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: 080801 «Прикладная информатика (в информационной сфере)»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема:
«Поиск геометрических ошибок на 2D-чертежах»

Заведующий кафедрой:
доцент, к. ф-м. н. Шапошников Д.Е.

_____
подпись

Выполнил:
студент группы 8409 Дёгтев А.С.

_____
подпись

Научный руководитель:
к.т.н., доц. каф. ИАНИ Васин Д. Ю.

_____
подпись

Нижний Новгород
2015

Аннотация

В работе содержится описание информационного, алгоритмического и программного обеспечения комплекса обнаружения геометрических ошибок на 2D машиностроительных чертежах в формате dxf. Описываются возможные ошибки, алгоритмы их обнаружения, библиотеки для работы с графической часть, условия функционирования комплекса, а также результаты полученные с его помощью.

Оглавление

Глоссарий

Действительное пересечение двух объектов – пересечение, при котором у обоих объектов есть общая точка.

Мнимое пересечение двух объектов – пересечение, при котором, при продлении объектов у них образуется общая точка.

Полностью совпадающие окружности – окружности с совпадающими координатами центров и равными радиусами.

Полностью совпадающие дуги – дуги с совпадающими координатами центров, равными радиусами, попарно равными начальными и конечными углами.

Полностью совпадающие эллипсы – эллипсы с совпадающими координатами центров, попарно равными длинами обоих осей, равными углами поворота.

Частично совпадающие окружность и дуга – окружность и дуга с совпадающими координатами центров и равными радиусами.

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.

Цель работы – разработка информационного, алгоритмического и программного обеспечения подсистемы обнаружения геометрических ошибок на векторных 2D-изображениях машиностроительных чертежей.

1. Теоритическая часть

1.1. 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 mental ray;
- включает в себя полный набор инструментов для комплексного трёхмерного моделирования (поддерживается твердотельное, поверхностное и полигональное моделирование).

Таким образом, развитых средств обнаружения геометрических ошибок на изображениях 2D-чертежей в САПР 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, развитых средств обнаружения геометрических ошибок на изображениях 2D-чертежей в САПР КОМПАС-3D не имеется.

1.2. Существующие системы поиска геометрических ошибок на 2D-чертежах

1.2.1. 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.

1.2.2. 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.

2. Постановка задачи

Задан векторный 2D-чертеж формата dxf версии AutoCAD 2007 и старше на листе формата А4.

Необходимо реализовать 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.

На рис. 1 приведён пример ошибки на машиностроительном чертеже.

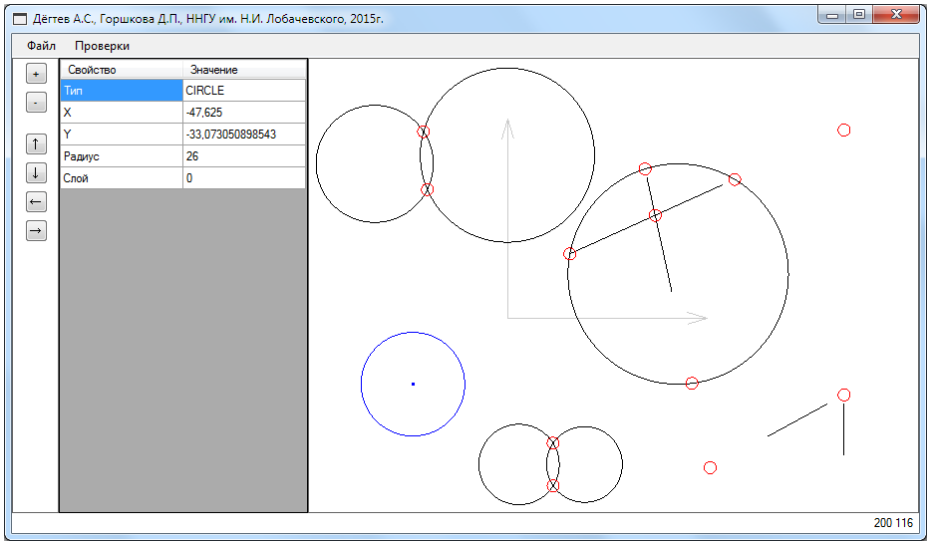


Рисунок 1. Пример ошибки

3. Алгоритмическое обеспечение

3.1. Пересечение двух отрезков

Пример пересечения двух отрезков приведён на рис. 2.

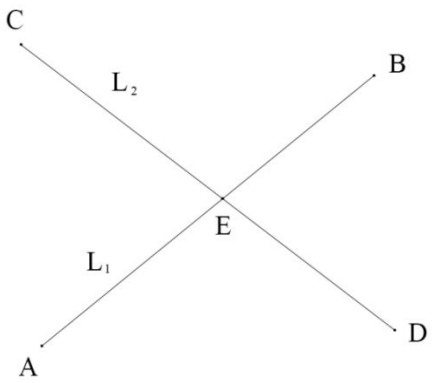


Рисунок 2. Пересечение двух отрезков

Уравнения отрезков имеют вид:

$$\boldsymbol{E}_{L_1} = \boldsymbol{A} + U_{L_1}(\boldsymbol{B} - \boldsymbol{A}) \quad (1)$$

$$\boldsymbol{E}_{L_2} = \boldsymbol{A} + U_{L_2}(\boldsymbol{B} - \boldsymbol{A}) \quad (2)$$

Так как $\boldsymbol{E}_{L_1}$ и $\boldsymbol{E}_{L_2}$ являются одной и той же точкой, то уравнения (1) и (2) равны, из чего можно получить следующую систему:

$$\begin{cases} x_1 + U_{L_1}(x_2 - x_1) = x_3 + U_{L_2}(x_4 - x_3) \\ y_1 + U_{L_1}(y_2 - y_1) = y_3 + U_{L_2}(y_4 - y_3) \end{cases} \quad (3)$$

Решая её относительно $\boldsymbol{U}_{L_1}$ и $\boldsymbol{U}_{L_2}$ получим:

$$\boldsymbol{U}_{L_1} = \frac{(x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)} \quad (4)$$

$$\boldsymbol{U}_{L_2} = \frac{(x_2 - x_1)(y_1 - y_3) - (y_2 - y_1)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)} \quad (5)$$

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

$$x = x_1 + \boldsymbol{U}_{L_1}(x_2 - x_1) \quad (6)$$

$$y = y_1 + \boldsymbol{U}_{L_2}(y_2 - y_1) \quad (7)$$

Если $\boldsymbol{U}_{L_1} \in [0; 1]$ и $\boldsymbol{U}_{L_2} \in [0; 1]$, то точка принадлежит и первому и второму отрезку, значит это действительное пересечение отрезков, иначе пересечение будет мнимым.

3.2. Пересечение отрезка и окружности

Пример пересечения отрезка и окружности приведён на рис. 3.

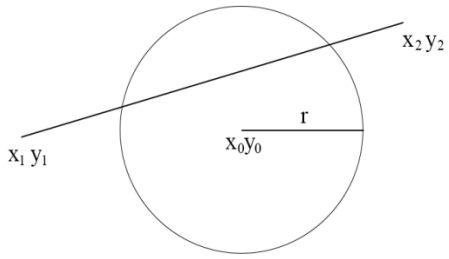


Рисунок 3. Пересечение отрезка и окружности

Для нахождения точек пересечения отрезка и окружности нужно решить систему:

$$\begin{cases} \dfrac{y - y_1}{y_2 - y_1} = \dfrac{x - x_1}{x_2 - x_1} & (8) \\ x^2 + y^2 = r^2 & (9) \end{cases}$$

Преобразуем уравнение (8) к уравнению вида

$$y = ax + b, \quad (10)$$

где

$$a = \frac{y_2 - y_1}{x_2 - x_1};\ b = y_1 - ax_1 \quad (11, 12)$$

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

Если $x \in [x_1; x_2]$ и $y \in [y_1; y_2]$, то пересечение будет действительным, иначе оно мнимое, т. к. точка лежит вне отрезка.

3.3. Пересечение отрезка и дуги

Пример пересечения отрезка и дуги приведён на рис. 4.

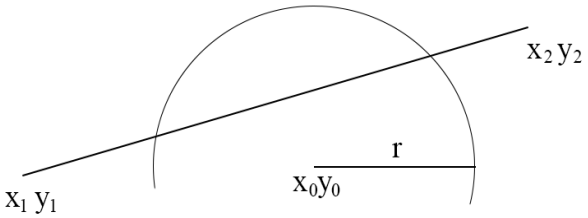


Рисунок 4. Пересечение отрезка и дуги

Поиск точек пересечения отрезка и дуги осуществляется точно так же как и при поиске пересечений отрезка и окружности.

Далее находим угол α между прямой, проходящей через точку $(x_0; y_0)$ и параллельной оси абсцисс, и прямой, проходящей через точки $(x_0; y_0)$ и $(x; y)$.

Если $\alpha \in [\alpha_{\text{нач}}; \alpha_{\text{кон}}]$, то эта точка является точкой пересечения отрезка и дуги.

Если $x \in [x_1; x_2]$ и $y \in [y_1; y_2]$, то пересечение будет действительным, иначе оно мнимое, т. к. точка лежит вне отрезка.

3.4. Пересечение отрезка и эллипса

Пусть изначально имеем следующее расположение отрезка и эллипса (рис. 5):

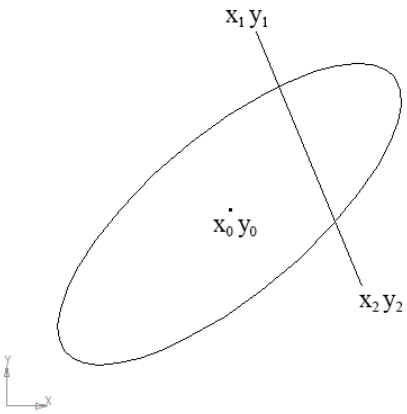


Рисунок 5. Пересечение отрезка и эллипса

Для упрощения вычислений выполним сдвиг системы координат так, чтобы центр эллипса оказался в точке $(0; 0)$.

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 (концов отрезка) координаты будут равны $(x_1''; y_1'')$ и $(x_2''; y_2'')$ соответственно.

И получим систему:

$$\begin{cases} \dfrac{y - y_1''}{y_2'' - y_1''} = \dfrac{x - x_1''}{x_2'' - x_1''} & (13) \\ \dfrac{x^2}{a^2} + \dfrac{y^2}{b^2} = 1 & (14) \end{cases}$$

Преобразуем уравнение (13) к виду:

$$y = kx + c, \quad (15)$$

где

$$k = \frac{y_2'' - y_1''}{x_2'' - x_1''} \quad (16)$$

$$c = y_1'' - kx_1'' \quad (17)$$

И уравнение (14) к виду:

$$b^2x^2 + a^2y^2 - a^2b^2 = 0 \quad (18)$$

Подстановкой (15) в (18) получим:

$$b^2x^2 + a^2(k^2x^2 + 2kxc + c^2) - a^2b^2 = 0 \quad (19)$$

$$(b^2 + a^2)x^2 + 2a^2kcx + a^2c^2 - a^2b^2 = 0 \quad (20)$$

Найдя корни уравнения (20) и подставив их в (15), получим координаты точек пересечения отрезка и эллипса $(x_{1t}''; y_{1t}'')$ и $(x_{2t}''; y_{2t}'')$ (рис. 6).

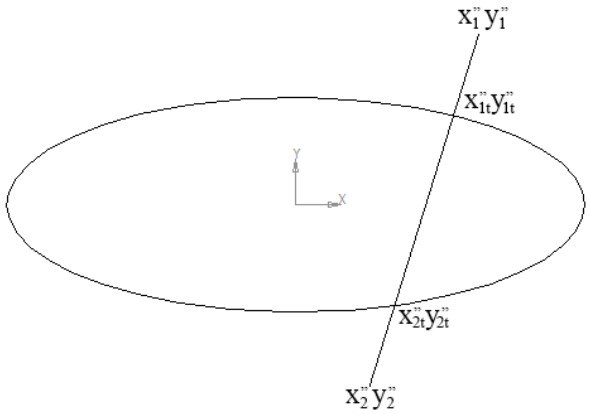


Рисунок 6. Пересечение отрезка и эллипса с найденными точками пересечения

Далее обратным преобразованием координат (поворот на угол $-\alpha$ и переноса центра координат на x_0 и y_0) получаем координаты точек пересечения в исходной системе координат $(x_{1t}; y_{1t})$ и $(x_{2t}; y_{2t})$ соответственно.

3.5. 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.

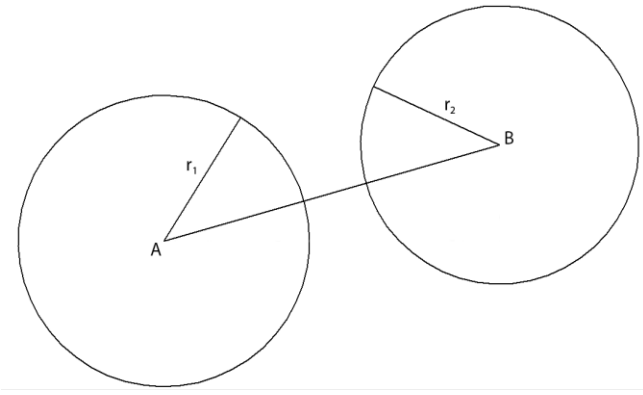


Рисунок 7. Расстояние между двумя окружностями

Расстояние между центрами двух окружностей (рис. 7) в евклидовом пространстве размерности 2 определяется по формуле:

$$AB = \sqrt{(A.x - B.x)^2 + (A.y - B.y)^2} \quad (21)$$

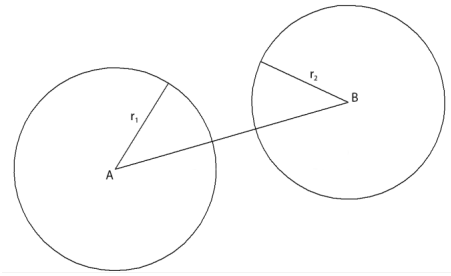


Рисунок 8. Окружности не имеют общих точек

Две окружности не будут иметь общих точек (рис. 8), если расстояние между их центрами больше суммы радиусов.

$$AB > r_1 + r_2 \quad (22)$$

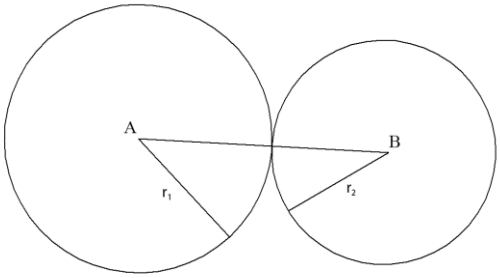


Рисунок 9. Окружности имеют одну общую точку

Две окружности будут иметь общую точку (рис. 9), если расстояние между их центрами равно сумме радиусов.

$$AB = r_1 + r_2 \quad (23)$$

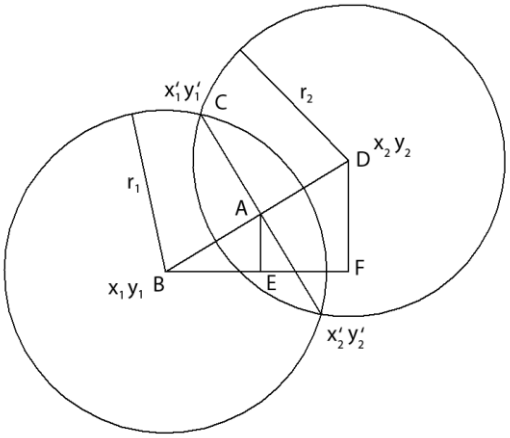


Рисунок 10. Окружности имеют две общих точки

Две окружности будут иметь две общих точки (рис. 10), если расстояние между их центрами меньше суммы радиусов.

$$BD < r_1 + r_2 \quad (24)$$

Одна окружность будет находиться внутри другой при $AB < |r_1 - r_2|$.

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

$$x = x_1 + \frac{(x_2 - x_1)r_1}{AB} \quad (25)$$

А координата y рассчитывается по формуле

$$y = y_1 + \frac{(y_2 - y_1)r_1}{AB} \quad (26)$$

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:

$$S = \frac{1}{2} BD\ AC \tag{27}$$

$$S = \sqrt{p(p - BC)(p - BD)(p - CD)}, \tag{28}$$

где

$$p = \frac{BC+BD+CD}{2}, \tag{29}$$

т.е.

$$AC = 2\frac{\sqrt{p(p-BC)(p-BD)(p-CD)}}{BD}. \tag{30}$$

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

$$AB = \sqrt{BC^2 - AC^2}\ . \tag{31}$$

Из подобия треугольников BDF и ABE находим AE

$$AE = \frac{DF\ AB}{BD} \tag{32}$$

Соответственно

$$BE = \sqrt{AB^2 - AE^2} \tag{33}$$

Таким образом, координаты точки A будут равны:

$$A_x = x_1 + BE \tag{34}$$

$$A_y = y_1 + AE \tag{35}$$

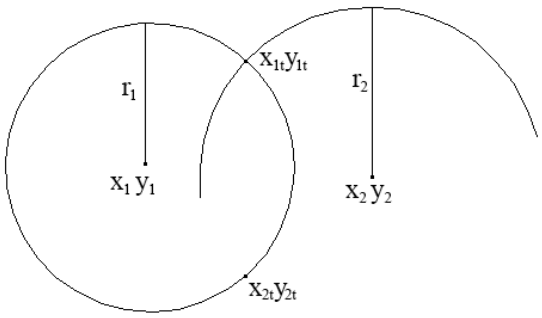
Координаты точек пересечения находятся следующим образом:

$$x_1' = A_x + \frac{(y_2 - y_1)AC}{BD};\ y_1' = A_y - \frac{(x_2 - x_1)AC}{BD} \tag{36}$$

$$x_2' = A_x - \frac{(y_2 - y_1)AC}{BD};\ y_2' = A_y + \frac{(x_2 - x_1)AC}{BD} \tag{37}$$

3.6. Пересечение окружности и дуги

Пример пересечения окружности и дуги приведён на рис. 11.



Поиск точек пересечения окружности и дуги осуществляется точно так же, как и при поиске пересечений двух окружностей.

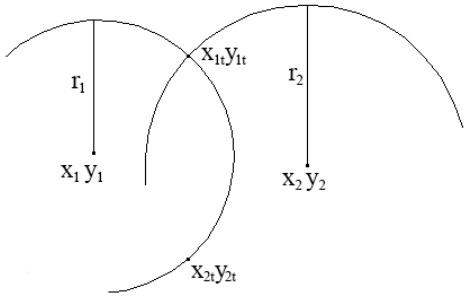
Далее находим угол α между прямой, проходящей через центр дуги и параллельной оси абсцисс, и прямой, проходящей через точки $(x_2; y_2)$ и $(x_{1t}; y_{1t})$.

Рисунок 11. Пересечение окружности и дуги

Если $\alpha \in [\alpha_{\text{нач}}; \alpha_{\text{кон}}]$, то эта точка является точкой пересечения отрезка и эллипса. Аналогично нужно поступить с точкой $(x_{2t}; y_{2t})$.

3.7. Пересечение дуг

Пример пересечения двух дуг приведён на рис. 12.



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 α_1 между прямой, параллельной оси абсцисс и проходящей через точку $(x_1; y_1)$, и прямой, проходящей через точки $(x_1; y_1)$ и $(x_{1t}; y_{1t})$; и угол α_2, между прямой, параллельной оси абсцисс и проходящей через точку $(x_2; y_2)$, и прямой, проходящей через точки $(x_2; y_2)$ и $(x_{1t}; y_{1t})$.

Рисунок 12. Пересечение двух дуг

Если $\alpha_1 \in [\alpha_{1\,\text{нач}}; \alpha_{1\,\text{кон}}]$ и $\alpha_2 \in [\alpha_{2\,\text{нач}}; \alpha_{2\,\text{кон}}]$, то найденная точка является точкой пересечения двух дуг.

Аналогичные действия выполняются для точки $(x_{2t}; y_{2t})$.

3.8. 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:

- $C \in [A; B]$;
- $B \in [C; D]$;
- Точки C и D лежат на прямой AB.

4. Информационное обеспечение

4.1. 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:

- C++ 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 [16] 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).

4.1.1. 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.

4.2. Информационное обеспечение комплекса

4.2.1. Базовые структуры данных библиотеки 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.

4.2.2. Базовые структуры данных библиотеки 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.

4.2.3. Базовые структуры данных библиотеки 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.

5. Разработанное программное обеспечение

5.1. Условия функционирования

Для использования созданного ПО необходимы:

- операционная система: OS Windows 7 64bit с архитектурой процессора x64;
- процессор 64-разрядный (x64) процессор (с 1 или более ядер) с тактовой частотой 1 ГГц или выше;
- 2 ГБ оперативной памяти (ОЗУ);
- монитор с пространственным разрешением не ниже 1024x768px;
- видеопроцессор Intel GMA HD или лучше;
- 104-клавишная клавиатура,
- 3-клавишный манипулятор «мышь»;
- установленные библиотеки OpenGL, версии не ниже 2.0;

Созданное ПО интегрально содержит 6 классов было разработано на языках C++/CLI и C# в среде Microsoft Visual Studio 2012. ПО интегрально содержит 6 классов, объём программного кода составляет ~ 4.5 тыс. операторов.

5.2. Библиотека работы с dxf-файлами

5.2.1. Назначение

Библиотека служит для выполнения базовых операций с файлами формата dxf, версии AutoCAD 2007, реализована в виде класса <DXF>.

5.2.2. Спецификация класса (файл Cl_DxfLib.h)

Спецификация класса приведена в Приложение А.

5.2.3. Приватные методы

Методы, перечисленные в этом разделе, объявлены приватными, так как не входят в интерфейс работы с рассматриваемой библиотекой и не предполагают непосредственной работы пользователя с ними.

Метод чтения отрезка из dxf-файла

Прототип: void ReadLine().

Входные и выходные параметры: отсутствуют.

Описание: метод void ReadLine() вызывается методом int Read() при обнаружении объекта LINE в обрабатываемом файле. Данный метод 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.

Метод чтения полилинии из dxf-файла

Прототип: void ReadPolyline().

Входные и выходные параметры: отсутствуют.

Описание: метод void ReadPolyline() вызывается методом int Read() при обнаружении объекта LWPOLYLINE в обрабатываемом файле. Данный метод создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор _Section.Entities.Polylines.

Метод чтения вспомогательной линии из dxf-файла

Прототип: void ReadXLine().

Входные и выходные параметры: отсутствуют.

Описание: метод void ReadXLine() вызывается методом int Read() при обнаружении объекта XLINE в обрабатываемом файле. Данный метод 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.

Метод чтения окружности из dxf-файла

Прототип: void ReadCircle().

Входные и выходные параметры: отсутствуют.

Описание: метод void ReadCircle() вызывается методом int Read() при обнаружении объекта CIRCLE в обрабатываемом файле. Данный метод создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор _Section.Entities.Circles.

Фрагмент кода метода:

```
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());

Метод чтения блокового объекта из dxf-файла

Прототип: void ReadBlock().

Входные и выходные параметры: отсутствуют.

Описание: метод void ReadBlock() вызывается методом int Read() при обнаружении объекта BLOCK в обрабатываемом файле. Данный метод создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор Blocks. Во время выполнения метода происходит анализ блока посредством чтения строк файла и при обнаружении объекта LINE вызывается метод void ReadLine().

Метод чтения объекта типа вставка из dxf-файла

Прототип: void ReadInsert().

Входные и выходные параметры: отсутствуют.

Описание: метод void ReadInsert() вызывается методом int Read() при обнаружении объекта INSERT в обрабатываемом файле. Данный метод создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор _Section.Entities.Inserts.

Метод чтения эллипса из dxf-файла

Прототип: void ReadEllipse().

Входные и выходные параметры: отсутствуют.

Описание: метод void ReadEllipse() вызывается методом int Read() при обнаружении объекта ELLIPSE в обрабатываемом файле. Данный метод создаёт временную переменную, считывает из файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор _Section.Entities.Ellipses.

Метод чтения дуги из dxf-файла

Прототип: void ReadArc().

Входные и выходные параметры: отсутствуют.

Описание: метод void ReadArc() вызывается методом int Read() при обнаружении объекта ARC в обрабатываемом файле. Данный метод создаёт временную переменную, считывает из

файла информацию об обнаруженном объекте во временную переменную и добавляет её в вектор _Section.Entities.Arcs.

Метод преобразования числового значения в строчном представлении в числовое значение в целочисленном представлении.

Прототип: int TextToHex(string str).

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.

Описание: метод преобразования шестнадцатеричного числа, представленного в строковом выражении, в шестнадцатеричное число типа integer.

Метод преобразования отрезка в строковое представление

Прототип: string LineToString(LINE l).

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.

Описание: метод преобразования объекта LINE в строковое представление для записи в файл dxf.

Метод преобразования полилинии в строковое представление

Прототип: string PolylineToString(LWPOLYLINE l).

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.

Описание: метод преобразования объекта LWPOLYLINE в строковое представление для записи в файл dxf.

Метод преобразования окружности в строковое представление

Прототип: string CircleToString(CIRCLE c).

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.

Описание: метод преобразования объекта CIRCLE в строковое представление для записи в файл dxf.

Метод преобразования отметки об ошибке в строковое представление

Прототип: string ErrPointToString(CROSSPOINT point).

Входные и выходные параметры: Входной параметр CROSSPOINT point с представляет собой объект типа CROSSPOINT.

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

Описание: метод преобразования объекта CROSSPOINT в строковое представление для записи в файл dxf.

Метод преобразования имени слоя в строковое представление слоя

Прототип: string LayerToString(string name).

Входные и выходные параметры: Входной параметр string name с представляет собой имя слоя.

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

Описание: метод преобразования имени слоя в его строковое представление для записи в файл dxf.

5.2.4. Публичные методы

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

Метод открытия файла

Прототип: bool Open(string dxfFileName).

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.

Описание: метод открытия файла по указанному имени и пути к файлу.

Метод чтения файла

Прототип: int Read().

Входные и выходные параметры: Входные параметры отсутствуют.

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

Описание: метод 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 соответствующий ему метод, который получает информацию о текущем объекте и записывает её в переменные.

Фрагмент кода метода:

```
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;
}
```

Метод записи отметок об ошибках в файл

Прототип: bool SaveErrorPoints(vector<CROSSPOINT> errPoints).

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.

Описание: метод сохранения файла с отметками о возможных ошибках.

Метод записи полилинии в файл

Прототип: bool SavePolyLine(string _saveFileName, double **points, int size, bool closed).

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.

Описание: метод сохранения файла с полилинией, передаваемой в параметрах.

Метод закрытия файла

Прототип: void Close().

Входные и выходные параметры: отсутствуют.

Описание: метод закрытия файла и обнуления использованных переменных.

Метод, возвращающий значение переменной _Section

Прототип: SECTION GetSection().

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.

Описание: метод предоставляет доступ к приватной переменной _Section.

5.3. Библиотека работы с объектами dxf-файла

5.3.1. Назначение

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, реализована в виде класса <OBJECTS>.

5.3.2. Спецификация класса (файл Cl_ObjectsLib.h)

Спецификация класса приведена в Приложение В.

5.3.3. Приватные методы

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

Метод определения пересечения двух отрезков

Прототип: void CrossLines(LINE line1, LINE line2, unsigned char crossKind).

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.

Описание: метод определения пересечения двух отрезков. Алгоритм её работы описан в разделе 3.1. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool line_line равным true в BITSFIELD bitsfield.

Метод определения пересечения отрезка и окружности

Прототип: void CrossLineCircle(LINE line, CIRCLE circle, unsigned char crossKind).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод определения пересечения отрезка и окружности. Алгоритм её работы описан в разделе 3.2. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool line_circle равным true в BITSFIELD bitsfield.

Фрагмент кода метода:

```
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;
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;
```

Метод определения пересечения отрезка и дуги

Прототип: void CrossLineArc(LINE line, ARC arc, unsigned char crossKind).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод определения пересечения отрезка и дуги. Алгоритм её работы описан в разделе 3.3. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool line_arc равным true в BITSFIELD bitsfield.

Метод определения пересечения отрезка и эллипса

Прототип: void CrossLineEllipse(LINE line, ELLIPSE ellipse, unsigned char crossKind).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод определения пересечения отрезка и эллипса. Алгоритм её работы описан в разделе 3.4. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool line_ellipse равным true в BITSFIELD bitsfield.

Метод определения пересечения двух окружностей

Прототип: void CrossCircles(CIRCLE circle1, CIRCLE circle2).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод определения пересечения двух окружностей. Алгоритм её работы описан в разделе 3.5. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool circle_circle равным true в BITSFIELD bitsfield.

Метод определения пересечения окружности и дуги

Прототип: void CrossCircleArc(CIRCLE c, ARC a).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод определения пересечения окружности и дуги. Алгоритм её работы описан в разделе 3.6. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool circle_arc равным true в BITSFIELD bitsfield.

Метод определения пересечения двух дуг

Прототип: void CrossArcs(ARC a1, ARC a2).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод определения пересечения двух дуг. Алгоритм её работы описан в разделе 3.7. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool arc_arc равным true в BITSFIELD bitsfield.

Метод определения совпадения двух отрезков

Прототип: void OverlapLines(LINE line1, LINE line2).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод определения совпадения (полного и частичного) двух отрезков. Алгоритм её работы описан в разделе 3.8. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool overlap_lines равным true в BITSFIELD bitsfield.

Метод определения совпадения двух окружностей

Прототип: void OverlapCircles(CIRCLE circle1, CIRCLE circle2).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод определения совпадения двух окружностей. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool overlap_circles равным true в BITSFIELD bitsfield.

Фрагмент кода метода:

```
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);}
```

Метод определения совпадения двух дуг

Прототип: void OverlapArcs(ARC a1, ARC a2).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод определения совпадения (полного и частичного) двух дуг. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool overlap_arcs равным true в BITSFIELD bitsfield.

Метод определения совпадения двух эллипсов

Прототип: void OverlapEllipses(ELLIPSE e1, ELLIPSE e2).

Входные и выходные параметры:

Входные параметры:

– e1 – первый объект типа ELLIPSE;

– e2 – второй объект типа ELLIPSE.

Выходные параметры отсутствуют.

Описание: метод определения совпадения двух эллипсов. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool overlap_ellipses равным true в BITSFIELD bitsfield.

Метод определения совпадения окружности и дуги

Прототип: void OverlapCircleArc(CIRCLE c, ARC a).

Входные и выходные параметры:

Входные параметры:

– c – первый объект типа CIRCLE;

– a – второй объект типа ARC.

Выходные параметры отсутствуют.

Описание: метод определения совпадения окружности и дуги. Вызывается из метода void CheckCross(BITSFIELD bitsfield) при bool overlap_circle_arc равным true в BITSFIELD bitsfield.

Метод вычисления детерминанта

Прототип: double Det(double x1, double x2, double x3, double x4).

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

Выходной параметр имеет тип double и является значением детерминанта матрицы.

Описание: метод вычисления детерминанта матрицы $\begin{pmatrix} x1 & x2 \\ x3 & x4 \end{pmatrix}$.

Метод определения угла между двумя прямыми

Прототип: double GetAngle(double x0, double y0, double x1, double y1).

Входные и выходные параметры:

Входные параметры:

– x0, y0 – координаты точки $(x0, y0)$;

– x1, y1 – координаты точки $(x1, y1)$.

Выходной параметр имеет тип double и является значением искомого угла.

Описание: метод определения угла между прямой, проведённой через точку $(x0, y0)$ и параллельной оси абсцисс, и прямой, проведённой через точки $(x0, y0)$ и $(x1, y1)$.

Метод определения принадлежности точки (x, y) линии

Прототип: bool IsPointOnLine(LINE line, double x, double y).

Входные и выходные параметры:

Входные параметры:

- line – линия;
- x, y – координаты точки (x, y).

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

Описание: метод определения принадлежности точки (x, y) линии line.

Метод отределения расстояния между двумя точками

Прототип: double GetDistance(double x0, double y0, double x1, double y1).

Входные и выходные параметры:

Входные параметры:

- x0, y0 – координаты точки $(x0, y)0$;
- x1, y1 – координаты точки $(x1, y1)$.

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

Описание: метод определения расстояния между точкой $(x0, y0)$ и точкой $(x1, y1)$.

5.3.4. Публичные методы

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

Метод проверки пересечений и совпадений объектов

Прототип: void CheckCross(BITSFIELD 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.

Описание: в методе происходит попарная передача объектов, содержащихся в переменной Section, в методы, где непосредственно выполняется проверка пересечений и совпадений, в соответствии с флагами, выставленными в передаваемом параметре bitsfield.

Метод инициализации структуры BITSFIELD

Прототип: void InitBitsField(BITSFIELD *field).

Входные и выходные параметры: Входной параметр BITSFIELD *field – указатель на структуру типа BITSFIELD.

Выходные параметры отсутствуют.

Описание: метод инициализации структуры BITSFIELD.

Метод обработки клика мыши по объекту

Прототип: void ClickOnObject(double x, double y, double scale, DataGridView^ table).

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.

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

5.4. Библиотека работы с графикой

5.4.1. Назначение

Библиотека служит для работы с графической частью программы: отображение, масштабирование и перемещение объектов чертежа посредством OpenGL, реализована в виде класса <GRAPHICS>.

5.4.2. Спецификация класса (файл Cl_ObjectsLib.h)

Спецификация класса приведена в Приложение С.

5.4.3. Публичные методы

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

Метод получающий элемент для работы с графикой

Прототип: void GetOpenGlControl(SimpleOpenGlControl^ simpleOpenGlControl).

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

Выходные параметры отсутствуют.

Описание: метод, получающий элемент Windows Forms, для дальнейшей работы с ним.

Метод инициализации OpenGL

Прототип: int InitGL().

Входные и выходные параметры: Входные параметры отсутствуют.

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

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

Метод изменения размеров области отрисовки

Прототип: void Resize(double width, double height).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод изменения размеров области отрисовки.

Метод задания текущего цвета для отрисовки элементов

Прототип: void Render().

Входные и выходные параметры: отсутствуют.

Описание: метод задания текущего цвета для отрисовки элементов, отрисовка осей абсцисс и ординат.

Фрагмент кода метода:

```
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();
```

Метод очистки буферов OpenGL

Прототип: void SwapBuffers().

Входные и выходные параметры: отсутствуют.

Описание: метод очистки буферов OpenGL. Посредством чего отрисовываются элементы, находившиеся в этих буферах.

Метод изменения масштаба отрисовываемых элементов

Прототип: void Scale(signed short int s).

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.

Описание: метод изменения масштаба отрисовываемых элементов.

Метод смещения отображаемых элементов

Прототип: void Translate(double x, double y).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод смещения отображаемых элементов в горизонтальном и вертикальном направления.

Метод отображения точки

Прототип: void DrawPoint(double x, double y, float size).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод отображения точки заданного размера в заданных координатах.

Метод отображения отрезка

Прототип: void DrawLine(double x1, double y1, double x2, double y2).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод отображения отрезка по заданным координатам.

Метод отображения окружности

Прототип: void DrawCircle(double x, double y, double r).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод отображения окружности с заданными координатами центра и радиусом.

Фрагмент кода метода:

```
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();
```

Метод отображения точки ошибки

Прототип: void DrawErrorPoint(double x, double y, double r).

Входные и выходные параметры:

Входные параметры:

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

Выходные параметры отсутствуют.

Описание: метод отображения точки пересечения (совпадения) с заданными координатами центра и радиусом. Принцип работы заключается в изменении цвета отображения и вызове метода DrawCircle.

Метод отображения дуги

Прототип: void DrawArc(double x, double y, double r, double begin, double end).

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.

Описание: метод отображения дуги с заданными координатами центра, радиуса, начальным и конечным углами.

Метод отображения эллипса

Прототип: void DrawEllipse(double x, double y, double a, double b, double angle).

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.

Описание: метод отображения эллипса с заданными координатами центра, длинами большой и малой полуосей, углом наклона.

Метод установки цвета

Прототип: void SetColor(float red, float green, float blue).

Входные и выходные параметры:

Входные параметры:

- red – интенсивность красного цвета;
- green – интенсивность зелёного цвета;

– blue – интенсивность голубого цвета.

Выходные параметры отсутствуют.

Описание: метод установки цвета модели RGB, которым будут отображаться объекты. Параметры могут принимать значения в диапазоне [0.0; 1.0].

Метод возвращающий значение переменной IsInit

Прототип: bool GetIsInit().

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр принимает значение boolean и является значением переменной IsInit.

Описание: метод, возвращающий значение переменной IsInit.

5.5. Используемые сторонние библиотеки

Библиотека The Tao Framework [11] – 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.

6. Руководство пользователя к разработанному программному продукту

6.1. Интерфейс

Запустив приложение, пользователь увидит главный интерфейс программы (рис. 13).

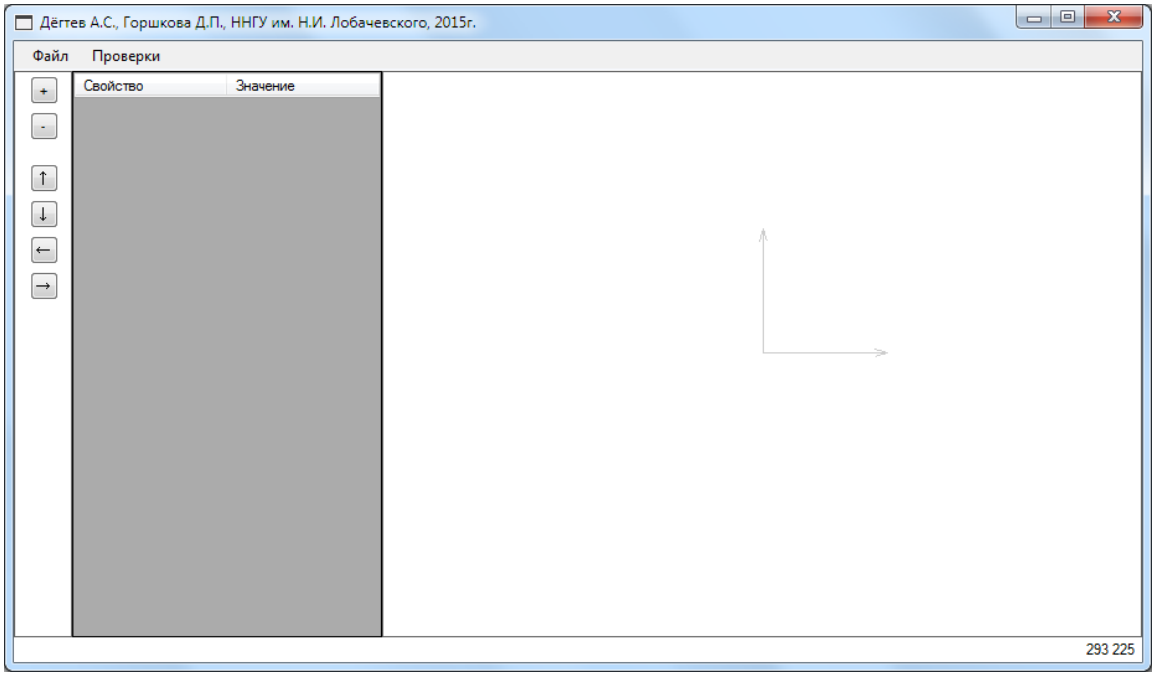


Рисунок 13. Главный интерфейс

6.2. Описание клавиш

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

«ФАЙЛ» содержит в себе пункты «ОТКРЫТЬ», «СОХРАНИТЬ», «ЗАКРЫТЬ», «ВЫХОД» (рис. 14):

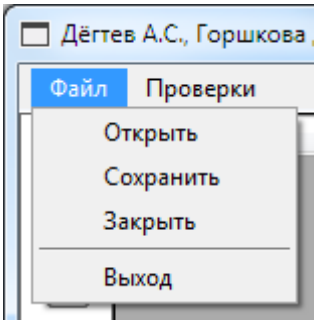


Рисунок 14. Вкладка "ФАЙЛ"

Клавиши «+» и «-» (рис. 15) отдаляют и приближают рисунок соответственно. При помощи клавиш со стрелками (рис. 15) можно перемещать чертёж по полю отображения.

Рисунок 15. Клавиши манипулирования чертежом

6.3. Открытие файла

Чтобы открыть файл формата dxf нужно выбрать пункт «ОТКРЫТЬ» и, выбрав с помощью открывшегося диалогового окна нужный файл, нажать кнопку «ОТКРЫТЬ» (рис. 16).

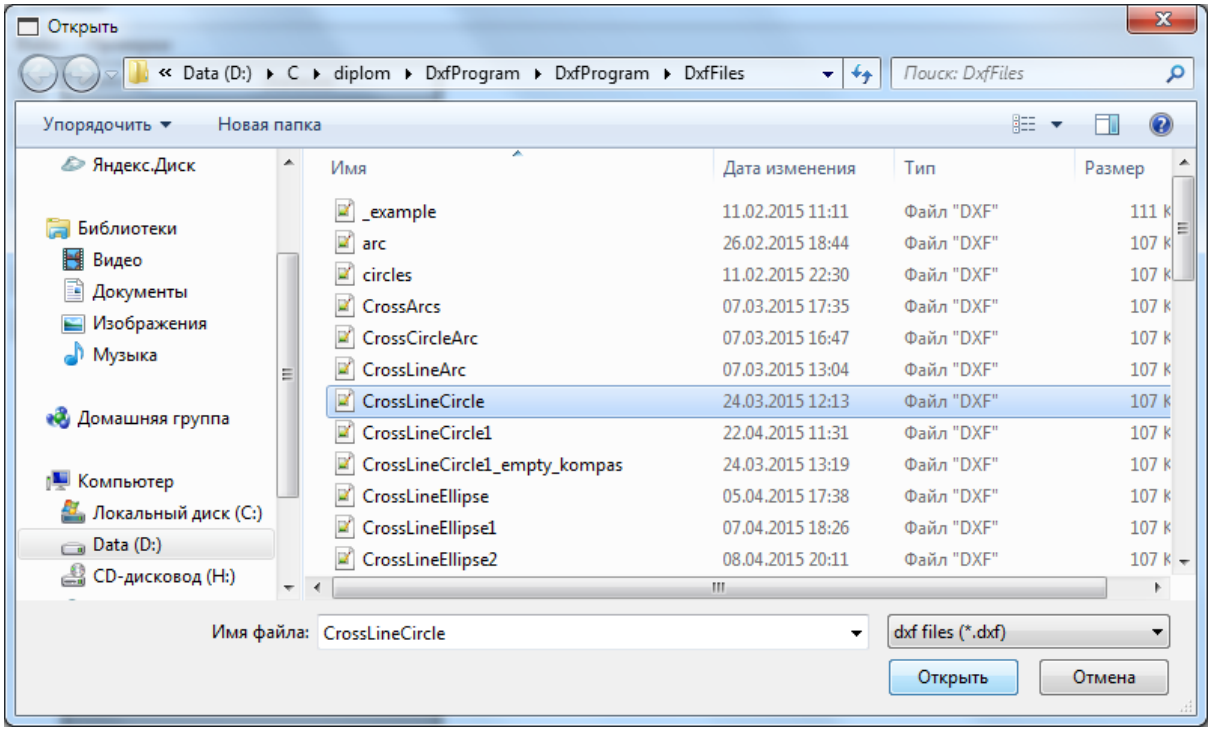


Рисунок 16. Диалог открытия файла

Загрузив файл, пользователь увидит следующее (рис. 17):

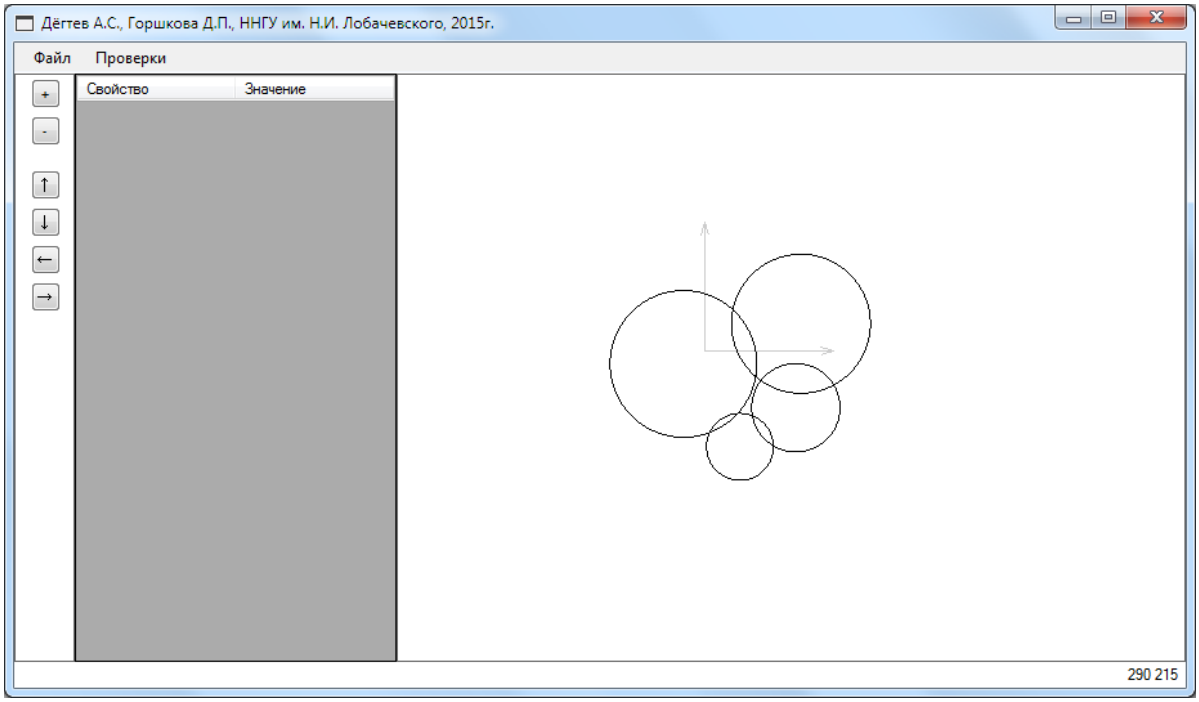


Рисунок 17. Открытый файл

В правом нижнем углу окна будут показаны координаты точки (рис. 18), на которую указывает мышь:

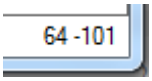


Рисунок 18. Координаты указателя мыши

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.

6.4. Проверка файла

На панели быстрого доступа размещена клавиши «ПРОВЕРКИ», нажав на которую пользователь увидит следующее (рис. 19):

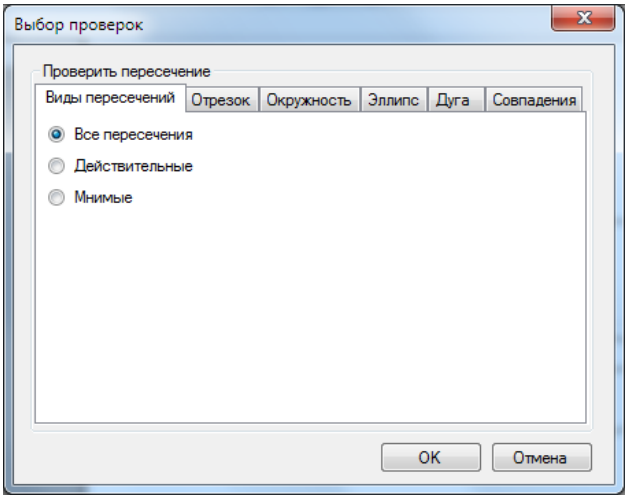


Рисунок 19. Окно выбора проверок

Если файл ещё не был открыт, то пользователь увидит сообщение об ошибке (рис. 20):

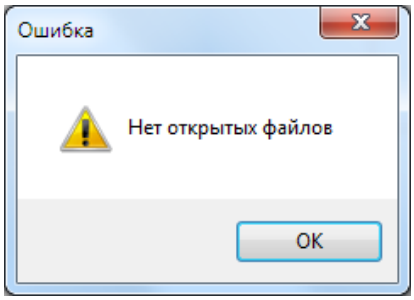


Рисунок 20. Ошибка: нет открытых файлов

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

Если же файл был открыт, то пользователь увидит окно (рис. 21), 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.

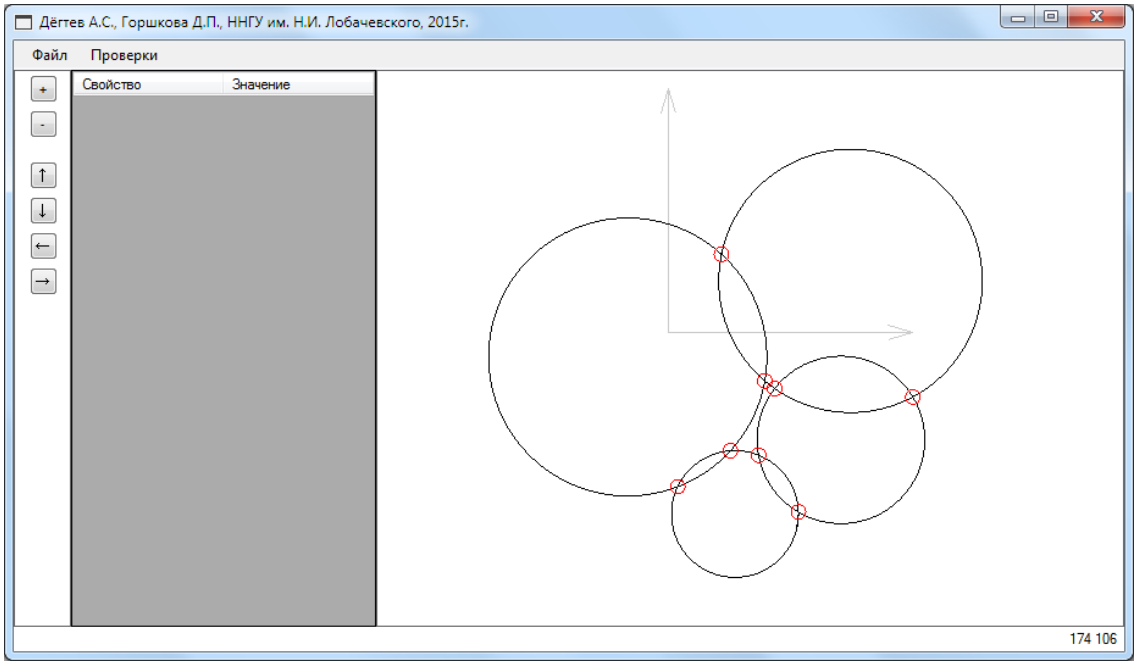


Рисунок 21. Отметки возможных ошибок

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 (рис. 22), а сам объект подсветится синим цветом:

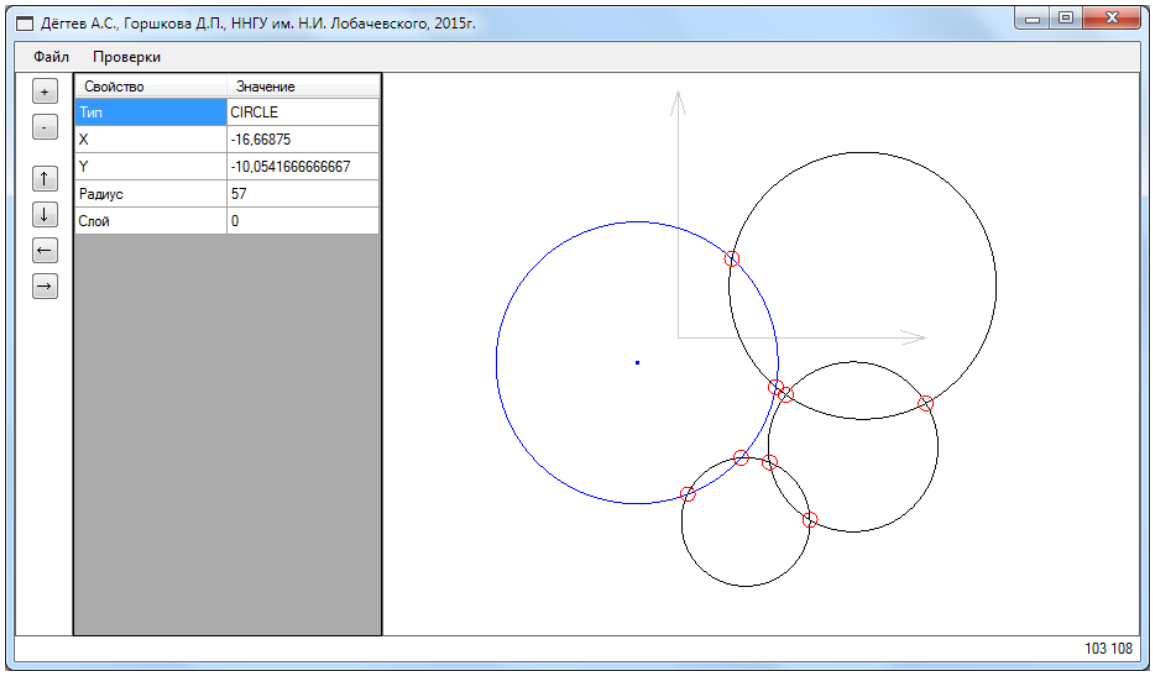


Рисунок 22. Свойства выбранного объекта

6.5. Сохранение файла

Для сохранения открытого нужно из пункта «ФАЙЛ» выбрать «СОХРАНИТЬ» (рис. 23).

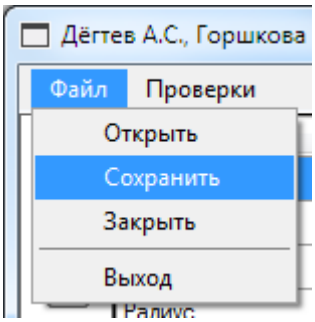


Рисунок 23. Сохранение файла

Если файл ещё не был открыт, то пользователь увидит сообщение об ошибке (рис. 24):

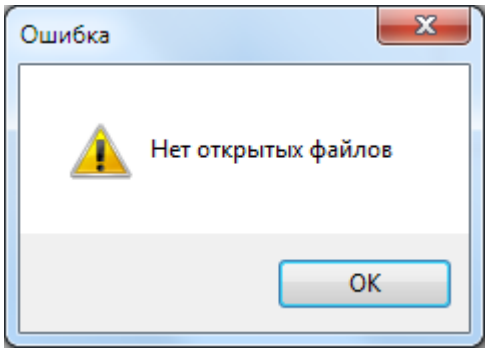


Рисунок 24. Ошибка: нет открытых файлов

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

Если файл был открыт, то пользователь увидит окно сохранения файла (рис. 25). Для того чтобы добавить информацию о чертеже в базу данных необходимо выбрать «Добавить файл в базу данных» и заполнить информационные поля.

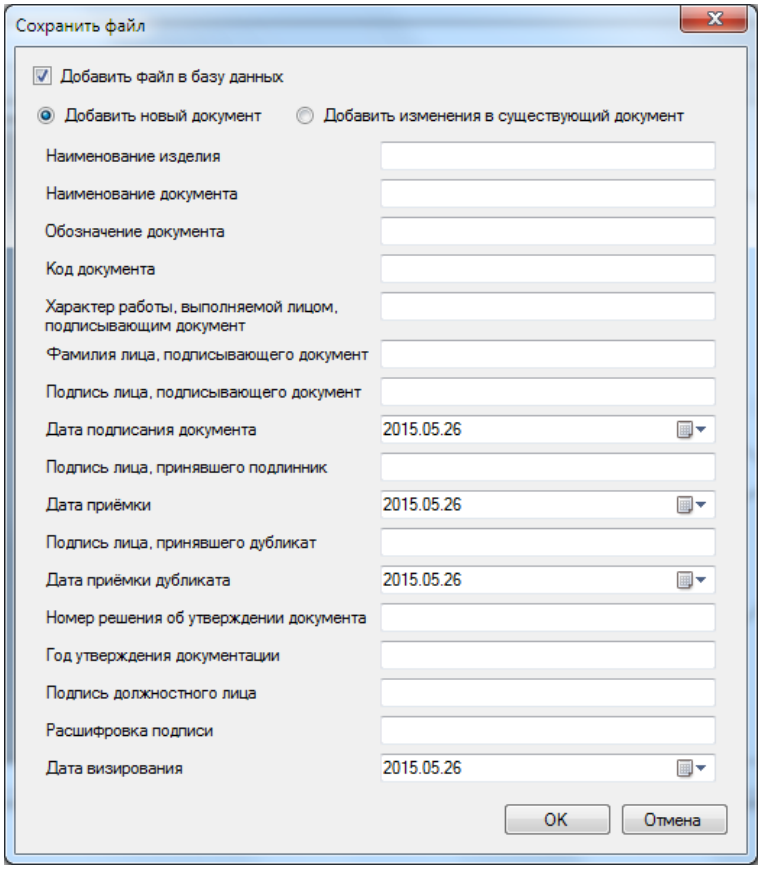


Рисунок 25. Добавление нового документа

Если же документ в базе уже существует, то нужно нажать на «Добавить изменения в существующий документ» (рис. 26).

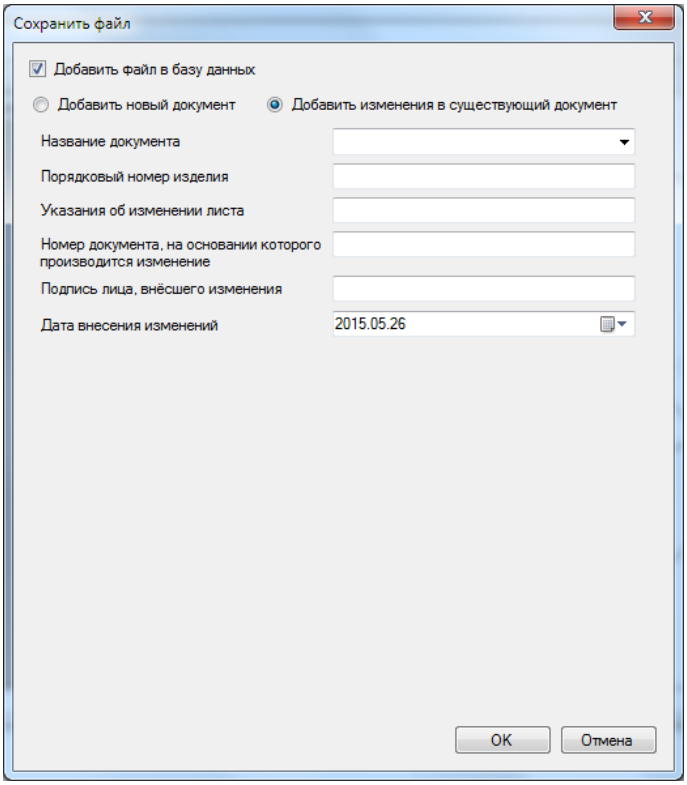


Рисунок 26. Добавить изменения в существующий документ

По окончании заполнения полей необходимо нажать клавишу «ОК». После чего информация будет занесена в электронный архив (посредством подсистемы, разработанной Горшковой Д.П.), 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 (рис. 27):

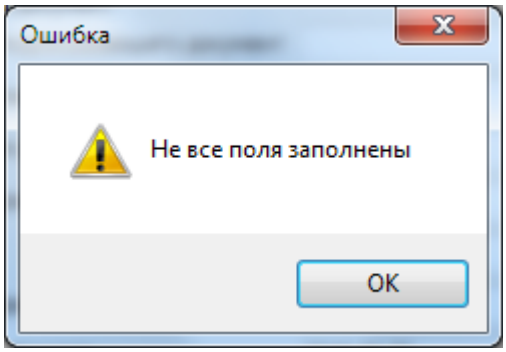


Рисунок 27. Ошибка: не все поля заполнены

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» (рис. 28).

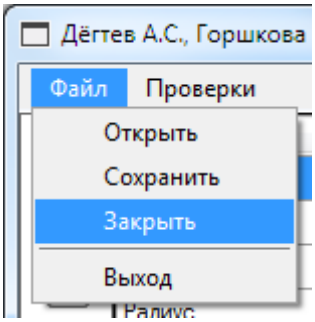


Рисунок 28. Закрытие файла

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

7. Редактирование обнаруженных геометрических ошибок в САПР КОМПАС-3D или AutoCAD

Разработанный программный комплекс предназначен только для нахождения возможных ошибок. Непосредственное же их исправление должно осуществляться в специализированных программах, поскольку существующие САПР существенно преуспели в разработке средств редактирования векторной графики, однако нахождение ошибок не является достаточно развитой их направленностью.

В результате работы программы будет получен dxf-файл с записанными координатами точек ошибок на отдельном слое «ErrorsLayer» (рис. 29).

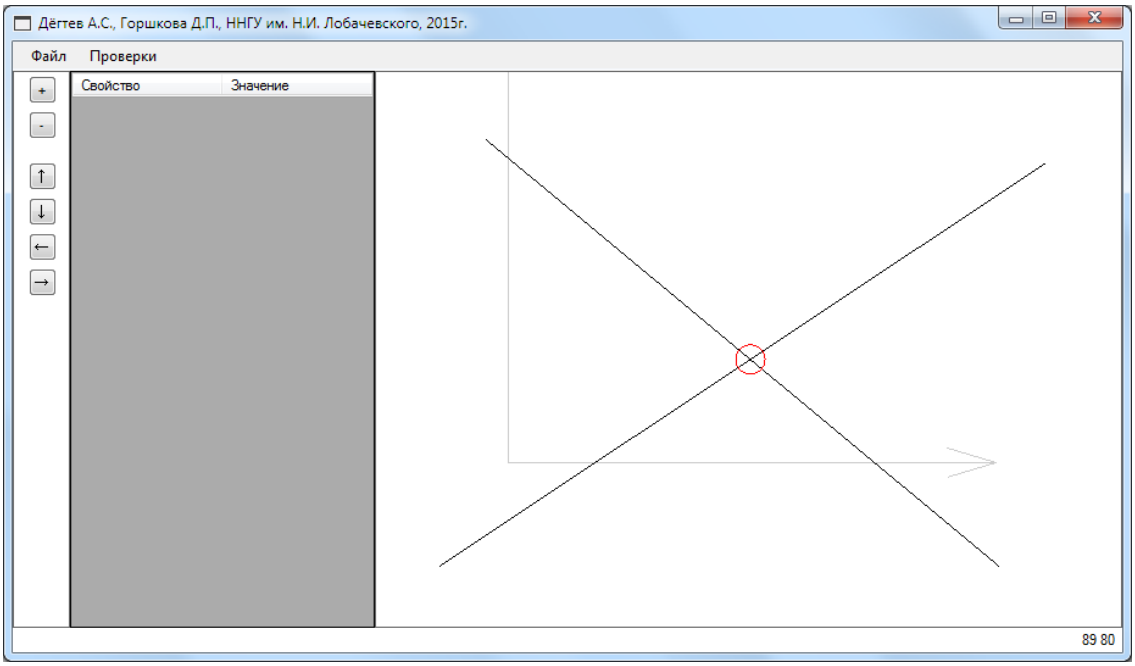


Рисунок 29. Результат работы программы с отрезками

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

Дополнительные примеры результатов работы программного комплекса можно увидеть на рис. 30, рис. 31, рис. 32.

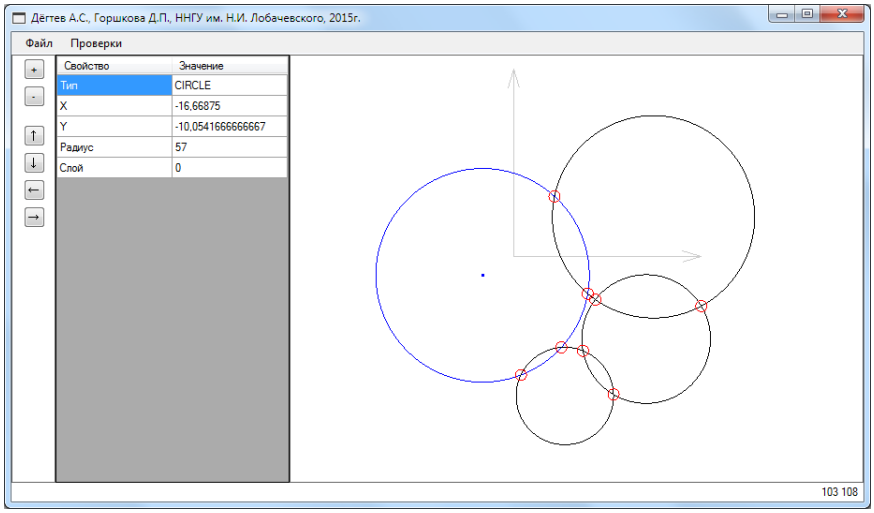


Рисунок 30. Пример результата работы программы с окружностями

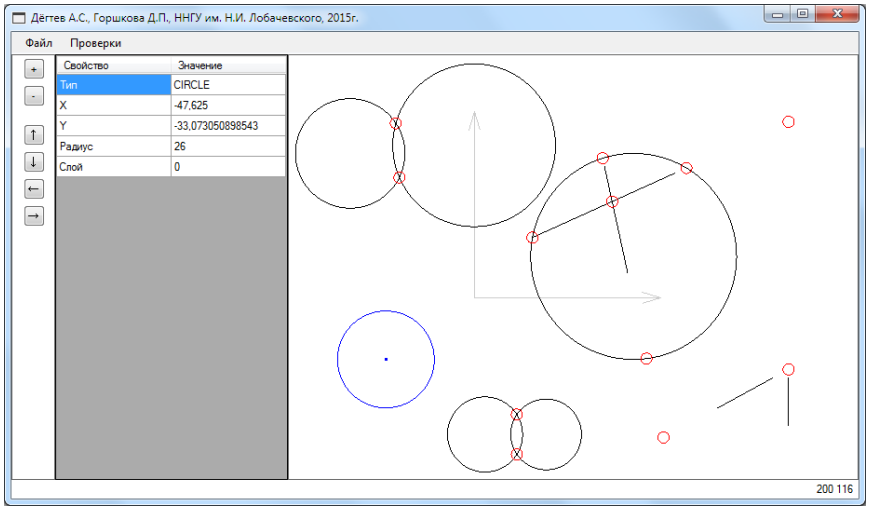


Рисунок 31. Пример результата работы программы с отрезками и окружностями

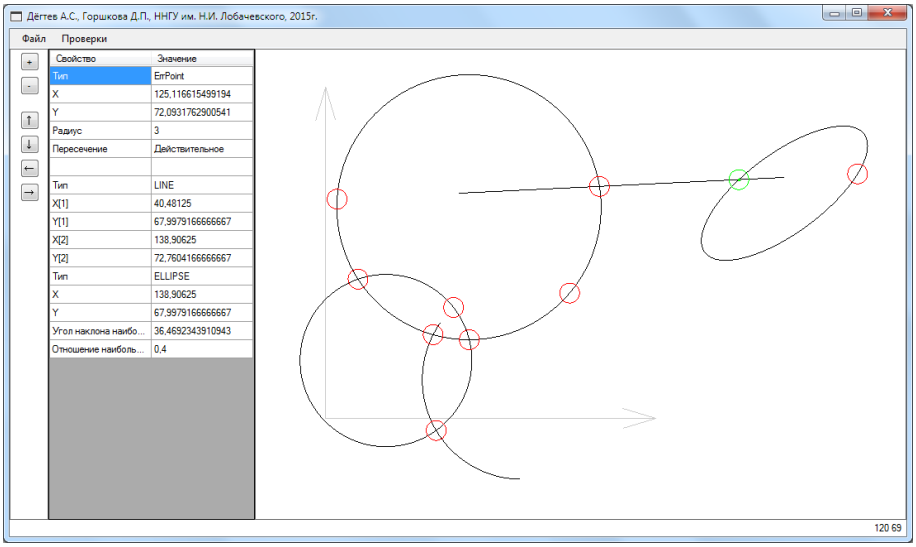


Рисунок 32. Пример результата работы программы с несколькими объектами

7.1. Результат работы разработанной программы в САПР КОМПАС-3D V13

После запуска САПР КОМПАС-3D на экране появится окно (рис. 33):

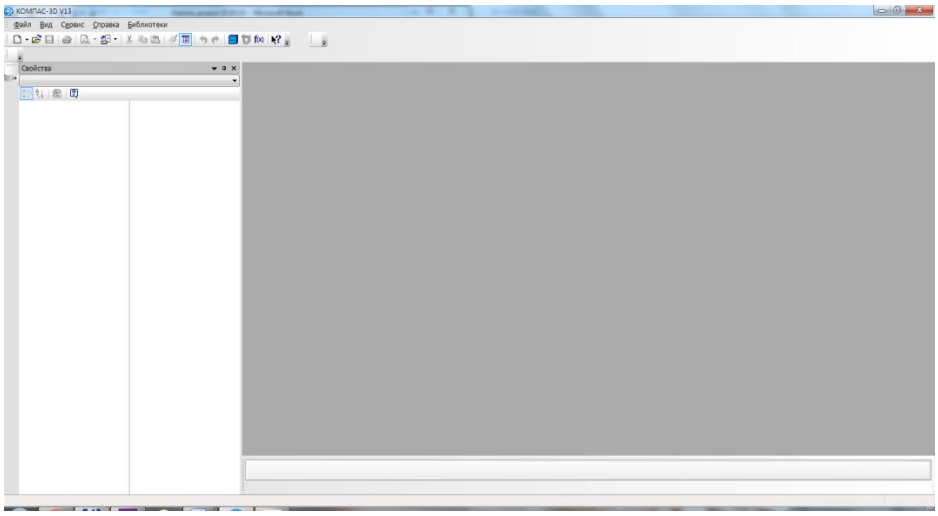


Рисунок 33. Интерфейс КОМПАС-3D

Чтобы открыть полученный файл в КОМПАС-3D нужно на вкладке «ФАЙЛ» выбрать пункт «Открыть…» (рис. 34).

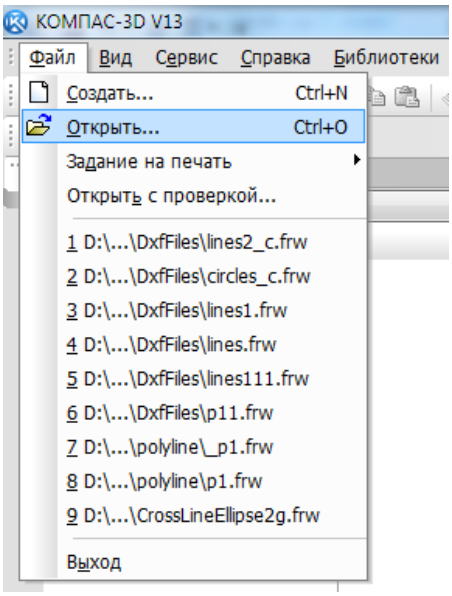


Рисунок 34. Пункт "Открыть..."

В открывшемся окне (рис. 35) выбрать нужный файл и нажать «ОТКРЫТЬ».

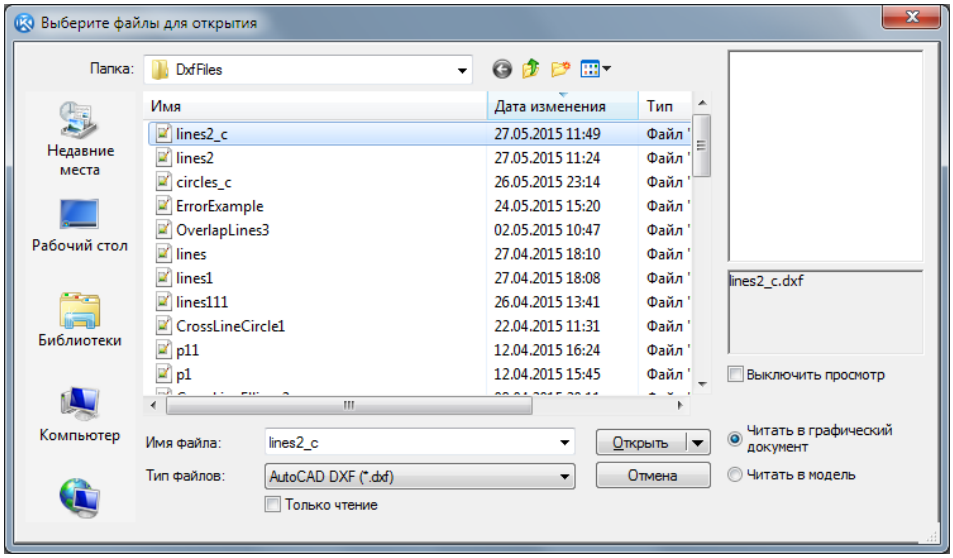


Рисунок 35. Диалог открытия файла

Чертёж с отметкой о возможной ошибке на отдельном слое открыт (рис. 36).

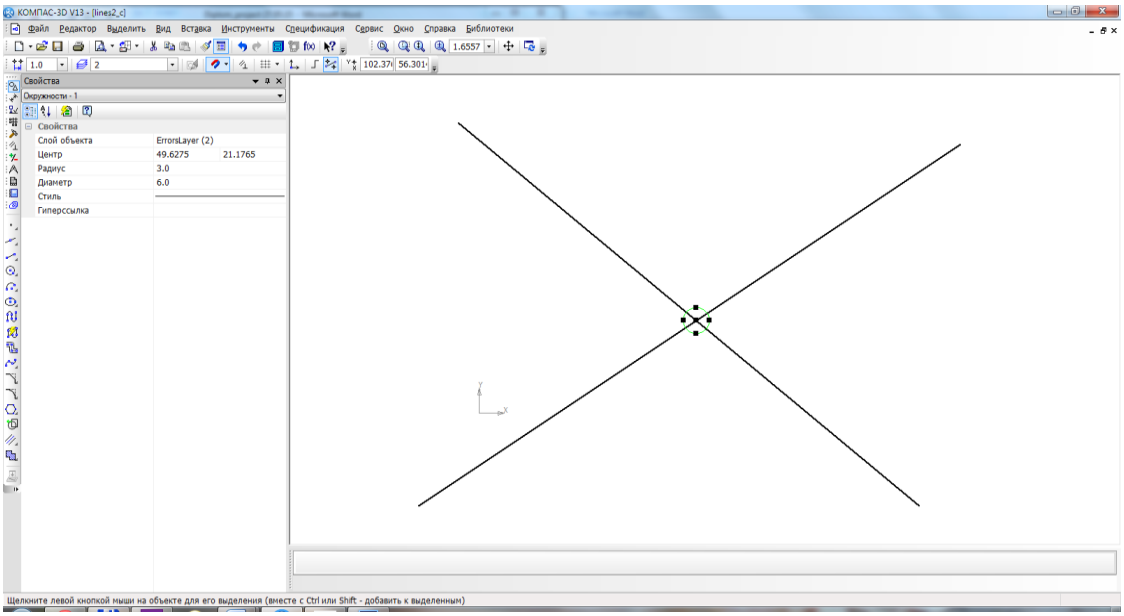


Рисунок 36. Чертёж, открытый в КОМПАС-3D

После исправления ошибки и удаления отметок со слоя ошибок получим (рис. 37):

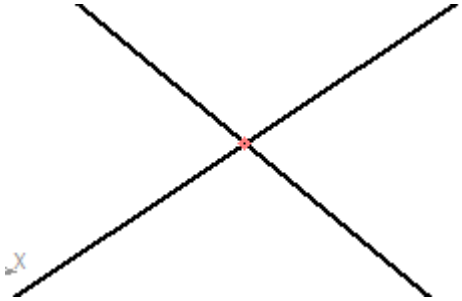


Рисунок 37. Исправленный чертёж

7.2. Результат работы разработанной программы в САПР AutoCAD 2015

После запуска программы САПР AutoCAD на экране появится окно (рис. 38):

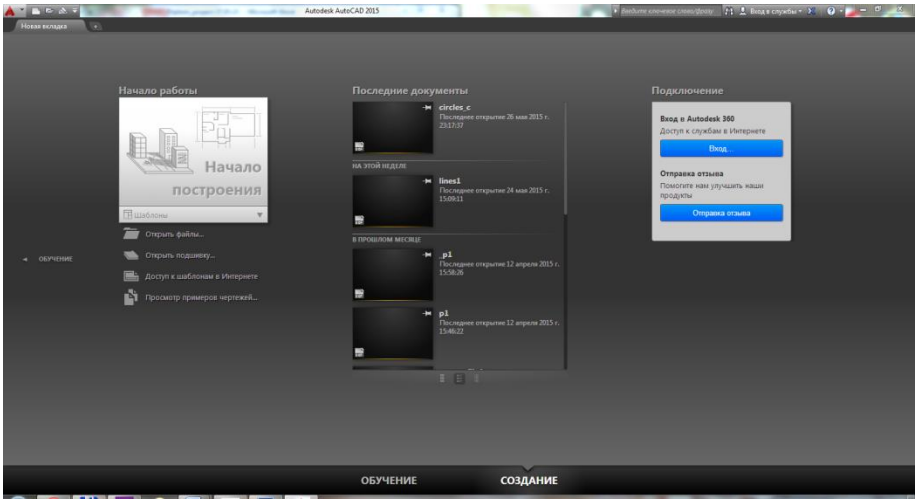


Рисунок 38. Интерфейс AutoCAD

Чтобы открыть полученный файл в КОМПАС-3D нужно нажать на значок программы, на вкладке «Открыть» выбрать пункт «Чертеж» (рис. 39).

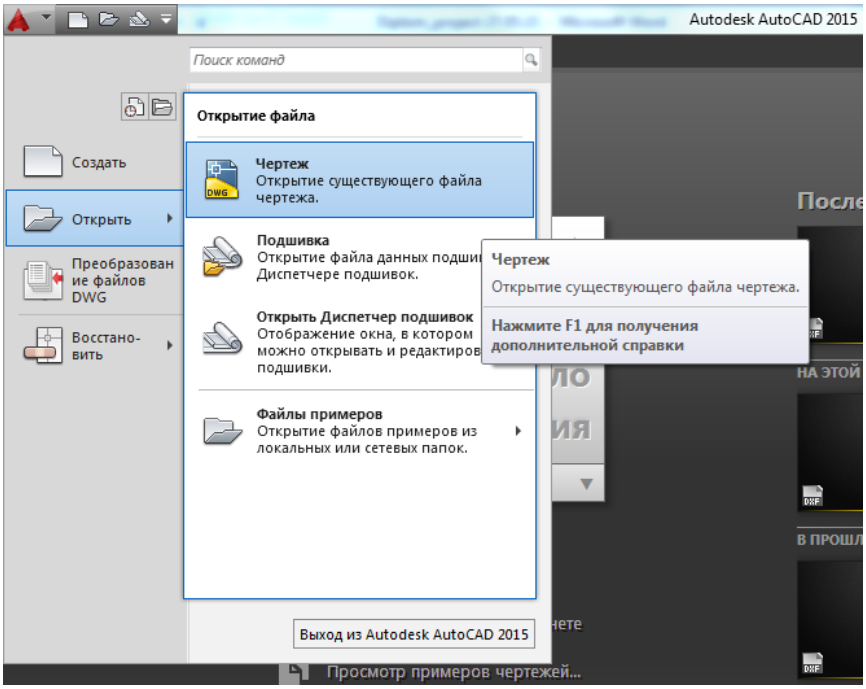


Рисунок 39. Вкладка "Открыть"

В открывшемся окне (рис. 40) выбрать нужный файл и нажать «ОТКРЫТЬ».

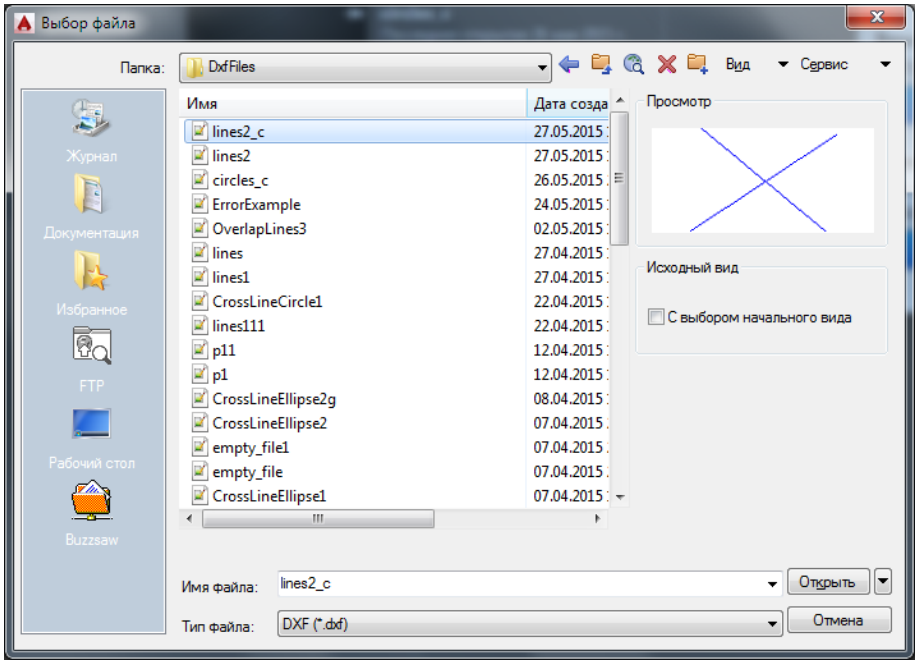


Рисунок 40. Диалог открытия файла

Чертёж с отметкой о возможной ошибке на отдельном слое открыт (рис. 41).

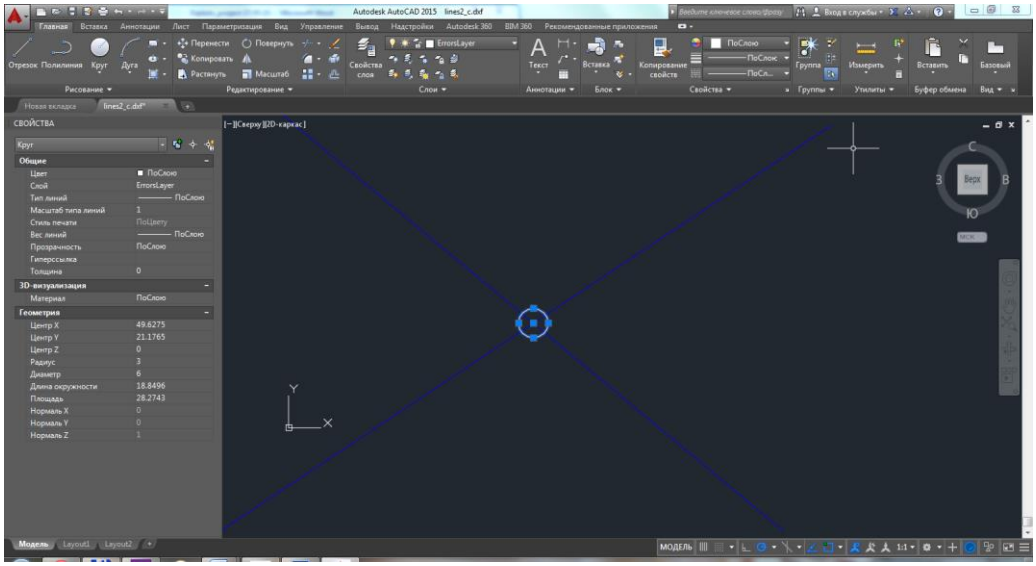


Рисунок 41. Чертёж, открытый в AutoCAD

После исправления ошибки и удаления отметок со слоя ошибок получим (рис. 42):

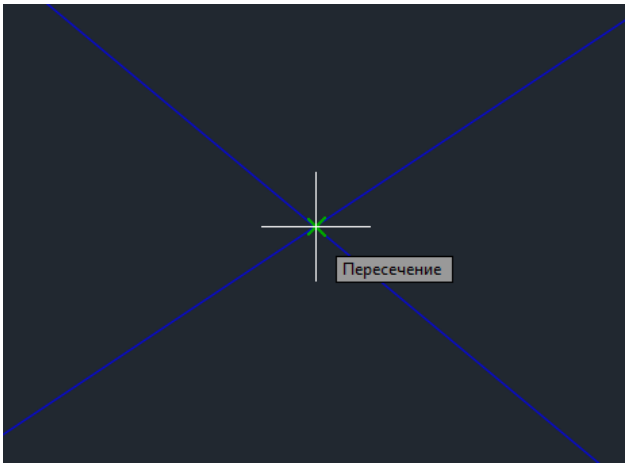


Рисунок 42. Исправленный чертёж

Заключение

По результатам выполнения выпускной квалификационной работы сформулируем основные выводы:

- в результате критического анализа литературных данных установлено, что задача нахождения геометрических ошибок на 2D-машиностроительных чертежах является актуальной, сложной, многоаспектной проблемой в технологиях создания электронных моделей изделий;
- изучены существующие САПР, форматы хранения данных, а также существующие аналоги;
- разработаны следующие алгоритмы:
  - нахождение точки пересечения двух отрезков;
  - нахождение точек пересечения отрезка и окружности;
  - нахождение точек пересечения отрезка и дуги;
  - нахождение точек пересечения отрезка и эллипса;
  - нахождение точек пересечения двух окружностей;
  - нахождение точек пересечения окружности и дуги
  - нахождение точек пересечения двух дуг
  - обнаружение совпадения двух отрезков;
  - обнаружение совпадения двух окружностей;
  - обнаружение совпадения двух дуг;
  - обнаружение совпадения окружности и дуги.
- реализованы библиотеки функций работы с файлами dxf, с объектами, полученными из файла, и с графической частью интерфейса;
- на базе созданных алгоритмов и библиотек создано ПО обнаружения геометрических ошибок на 2D-машиностроительных чертежах;
- проведена практическая апробация программного комплекса и выявлена его работоспособность на реальных данных.

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. – Загл. с экрана.
9. Что такое TrustedDWG? – Режим доступа: http://www.autodesk.ru/adsk/servlet/item?siteID=871736&id=10687507. – Загл. с экрана.
10. Описание формата dxf – Режим доступа: http://www.autodesk.com/techpubs/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. – Загл. с экрана.
15. DXF Reference – Режим доступа: http://www.autodesk.com/techpubs/autocad/acadr14/dxf/. – Загл. с экрана.
16. Использование DXF-файла – Режим доступа: http://gor.h1.ru/15bt1/ebook_1590/theory/part7.htm. – Загл. с экрана.

Приложения

Приложение А. Спецификация класса (файл Cl_DxfLib.h)

```
#ifndef CL_DXFLIB_H
#define CL_DXFLIB_H
#include <stdio.h>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>
#include <math.h>
#include <string>
using namespace std;
#define EPS 1e-9
#define PI 3.1415926535
#define ErrorLayerName "ErrorsLayer"
#define ErrPointRadius 3.0
namespace DXF_SPACE {
	struct POINT {
		double x;
		double y;
		string layer;
	};
	struct LINE {
		POINT p[2];
		unsigned short int type;
		string layer;
		bool current;
	};
	struct LWPOLYLINE {
		vector<POINT> p;
```

```
        string layer;
        bool current;
        bool closed;
        double width;
};
struct CIRCLE {
        POINT p;
        double r;
        string layer;
        bool current;
};
struct ELLIPSE {
        POINT p;
        double ratio;
        double width;
        double height;
        double angle;
        string layer;
};
struct ARC {
        POINT p;
        double r;
        double angleStart;
        double angleEnd;
        string layer;
        bool current;
};
struct BLOCK {
        int beginAddr;
        string number;
};
struct INSERT {
```

```
        string layer;
        string blockNumber;
};
struct ENTITIES {
        vector<POINT> Points;
        vector<LINE> Lines;
        vector<LWPOLYLINE> Polylines;
        vector<CIRCLE> Circles;
        vector<ELLIPSE> Ellipses;
        vector<ARC> Arcs;
        vector<INSERT> Inserts;
};
struct SECTION {
        ENTITIES Entities;
};
struct CROSSPOINT {
        double x;
        double y;
        bool type;
        bool current;
        bool isCrossPoint;
        vector<LINE> Lines;
        vector<CIRCLE> Circles;
        vector<ELLIPSE> Ellipses;
        vector<ARC> Arcs;
};
class DXF {
private:
        fstream file;
        string fileName;
        ofstream saveFile;
        string saveFileName;
```

```
        int LastNumberOfObject;
        int EndEntitiesPointer;
        SECTION _Section;
        int TextToHex(string str);
        string LineToString(LINE l);
        string PolylineToString(LWPOLYLINE l);
        string CircleToString(CIRCLE c);
        string ErrPointToString(CROSSPOINT point);
        string LayerToString(string name);
        void ReadLine();
        void ReadPolyline();
        void ReadXLine();
        void ReadCircle();
        void ReadBlock();
        void ReadInsert();
        void ReadEllipse();
        void ReadArc();
    public:
        int NumBlock;
        vector<BLOCK> Blocks;
        DXF();
        ~DXF();
        bool Open(string dxfFileName);
        int Read();
        bool SaveErrorPoints(vector<CROSSPOINT> errPoints);
        bool SavePolyLine(string _saveFileName, double **points, int size, bool closed);
        void Close();
        SECTION GetSection() const { return _Section; }
        string GetSaveFilePath() const { return saveFileName; }
    };
}
#endif
```

Приложение В. Спецификация класса (файл Cl_ObjectsLib.h)

```
#ifndef CL_OBJECTSLIB_H
#define CL_OBJECTSLIB_H
#include "Cl_DxfLib.h"
#include "Cl_GraphicsLib.h"
namespace OBJECTS_SPACE {
    using namespace System;
    using namespace System::Windows::Forms;
    using namespace DXF_SPACE;
    class OBJECTS: public DXF {
    private:
        double Det(double x1, double x2, double x3, double x4);
        double GetAngle(double x0, double y0, double x1, double y1);
        double GetAngle1(double x1, double y1, double c);
        bool IsPointOnLine(LINE line, double x, double y);
        double GetDistance(double x0, double y0, double x1, double y1);
        void CrossLines(LINE line1, LINE line2, unsigned char crossKind);
        void CrossLineCircle(LINE line, CIRCLE circle, unsigned char crossKind);
        void CrossLineEllipse(LINE line, ELLIPSE ellipse, unsigned char crossKind);
        void CrossLineArc(LINE line, ARC arc, unsigned char crossKind);
        void CrossCircles(CIRCLE circle1, CIRCLE circle2);
        void CrossCircleArc(CIRCLE c, ARC a);
        void CrossArcs(ARC a1, ARC a2);
        void OverlapLines(LINE line1, LINE line2);
        void OverlapCircles(CIRCLE circle1, CIRCLE circle2);
        void OverlapEllipses(ELLIPSE e1, ELLIPSE e2);
        void OverlapArcs(ARC a1, ARC a2);
        void OverlapCircleArc(CIRCLE c, ARC a);
        bool IsLineCircleCross(LINE line, CIRCLE circle);
        bool IsCirclesCross(CIRCLE c1, CIRCLE c2);
        bool IsArcCircleCross(ARC arc, CIRCLE circle);
    public:
```

```
            struct BITSFIELD {
                unsigned char crossKind;
                bool line_line;
                bool line_circle;
                bool line_ellipse;
                bool line_arc;
                bool circle_circle;
                bool circle_ellipse;
                bool circle_arc;
                bool ellipse_ellipse;
                bool ellipse_arc;
                bool arc_arc;
                bool overlap_lines;
                bool overlap_circles;
                bool overlap_ellipses;
                bool overlap_arcs;
                bool overlap_circle_arc;
            };
            SECTION Section;
            vector<CROSSPOINT> ErrPoints;
            OBJECTS(SECTION s);
            void CheckCross(BITSFIELD bitsfield);
            void InitBitsField(BITSFIELD *field);
            void ClickOnObject(double x, double y, double scale, DataGridView^ table);
        };
}
#endif
```

Приложение С. Спецификация класса (файл Cl_GraphicsLib.h)

```
#ifndef CL_GRAPHICS_H
#define CL_GRAPHICS_H
#include "Cl_DxfLib.h"
```

```
namespace GRAPHICS_SPACE {
	using namespace System::Windows::Forms;
	using namespace Tao::OpenGl;
	using namespace Tao::FreeGlut;
	using namespace Tao::Platform::Windows;
	ref class GRAPHICS {
	private:
		SimpleOpenGlControl^ OpenGlControl;
		bool IsInit;
	public:
		GRAPHICS();
		~GRAPHICS();
		void GetOpenGlControl(SimpleOpenGlControl^ simpleOpenGlControl);
		int InitGL();
		void Resize(double width, double height);
		void Render();
		void SwapBuffers();
		void Scale(signed short int s);
		void Translate(double x, double y);
		void DrawPoint(double x, double y, float size);
		void DrawLine(double x1, double y1, double x2, double y2);
		void DrawCircle(double x, double y, double r);
		void DrawErrorPoint(double x, double y, double r);
		void DrawArc(double x, double y, double r, double begin, double end);
		void DrawEllipse(double x, double y, double a, double b, double angle);
		void SetColor(float red, float green, float blue);
		bool GetIsInit() { return IsInit; }
	};
}
#endif
```