      if((Application->MessageBox("Вы действительно хотите выйти?\nВсе сохранённые
данные пропадут!",
                  "Сообщение",MB_YESNO | MB_ICONQUESTION))==IDYES){
            Form2->Close();
      }
}
//---------------------------------------------------------------------------

void __fastcall TForm4::FormCloseQuery(TObject *Sender, bool &CanClose)
{
      if((Application->MessageBox("Вы действительно хотите выйти?\nВсе сохранённые
данные пропадут!",
                  "Сообщение",MB_YESNO | MB_ICONQUESTION))==IDYES){
            Form2->Close();
            CanClose=true;
      }else CanClose=false;
}
//---------------------------------------------------------------------------

void __fastcall TForm4::N6Click(TObject *Sender)
{
      vis=0;
      Form8->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm4::N8Click(TObject *Sender)
{
      MessageBox(Handle, "Приложение маникен.\nАвтор: Худжанов Дмитрий.\nГруппа
ЕТ-484\nЮУрГУ,2017",
                  "О программе",MB_ICONASTERISK);
}
//---------------------------------------------------------------------------
```

Image_front.h – заголовочный файл.

```
//---------------------------------------------------------------------------
#ifndef Image_frontH
#define Image_frontH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
```

```
#include <Menus.hpp>
//---------------------------------------------------------------------------
class TForm4 : public TForm
{
__published:      // IDE-managed Components
      TImage *Image1;
      TImage *Image2;
      TButton *Button1;
      TLabel *Label1;
      TLabel *Label2;
      TMainMenu *MainMenu1;
      TMenuItem *N1;
      TMenuItem *N4;
      TMenuItem *N5;
      TMenuItem *N6;
      TMenuItem *N7;
      TMenuItem *N8;
      void __fastcall FormActivate(TObject *Sender);
      void __fastcall Button1Click(TObject *Sender);
      void __fastcall Image1MouseDown(TObject *Sender, TMouseButton Button,
            TShiftState Shift, int X, int Y);
      void __fastcall Image1MouseMove(TObject *Sender, TShiftState Shift, int X,
          int Y);
      void __fastcall Image1MouseUp(TObject *Sender, TMouseButton Button,
          TShiftState Shift, int X, int Y);
      void __fastcall N4Click(TObject *Sender);
      void __fastcall FormCloseQuery(TObject *Sender, bool &CanClose);
      void __fastcall N6Click(TObject *Sender);
      void __fastcall N8Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
      __fastcall TForm4(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm4 *Form4;
//---------------------------------------------------------------------------
#endif
```

Image_side.cpp – форма расстановки точек (вид сбоку).

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Image_side.h"
#include "Image_back.h"
#include "Image_front.h"
#include "Instruction.h"
#include "AddFhoto.h"
#include "Start.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
int perem=0;      //флаг перемещения точки
int j=0,          //текущее количество точек
      x1,y1,            //переменные для хранения координат точки
      schet=0,    //флаг попадания в точку
      x2,y2;            //переменные для хранения координат точки
int Per=-1;     //переменная для переноса точки
const int length2=35;  //количество точек на фигуре
int Xob,Zob,    //середина картинки по оси OX и OZ
      sizex2,sizey2;  //размер image с загруженной картинкой
```

```
int mas2[length2][2];  //массив точек
Graphics::TBitmap *img = new Graphics::TBitmap;
extern int vis;  //переменная хранящая расположение фигуры
//---------------------------------------------------------------------------
__fastcall TForm5::TForm5(TComponent* Owner)
      : TForm(Owner)
{
      Form5->Image2->Center= true;
      Form5->Image2->Picture->LoadFromFile("Картинки/2.bmp");
}
//---------------------------------------------------------------------------
void draw2(){
      //рисование точек
      for(int i=0;i<j;i++){
            Form5->Image1->Canvas->Brush->Style = bsClear;
            Form5->Image1->Canvas->Font->Color=bsClear;
            if(i<9){
                  Form5->Image1->Canvas->TextOutA(mas2[i][0]-2,mas2[i][1]-16,i+1);
            }else{
                  Form5->Image1->Canvas->TextOutA(mas2[i][0]-5,mas2[i][1]-16,i+1);
            }
            Form5->Image1->Canvas->Brush->Color=clBlack;
            Form5->Image1->Canvas->Ellipse(mas2[i][0]-3,mas2[i][1]-3,mas2[i]
[0]+3,mas2[i][1]+3);
            Form5->Image1->Canvas->FloodFill(mas2[i][0],mas2[i]
[1],clBlack,fsSurface);
            Form5->Image1->Canvas->Pen->Mode=pmCopy;
      }
}
//---------------------------------------------------------------------------
void drawrec2(){
      //перерисовка точек при перемещениии
      Form5->Image1->Canvas->Brush->Style = bsClear;
      PatBlt(Form5->Image1->Canvas->Handle,  0,  0,  Form5->Image1->Width,  Form5-
>Image1->Height, WHITENESS);
      Form5->Image1->Canvas->Draw(0,0,img);
      for(int i=0;i<j;i++){
            Form5->Image1->Canvas->Brush->Style = bsClear;
            Form5->Image1->Canvas->Font->Color=bsClear;
            if(i<9){
                  Form5->Image1->Canvas->TextOutA(mas2[i][0]-2,mas2[i][1]-16,i+1);
            }else{
                  Form5->Image1->Canvas->TextOutA(mas2[i][0]-5,mas2[i][1]-16,i+1);
            }
            Form5->Image1->Canvas->Brush->Color=clBlack;
            Form5->Image1->Canvas->Ellipse(mas2[i][0]-3,mas2[i][1]-3,mas2[i]
[0]+3,mas2[i][1]+3);
            Form5->Image1->Canvas->FloodFill(mas2[i][0],mas2[i]
[1],clBlack,fsSurface);
            Form5->Image1->Canvas->Pen->Mode=pmCopy;
      }
}
//---------------------------------------------------------------------------
void __fastcall TForm5::FormActivate(TObject *Sender)
{
      Form5->Image1->Canvas->Brush->Style = bsClear;
      Form5->Image1->AutoSize=true;
      img->LoadFromFile(image_side);
      Form5->Image1->Picture->LoadFromFile(image_side);
      Xob=(Form5->Image1->Width)/2;
      Zob=(Form5->Image1->Height)/2;
      sizex2=Form5->Image1->Width;
```

```
      sizey2=Form5->Image1->Height;
      draw2();
}
//---------------------------------------------------------------------------

void __fastcall TForm5::Button2Click(TObject *Sender)
{
      vis=0;
      if(Form8->Visible==true){
            Form8->Show();
      }
      Form4->Visible=true;
      Form5->Visible=false;
}
//---------------------------------------------------------------------------

void __fastcall TForm5::Button1Click(TObject *Sender)
{
      vis=2;
      if(Form8->Visible==true){
            Form8->Show();
      }
      Form6->Show();
      Form5->Visible=false;
}
//---------------------------------------------------------------------------


void __fastcall TForm5::Image1MouseDown(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
      if (j<length2) {
            x1=X;
            y1=Y;
            int k=j;
            int est=0;
            //добавление новой точки
            for (int i = 0; i <= k; i++) {
                  if                  (((abs(x1-(mas2[i][0]+5))+abs(x1-(mas2[i][0]-
5))==abs((mas2[i][0]+5)-(mas2[i][0]-5))))     &&     (abs(y1-(mas2[i][1]+5))+abs(y1-
(mas2[i][1]-5))==abs((mas2[i][1]+5)-(mas2[i][1]-5)))){
                        Per=i;
                        est=1;
                  }
            }
            if(est==0){
                  mas2[j][0]=x1;
                  mas2[j][1]=y1;
                  j++;
            }
            draw2();
      }else{
            x1=X;
            y1=Y;
            Per=-1;
            for (int i = 0; i <= j; i++) {
                  if                  (((abs(x1-(mas2[i][0]+5))+abs(x1-(mas2[i][0]-
5))==abs((mas2[i][0]+5)-(mas2[i][0]-5))))     &&     (abs(y1-(mas2[i][1]+5))+abs(y1-
(mas2[i][1]-5))==abs((mas2[i][1]+5)-(mas2[i][1]-5)))){
                        Per=i;
                  }
            }
```

```
      }
}
//---------------------------------------------------------------------------

void __fastcall TForm5::Image1MouseMove(TObject *Sender, TShiftState Shift,
       int X, int Y)
{
       if(X<Form5->Image1->Width && Y<Form5->Image1->Height && X>0 && Y>0){
       if(Shift==TShiftState()<<ssLeft){
       x2=X;
       y2=Y;
       //перемещение уже поставленной точки
       if(schet==0){
            for (int i = 0; i <= j; i++){
                              if      (((abs(x1-(mas2[i][0]+5))+abs(x1-(mas2[i][0]-
5))==abs((mas2[i][0]+5)-(mas2[i][0]-5))))     &&     (abs(y1-(mas2[i][1]+5))+abs(y1-
(mas2[i][1]-5))==abs((mas2[i][1]+5)-(mas2[i][1]-5)))){
                         Per=i;
                         schet++;
                    }
            }
       }
            PatBlt(Form5->Image1->Canvas->Handle,  mas2[Per][0]-3,  mas2[Per][1]-3,
6, 6, WHITENESS);
                  mas2[Per][0]=X;
                  mas2[Per][1]=Y;
            drawrec2();
            perem=1;
      }
      schet=0;
      }
}
//---------------------------------------------------------------------------


void __fastcall TForm5::Image1MouseUp(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
      if(perem==1){perem=0;draw2();}
}
//---------------------------------------------------------------------------

void __fastcall TForm5::N4Click(TObject *Sender)
{
      if((Application->MessageBox("Вы действительно хотите выйти?\nВсе сохранённые
данные пропадут!",
                  "Сообщение",MB_YESNO | MB_ICONQUESTION))==IDYES){
            Form1->Close();
      }
}
//---------------------------------------------------------------------------

void __fastcall TForm5::FormCloseQuery(TObject *Sender, bool &CanClose)
{
     if((Application->MessageBox("Вы действительно хотите выйти?\nВсе сохранённые
данные пропадут!",
                  "Сообщение",MB_YESNO | MB_ICONQUESTION))==IDYES){
            Form1->Close();
            CanClose=true;
      }else CanClose=false;
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm5::N6Click(TObject *Sender)
{
      Form8->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm5::N8Click(TObject *Sender)
{
      MessageBox(Handle, "Приложение маникен.\nАвтор: Худжанов Дмитрий.\nГруппа
ЕТ-484\nЮУрГУ,2017",
                  "О программе",MB_ICONASTERISK);
}
//---------------------------------------------------------------------------
```

Image_side.h – заголовочный файл.

```
//---------------------------------------------------------------------------
#ifndef Image_sideH
#define Image_sideH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Menus.hpp>
//---------------------------------------------------------------------------
class TForm5 : public TForm
{
__published:      // IDE-managed Components
      TImage *Image1;
      TImage *Image2;
      TButton *Button1;
      TLabel *Label1;
      TLabel *Label2;
      TButton *Button2;
      TMainMenu *MainMenu1;
      TMenuItem *N1;
      TMenuItem *N4;
      TMenuItem *N5;
      TMenuItem *N6;
      TMenuItem *N7;
      TMenuItem *N8;
      void __fastcall FormActivate(TObject *Sender);
      void __fastcall Button2Click(TObject *Sender);
      void __fastcall Button1Click(TObject *Sender);
      void __fastcall Image1MouseDown(TObject *Sender, TMouseButton Button,
          TShiftState Shift, int X, int Y);
      void __fastcall Image1MouseMove(TObject *Sender, TShiftState Shift, int X,
          int Y);
      void __fastcall Image1MouseUp(TObject *Sender, TMouseButton Button,
          TShiftState Shift, int X, int Y);
      void __fastcall N4Click(TObject *Sender);
      void __fastcall FormCloseQuery(TObject *Sender, bool &CanClose);
      void __fastcall N6Click(TObject *Sender);
      void __fastcall N8Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
      __fastcall TForm5(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm5 *Form5;
```

```
//---------------------------------------------------------------------------
#endif
```

Image_back.cpp – форма расстановки точек (вид сзади).

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Image_back.h"
#include "Image_side.h"
#include "Instruction.h"
#include "Model.h"
#include "AddFhoto.h"
#include "Start.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm6 *Form6;
int perem=0;      //флаг перемещения точки
int j=0,          //текущее количество точек
      x1,y1,            //переменные для хранения координат точки
      schet=0,    //флаг попадания в точку
      x2,y2;            //переменные для хранения координат точки
int Per=-1;     //переменная для переноса точки
const int length3=22;  //количество точек на фигуре
int mas3[length3][2];   //массив точек
int sizex3,sizey3;  //размер image с загруженной картинкой
extern int vis;     //переменная хранящая расположение фигуры
Graphics::TBitmap *img = new Graphics::TBitmap;
//---------------------------------------------------------------------------
__fastcall TForm6::TForm6(TComponent* Owner)
      : TForm(Owner)
{
      Form6->Image2->Center= true;
      Form6->Image2->Picture->LoadFromFile("Картинки/3.bmp");
}
//---------------------------------------------------------------------------
void draw3(){
      //рисование точек
      for(int i=0;i<j;i++){
            Form6->Image1->Canvas->Brush->Style = bsClear;
            Form6->Image1->Canvas->Font->Color=bsClear;
            if(i<9){
                  Form6->Image1->Canvas->TextOutA(mas3[i][0]-2,mas3[i][1]-16,i+1);
            }else{
                  Form6->Image1->Canvas->TextOutA(mas3[i][0]-5,mas3[i][1]-16,i+1);
            }
            Form6->Image1->Canvas->Brush->Color=clBlack;
            Form6->Image1->Canvas->Ellipse(mas3[i][0]-3,mas3[i][1]-3,mas3[i]
[0]+3,mas3[i][1]+3);
            Form6->Image1->Canvas->FloodFill(mas3[i][0],mas3[i]
[1],clBlack,fsSurface);
            Form6->Image1->Canvas->Pen->Mode=pmCopy;
      }
}
//---------------------------------------------------------------------------
void drawrec3(){
      //перерисовка точек при перемещениии
      Form6->Image1->Canvas->Brush->Style = bsClear;
      PatBlt(Form6->Image1->Canvas->Handle,  0,  0,  Form6->Image1->Width,  Form6-
>Image1->Height, WHITENESS);
      Form6->Image1->Canvas->Draw(0,0,img);
```

```
      for(int i=0;i<j;i++){
            Form6->Image1->Canvas->Brush->Style = bsClear;
            Form6->Image1->Canvas->Font->Color=bsClear;
            if(i<9){
                  Form6->Image1->Canvas->TextOutA(mas3[i][0]-2,mas3[i][1]-16,i+1);
            }else{
                  Form6->Image1->Canvas->TextOutA(mas3[i][0]-5,mas3[i][1]-16,i+1);
            }
            Form6->Image1->Canvas->Brush->Color=clBlack;
            Form6->Image1->Canvas->Ellipse(mas3[i][0]-3,mas3[i][1]-3,mas3[i]
[0]+3,mas3[i][1]+3);
            Form6->Image1->Canvas->FloodFill(mas3[i][0],mas3[i]
[1],clBlack,fsSurface);
            Form6->Image1->Canvas->Pen->Mode=pmCopy;
      }
}
//---------------------------------------------------------------------------
void __fastcall TForm6::FormActivate(TObject *Sender)
{
      Form6->Image1->Canvas->Brush->Style = bsClear;
      Form6->Image1->AutoSize=true;
      img->LoadFromFile(image_back);
      Form6->Image1->Picture->LoadFromFile(image_back);
      sizex3=Form6->Image1->Width;
      sizey3=Form6->Image1->Height;
      draw3();
}
//---------------------------------------------------------------------------


void __fastcall TForm6::Button1Click(TObject *Sender)
{
      Form3->Show();
      Form6->Visible=false;
      Form8->Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm6::Button2Click(TObject *Sender)
{
      vis=1;
      if(Form8->Visible==true){
            Form8->Show();
      }
      Form5->Visible=true;
      Form6->Visible=false;
}
//---------------------------------------------------------------------------

void __fastcall TForm6::Image1MouseDown(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
      if (j<length3) {
            x1=X;
            y1=Y;
            int k=j;
            int est=0;
            //добавление новой точки
            for (int i = 0; i <= k; i++) {
                  if                  (((abs(x1-(mas3[i][0]+5))+abs(x1-(mas3[i][0]-
5))==abs((mas3[i][0]+5)-(mas3[i][0]-5))))     &&     (abs(y1-(mas3[i][1]+5))+abs(y1-
(mas3[i][1]-5))==abs((mas3[i][1]+5)-(mas3[i][1]-5)))){
```

```
                        Per=i;
                        est=1;
                  }
            }
            if(est==0){
                  mas3[j][0]=x1;
                  mas3[j][1]=y1;
                  j++;
            }
            draw3();
      }else{
            x1=X;
            y1=Y;
            Per=-1;
            for (int i = 0; i <= j; i++) {
                  if                  (((abs(x1-(mas3[i][0]+5))+abs(x1-(mas3[i][0]-
5))==abs((mas3[i][0]+5)-(mas3[i][0]-5))))     &&     (abs(y1-(mas3[i][1]+5))+abs(y1-
(mas3[i][1]-5))==abs((mas3[i][1]+5)-(mas3[i][1]-5)))){
                        Per=i;
                  }
            }
      }
}
//---------------------------------------------------------------------------

void __fastcall TForm6::Image1MouseMove(TObject *Sender, TShiftState Shift,
      int X, int Y)
{
       if(X<Form6->Image1->Width && Y<Form6->Image1->Height && X>0 && Y>0){
       if(Shift==TShiftState()<<ssLeft){
       x2=X;
       y2=Y;
       //перемещение уже поставленной точки
       if(schet==0){
            for (int i = 0; i <= j; i++){
                              if      (((abs(x1-(mas3[i][0]+5))+abs(x1-(mas3[i][0]-
5))==abs((mas3[i][0]+5)-(mas3[i][0]-5))))     &&     (abs(y1-(mas3[i][1]+5))+abs(y1-
(mas3[i][1]-5))==abs((mas3[i][1]+5)-(mas3[i][1]-5)))){
                        Per=i;
                        schet++;
                     }
            }
       }
            PatBlt(Form6->Image1->Canvas->Handle,  mas3[Per][0]-3,  mas3[Per][1]-3,
6, 6, WHITENESS);
                  mas3[Per][0]=X;
                  mas3[Per][1]=Y;
            drawrec3();
            perem=1;
      }
      schet=0;
      }
}
//---------------------------------------------------------------------------

void __fastcall TForm6::Image1MouseUp(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
      if(perem==1){perem=0;draw3();}
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm6::FormCloseQuery(TObject *Sender, bool &CanClose)
{
      if((Application->MessageBox("Вы действительно хотите выйти?\nВсе сохранённые
данные пропадут!",
                  "Сообщение",MB_YESNO | MB_ICONQUESTION))==IDYES){
            Form1->Close();
            CanClose=true;
      }else CanClose=false;
}
//---------------------------------------------------------------------------

void __fastcall TForm6::N4Click(TObject *Sender)
{
      if((Application->MessageBox("Вы действительно хотите выйти?\nВсе сохранённые
данные пропадут!",
                  "Сообщение",MB_YESNO | MB_ICONQUESTION))==IDYES){
            Form1->Close();
      }
}
//---------------------------------------------------------------------------

void __fastcall TForm6::N6Click(TObject *Sender)
{
      Form8->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm6::N8Click(TObject *Sender)
{
      MessageBox(Handle, "Приложение маникен.\nАвтор: Худжанов Дмитрий.\nГруппа
ЕТ-484\nЮУрГУ,2017",
                  "О программе",MB_ICONASTERISK);
}
//---------------------------------------------------------------------------

void __fastcall TForm6::N2Click(TObject *Sender)
{
      if((Application->MessageBox("Вы действительно хотите выйти?\nВсе сохранённые
данные пропадут!",
                  "Сообщение",MB_YESNO | MB_ICONQUESTION))==IDYES){
            Form1->Close();
      }
}
//---------------------------------------------------------------------------
```

Image_back.h – заголовочный файл.

```
//---------------------------------------------------------------------------
#ifndef Image_backH
#define Image_backH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Menus.hpp>
//---------------------------------------------------------------------------
class TForm6 : public TForm
{
__published:      // IDE-managed Components
      TImage *Image1;
      TImage *Image2;
```

```
        TButton *Button1;
        TLabel *Label1;
        TLabel *Label2;
        TButton *Button2;
        TMainMenu *MainMenu1;
        TMenuItem *N1;
        TMenuItem *N2;
        TMenuItem *N5;
        TMenuItem *N6;
        TMenuItem *N7;
        TMenuItem *N8;
        void __fastcall FormActivate(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Image1MouseDown(TObject *Sender, TMouseButton Button,
            TShiftState Shift, int X, int Y);
        void __fastcall Image1MouseMove(TObject *Sender, TShiftState Shift, int X,
            int Y);
        void __fastcall Image1MouseUp(TObject *Sender, TMouseButton Button,
            TShiftState Shift, int X, int Y);
        void __fastcall FormCloseQuery(TObject *Sender, bool &CanClose);
        void __fastcall N4Click(TObject *Sender);
        void __fastcall N6Click(TObject *Sender);
        void __fastcall N8Click(TObject *Sender);
        void __fastcall N2Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm6(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm6 *Form6;
//---------------------------------------------------------------------------
#endif
```

Instruction.cpp – форма с инструкцией по расстановке точек.

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Instruction.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm8 *Form8;
extern int vis;
//---------------------------------------------------------------------------
__fastcall TForm8::TForm8(TComponent* Owner)
        : TForm(Owner)
{
}
void Label_Caption(){
   Form8->Label1->Font->Size=14;
   Form8->Label1->Font->Style = TFontStyles() << fsBold;
   Form8->Label1->Caption="Инструкция по расстановке точек ";
   //информация от точках вид спереди
   if (vis==0) {
      Form8->Width=537;
      Form8->Height=397;
      Form8->Label1->Caption=Form8->Label1->Caption+"(вид спереди):";
      Form8->Label2->Font->Size=12;
      Form8->Label2->Font->Style = TFontStyles();
      Form8->Label2->Caption="1 - верх головы\n";
```

```
  Form8->Label2->Caption=Form8->Label2->Caption+"2 - середина между бровями\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"3 - центр подборотка\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"4 - центр верха шеи\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"5 - центр низа шеи\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"6 - точка меджу рукой и грудью
  (подмышка)\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"7   -   самая   выпуклая   часть
  груди\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"8 - центр локтя\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"9   -   самая   выпуклая   часть
  живота\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"10 - центр запястья\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"11 - центр коленки\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"12 - центр щиколотки\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"13 - левый край талии\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"14   -   левый   верхний   угол
  купюры\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"15   -   левый   нижний   угол
  купюры\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"16   -   правый   верхний   угол
  купюры";
}
    //информация от точках вид сбоку
if (vis==1) {
  Form8->Width=537;
  Form8->Height=650;
  Form8->Label1->Caption=Form8->Label1->Caption+"(вид сбоку):";
  Form8->Label2->Font->Size=10;
  Form8->Label2->Font->Style = TFontStyles();
  Form8->Label2->Caption="1 - верх головы\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"2   -   самая   выпуклая   часть
  головы слева\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"3 - точка над ухом\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"4  -  самая  правая  точка  на
  уровне бровей\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"5   -   самая   правая   точка
  подбородка\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"6   -   правая   верхняя   точка
  шеи\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"7   -   правая   нижняя   точка
  шеи\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"8 - левая нижняя точка шеи\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"9 - верхняя точка плеча\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"10  -  самая  выпуклая  точка  на
  спине\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"11   -   самая   левая   точка
  подмышки\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"12   -   самая   правая   точка
  подмышки\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"13   -   самая   выпуклая   часть
  груди\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"14   -   самая   вогнутая   часть
  спины\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"15   -   самая   левая   точка
  локтя\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"16 - центр локтя\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"17   -   самая   правая   точка
  локтя\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"18   -   самая   выпуклая   точка
  живота\n";
  Form8->Label2->Caption=Form8->Label2->Caption+"19   -   самая   выпуклая   точка
  ягодиц\n";
```

```
      Form8->Label2->Caption=Form8->Label2->Caption+"20 - центр на уровне бёдер\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"21   -   самая   левая   точка
      запястья\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"22 - центр запястья\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"23   -   самая   правая   точка
      запястья\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"24 - нижняя точка ягодиц\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"25   -   самая   левая   точка
      колена\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"26 - центр колена\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"27   -   самая   правая   точка
      колена\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"28   -   самая   левая   точка
      щиколотки\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"29 - центр щиколотки\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"30   -   самая   правая   точка
      щиколотки\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"31   -   центр   нижней   части
      шеи\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"32 - бок на уровне живота\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"33   -   левый   верхний   угол
      купюры\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"34   -   левый   нижний   угол
      купюры\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"35   -   правый   верхний   угол
      купюры";
       }
//информация от точках вид сзади
  if (vis==2) {
      Form8->Width=600;
      Form8->Height=510;
      Form8->Label1->Caption=Form8->Label1->Caption+"(вид сзади):";
      Form8->Label2->Font->Size=12;
      Form8->Label2->Font->Style = TFontStyles();
      Form8->Label2->Caption="1 - центр головы\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"2  -  самая  правая  точка  над
      правым ухом\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"3 - центр нижней части шеи\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"4 - верхняя точка плеча\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"5   -   точка   соответствующая
      самой выпуклой точке на спине(виде сбоку)\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"6 - точка меджу рукой и грудью
      (подмышка)\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"7   -   точка   соответствующая
      самой вогнутой точке на спине(виде сбоку)\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"8 - центр локтя\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"9   -   самая   правая   точка
      локтя\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"10 - центр правой ягодицы\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"11  -  самая  правая  точка  на
      уровне бедра\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"12 - центр запястья\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"13   -   самая   правая   точка
      запястья\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"14  -  центральная  точка  под
      правой ягодицей\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"15 - центр колена\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"16   -   самая   правая   точка
      колена\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"17 - центр щиколотки\n";
      Form8->Label2->Caption=Form8->Label2->Caption+"18   -   самая   правая   точка
      щиколотки\n";
```

```
     Form8->Label2->Caption=Form8->Label2->Caption+"19 - самая правая точка нижней
     части шеи\n";
     Form8->Label2->Caption=Form8->Label2->Caption+"20   -   левый   верхний   угол
     купюры\n";
     Form8->Label2->Caption=Form8->Label2->Caption+"21   -   левый   нижний   угол
     купюры\n";
     Form8->Label2->Caption=Form8->Label2->Caption+"22   -   правый   верхний   угол
     купюры";
   }
}
//---------------------------------------------------------------------------

void __fastcall TForm8::FormActivate(TObject *Sender)
{
      Label_Caption();
}
//---------------------------------------------------------------------------
```

Instruction.h – заголовочный файл.

```
//---------------------------------------------------------------------------

#ifndef InstructionH
#define InstructionH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TForm8 : public TForm
{
__published:      // IDE-managed Components
      TLabel *Label1;
      TLabel *Label2;
      void __fastcall FormActivate(TObject *Sender);
private:    // User declarations
public:           // User declarations
      //void Label_Caption();
      __fastcall TForm8(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm8 *Form8;
//---------------------------------------------------------------------------
#endif
```