МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Факультет вычислительной математики и кибернетики**

**Кафедра: Центр прикладной информатики**

Направление подготовки: «Прикладная информатика»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

**Тема:**

**«Поиск геометрических ошибок на 2D чертежах»**

Выполнил:

Заведующий кафедрой: студент группы

доцент, к. ф-м. н. Шапошников Д.Е. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ученая степень, ученое звание, ф.и.о. ф.и.о.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись подпись

Научный руководитель:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ученая степень, ученое звание, ф.и.о.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Рецензент:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ученая степень, ученое звание, ф.и.о.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Нижний Новгород

2015

**Аннотация**

Выпускная квалификационная работа бакалавра «Поиск геометрических ошибок на 2D чертежах».

В данной работе рассматривается создание программного комплекса для обнаружения геометрических ошибок на 2D машиностроительных dxf-чертежах. Описываются возможные ошибки, алгоритмы их обнаружения, информационное и программное обеспечение комплекса, условия его функционирования, а также результаты, полученные с его помощью.

Оглавление

[Глоссарий 5](#_Toc421097289)

[Введение 6](#_Toc421097290)

[1. Теоритическая часть 8](#_Toc421097291)

[1.1. Обзор функциональных возможностей существующих САПР 8](#_Toc421097292)

[1.2. Существующие аналоги 10](#_Toc421097293)

[1.2.1. Autodesk Navisworks 10](#_Toc421097295)

[1.2.2. CorelCAD 10](#_Toc421097296)

[2. Постановка задачи 12](#_Toc421097297)

[3. Алгоритмическое обеспечение 13](#_Toc421097298)

[3.1. Пересечение двух отрезков 13](#_Toc421097299)

[3.2. Пересечение отрезка и окружности 13](#_Toc421097300)

[3.3. Пересечение отрезка и дуги 14](#_Toc421097301)

[3.4. Пересечение отрезка и эллипса 14](#_Toc421097302)

[3.5. Пересечение двух окружностей 15](#_Toc421097303)

[3.6. Пересечение окружности и дуги 18](#_Toc421097304)

[3.7. Пересечение дуг 18](#_Toc421097305)

[3.8. Совпадение двух отрезков 19](#_Toc421097306)

[4. Информационное обеспечение 20](#_Toc421097307)

[4.1. Базовые форматы хранения САПР 20](#_Toc421097308)

[4.1.1. Общая структура dxf-файла 21](#_Toc421097309)

[4.2. Информационное обеспечение комплекса 23](#_Toc421097310)

[4.2.1. Библиотека работы с dxf-файлами 23](#_Toc421097311)

[4.2.2. Библиотека работы с объектами 26](#_Toc421097312)

[4.2.3. Библиотека работы с графикой 27](#_Toc421097313)

[5. Программное обеспечение 28](#_Toc421097314)

[5.1. Условия функционирования программного обеспечения 28](#_Toc421097315)

[5.2. Разработанная библиотека работы с dxf-файлами 28](#_Toc421097316)

[5.2.1. Публичные функции 28](#_Toc421097317)

[5.2.2. Приватные функции 30](#_Toc421097318)

[5.3. Разработанная библиотека для работы с объектами 33](#_Toc421097319)

[5.3.1. Публичные функции 34](#_Toc421097320)

[5.3.2. Приватные функции 34](#_Toc421097321)

[5.4. Разработанная библиотека для работы с графикой 40](#_Toc421097322)

[5.4.1. Публичные функции 40](#_Toc421097323)

[5.5. Используемые сторонние библиотеки 44](#_Toc421097324)

[6. Руководство пользователя к разработанному программному продукту 45](#_Toc421097325)

[6.1. Интерфейс 45](#_Toc421097326)

[6.2. Описание клавишей 45](#_Toc421097327)

[6.3. Открытие файла 46](#_Toc421097328)

[6.4. Проверка файла 47](#_Toc421097329)

[6.5. Сохранение файла 50](#_Toc421097330)

[7. Взаимосвязь результатов работы комплекса с САПР КОМПАС-3D и AutoCAD 54](#_Toc421097331)

[7.1. Результат работы разработанной программы в САПР КОМПАС-3D 54](#_Toc421097332)

[7.2. Результат работы разработанной программы в САПР AutoCAD 56](#_Toc421097333)

[Заключение 59](#_Toc421097334)

[Литература 60](#_Toc421097335)

Глоссарий

Будем называть действительным пересечением двух объектов, пересечение, при котором у обоих объектов есть общая точка.

Будем называть мнимым пересечением двух объектов пересечение, если при продлении одного объекта у него образуется точка пересечения с другим объектом.

Две окружности совпадают друг с другом, если координаты их центров, а также радиусы равны.

Две дуги полностью совпадают друг с другом, если равны координаты их центров, радиусы, а также начальные и конечные углы.

Два эллипса совпадают друг с другом, если у них равны координаты центра, длины обоих осей и углы поворота.

Окружность и эллипс совпадают друг с другом, если координаты их центров и радиусы равны между собой.

Введение

Различные возможности и границы применения вычислительной техники для автоматизации проектирования определяются уровнем формализации научно-технических знаний в конкретной отрасли. Чем глубже разработана теория того или иного класса технических систем, тем большие возможности объективно существуют для автоматизации процесса их проектирования.

Применение ЭВМ при проектно-конструкторских работах в своем развитии прошло несколько стадий и претерпело значительные изменения. С появлением вычислительной техники был сделан акцент на автоматизацию проектных задач, имеющих четко выраженный расчетный характер, когда реализовывались методики, ориентированные на ручное проектирование. Затем, по мере накопления опыта, стали создавать программы автоматизированных расчетов на основе методов вычислительной математики (параметрическая оптимизация, метод конечных элементов и т. п.). С внедрением специализированных терминальных устройств появляются универсальные программы для ЭВМ для решения как расчетных, так и некоторых рутинных проектных задач (изготовление чертежей, спецификаций, текстовых документов и т. п.).

Решение проблем автоматизации проектирования с помощью ЭВМ основывается на системном подходе, т. е. на создании и внедрении САПР – систем автоматизированного проектирования технических объектов, которые решают весь комплекс задач от анализа задания до разработки полного объема конструкторской и технологической документации. Это достигается за счет объединения современных технических средств и математического обеспечения, параметры и характеристики которых выбираются с максимальным учетом особенностей задач проектно-конструкторского процесса. Примерами таких систем могут служить AutoCAD, КОМПАС-3D, CorelCAD. Однако следует заметить, что основной целью указанных САПР является создание конструкторских документов, но важной частью разработки является и анализ готовых машиностроительных чертежей, в том числе и поиск ошибок. Указанные САПР не имеют возможности анализировать чертеж на предмет ошибок, что является существенным недостатком этих систем, так как ошибка в чертеже – это неверно спроектированный объект. Не стоит забывать о том, что многие конструкторские документы, в том числе и чертежи, создаются вручную с помощью кульмана. Проанализировать данный чертеж на предмет ошибок достаточно проблематично, так же, как и оцифрованные с помощью сканера. Важной областью разработки машиностроительной документации является хранение.

В последнее время наметилась тенденция на переход к электронным архивам хранения. Следовательно, проблема обнаружения ошибок на чертежах, так же как и занесения в электронный архив является актуальной.

Из всего вышесказанного можно сделать следующий вывод, что целью данной работы является разработка информационного, алгоритмического и программного обеспечения подсистемы оценки качества формируемых документов типа машиностроительный чертёж.

1. Теоритическая часть
   1. Обзор функциональных возможностей существующих САПР

AutoCAD – двух- и трёхмерная система автоматизированного проектирования и черчения, разработанная компанией Autodesk. Первая версия системы была выпущена в 1982 году.[3]

Функциональные возможности последних версий программы:

* Инструменты работы с произвольными формами позволяющими создавать и анализировать сложные трёхмерные объекты;
* 3D-печать;
* Использование динамических блоков;
* Масштабирование аннотативных объектов на видовых экранах или в пространстве модели;
* Запись операций в виде макроса;
* Диспетчер подшивок для организации листов чертежей;
* Инструменты упрощённой трёхмерной навигации;
* Инструмент «аниматор движения»;
* Поддержка возможности настройки интерфейса под потребности конкретной отрасли;
* Возможность динамической связи чертежа с реальными картографическими данными (GeoLocation API);
* Механизм внешних ссылок (XRef), позволяющий разбивать чертеж на составные файлы;
* Высококачественная визуализация моделей с помощью системы [рендеринга](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BD%D0%B4%D0%B5%D1%80%D0%B8%D0%BD%D0%B3) [mental ray](http://ru.wikipedia.org/wiki/Mental_ray);
* Включает в себя полный набор инструментов для комплексного трёхмерного моделирования (поддерживается твердотельное, поверхностное и полигональное моделирование).

Таким образом, каких-либо средств обнаружения ошибок на чертежах в AutoCAD не имеется.

КОМПАС-3D – система трехмерного твердотельного моделирования, универсальная система автоматизированного проектирования.[4] Система КОМПАС-3D предназначена для создания трехмерных ассоциативных моделей отдельных деталей и сборочных единиц, содержащих как оригинальные, так и стандартизованные конструктивные элементы. Система позволяет реализовать классический процесс трехмерного параметрического проектирования от идеи к ассоциативной объемной модели, от модели к конструкторской документации.

Базовый функционал системы включает в себя:

* Средства работы над проектами, включающими подсбороки, деталей и стандартных изделий;
* Функционал моделирования деталей из листового материала: команды создания листового тела, сгибов, отверстий, жалюзи, буртиков, штамповок и вырезов в листовом теле, замыкания углов и т.д., а также выполнения развертки полученного листового тела (в том числе формирования ассоциативного чертежа развертки);
* Средства создания поверхностей;
* Инструменты создания пользовательских параметрических библиотек типовых элементов;
* Возможность получения конструкторской и технологической документации: встроенная система КОМПАС-График позволяет выпускать чертежи, спецификации, схемы, таблицы, текстовые документы;
* Возможность простановки размеров и обозначений в трехмерных моделях (поддержка стандарта ГОСТ 2.052-2006 «ЕСКД. Электронная модель изделия»);
* Поддержка стандарта Unicode;
* Средства интеграции с различными CAD/CAM/CAE системами;
* Средства защиты пользовательских данных, интеллектуальной собственности и сведений, составляющих коммерческую и государственную тайну (реализовано отдельным программным модулем КОМПАС-Защита);
* Система КОМПАС-График, позволяющая в скоростном режиме выпускать чертежи изделий, схемы, спецификации, различные текстовые документы, таблицы, инструкции и прочие документы.

Система изначально ориентирована на полную поддержку стандартов ЕСКД. При этом она обладает возможностью гибкой настройки на стандарты предприятия.

Таким образом, каких-либо средств обнаружения ошибок на чертежах в КОМПАС-3D не имеется.

* 1. Существующие аналоги

1. * 1. Autodesk Navisworks

Программа Autodesk Navisworks служит для экспертизы архитектурно-строительных проектов.[14] С её помощью можно координировать выполняемые работы, моделировать процесс строительства и проводить комплексный анализ. В данной программе имеются инструменты моделирования и оптимизации строительных графиков, выявления коллизий и пересечений, совместной работы (посредством добавления комментариев) и обнаружения потенциальных проблем, выявления конфликтов. Имеется поддержка приложений сторонних разработчиков.

В Autodesk Navisworks используются собственные форматы файлов – .nwd, .nwf, .nwc; имеется поддержка приложений AutoCAD, Revit, 3ds Max и продуктов на их основе, а также многих из широко используемых форматов в области САПР.

Autodesk Navisworks устанавливается отдельно от AutoCAD.

Данная программа платная, но имеются демо-версии (30 дней) и версия для студентов (бесплатная лицензия, действительная в течение 3 лет).

Данный аналог удовлетворяет поставленным требованиям, однако распространяется на платной основе.

* + 1. CorelCAD

CorelCAD – это эффективное, высокопроизводительное и экономичное решение САПР для выполнения повседневных работ по проектированию, требующих высокой точности и тщательной проработки деталей.[7]

Данная программа позволяет проверить чертёж и выявить нарушения отраслевых, корпоративных и проектных стандартов. Команда «Проверка стандартов» инспектирует следующие параметры на соответствие стандартам наименований из файла чертёжных стандартов:

* Свойства слоя;
* Свойства линий;
* Размер;
* Текст.

При нахождении ошибки в командном окне появляется ее описание. Данная команда создает ASCII файл с расширением .adt, который содержит отчет об обнаруженных ошибках. Файл .adt находится в одной папке с чертежом.[12]

Имеется возможность заменить несоответствующие стандартам элементы.

Данная функция интегрирована в среду проектирования и не требует установки дополнительных компонент.

Программа CorelCAD распространяется на платной основе. Также имеется 30-ти дневная ознакомительная версия.

Данный аналог обнаруживает ошибки только касательно стандартов, что не соответствует поставленным целям.

Из всех рассмотренных программных продуктов, наиболее удовлетворяющим заданным требованиям является Autodesk Navisworks. Однако данный продукт распространяется на платной основе. Таким образом, рассматриваемая проблема поиска ошибок в строительных (и не только) чертежах на данный момент и в будущем будет актуальна в связи с всё большим распространением электронных средств проектирования и хранения чертежей.

1. Постановка задачи

Разработать информационное, алгоритмическое и программное обеспечение автоматизированной подсистемы нахождения геометрических ошибок следующих типов:

* Пересечение двух отрезков;
* Пересечение отрезка и окружности;
* Пересечение отрезка и дуги;
* Пересечение отрезка и эллипса;
* Пересечение двух окружностей;
* Пересечение окружности и дуги;
* Пересечение дуг;
* Совпадение двух отрезков;
* Совпадение двух окружностей;
* Совпадение двух дуг;
* Совпадение двух эллипсов;
* Совпадение окружности и дуги;

на машиностроительных 2D-чертежах, представленных в DXF-формате версии AutoCAD 2007.

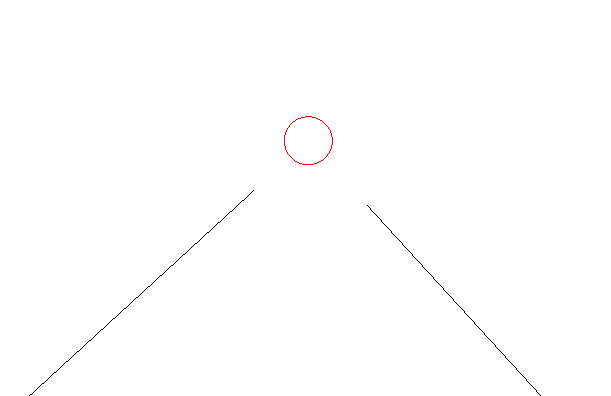


Рисунок . Пример ошибки

1. Алгоритмическое обеспечение
   1. Пересечение двух отрезков

|  |  |
| --- | --- |
| D:\Диплом\p1.png  Рисунок . Пересечение двух отрезков | Уравнения отрезков имеют вид:  (1)  (2)  Так как и являются одной и той же точкой, то уравнения (1) и (2) равны, из чего можно получить следующую систему:  (3)  Решая её относительно и получим: |

(4)

(5)

Подставляя полученные значения в начальное уравнение, получим координаты точки пересечения:

(6)

(7)

Если и , то точка принадлежит и первому и второму отрезку, значит это действительное пересечение отрезков, иначе пересечение будет мнимым.

* 1. Пересечение отрезка и окружности

|  |  |
| --- | --- |
| D:\Диплом\p2.png  Рисунок . Пересечение отрезка и окружности | Для нахождения точек пересечения отрезка и окружности нужно решить систему: |

Преобразуем уравнение (8) к уравнению вида

(10)

где

(11, 12)

Подставив полученное уравнение в (9) получим координаты и точки пересечения.

Если и , то пересечение будет действительным, иначе оно мнимое, т. к. точка лежит вне отрезка.

* 1. Пересечение отрезка и дуги

|  |  |
| --- | --- |
| D:\Диплом\p3.png  Рисунок . Пересечение отрезка и дуги | Поиск точек пересечения отрезка и эллипса осуществляется точно так же как и при поиске пересечений отрезка и окружности.  Далее находим угол между прямой, проходящей через точку и параллельной оси абсцисс, и прямой, проходящей через точки и . |

Если , то эта точка является точкой пересечения отрезка и эллипса.

Если и , то пересечение будет действительным, иначе оно мнимое, т. к. точка лежит вне отрезка.

* 1. Пересечение отрезка и эллипса

Пусть изначально имеем следующее расположение отрезка и эллипса:

|  |  |
| --- | --- |
| D:\Диплом\p5_1.png  Рисунок . Пересечение отрезка и эллипса | Для упрощения вычислений выполним сдвиг системы координат так, чтобы центр эллипса оказался в точке .  И выполним поворот системы координат так, чтобы большая ось эллипса лежала на оси абсцисс и меньшая ось лежала на оси ординат.  Таким образом, у точек A и B координаты будут равны и соответственно. |

И получим систему:

Преобразуем уравнение (13) к виду:

, (15)

где

(16)

(17)

И уравнение (14) к виду:

(18)

Подстановкой (15) в (18) получим:

(19)

(20)

Найдя корни уравнения (20) и подставив их в (15), получим координаты точек пересечения отрезка и эллипса и .



Рисунок . Пересечение отрезка и эллипса с найденными точками пересечения

Далее обратным преобразованием координат (поворот на угол и переноса центра координат на и ) получаем координаты точек пересечения в исходной системе координат и соответственно.

* 1. Пересечение двух окружностей

Рассматриваются 3 случая взаимного расположения двух окружностей:

* Окружности не пересекаются;
* Окружности пересекаются (1 или 2 точки пересечения);
* Одна окружность находится внутри другой окружности.

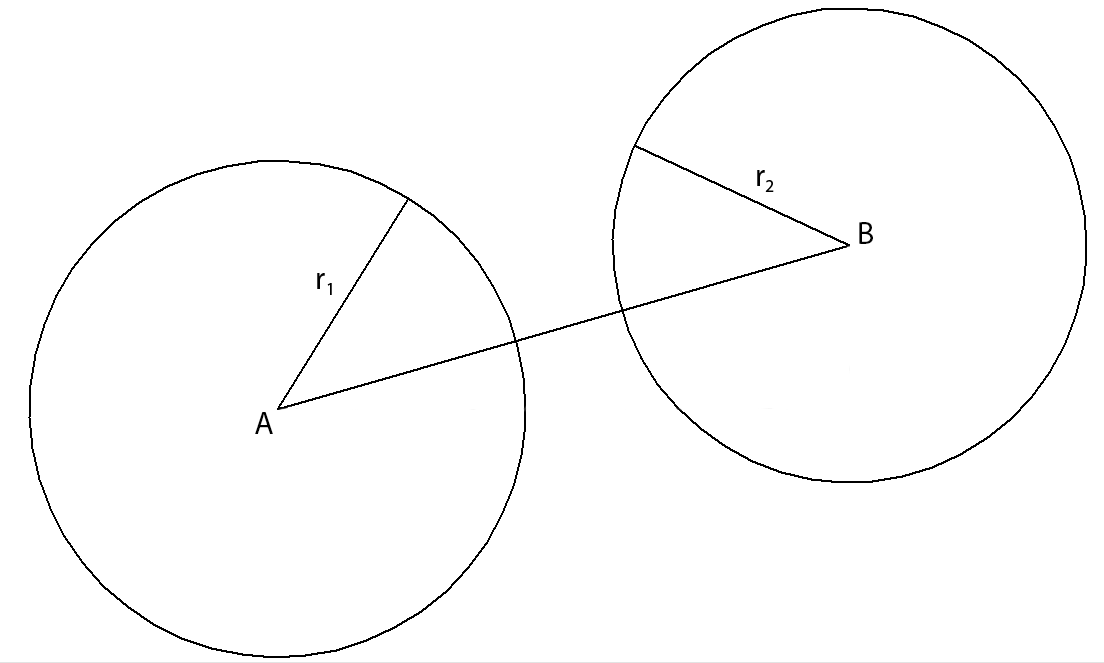


Рисунок . Расстояние между двумя окружностями

Расстояние между центрами двух окружностей определяется по формуле

(21)

|  |  |
| --- | --- |
| D:\Диплом\CircleCircle4.png  Рисунок . Окружности не имеют общих точек | Две окружности не будут иметь общих точек, если расстояние между их центрами больше суммы радиусов.  (22) |
| D:\Диплом\CircleCircle5.png  Рисунок . Окружности имеют одну общую точку | Две окружности будут иметь общую точку, если расстояние между их центрами равно сумме радиусов.  (23) |

|  |  |
| --- | --- |
| D:\Диплом\CircleCircle3.png  Рисунок . Окружности имеют две общих точки | Две окружности будут иметь две общих точки, если расстояние между их центрами меньше суммы радиусов.  (24) |

В случае, когда окружности имеют одну общую точку, координата x рассчитывается по формуле

(25)

А координата y рассчитывается по формуле

(26)

В случае, когда окружности имеют две общие точки, координаты точек пересечения рассчитываются следующим образом.

AC можно найти, выразив высоту треугольника BCD через формулы для нахождения площади треугольника:

(27)

,(28)

где

, (29)

т.е.

. (30)

Отрезок AB можно найти из прямоугольного треугольника ABC, т.е.

. (31)

Из подобия треугольников BDF и ABE находим AE

(32)

Соответственно

(33)

Таким образом, координаты точки A будут равны:

(34)

(35)

Координаты точек пересечения находятся следующим образом:

(36)

(37)

* 1. Пересечение окружности и дуги

|  |  |
| --- | --- |
| D:\Диплом\CircleArc.png  Рисунок . Пересечение окружности и дуги | Поиск точек пересечения окружности и эллипса осуществляется точно так же, как и при поиске пересечений двух окружностей.  Далее находим угол между прямой, проходящей через центр дуги и параллельной оси абсцисс, и прямой, проходящей через точки и . |

Если , то эта точка является точкой пересечения отрезка и эллипса. Аналогично нужно поступить с точкой .

* 1. Пересечение дуг

|  |  |
| --- | --- |
| D:\Диплом\ArcArc.png  Рисунок . Пересечение двух дуг | Поиск точек пересечения двух дуг осуществляется точно так же, как и при поиске пересечений двух окружностей.  Затем необходимо определить, принадлежат ли найденные точки обоим дугам. Для этого нужно найти угол между прямой, параллельной оси абсцисс и проходящей через |

точку , и прямой, проходящей через точки и ; и угол , между прямой, параллельной оси абсцисс и проходящей через точку , и прямой, проходящей через точки и .

Если и , то найденная точка является точкой пересечения двух дуг.

Аналогичные действия выполняются для точки .

* 1. Совпадение двух отрезков

Два отрезка полностью совпадают друг с другом, если их начальные и конечные координаты попарно равны или если начальные координаты первого отрезка равны конечным координатам второго отрезка и конечные координаты первого отрезка равны начальным координатам второго отрезка.

Два отрезка частично совпадают друг с другом, если:

* ;
* ;
* Точки C и D лежат на прямой AB.

1. Информационное обеспечение
   1. Базовые форматы хранения САПР

**DWF** – формат файла [8], развитый Autodesk для того, чтобы представить данные проекта в виде, который был бы независим от оригинального прикладного программного обеспечения, аппаратных средств, или операционной системы, с помощью которой создавались данные проекта. Файл DWF может описать данные проекта, содержащие любую комбинацию текста, графики, и изображений в независимом устройстве. Эти файлы могут быть одним листом или многократными листами, очень простыми или чрезвычайно сложными с богатым использованием шрифтов, графики, цвета, и изображений.

Файлы DWF призваны не для замены родных форматов CAD, таких как рисунки AutoCAD (DWG). Единственная цель DWF состоит в том, чтобы позволить проектировщикам, инженерам и их коллегам сообщать информацию проекта и содержание проекта к любому человеку, которому необходимо рассмотреть или напечатать информацию проекта – без этих участников команды, знающих AutoCAD или другое программное обеспечение для проектирования.

Технология DWF файлов базируется на трех компонентах:

* С++ библиотеки для разработки;
* программа для просмотра DWF файлов, которая предоставляет пользователям без знания AutoCAD просматривать файлы этого формата;
* программа для создания DWF файлов, позволяющая пользователям создавать файлы этого формата.

Кроме того, DWF основан на другим стандартах, таких как ZLIB, XML и общие форматы изображения.

Файлы DWF (начиная с версии 6.0) являются ZIP контейнерами для файлов рисунка; несмотря на малое количество первых байтов файла, содержащего заголовок DWF, переименование \*.dwf файл в \*.zip, позволит составляющим файлам внутри просматриваться с помощью программного обеспечения, таким как Winzip.

DWF может соединяться с .NET Библиотеками.

**DWG** – это универсальный формат для хранения данных 2D- и 3D-чертежей [9], название которого происходит от английского слова Drawing–чертеж. DWG – бинарный формат файла, используемый для хранения двухмерных (2D) и трёхмерных (3D) проектных данных и метаданных. Является основным форматом для некоторых САПР-программ. Формат DWG поддерживается многими САПР-приложениями косвенно: то есть данные из одного формата данных перемещаются в другой через функции импорт-экспорт. Форматы .dws («drawing standards» – стандарты чертежа), .dwt («drawing template» – шаблон чертежа) также являются форматом DWG. Современные программы, работающие в dwg, часто создают на диске временные и резервные копии документов в формате dwg в файлах с расширениями .sv$ («temporary automatic save» – временное автоматическое сохранение) и .bak («backup» – резервная копия).

**DXF –** открытый формат файлов для обмена графической информацией между приложениями САПР. Был создан фирмой Autodesk для системы AutoCAD. Поддерживается практически всеми CAD-системами на платформе PC.

DXF был впервые представлен в декабре 1982 года как часть AutoCAD 1.0, в качестве обменного формата данных, предоставляющего ту же информацию, что и закрытый внутренний формат AutoCAD – DWG, спецификация на который никогда не предоставлялась. В настоящее время на сайте Autodesk можно найти спецификации всех версий DXF, начиная с AutoCAD Release 13 (ноябрь 1994 г.) по AutoCAD 2013 (2012 г.). Начиная с AutoCAD Release 10 (октябрь 1988 г.) помимо текстового варианта DXF, поддерживается также и двоичная версия – DXB.

Однако, для большинства практических нужд вновь вводимые объекты, такие как 3D-расширения, не являются необходимыми. Например, в соответствии с требованием ЕСКД, чертёж любого изделия является двумерным контурным изображением (ГОСТ 2.301-68, Форматы). Поэтому DXF не только не исчез, но стал де-факто одним из двух стандартов для векторных изображений в открытых операционных системах и приложениях (второй стандарт – SVG).

* + 1. Общая структура dxf-файла

Файл обмена чертежами представляет собой обычный текстовый файл типа "DXF" в кодах ASCII, в котором находится текстовая информация в специально заданном формате. Рассматриваются dxf-файлы версии AutoCAD 2007. Файл DXF организован следующим образом:[15]

1. Раздел ЗАГОЛОВКА /HEADER/ – в данном разделе файла DXF содержится общая информация о чертеже. Каждый параметр имеет имя переменной и соответствующее ей значение.

2. Раздел ТАБЛИЦ /TABLES/ – в данном разделе содержатся определения именованных элементов.

* Таблица типов линий (LTYPE)
* Таблица слоев
* Таблица типов шрифтов
* Таблица видов

3. Раздел БЛОКОВ /BLOCKS/ – В данном разделе содержатся графические примитивы определений блоков, которые описывают примитивы, входящие в состав каждого блока изображения.

4. Раздел ПРИМИТИВОВ /ENTITIES/ – В данном разделе содержатся графические примитивы чертежа, включая любые ссылки на блоки.

5. КОНЕЦ ФАЙЛА.

Если вы используете опцию «Entities» команды DXFOUT, выходной DXF файл будет содержать только секции ПРИМИТИВЫ и КОНЕЦ ФАЙЛА, а раздел ПРИМИТИВЫ будет отражать только выбранные вами для вывода примитивы.

Файл DXF состоит из множества групп, каждая из которых занимает две строки в файле DXF. В первой строке размещается код группы, который представляет собой положительное ненулевое целое число, имеющее формат «I3» языка ФОРТРАН (то есть выровненное вправо число, заполненное пробелами в трехсимвольном поле).

Вторая строка группы представляет собой значение группы, имеющее формат, который зависит от типа группы, задаваемого кодом группы.

Присвоение кодов групп зависит от того, какой элемент описывается в файле. Тип значения, которое дает эта группа, определяется из кода группы следующим образом:

* 0 – 9 – строковое;
* 10 – 59 – с плавающей запятой;
* 60 – 79 – целое.

Таким образом, программа может легко прочитать значение, соответствующее коду группы, не имея информации о конкретном использовании этой группы в элементе файла. Форма представления значений в файле DXF не зависит от установки параметров в команде UNITS: координаты всегда представляются в виде десятичных чисел (или если они очень большие, то в виде научных обозначений), а углы всегда представляются в виде десятичных градусов, причем отсчет ведется от направления на восток от начала координат.  
 Переменные, точки входа в элементы таблицы и графические примитивы описываются с помощью группы, которая представляет элемент, задавая его тип и/или имя, после чего следуют многочисленные группы, которые представляют собой значения, связанные с этим элементом. Кроме того, для разделителей файлов, таких как метки конца и начала разделов, таблиц и самих файлов, используются специальные группы.

Графические примитивы, точки входа в таблицы и разделители файлов всегда вводятся с помощью группы 0, за которой следует имя, описывающее элемент.

* 1. Информационное обеспечение комплекса
     1. Библиотека работы с dxf-файлами

Ниже перечислены используемые структуры данных в рассматриваемой библиотеке.

struct POINT {

double x, y; // Координаты точки.

string layer; // Название слоя, на котором расположена точка.

};

struct LINE {

POINT p[2]; // Массив, состоящий из двух элементов, которые содержат координаты конечных точек отрезка.

unsigned short int type; // Тип линии. Содержит следующие значения: 0 – обычный отрезок, 1 – направляющая прямая.

string layer; // Название слоя, на котором расположен отрезок.

bool current; // Содержит следующие значения: true, если объект выделен, и false в противном случае.

};

struct LWPOLYLINE {

vector<POINT> p; // Вектор, содержащий координаты точек, из которых состоит полилиния.

string layer; // Название слоя, на котором расположена полилиния.

bool current; // Содержит следующие значения: true, если объект выделен, и false в противном случае.

bool closed; // Содержит следующие значения: true, если полилиния замкнута, и false в противном случае.

double width; // Толщина полилинии.

};

struct CIRCLE {

POINT p; // Точка, содержащая координаты центра окружности.

double r; // Радиус окружности.

string layer; // Название слоя, на котором расположена окружность.

bool current; // Содержит следующие значения: true, если объект выделен, и false в противном случае.

};

struct ELLIPSE {

POINT p; // Точка, содержащая координаты центра эллипса.

double ratio; // Отношение длинны наибольшей оси к наименьшей.

double width; // Длина наибольшей полуоси.

double height; // Длина наименьшей полуоси.

double angle; // Угол поворота большей оси относительно OX.

string layer; // Название слоя, на котором расположен эллипс.

};

struct ARC {

POINT p; // Точка, содержащая координаты центра дуги.

double r; // Диаметр дуги.

double angleStart; // Начальный угол дуги.

double angleEnd; // Конечный угол дуги.

string layer; // Название слоя, на котором расположена дуга.

bool current; // Содержит следующие значения: true, если объект выделен, и false в противном случае.

};

struct BLOCK {

int beginAddr; // Позиция начала блока в файле.

string number; // Шестнадцатеричный номер блока.

};

struct INSERT {

string layer; // Название слоя, на котором расположен блок для вставки.

string blockNumber; // Номер блока для вставки.

};

struct ENTITIES {

vector<POINT> Points; // Вектор точек, входящих в чертёж.

vector<LINE> Lines; // Вектор отрезков, входящих в чертёж.

vector<LWPOLYLINE> Polylines; // Вектор полилиний, входящих в чертёж.

vector<CIRCLE> Circles; // Вектор окружностей, входящих в чертёж.

vector<ELLIPSE> Ellipses; // Вектор эллипсов, входящих в чертёж.

vector<ARC> Arcs; // Вектор дуг, входящих в чертёж.

vector<INSERT> Inserts; // Вектор блоков для вставки, входящих в чертёж.

};

struct SECTION {

ENTITIES Entities; // Структура, в которой содержатся объекты, входящие в секцию ENTITIES чертежа.

};

struct CROSSPOINT {

double x, y; // Координаты точки пересечения.

bool type; // Содержит следующие значения: true, если пересечение действительное, и false в противном случае.

bool current; // Содержит следующие значения: true, если объект выделен, и false в противном случае.

bool isCrossPoint; // Содержит следующие значения: true, если объект является точкой печесечения двух других объектов, и false, если объект является наложением друг на друга двух других объектов.

vector<LINE> Lines; // Вектор, содержащий два пересекающихся или накладывающихся друг на друга отрезка.

vector<CIRCLE> Circles; // Вектор, содержащий две пересекающиеся или накладывающиеся друг на друга окружности.

vector<ELLIPSE> Ellipses; // Вектор, содержащий два пересекающихся или накладывающихся друг на друга эллипса.

vector<ARC> Arcs; // Вектор, содержащий две пересекающиеся или накладывающиеся друг на друга дуги.

};

fstream file – поток файла для чтения.

string fileName – имя файла для чтения.

ofstream saveFile – поток файла для записи.

string saveFileName – имя файла для записи.

int LastNumberOfObject – последний порядковый номер объектов.

int EndEntitiesPointer – позиция конца блока ENTITIES в исходном файле.

SECTION \_Section – структура для записи объектов.

vector<BLOCK> Blocks – вектор объектов типа BLOCK, входящих в чертёж.

int NumBlock – число объектов типа BLOCK, входящих в чертёж.

* + 1. Библиотека работы с объектами

Ниже перечислены используемые структуры данных в рассматриваемой библиотеке.

struct BITSFIELD {

unsigned char crossKind; // Виды пересечений, которые должны быть обработаны. Может содержать следующие значения: 0 - все пересечения, 1 - только действительные, 2 - только мнимые.

bool line\_line; // Содержит true, если необходимо проверить пересечение двух отрезков.

bool line\_circle; // Содержит true, если необходимо проверить пересечение отрезка и окружности.

bool line\_ellipse; // Содержит true, если необходимо проверить пересечение отрезка и эллипса.

bool line\_arc; // Содержит true, если необходимо проверить пересечение отрезка и дуги.

bool circle\_circle; // Содержит true, если необходимо проверить пересечение двух окружностей.

bool circle\_ellipse; // Содержит true, если необходимо проверить пересечение окружности и эллипса.

bool circle\_arc; // Содержит true, если необходимо проверить пересечение окружности и дуги.

bool ellipse\_ellipse; // Содержит true, если необходимо проверить пересечение двух эллипсов.

bool ellipse\_arc; // Содержит true, если необходимо проверить пересечение эллипса и дуги.

bool arc\_arc; // Содержит true, если необходимо проверить пересечение двух дуг.

bool overlap\_lines; // Содержит true, если необходимо проверить совпадение двух отрезков.

bool overlap\_circles; // Содержит true, если необходимо проверить совпадение двух окружностей.

bool overlap\_ellipses; // Содержит true, если необходимо проверить совпадение двух эллипсов.

bool overlap\_arcs; // Содержит true, если необходимо проверить совпадение двух дуг.

bool overlap\_circle\_arc; // Содержит true, если необходимо проверить совпадение окружности и дуги.

};

SECTION Section – структура, в которой содержатся ранее считанные из dxf-файла объекты чертежа.

vector<CROSSPOINT> ErrPoints – вектор, содержащий точки пересечений и совпадений объектов.

* + 1. Библиотека работы с графикой

Ниже перечислены используемые структуры данных в рассматриваемой библиотеке.

SimpleOpenGlControl^ OpenGlControl – элемент Windows Forms, позволяющий работать с графикой посредством OpenGL.

bool IsInit – переменная, содержащая true, если была выполнена инициализация OpenGlControl, и false в противном случае.

1. Программное обеспечение
   1. Условия функционирования программного обеспечения

Операционная система: OS Windows 7 64bit с архитектурой процессора x64.

Для использования программы необходимы процессор 64-разрядный (x64) процессор (с 1 или более ядер) с тактовой частотой 1 гигагерц (ГГц) или выше, 2 ГБ оперативной памяти (ОЗУ), монитор (рекомендуемое разрешение не ниже 1024х768px), 104-клавишная клавиатура, мышь, установленные библиотеки OpenGL не ниже версии 2.0, видеопроцессор Intel GMA HD или лучше.

Программное обеспечение было разработано на языках C++/CLI и C# в среде Microsoft Visual Studio 2012.

* 1. Разработанная библиотека работы с dxf-файлами

Данная библиотека служит для работы с файлами формата dxf (открытие, чтение, сохранение и закрытие файла).

* + 1. Публичные функции

Функции, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемой библиотекой.

**bool Open(string dxfFileName)**

Функция открытия файла по указанному имени и пути к файлу.

**Входные и выходные параметры.**

Входной параметр string dxfFileName – имя файла без указания пути к нему, если он находится в той же папке, что и программа. Если же программа и файл находятся в разных папках, то необходимо указать полный путь к файлу.

Выходной параметр имеет тип boolean, содержащий значение true в случае успешного открытия файла или false в случае возникновения какой-либо ошибки.

**int Read()**

Функция чтения информации о чертеже из ранее открытого файла.

При построчном чтении файла считывается общая информация о чертеже и его объекты. При обнаружении каждого из возможных объектов, вызывается соответствующая ему функция, которая получает информацию о текущем объекте и записывает её в переменные.

**Входные и выходные параметры.**

Входные параметры отсутствуют.

Выходной параметр имеет тип integer. Может содежать следуюшие значения: 0 – при успешном выполнении функции, 1 – при не обнаружении секции ENTITIES в файле.

**Фрагмент кода функции.**

file.seekg(beginEntitiesPointer);

while(line != "EOF") {

if(line == "LINE") ReadLine();

if(line == "LWPOLYLINE") ReadPolyline();

if(line == "XLINE") ReadXLine();

if(line == "CIRCLE") ReadCircle();

if(line == "ELLIPSE") ReadEllipse();

if(line == "INSERT") ReadInsert();

if(line == "ARC") ReadArc();

file >> line;

}

**bool SaveErrorPoints(vector<CROSSPOINT> errPoints)**

Функция сохранения файла с отметками о возможных ошибках.

**Входные и выходные параметры.**

Входной параметр vector<CROSSPOINT> errPoints – вектор структур CROSSPOINT, которые содержат информацию о точках возможных ошибок.

Выходной параметр имеет тип boolean. Содержит значение true в случае успешного сохранения файла или false в случае возникновения какой-либо ошибки.

**bool SavePolyLine(string \_saveFileName, double \*\*points, int size, bool closed)**

Функция сохранения файла с полилинией, передаваемой в параметрах.

**Входные и выходные параметры.**

Входные параметры:

* string \_saveFileName – содержит имя сохраняемого файла;
* double \*\*points – массив точек полилинии, где points[i][0] - координата X, points[i][1] - координата Y;
* int size – толщина полилинии;
* bool closed – содержит true, если полилиния замкнута, и false в противном случае.

Выходной параметр имеет тип boolean. Содержит значение true в случае успешного сохранения файла или false в случае возникновения какой-либо ошибки.

**void Close()**

Функция закрытия файла и обнуления использованных переменных.

**Входные и выходные параметры.**

Входные и выходные параметры отсутствуют.

**SECTION GetSection()**

Функция предоставляет доступ к приватной переменной \_Section.

**Входные и выходные параметры.**

Входные параметры отсутствуют.

Выходной параметр имеет тип SECTION. В данном параметре содержится приватная переменная \_Section, в которой содержатся полученные из файла объекты.

* + 1. Приватные функции

Функции, объявленные в этом разделе, являются приватными, так как не входят в интерфейс работы с рассматриваемой библиотекой и не предполагают непосредственной работы пользователя с ними.

**void ReadLine()**

Функция void ReadLine() вызывается функцией int Read() при обнаружении объекта LINE в обрабатываемом файле. Данная функция создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор \_Section.Entities.Lines с параметром type, равным 0.

**Входные и выходные параметры.**

Входные и выходные параметры отсутствуют.

**void ReadPolyline()**

Функция void ReadPolyline() вызывается функцией int Read() при обнаружении объекта LWPOLYLINE в обрабатываемом файле. Данная функция создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор \_Section.Entities.Polylines.

**Входные и выходные параметры.**

Входные и выходные параметры отсутствуют.

**void ReadXLine()**

Функция void ReadXLine() вызывается функцией int Read() при обнаружении объекта XLINE в обрабатываемом файле. Данная функция создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор \_Section.Entities.Lines с параметром type, равным 1.

**Входные и выходные параметры.**

Входные и выходные параметры отсутствуют.

**void ReadCircle()**

Функция void ReadCircle() вызывается функцией int Read() при обнаружении объекта CIRCLE в обрабатываемом файле. Данная функция создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор \_Section.Entities.Circles.

**Входные и выходные параметры.**

Входные и выходные параметры отсутствуют.

**Фрагмент кода функции.**

while(line != "AcDbCircle") file >> line;

file >> line >> line;

tmp.p.x = atof(line.c\_str());

file >> line >> line;

tmp.p.y = atof(line.c\_str());

file >> line >> line >> line >> line;

tmp.r = atof(line.c\_str());

**void ReadBlock()**

Функция void ReadBlock() вызывается функцией int Read() при обнаружении объекта BLOCK в обрабатываемом файле. Данная функция создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор Blocks. Во время выполнения функции происходит анализ блока посредством чтения строк файла и при обнаружении объекта LINE вызывается функция void ReadLine().

**Входные и выходные параметры.**

Входные и выходные параметры отсутствуют.

**void ReadInsert()**

Функция void ReadInsert() вызывается функцией int Read() при обнаружении объекта INSERT в обрабатываемом файле. Данная функция создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор \_Section.Entities.Inserts.

**Входные и выходные параметры.**

Входные и выходные параметры отсутствуют.

**void ReadEllipse()**

Функция void ReadEllipse() вызывается функцией int Read() при обнаружении объекта ELLIPSE в обрабатываемом файле. Данная функция создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор \_Section.Entities.Ellipses.

**Входные и выходные параметры.**

Входные и выходные параметры отсутствуют.

**void ReadArc()**

Функция void ReadArc() вызывается функцией int Read() при обнаружении объекта ARC в обрабатываемом файле. Данная функция создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор \_Section.Entities.Arcs.

**Входные и выходные параметры.**

Входные и выходные параметры отсутствуют.

**int TextToHex(string str)**

Функция преобразования шестнадцатеричного числа, представленного в строковом выражении, в шестнадцатеричное число типа integer.

**Входные и выходные параметры.**

Входной параметр string str представляет собой строку, в которой содержится шестнадцатеричное число, представленное в строковом выражении.

Выходной параметр типа integer представляет собой шестнадцатеричное число в целочисленном формате.

**string LineToString(LINE l)**

Функция преобразования объекта LINE в строковое представление для записи в файл dxf.

**Входные и выходные параметры.**

Входной параметр LINE l представляет собой объект типа LINE.

Выходной параметр типа string имеет строковое представление для записи в файл dxf.

**string PolylineToString(LWPOLYLINE l)**

Функция преобразования объекта LWPOLYLINE в строковое представление для записи в файл dxf.

**Входные и выходные параметры.**

Входной параметр LWPOLYLINE l представляет собой объект типа LWPOLYLINE.

Выходной параметр типа string имеет строковое представление для записи в файл dxf.

**string CircleToString(CIRCLE c)**

Функция преобразования объекта CIRCLE в строковое представление для записи в файл dxf.

**Входные и выходные параметры.**

Входной параметр CIRCLE c представляет собой объект типа CIRCLE.

Выходной параметр типа string имеет строковое представление для записи в файл dxf.

**string ErrPointToString(CROSSPOINT point)**

Функция преобразования объекта CROSSPOINT в строковое представление для записи в файл dxf.

**Входные и выходные параметры.**

Входной параметр CROSSPOINT point c представляет собой объект типа CROSSPOINT.

Выходной параметр типа string имеет строковое представление для записи в файл dxf.

**string LayerToString(string name)**

Функция составления имени слоя в его строковое представление для записи в файл dxf.

**Входные и выходные параметры.**

Входной параметр string name c представляет собой имя слоя.

Выходной параметр типа string имеет строковое представление для записи в файл dxf.

* 1. Разработанная библиотека для работы с объектами

Данная библиотека служит для обнаружения пересечений и совпадений объектов, полученных из dxf-файла, а так же для обнаружения взаимодействия с объектом, посредством клика по нему мышью.

* + 1. Публичные функции

Функции, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемой библиотекой.

**void CheckCross(BITSFIELD bitsfield)**

В рассматриваемой функции происходит попарная передача объектов, содержащихся в переменной Section, в функции, где непосредственно выполняется проверка пересечений и совпадений, в соответствии с флагами, выставленными в передаваемом параметре bitsfield.

**Входные и выходные параметры.**

Входной параметр BITSFIELD bitsfield. В нём передаются флаги для определения того, какие объекты необходимо проверить на пересечение и совпадение.

Выходные параметры отсутствуют.

**void InitBitsField(BITSFIELD \*field)**

Функция инициализации структуры BITSFIELD.

**Входные и выходные параметры.**

Входной параметр BITSFIELD \*field – указатель на структуру типа BITSFIELD.

Выходные параметры отсутствуют.

**void ClickOnObject(double x, double y, double scale, DataGridView^ table)**

Функция распознавания объекта, по которому был совершён клик мыши, вывода информации об этом объекте и изменения параметра current этого объекта.

**Входные и выходные параметры.**

Входные параметры:

* x, y – координаты места, по которому был совершён клик мыши;
* scale – коэффициент масштабирования при отображении объектов на экране;
* table – таблица, в которую будет выведена информация о выбранном файле.

Выходные параметры отсутствуют.

* + 1. Приватные функции

Функции, объявленные в этом разделе, являются приватными, так как не входят в интерфейс работы с рассматриваемой библиотекой и не предполагают непосредственной работы пользователя с ними.

**void CrossLines(LINE line1, LINE line2, unsigned char crossKind)**

Функция определения пересечения двух отрезков. Алгоритм её работы описан в разделе 3.1. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool line\_line равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* line1 – первый объект типа LINE;
* line2 – второй объект типа LINE;
* crossKind – виды пересечений, которые должны быть обработаны.

Выходные параметры отсутствуют.

**void CrossLineCircle(LINE line, CIRCLE circle, unsigned char crossKind)**

Функция определения пересечения отрезка и окружности. Алгоритм её работы описан в разделе 3.2. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool line\_circle равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* line – объект типа LINE;
* circle – объект типа CIRCLE;
* crossKind – виды пересечений, которые должны быть обработаны.

Выходные параметры отсутствуют.

**Фрагмент кода функции.**

double a = (y2 - y1) / (x2 - x1);

double b = y1 - x1 \* a;

double alpha = a \* a + 1;

double beta = 2 \* (a \* (b - y0) - x0);

double gamma = x0 \* x0 + pow(b - y0, 2) - r \* r;

long double det = beta \* beta - 4 \* alpha \* gamma;

if(det < 0) return; //Не пересекаются

long double tx1 = (-beta + sqrt(det)) / (2 \* alpha);

long double tx2 = (-beta - sqrt(det)) / (2 \* alpha);

long double ty1 = a \* tx1 + b;

long double ty2 = a \* tx2 + b;

**void CrossLineArc(LINE line, ARC arc, unsigned char crossKind)**

Функция определения пересечения отрезка и дуги. Алгоритм её работы описан в разделе 3.3. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool line\_arc равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* line – объект типа LINE;
* arc – объект типа ARC;
* crossKind – виды пересечений, которые должны быть обработаны.

Выходные параметры отсутствуют.

**void CrossLineEllipse(LINE line, ELLIPSE ellipse, unsigned char crossKind)**

Функция определения пересечения отрезка и эллипса. Алгоритм её работы описан в разделе 3.4. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool line\_ellipse равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* line – объект типа LINE;
* ellipse – объект типа ELLIPSE;
* crossKind – виды пересечений, которые должны быть обработаны.

Выходные параметры отсутствуют.

**void CrossCircles(CIRCLE circle1, CIRCLE circle2)**

Функция определения пересечения двух окружностей. Алгоритм её работы описан в разделе 3.5. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool circle\_circle равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* circle1– первый объект типа CIRCLE;
* circle2– второй объект типа CIRCLE.

Выходные параметры отсутствуют.

**void CrossCircleArc(CIRCLE c, ARC a)**

Функция определения пересечения окружности и дуги. Алгоритм её работы описан в разделе 3.6. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool circle\_arc равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* c – первый объект типа CIRCLE;
* a – второй объект типа ARC.

Выходные параметры отсутствуют.

**void CrossArcs(ARC a1, ARC a2)**

Функция определения пересечения двух дуг. Алгоритм её работы описан в разделе 3.7. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool arc\_arc равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* a1 – первый объект типа ARC;
* a2 – второй объект типа ARC.

Выходные параметры отсутствуют.

**void OverlapLines(LINE line1, LINE line2)**

Функция определения совпадения (полного и частичного) двух отрезков. Алгоритм её работы описан в разделе 3.8. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool overlap\_lines равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* line1 – первый объект типа LINE;
* line2 – второй объект типа LINE.

Выходные параметры отсутствуют.

**void OverlapCircles(CIRCLE circle1, CIRCLE circle2)**

Функция определения совпадения двух окружностей. Алгоритм её работы описан в разделе 3.8. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool overlap\_circles равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* circle1 – первый объект типа CIRCLE;
* circle2 – второй объект типа CIRCLE.

Выходные параметры отсутствуют.

**Фрагмент кода функции.**

if((circle1.p.x == circle2.p.x) && (circle1.p.y == circle2.p.y) && (circle1.r == circle2.r)) {

CROSSPOINT tmp;

tmp.current = false;

tmp.isCrossPoint = false;

tmp.Lines.clear();

tmp.Circles.clear();

tmp.Ellipses.clear();

tmp.Arcs.clear();

tmp.Circles.push\_back(circle1);

ErrPoints.push\_back(tmp);}

**void OverlapArcs(ARC a1, ARC a2)**

Функция определения совпадения (полного и частичного) двух дуг. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool overlap\_arcs равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* a1 – первый объект типа ARC;
* a2 – второй объект типа ARC.

Выходные параметры отсутствуют.

**void OverlapEllipses(ELLIPSE e1, ELLIPSE e2)**

Функция определения совпадения двух эллипсов. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool overlap\_ellipses равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* e1 – первый объект типа ELLIPSE;
* e2 – второй объект типа ELLIPSE.

Выходные параметры отсутствуют.

**void OverlapCircleArc(CIRCLE c, ARC a)**

Функция определения совпадения окружности и дуги. Вызывается из функции void CheckCross(BITSFIELD bitsfield) при bool overlap\_circle\_arc равным true в BITSFIELD bitsfield.

**Входные и выходные параметры.**

Входные параметры:

* c – первый объект типа CIRCLE;
* a – второй объект типа ARC.

Выходные параметры отсутствуют.

**double Det(double x1, double x2, double x3, double x4)**

Функция вычисления детерминанта матрицы:

**Входные и выходные параметры.**

Входные параметры: x1, x2, x3, x4 – параметры матрицы.

Выходной параметр имеет тип double и является значением детерминанта матрицы.

**double GetAngle(double x0, double y0, double x1, double y1)**

Функция определения угла между прямой, проведённой через точку и параллельной оси абсцисс, и прямой, проведённой через точки и .

**Входные и выходные параметры.**

Входные параметры:

* x0, y0 – координаты точки ;
* x1, y1 – координаты точки .

Выходной параметр имеет тип double и является значением искомого угла.

**bool IsPointOnLine(LINE line, double x, double y)**

Функция определения принадлежности точки линии line.

**Входные и выходные параметры.**

Входные параметры:

* line – линия;
* x, y – координаты точки .

Выходной параметр имеет тип bool и принимает значение true, если точка принадлежит отрезку, иначе false.

**double GetDistance(double x0, double y0, double x1, double y1)**

Функция определения расстояния между точкой и точкой .

**Входные и выходные параметры.**

Входные параметры:

* x0, y0 – координаты точки ;
* x1, y1 – координаты точки .

Выходной параметр имеет тип double и является расстоянием между рассматриваемыми точками.

* 1. Разработанная библиотека для работы с графикой

Данная библиотека служит для работы с графической частью программы: отображение, масштабирование и перемещение объектов чертежа посредством OpenGL.

* + 1. Публичные функции

Функции, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемой библиотекой.

**void GetOpenGlControl(SimpleOpenGlControl^ simpleOpenGlControl)**

Функция, получающая элемент Windows Forms, для дальнейшей работы с ним.

**Входные и выходные параметры.**

Входной параметр SimpleOpenGlControl^ simpleOpenGlControl. В данном параметре передаётся элемент Windows Forms, для дальнейшей работы библиотеки с ним.

Выходные параметры отсутствуют.

**int InitGL()**

Функция инициализации OpenGL, задание цветов отчистки, используемой цветовой модели и размера буфера; задание размеров области вывода.

**Входные и выходные параметры.**

Входные параметры отсутствуют.

Выходной параметр имеет тип integer и принимает значение 0, при успешном выполнении функции, или -1, если в библиотеку не был передан элемент формы.

**void Resize(double width, double height)**

Функция изменения размеров области отрисовки.

**Входные и выходные параметры.**

Входные параметры:

* width – новая ширина области отрисовки;
* height – новая высота области отрисовки.

Выходные параметры отсутствуют.

**void Render()**

Функция задания текущего цвета для отрисовки элементов, отрисовка осей абсцисс и ординат.

**Входные и выходные параметры.**

Входные и выходные параметры отсутствуют.

**Фрагмент кода функции.**

Gl::glColor3f(0.8f, 0.8f, 0.8f);

Gl::glBegin(Gl::GL\_LINE\_STRIP);

Gl::glVertex2f(0,0);

Gl::glVertex2f(100,0);

Gl::glEnd();

Gl::glBegin(Gl::GL\_LINE\_STRIP);

Gl::glVertex2f(90,3);

Gl::glVertex2f(100,0);

Gl::glVertex2f(90,-3);

Gl::glEnd();

**void SwapBuffers()**

Очистка буферов OpenGL. Посредством чего отрисовываются элементы, находившиеся в этих буферах.

**Входные и выходные параметры.**

Входные и выходные параметры отсутствуют.

**void Scale(signed short int s)**

Функция изменения масштаба отрисовываемых элементов.

**Входные и выходные параметры.**

Входной параметр s принимает значения 1, для увеличения масштаба в 2 раза, или -1 для уменьшения масштаба в 2 раза.

Выходные параметры отсутствуют.

**void Translate(double x, double y)**

Функция смещения отображаемых элементов в горизонтальном и вертикальном направления.

**Входные и выходные параметры.**

Входные параметры:

* x – сдвиг по оси абсцисс;
* y – сдвиг по оси ординат.

Выходные параметры отсутствуют.

**void DrawPoint(double x, double y, float size)**

Функция отображения точки заданного размера в заданных координатах.

**Входные и выходные параметры.**

Входные параметры:

* x, y – координаты точки;
* size – размер точки в пикселях.

Выходные параметры отсутствуют.

**void DrawLine(double x1, double y1, double x2, double y2)**

Функция отображения отрезка по заданным координатам.

**Входные и выходные параметры.**

Входные параметры:

* x1, y1 – координаты первой точки;
* x2, y2 – координаты второй точки.

Выходные параметры отсутствуют.

**void DrawCircle(double x, double y, double r)**

Функция отображения окружности с заданными координатами центра и радиусом.

**Входные и выходные параметры.**

Входные параметры:

* x, y – координаты центра окружности;
* r – радиус окружности.

Выходные параметры отсутствуют.

**Фрагмент кода функции.**

Gl::glBegin(Gl::GL\_LINE\_LOOP);

for(int i=0; i < segments; i++) {

double alpha = 2 \* PI \* i / segments;

double dx = r \* cos(alpha);

double dy = r \* sin(alpha);

Gl::glVertex2d(x + dx, y + dy);

}

Gl::glEnd();

**void DrawErrorPoint(double x, double y, double r)**

Функция отображения точки пересечения (совпадения) с заданными координатами центра и радиусом. Принцип работы заключается в изменении цвета отображения и вызове функции DrawCircle.

**Входные и выходные параметры.**

Входные параметры:

* x, y – координаты центра окружности;
* r – радиус окружности.

Выходные параметры отсутствуют.

**void DrawArc(double x, double y, double r, double begin, double end)**

Функция отображения дуги с заданными координатами центра, радиуса, начальным и конечным углами.

**Входные и выходные параметры.**

Входные параметры:

* x, y – координаты центра дуги;
* r – радиус дуги;
* begin – начальный угол дуги;
* end – конечный угол дуги.

Выходные параметры отсутствуют.

**void DrawEllipse(double x, double y, double a, double b, double angle)**

Функция отображения эллипса с заданными координатами центра, длинами большой и малой полуосей, углом наклона.

**Входные и выходные параметры.**

Входные параметры:

* x, y – координаты центра эллипса;
* a – длина большей полуоси;
* b – длина меньшей полуоси;
* angle – угол наклона эллипса относительно оси абсцисс.

Выходные параметры отсутствуют.

**void SetColor(float red, float green, float blue)**

Функция установки цвета модели RGB, которым будут отображаться объекты. Параметры могут принимать значения в диапазоне .

**Входные и выходные параметры.**

Входные параметры:

* red – интенсивность красного цвета;
* green – интенсивность зелёного цвета;
* blue – интенсивность голубого цвета.

Выходные параметры отсутствуют.

**bool GetIsInit()**

Функция, возвращающая значение переменной IsInit.

**Входные и выходные параметры.**

Входные параметры отсутствуют.

Выходной параметр принимает значение boolean и является значением переменной IsInit.

* 1. Используемые сторонние библиотеки

**Библиотека The Tao Framework –** данная библиотека распространяется по лицензии MIT, используется для отрисовки объектов чертежа посредством OpenGL на элементах Windows Forms. Использование OpenGL в программе является затруднительным, поскольку при разработке программы интерфейс был выполнен с использованием Windows Forms.

**Библиотека MySql.Data** – данная библиотека используется для работы с базой данных MySQL. Данная библиотека входит в пакет Connector.NET, поставляемый с СУБД MySQL.

1. Руководство пользователя к разработанному программному продукту
   1. Интерфейс

Запустив приложение, пользователь увидит главный интерфейс программы.

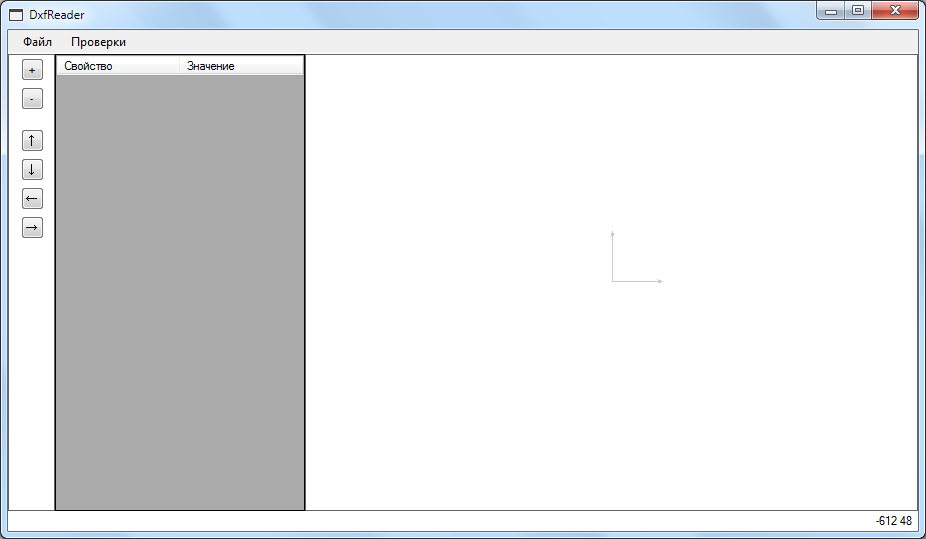


Рисунок . Главный интерфейс

* 1. Описание клавишей

На панели быстрого доступа расположены вкладка «ФАЙЛ» и кнопка «ПРОВЕРИТЬ».

«ФАЙЛ» содержит в себе пункты «ОТКРЫТЬ», «СОХРАНИТЬ», «ЗАКРЫТЬ», «ВЫХОД»:

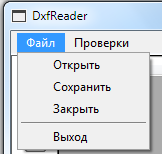


Рисунок . Вкладка "ФАЙЛ"

Клавиши «+» и «-» отдаляют и приближают рисунок соответственно. При помощи клавиш со стрелками можно перемещать чертёж по полю отображения.

D:\Диплом\2015-05-25_102923.png

Рисунок . Клавиши манипулирования чертежом

* 1. Открытие файла

Чтобы открыть файл формата dxf нужно выбрать пункт «ОТКРЫТЬ» и, выбрав с помощью открывшегося диалогового окна нужный файл, нажать кнопку «ОТКРЫТЬ».

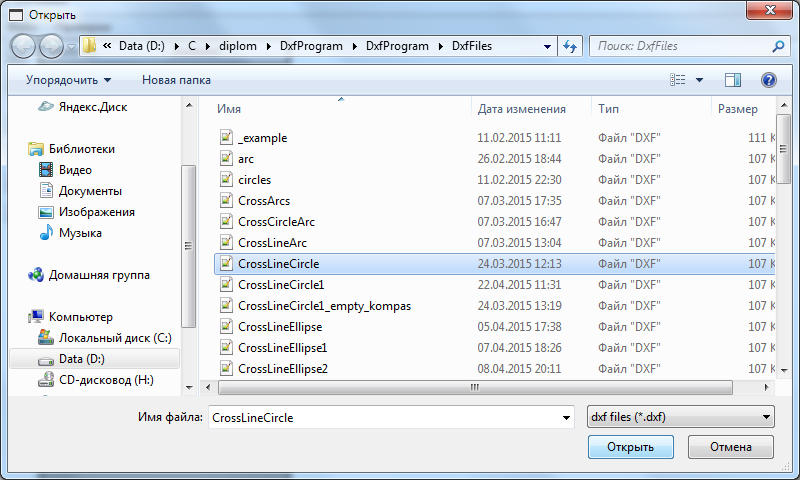


Рисунок . Диалог открытия файла

Загрузив файл, пользователь увидит следующее:

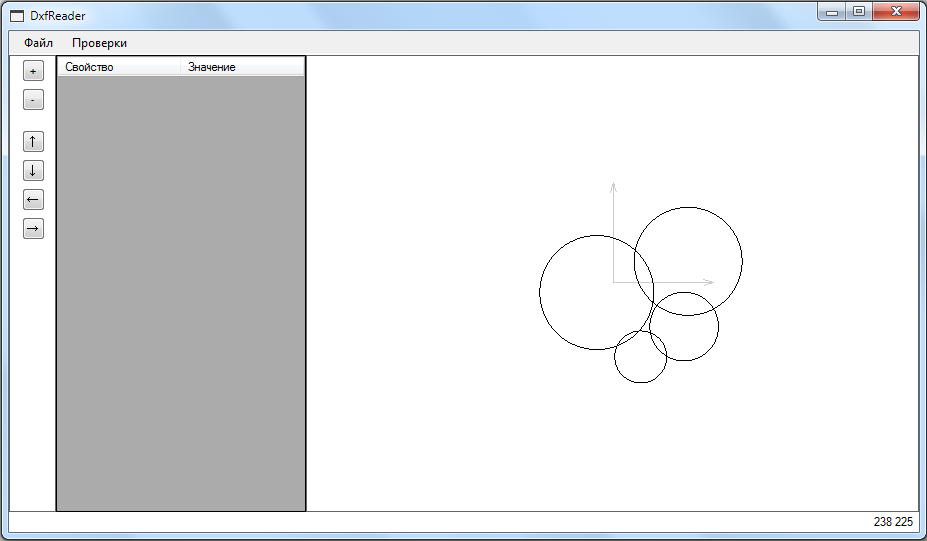


Рисунок . Открытый файл

В правом нижнем углу окна будут показаны координаты точки, на которую указывает мышь:

D:\Диплом\2015-05-25_1103121.png

Рисунок . Координаты указателя мыши

Манипулировать чертежом кроме клавиш (Рис. 15) можно и при помощи мыши. Для перемещения чертежа нужно нажать колёсико мыши и передвигать чертёж в нужную сторону. Для увеличения или уменьшения чертежа необходимо прокрутить колёсико мыши.

* 1. Проверка файла

На панели быстрого доступа размещена клавиши «ПРОВЕРКИ», нажав на которую пользователь увидит следующее:

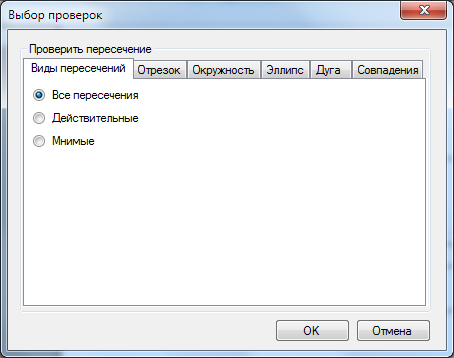


Рисунок . Окно выбора проверок

Если файл ещё не был открыт, то пользователь увидит сообщение об ошибке:

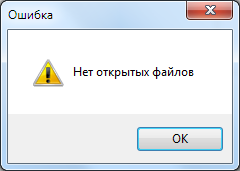


Рисунок . Ошибка: нет открытых файлов

После чего нужно нажать на кнопку «ОК» и открыть нужный файл.

Если же файл был открыт, то пользователь увидит окно, в котором может выбрать виды пересечений, а также пары объектов, у которых необходимо обнаружить пересечения или совпадения. Для окончания выбора следует нажать клавишу «ОК», после чего на чертеже появятся красные отметки, которыми отмечены возможные ошибки.

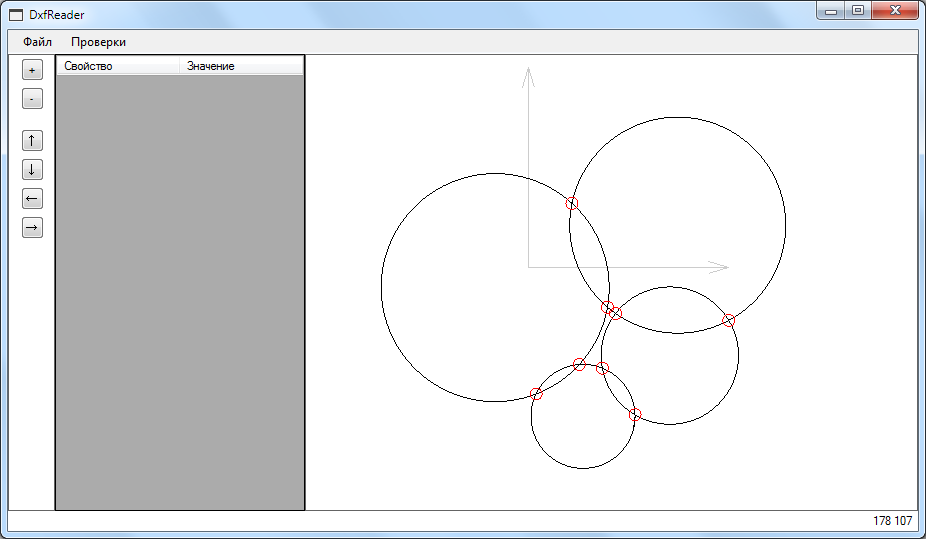


Рисунок . Отметки возможных ошибок

Присутствует возможность получить информацию об отображаемом объекте. Для этого необходимо нажать левой кнопкой мыши на нужный объект, после чего отобразятся свойства выбранного объекта и их значения, а сам объект подсветится синим цветом:

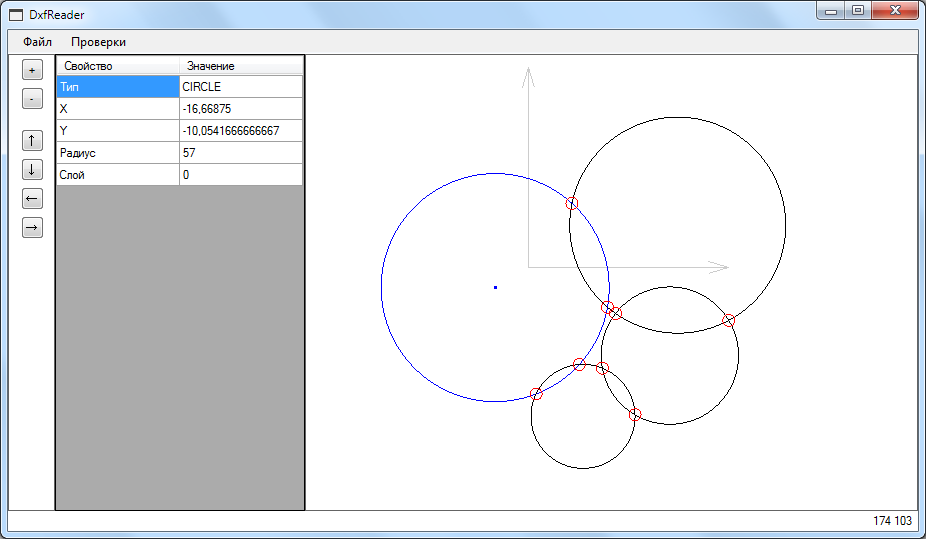


Рисунок . Свойства выбранного объекта

* 1. Сохранение файла

Для сохранения открытого нужно из пункта «ФАЙЛ» выбрать «СОХРАНИТЬ».

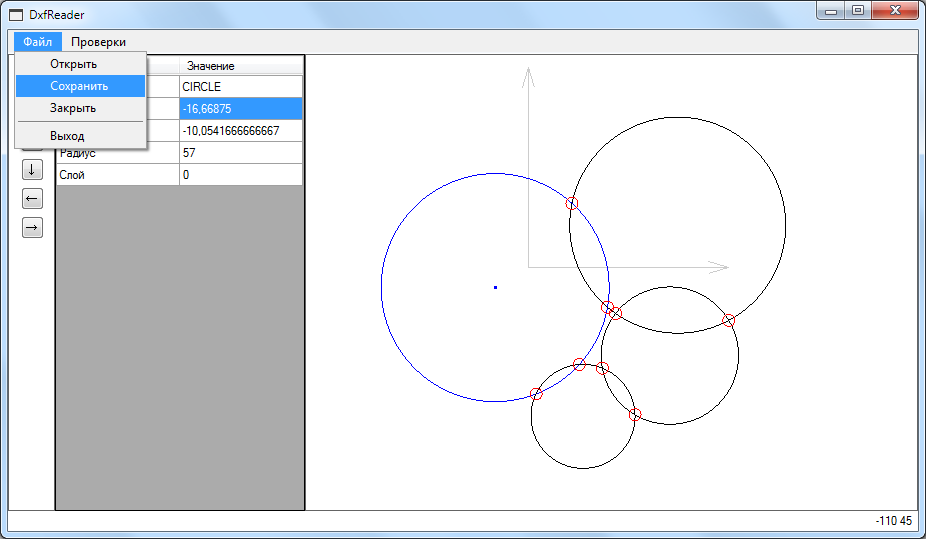


Рисунок . Сохранение файла

Если файл ещё не был открыт, то пользователь увидит сообщение об ошибке:

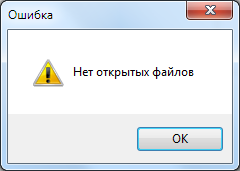


Рисунок . Ошибка: нет открытых файлов

После чего нужно нажать на кнопку «ОК» и открыть нужный файл.

Если файл был открыт, то пользователь увидит окно сохранения файла. Для того чтобы добавить информацию о чертеже в базу данных необходимо выбрать «Добавить файл в базу данных» и заполнить информационные поля.

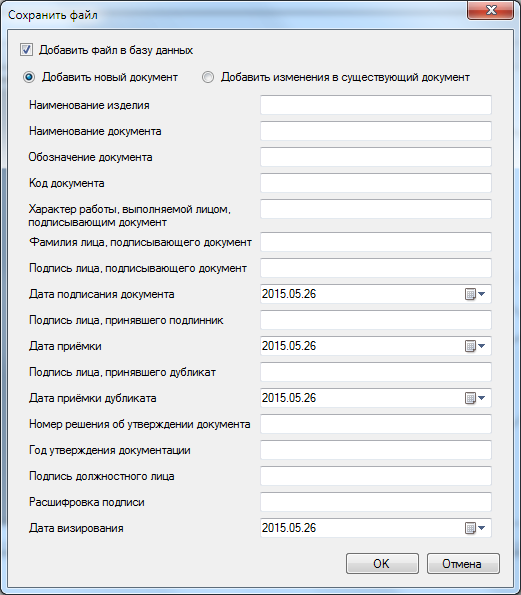


Рисунок . Добавление нового документа

Если же документ в базе уже существует, то нужно нажать на «Добавить изменения в существующий документ».

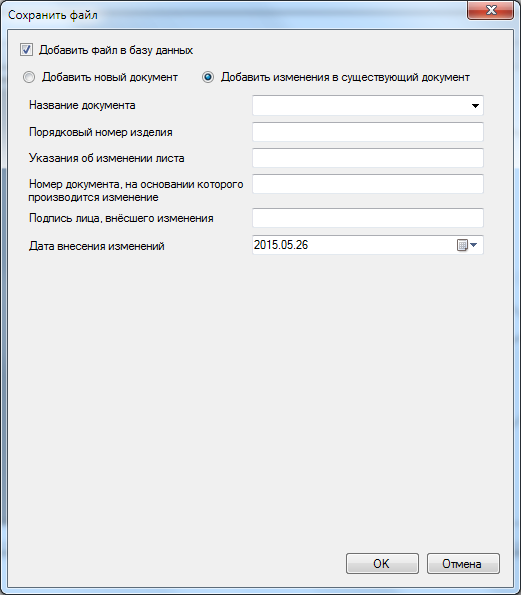


Рисунок . Добавить изменения в существующий документ

По окончании заполнения полей необходимо нажать клавишу «ОК». После чего информация будет занесена в базу данных, а файл с отметками о возможных ошибках будет сохранён в той же папке, что и чертёж с именем «имя\_файла»\_c.dxf.

Ели же не все поля будут заполнены, то будет показано соответствующее сообщение об ошибке:

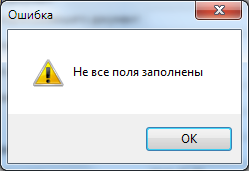


Рисунок . Ошибка: не все поля заполнены

После чего необходимо нажать на кнопку «ОК» и заполнить пустые поля, либо нажать кнопку «ОТМЕНА» в окне сохранения файла.

Для того чтобы закрыть открытый ранее чертёж нужно из пункта «ФАЙЛ» выбрать «ЗАКРЫТЬ».

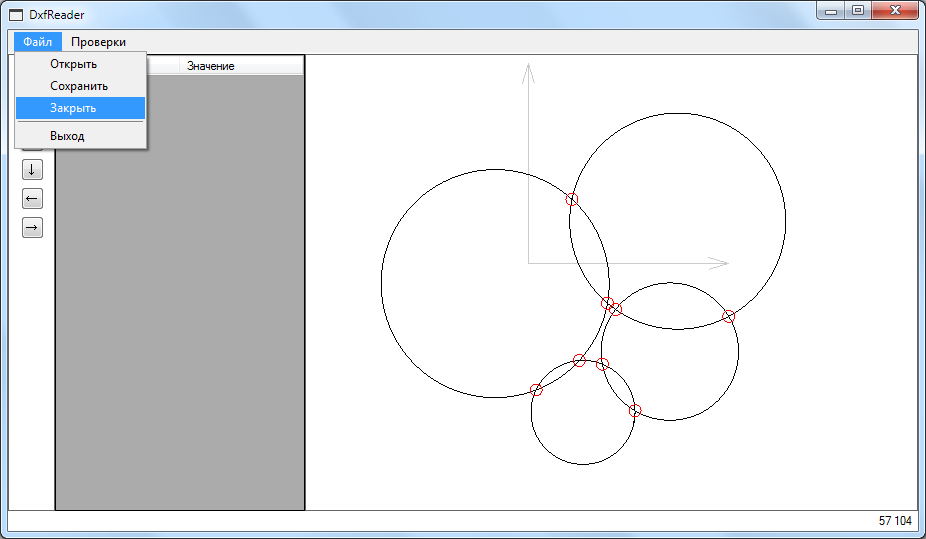


Рисунок . Закрытие файла

Для того чтобы выйти из программы необходимо из пункта «ФАЙЛ» выбрать «ВЫХОД» или нажать на кнопку «Х».

1. Взаимосвязь результатов работы комплекса с САПР КОМПАС-3D и AutoCAD

Разработанный программный комплекс предназначен для нахождения возможных ошибок на машиностроительных 2D-чертежах, исправление же ошибок осуществляется в САПР.

В результате работы программы будет получен dxf-файл с записанными координатами точек ошибок на отдельном слое «ErrorsLayer».

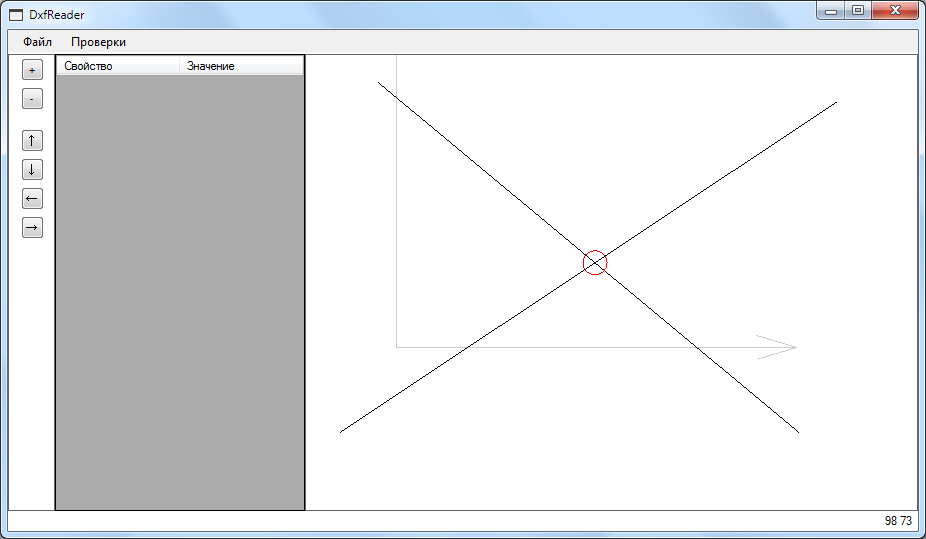


Рисунок . Результат работы программы

Как можно видеть, точка, обозначающая пересечение двух отрезков отсутсвует.

* 1. Результат работы разработанной программы в САПР КОМПАС-3D

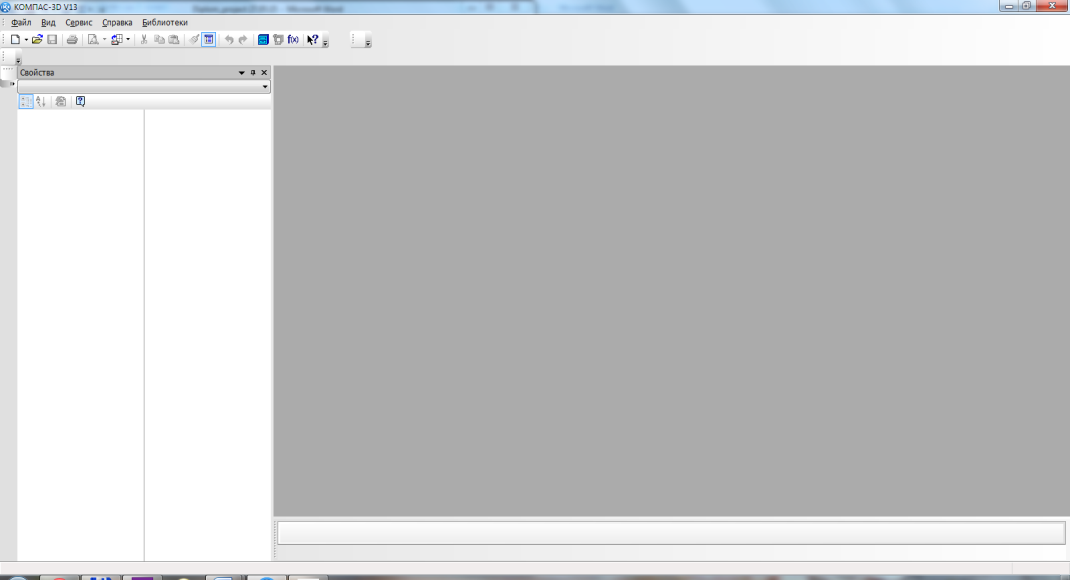


Рисунок . Интерфейс КОМПАС-3D

Чтобы открыть полученный файл в КОМПАС-3D нужно на вкладке «ФАЙЛ» выбрать пункт «Открыть…».

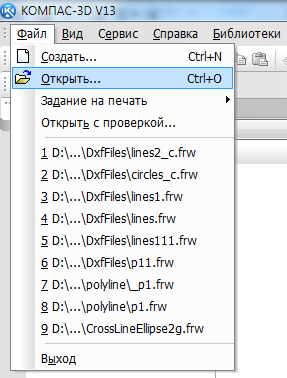


Рисунок . Пункт "Открыть..."

В открывшемся окне выбрать нужный файл и нажать «ОТКРЫТЬ».

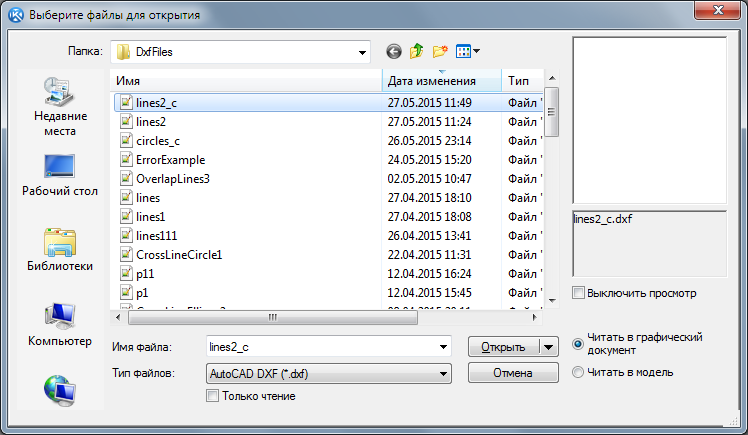


Рисунок . Диалог открытия файла

Чертёж с отметкой о возможной ошибке на отдельном слое открыт.

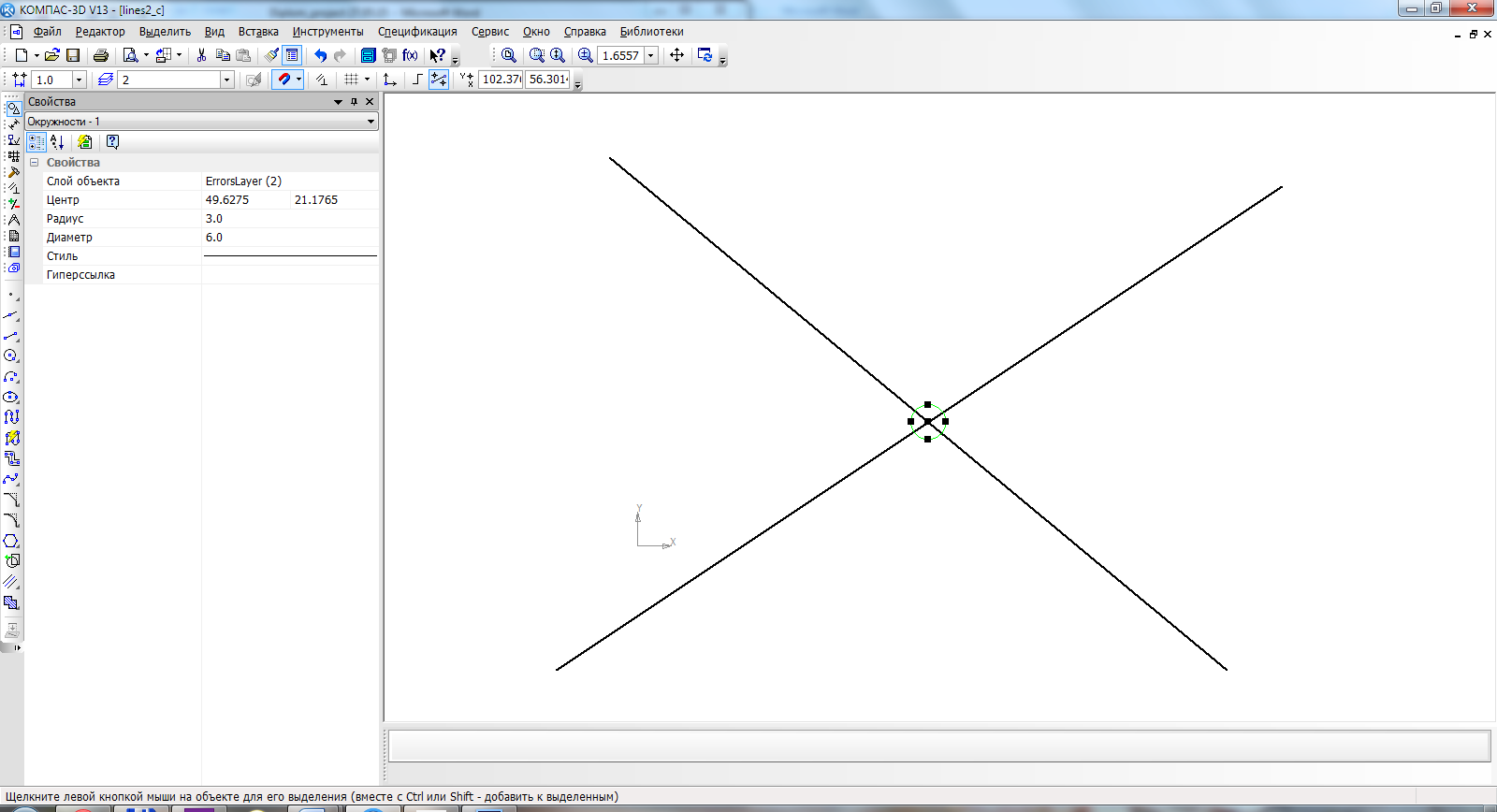


Рисунок . Чертёж, открытый в КОМПАС-3D

После исправления ошибки и удаления отметок со слоя ошибок получим:

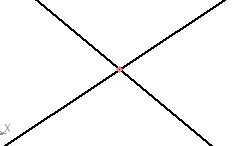


Рисунок . Исправленный чертёж

* 1. Результат работы разработанной программы в САПР AutoCAD

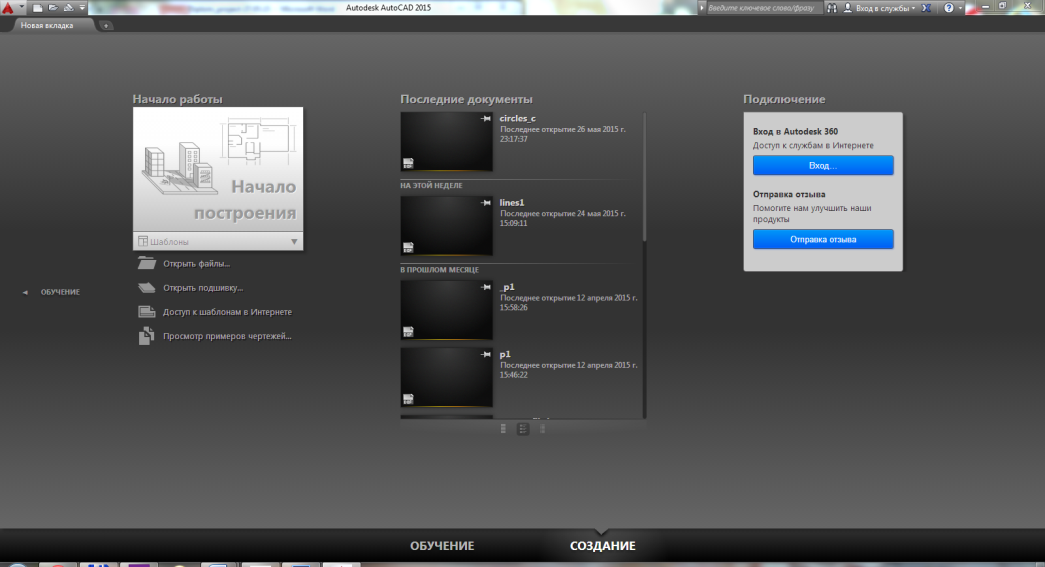


Рисунок . Интерфейс AutoCAD

Чтобы открыть полученный файл в КОМПАС-3D нужно нажать на значок программы, на вкладке «Открыть» выбрать пункт «Чертеж».

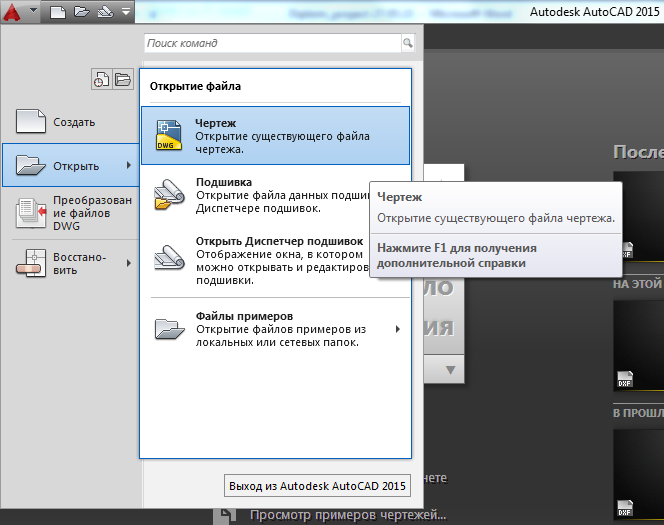


Рисунок . Вкладка "Открыть"

В открывшемся окне выбрать нужный файл и нажать «ОТКРЫТЬ».

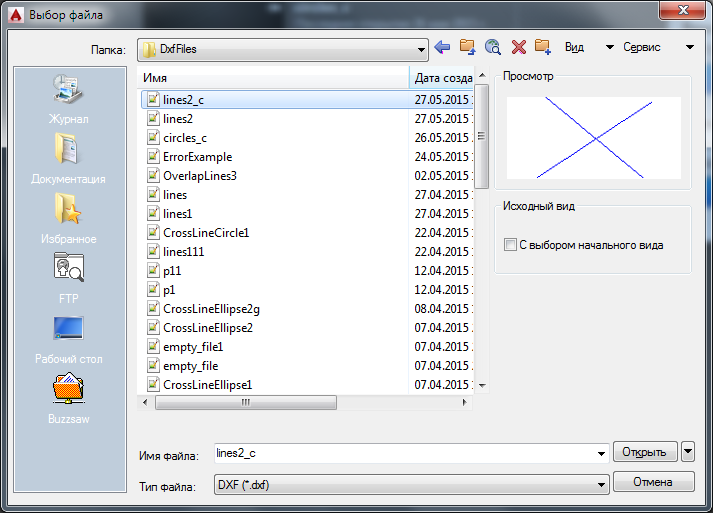


Рисунок . Диалог открытия файла

Чертёж с отметкой о возможной ошибке на отдельном слое открыт.

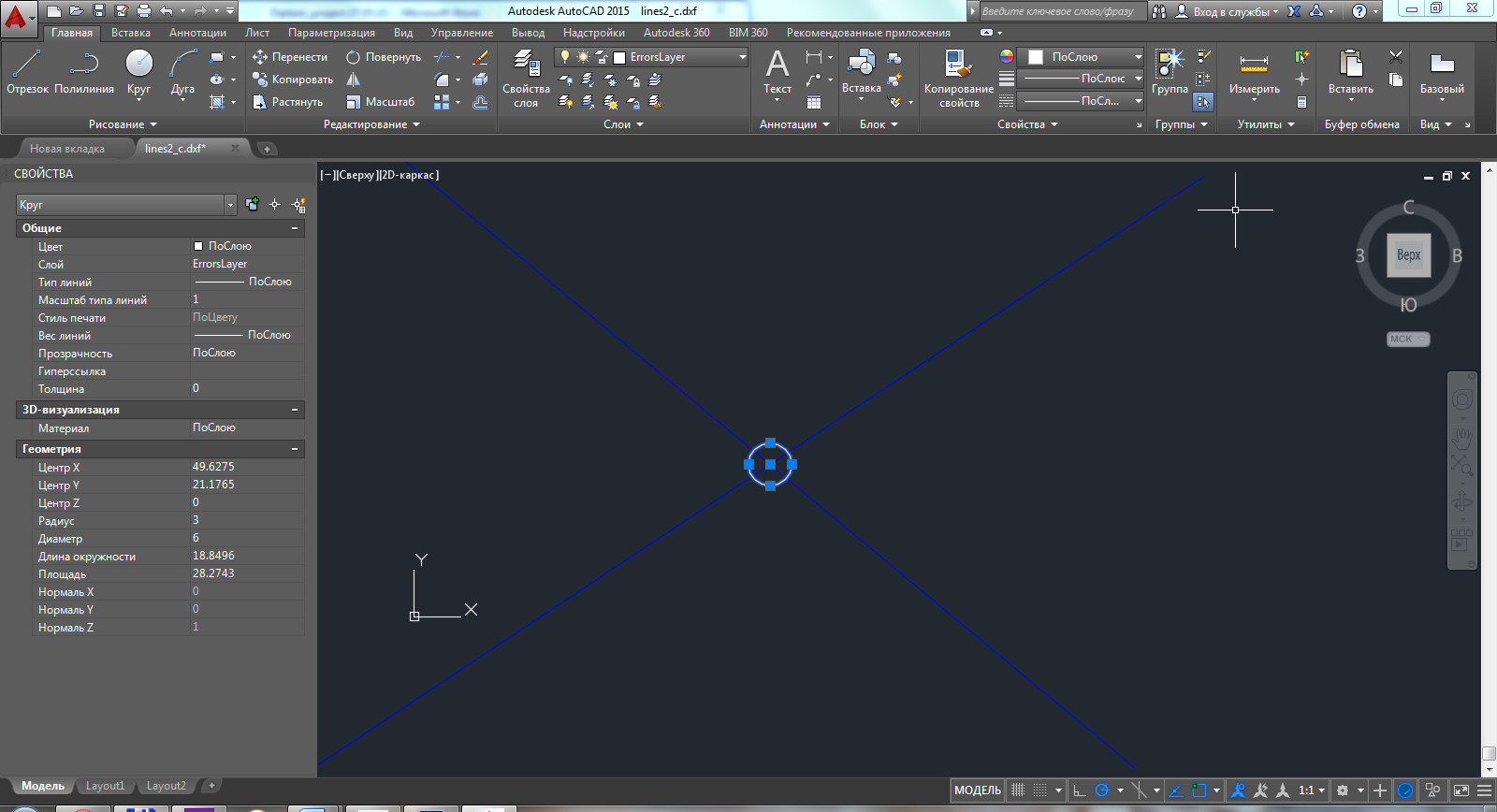


Рисунок . Чертёж, открытый в AutoCAD

После исправления ошибки и удаления отметок со слоя ошибок получим:

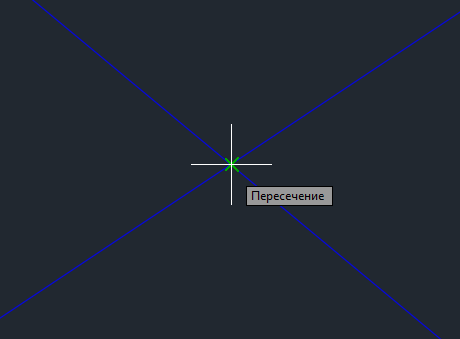


Рисунок . Исправленный чертёж

Заключение

По результатам выполнения выпускной квалификационной работы можно сделать следующие выводы:

* Изучены существующие САПР, форматы хранения данных, а также существующие аналоги.
* Спроектировано информационное, алгоритмическое и программное обеспечение системы.
* Реализован программный комплекс в целом (реализованы библиотеки работы с файлами dxf, с геометрическими объектами и графической частью интерфейса).
* Проведена апробация программного комплекса.

Литература

1. Петров, А.В. Разработка САПР / А.В. Петров. – М.: Высш. шк., 1990
2. 3D-технология построения чертежа в AutoCAD / А.Л. Хейфец [и др.] – СПб.: БХВ-Петербург, 2005. – 256 с.
3. Климачева, Т. Н. Трехмерная компьютерная графика и автоматизация проектирования в AutoCAD / Т.Н. Климатчева. – М.: ДМК Пресс. 2007. – 464 с.
4. Бочков, А.Л. Трехмерное моделирование в системе Компас-3D (практическое руководство) / А.Л. Бочков. – СПб: СПбГУ ИТМО, 2007. – 84 с.
5. Норенков, И.П. Основы автоматизированного проектирования / И.П. Норенков. – М.: Издательство МГТУ имени Н.Э. Баумана, 2009. – 410 с.
6. Обмен данными между системами КОМПАС-3D V10 и AutoCAD / ЗАО АСКОН, 2009. – 29 с.
7. CorelCAD 2014 Руководство обозревателя
8. Autodesk расширяет возможности коллективной работы при конструировании промышленных изделий – Режим доступа: [http://www.autodesk.ru/adsk/servlet/ item?siteID=871736&id=6396852](http://www.autodesk.ru/adsk/servlet/%20item?siteID=871736&id=6396852). – Загл. с экрана.
9. Что такое TrustedDWG? – Режим доступа: [http://www.autodesk.ru/adsk/servlet/ item?siteID=871736&id=10687507](http://www.autodesk.ru/adsk/servlet/%20item?siteID=871736&id=10687507). – Загл. с экрана.
10. Описание формата dxf – Режим доступа: [http://www.autodesk.com/techpubs /autocad/acadr14/dxf/](http://www.autodesk.com/techpubs%20/autocad/acadr14/dxf/). – Загл. с экрана.
11. The Tao Framework – Режим доступа: <http://sourceforge.net/projects/taoframework/>
12. Руководство пользователя CorelCAD – Режим доступа: <http://corel-cad.ru/html/hlpid_file_audit.htm>. – Загл. с экрана.
13. Функции CorelCAD – Режим доступа: <http://www.coreldraw.com/ru/product/cad-software/#tab2>. – Загл. с экрана.
14. Autodesk Nevisworks – Режим доступа: [http://www.autodesk.ru/products/ navisworks/overview](http://www.autodesk.ru/products/%20navisworks/overview). – Загл. с экрана.
15. DXF Reference – Режим доступа: [http://www.autodesk.com/techpubs/autocad/ acadr14/dxf/](http://www.autodesk.com/techpubs/autocad/%20acadr14/dxf/). – Загл. с экрана.