

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра анализа данных и искусственного интеллекта

Допустить к защите
заведующий кафедрой
д-р тех. наук, доцент

А.В. Коваленко

2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

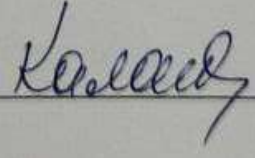
**РАЗРАБОТКА И АВТОМАТИЗАЦИЯ ПРОГРАММНОГО
ОБЕСПЕЧЕНИЯ ДЛЯ ЧПУ ОБОРУДОВАНИЯ В
ПРОИЗВОДСТВЕННЫХ ПРЕДПРИЯТИЯХ**

Работу выполнил(а)  Д. Д. Михеев

Направление подготовки 09.03.03 Прикладная информатика

Направленность (профиль) Прикладная информатика в экономике

Научный руководитель
д-р тех. наук, доцент  А. В. Коваленко

Нормоконтролер
канд. физ.-мат. наук, доцент  Г.В. Калайдина

Краснодар
2024

РЕФЕРАТ

Выпускная квалификационная работа 50 с., 48 рис., 10 источн.
ЧПУ, РАЗРАБОТКА, C#, WINDOWS FORMS, ФРЕЗЕРНЫЙ СТАНОК,
ТЕСТИРОВАНИЕ, АВТОМАТИЗАЦИЯ

В данной работе были изучены теоретические основы разработки программного обеспечения, выбран инструментарий и выполнена реализация приложения с помощью C#.

Цель – разработка и оптимизация программного обеспечения для ЧПУ оборудования в производственных предприятиях.

Задачи:

- применить и адаптировать основы разработки программного обеспечения для ЧПУ оборудования;
- провести сравнительный анализ существующих на рынке программных решений;
- разработать алгоритм генерации G-code;
- разработать и протестировать программное обеспечение.

Объект исследования – программное обеспечение для ЧПУ. Предмет исследования – C#. Итог проделанной работы – создание программного обеспечения на C#, которое пишет готовый G-code.

СОДЕРЖАНИЕ

Введение	4
1 Теоретические аспекты ЧПУ оборудования	5
1.1 Что такое ЧПУ оборудование	5
1.2 Исторический обзор развития ЧПУ технологий	6
1.3 Основные принципы работы фрезерного станка ЧПУ	8
1.4 G-code – язык ЧПУ	9
2 Анализ существующих программных решений для ЧПУ оборудования и проектирование программного обеспечения	11
2.1 Обзор основных программных продуктов на рынке	11
2.2 Сравнительный анализ функциональности, производительности, стоимости	14
2.3 Оценка преимуществ и недостатков различных решений	15
2.4 Определение требований к программному обеспечению	17
2.5 Разработка архитектуры системы	19
2.6 Выбор технологий и инструментов разработки	20
3 Практическая реализация программного обеспечения	23
3.1 Реализация пользовательского интерфейса	23
3.2 Реализация основных функциональных блоков	27
3.3 Отладка и тестирование программы на примере текстовых данных	37
3.4 Отладка и тестирование программы на примере изображения	42
3.5 Описание разработанного алгоритма	45
Заключение	49
Список использованных источников	50

ВВЕДЕНИЕ

В современном производственном секторе автоматизация играет ключевую роль в повышении эффективности и конкурентоспособности предприятий. Особое внимание уделяется разработке и оптимизации программного обеспечения для систем ЧПУ (числового программного управления), которые управляют оборудованием на производственных линиях. Эти системы обеспечивают точное и эффективное выполнение различных задач, от обработки материалов до сборки изделий. В данном дипломном проекте рассматривается процесс разработки и оптимизации программного обеспечения для ЧПУ оборудования с целью повышения производительности и качества производства на промышленных предприятиях.

Целью выпускной квалификационной работы является разработка и оптимизация программного обеспечения для ЧПУ оборудования в производственных предприятиях. Результаты этого исследования представляют собой важный вклад в развитие современных технологий производства и автоматизации, способствуя повышению эффективности и конкурентоспособности предприятий.

Выпускная квалификационная работа состоит из трех разделов, введения, заключения, списка использованных источников. В первом разделе подробно описаны теоретические аспекты ЧПУ оборудования, история развития и основы, заложенные в разработку программного обеспечения. Во втором разделе был проведен анализ существующих систем для работы с производственным оборудованием и выявления преимуществ и недостатков этих систем. В третьем разделе описаны и обоснованы выбранные решения для разработки и спроектирована структура программного обеспечения. В последнем разделе был разработан пользовательский интерфейс и само программное обеспечение.

1 Теоретические аспекты ЧПУ оборудования

1.1 Что такое ЧПУ оборудование

Оборудование с ЧПУ (числовым программным управлением) – это система автоматического управления, которая используется для управления различными типами оборудования в производственных процессах, включая станки с ЧПУ и роботов. Эта технология позволяет выполнять точные и сложные операции с материалами или изделиями с помощью предварительно запрограммированного программного обеспечения, устраняя необходимость в ручном труде [2].

Система ЧПУ обычно состоит из компьютера или контроллера, программного обеспечения для создания программ и управления ими, а также устройств ввода-вывода для связи с оборудованием (Рисунок 1).



Рисунок 1 – Контроллер ЧПУ

Особенности современного этапа развития машиностроения характеризуется значительным распространением и использованием многофункциональных станков с ЧПУ. Применение такого типа оборудования позволяет значительно повысить производительность обработки и улучшить качество изготавливаемых деталей. Главная особенность этого оборудования

состоит в том, что движение инструмента относительно обрабатываемой заготовки заранее программируется и записывается в числовой форме [1].

Одним из наиболее существенных преимуществ оборудования с ЧПУ является его гибкость и способность быстро перенастраиваться для решения различных задач, что делает его важнейшим инструментом в современной обрабатывающей промышленности.

В последнее время микропроцессорная технология сделала управление ЧПУ еще дешевле, что привело к появлению ЧПУ для хобби и персонального рынка ЧПУ. Доступное оборудование с ЧПУ также проложило путь к использованию ЧПУ в прототипировании наряду с 3D-печатью [3].

1.2 Исторический обзор развития ЧПУ технологий

В начале XVIII века изобретательный француз Жозеф Мари Жаккард создал прототип станка с ЧПУ. Его ткацкий станок (жаккардовая машина) управлялся куском картона с пробитыми в нужных местах отверстиями. Современный этап истории станков с ЧПУ начался в Соединенных Штатах Америки в конце 40-х годов прошлого века, через полтора столетия после изобретения жаккардовой машины и после окончания Второй мировой войны.

Джон Парсонс, сын владельца Parsons Incorporated и работавший в инженерном отделе компании своего отца, первым запатентовал идею использования станков для обработки таких материалов, как пропеллеры, с помощью программы, которая считывает необходимую информацию с перфокарты и запускает ее [7].

В начале 1949 года ВВС США начали финансировать компанию Parsons, специализирующуюся на производстве лопастей и пропеллеров для вертолетов, для разработки и производства станков, способных обрабатывать по контуру детали сложной формы, производимые для вертолетов, самолетов и других летательных аппаратов.

Компания Парсонса сотрудничала с Массачусетским технологическим институтом на протяжении 1950-х годов. Сотрудники лаборатории сервомеханики института так и не смогли усовершенствовать попавшие к ним в руки конструкции, и Парсонс вскоре был забыт. А его идеи - нет. Однако наибольшего успеха добились конструкторы компании Bendix. Первый станок с ЧПУ был спроектирован и построен в 1954 году. С 1955 года станок поступил в производство и начал активно использоваться (Рисунок 2).

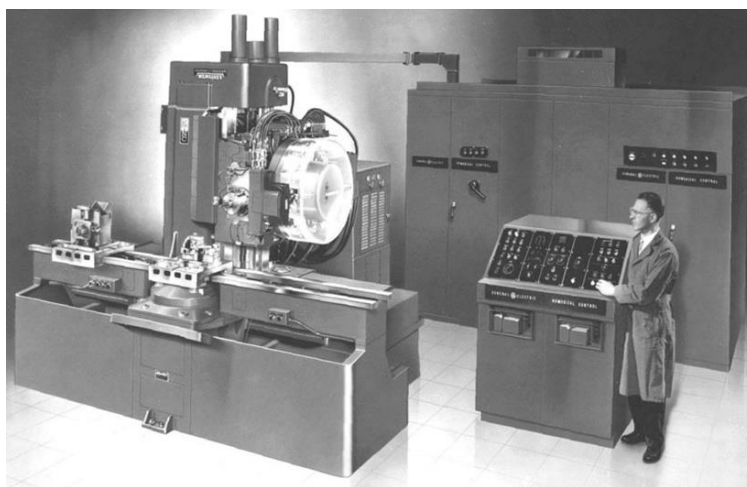


Рисунок 2 – Один из первых ЧПУ станков

Станки с ЧПУ медленно набирали обороты. Предприниматели с недоверием относились к новой технологии. К 1970 году экономика большинства западных стран замедлилась, а стоимость занятости росла; начиная с 1960-х годов станки с ЧПУ, обеспечившие столь необходимую прочную техническую основу, вышли на рынок и стали уверенно вытеснять более старые технологии, такие как гидравлические трассировщики и ручная обработка. Начался взлет. Доступное оборудование с ЧПУ открыло путь к использованию ЧПУ для создания прототипов, наряду с 3D-печатью. Ранее использование ЧПУ ограничивалось в основном производственными предприятиями [7].

1.3 Основные принципы работы фрезерного станка ЧПУ

Наличие числового программного обеспечения в оборудовании освобождает оператора от необходимости следить за процессом изготовления продукта. Станки, оснащенные электронной системой ЧПУ способны работать в режиме автомата, тем самым сокращая время на обработку и исключая возможность постороннего вмешательства. Работа осуществляется при помощи специально разработанных программ для различного вида операций, которые загружаются в систему ЧПУ. На приборной панели оператор может следить за ходом выполняемой операции и вручную контролировать некоторые действия, а также, в случае возникновения аварийной ситуации, отключить приборную панель и систему питания.

Фрезерные станки с чпу осуществляют работу высочайшей точности. Орудия в трехмерном пространстве тремя электромоторами, шпиндель перемещается по оси заготовки, согласно заданному алгоритму программного обеспечения (Рисунок 3). Режущая часть фрезы снимает ненужные слои материала и обрабатывает деталь в соответствии с заранее разработанной схемой [5].

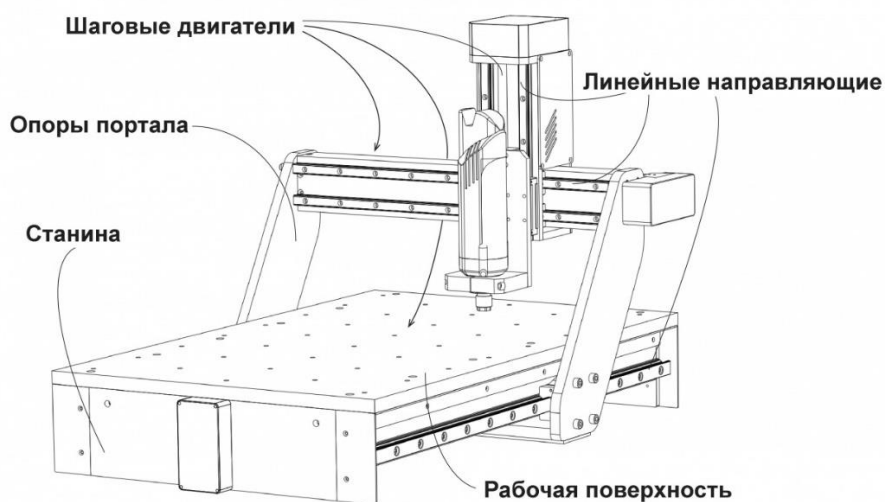


Рисунок 3 – Фрезерный станок

Благодаря фрезерному станку с чпу, оснащеному электронным оборудованием, выполняются самые различные виды работ по заготовке металлических, деревянных, стеклянных деталей для разного рода отраслей. Например, из отраслей следующего вида:

- промышленного производства;
- для судостроительного оборудования;
- авиационной промышленности;
- столярных и сталелитейных цехов.

В числовом управлении также предусмотрена функция смены режущего инструмента. Для этого в конструкцию добавляется механизм автоматической замены и набор инструментов, необходимых для работы. Двигаясь в трех основных направлениях (X, Y, Z) шпиндель способен перемещаться и по дополнительной траектории, согласно указанным координатам в шаблоне [5].

1.4 G-code – язык ЧПУ

Gcode представляет собой язык программирования, предназначенный для станков ЧПУ. Он же применяется и в 3Д принтерах. Этот код формирует специальная программа – слайсер. В нее загружается модель, задаются необходимые параметры, и в результате формируется джи код [10].

Как правило, каждая команда G-кода начинается с обозначения действия, сопровождаемого параметрами, которые уточняют это действие. Например, в команде "g00 x 80 y0" g00 указывает на перемещение на холостом ходу, а x и y обозначают движение головки машины вправо на 80 мм. G-коды также позволяют изменять температуру экструдера, управлять осями и выполнять другие операции.

Чтобы узнать коды для определенного оборудования, создатели станков прикладывают документацию с наименованием всех возможных кодов. Пример такой документации изображен на рисунке 4.

Г-код	Группа	Функция
G00	01	Быстрое позиционирование
G01	01	Линейная интерполяция
G01	01	Автоматическое скругление углов и снятие фасок
G02	01	Дуговая интерполяция по часовой стрелке (CW)
G03	01	Дуговая интерполяция против часовой стрелки (CCW)
G04	00	Пауза
G05	00	Изменение группы параметров
G09	00	Точный останов
G10	00	Включение режима ввода программируемых данных
G11	00	Выключение режима ввода программируемых данных
G15	16	Выключение режима полярных координат
G16	16	Включение режима полярных координат
G17	02	Выбор рабочей плоскости XY
G18	02	Выбор рабочей плоскости XZ
G19	02	Выбор рабочей плоскости YZ
G20	06	Выбор дюймовой системы измерения
G21	06	Выбор метрической системы измерения
G24	17	Включение зеркалирования
G25	17	Выключение зеркалирования
G28	00	Возврат через заданную точку
G29	00	Возврат к начальной точке
G30	00	Автовозврат через вторую, третью и четвертую заданные точки
G31	00	Функция пропуска

Рисунок 4 – Пример документации

Используя G-код, программист создает программы, которые затем загружаются на оборудование для выполнения различных производственных операций с металлом, деревом, пластиком и другими материалами. Однако написание таких программ требует учета множества параметров, что делает процесс сложным и неудобным. Поэтому разработчики создают специализированное программное обеспечение для упрощения этой задачи.

2 Анализ существующих программных решений для ЧПУ оборудования и проектирование программного обеспечения

2.1 Обзор основных программных продуктов на рынке

На рынке систем числового программного управления (ЧПУ) представлено множество программных продуктов, различающихся по функционалу и сложности. Эти программы служат для создания, редактирования и управления программами ЧПУ.

Одним из таких решений является Autodesk Fusion 360 (Рисунок 5). Это интегрированное ПО предлагает функции 3D-моделирования, симуляции и автоматизированного производства, а также программирования ЧПУ. В его арсенале – возможности для создания гибких траекторий движения инструмента и имитации процессов обработки.

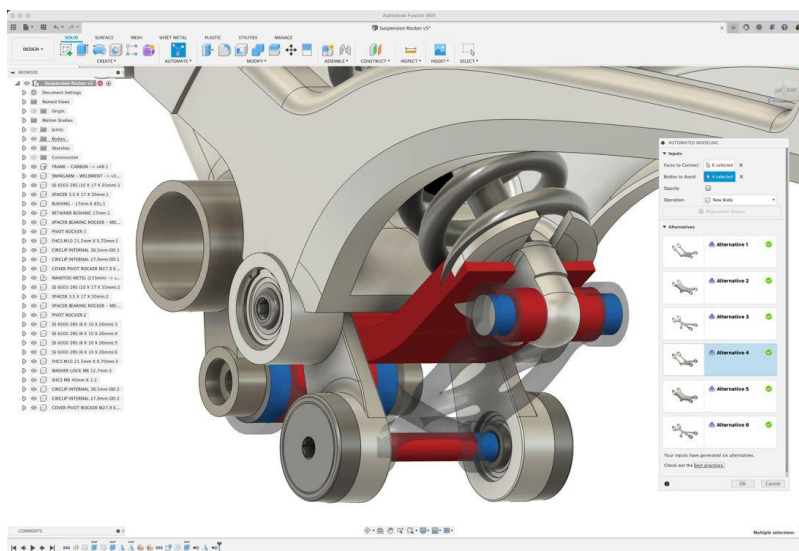


Рисунок 5 – Autodesk Fusion 360 интерфейс

Еще одним известным программным продуктом является SolidCAM, который работает в интеграции с SolidWorks. Эта система автоматизированного производства предоставляет мощные инструменты для создания и редактирования программ ЧПУ для различных станков.

Mastercam (Рисунок 6) – хорошо известное программное обеспечение для программирования ЧПУ, обладающее широким набором функций. Оно позволяет пользователям разрабатывать и редактировать траектории движения инструмента, генерировать код ЧПУ и моделировать операции обработки.

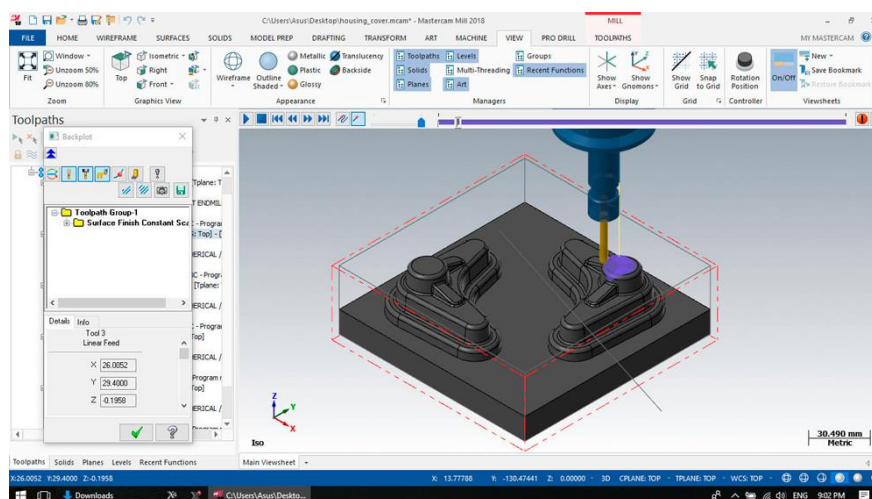


Рисунок 6 – Mastercam интерфейс

Siemens NX CAM (Рисунок 7) – комплексное решение, интегрирующееся с системой Siemens NX CAD/CAM. Оно предлагает множество инструментов для создания, редактирования и оптимизации программ ЧПУ, используемых в производственных процессах.

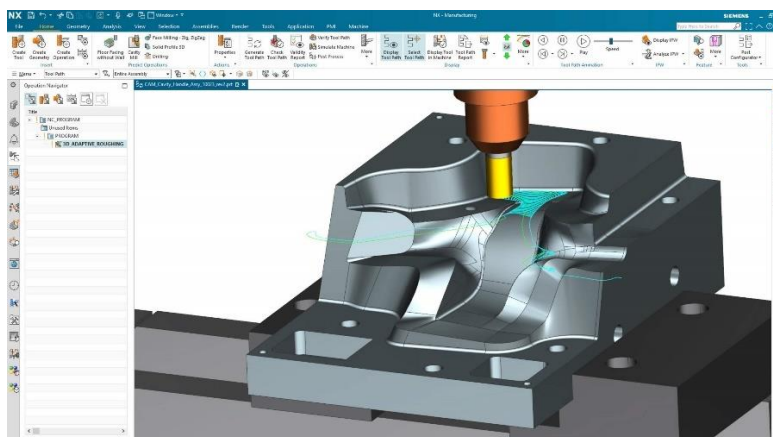


Рисунок 7 – Siemens NX CAM интерфейс

OpenCAM – бесплатное ПО с открытым исходным кодом, базирующееся на открытых стандартах и библиотеках. Оно предоставляет базовые возможности для создания и модификации программ ЧПУ и подходит для пользователей, ищущих экономичное решение.

BobCAD-CAM (Рисунок 8) – доступный и удобный инструмент для программирования ЧПУ, поддерживающий широкий спектр оборудования и предоставляющий инструменты для создания эффективных кодов ЧПУ для различных производственных задач.

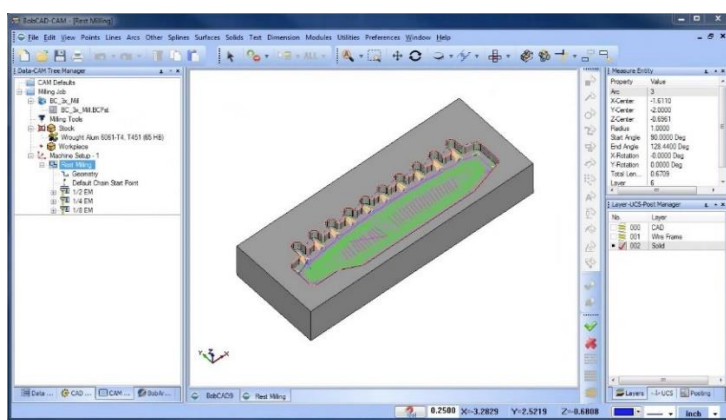


Рисунок 8 – BobCad CAM интерфейс

EdgeCAM (Рисунок 9) предлагает решения для 3D-моделирования, автоматизированного проектирования (CAD) и автоматизированного производства (CAM), обеспечивая точность и эффективность обработки деталей на станках с ЧПУ.

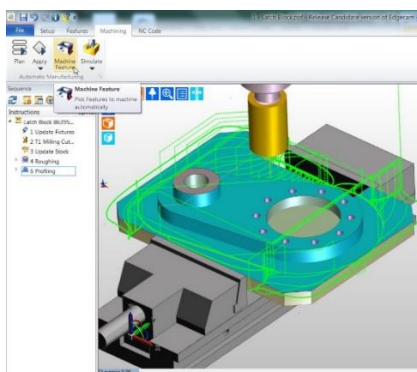


Рисунок 9 – EdgeCAM интерфейс

2.2 Сравнительный анализ функциональности, производительности, стоимости

Сравнение программного обеспечения для CNC может быть сложной задачей из-за разнообразия функций, целевых рынков и возможностей, предлагаемых различными продуктами. Рассмотрим основные факторы, которые учитываются при выборе такого ПО:

- 1) Функциональность:
 - возможности моделирования и создания геометрических узоров;
 - гибкость и эффективность в создании траекторий движения инструментов;
 - возможности симуляции технологических процессов;
 - интеграция с системами автоматизированного проектирования;
 - совместимость с различными типами оборудования;
 - удобство использования.
- 2) Эффективность:
 - скорость генерации ЧПУ-кода;
 - эффективность оптимизации траекторий;
 - возможности параллельной обработки и распределённых вычислений;
 - стабильность и надёжность работы программы.
- 3) Стоимость:
 - модели ценообразования и затраты на лицензирование;
 - расходы на обучение и поддержку;
 - потенциальные затраты на модернизацию.

Рассмотрим сравнение нескольких популярных программных решений:

Fusion 360: предлагает функции CAD и CAM, а также моделирование производственных операций. Имеет высокую производительность и гибкие тарифные планы на основе подписки, включая бесплатный доступ для

студентов и некоммерческих пользователей.

SolidCAM и SolidWorks: Взаимодополняющие инструменты, предоставляющие расширенные возможности программирования ЧПУ. Обычно требуют платного доступа, но студенты могут получить их бесплатно.

Mastercam: Известен своими многочисленными функциями для программирования ЧПУ. Доступен на основе подписки, включает поддержку и обновления.

Siemens NX CAM: Комплексное решение, интегрируемое с Siemens NX CAD/CAM. Предлагает разнообразные варианты лицензирования.

OpenCAM: Программное обеспечение с открытым исходным кодом, доступное всем. Может иметь ограничения по функциональности и поддержке.

BobCAD-CAM и EdgeCAM: Другие варианты программного обеспечения для ЧПУ, предоставляющие аналогичные функции. Их характеристики и цены могут варьироваться в зависимости от предпочтений и потребностей пользователя.

2.3 Оценка преимуществ и недостатков различных решений

Разберем плюсы и минусы каждой из перечисленных программ.

Облачные функции autodesk fusion 360 обеспечивают значительные преимущества при совместной работе с другими пользователями и проектами. Благодаря удобному и понятному интерфейсу программы для станков пишутся легко. Обширные возможности моделирования программного обеспечения позволяют пользователям проектировать сложные компоненты и тестировать их перед передачей на машину, но некоторым пользователям может быть сложно работать над более крупными проектами из-за недостаточной вычислительной мощности и места для хранения в облаке.

SolidCam легко интегрируется с SolidWorks, создавая единый интерфейс от проектирования к производству. Содержит разнообразные методы обработки, позволяющие пользователям оптимизировать производство различных материалов и деталей.

Недостатком является то, что SolidWorks не полностью с ним совместим, что делает его непригодным для тех, кто предпочитает другое программное обеспечение.

Mastercam, широко используемый в отрасли, и имеет большую базу пользователей, что упрощает доступ к ресурсам и поддержке. Программа универсальна по своей природе, с модулями, доступными для различных машин и методов обработки. Из-за неудобного нагруженного интерфейса работа с программой может быть очень трудоемкой.

Siemens NX CAM – программное обеспечение CAD/CAM, с помощью которого можно работать с другими продуктами Siemens, проектировать и оптимизировать сложные детали и процессы обработки на станках, но понадобится потратить не мало времени на изучение программы, так как она очень загружена.

OpenCam – единственное в списке программное обеспечение с открытым исходным кодом, что делает его очень гибким в настройке. Приложение можно подстроить под определенные задачи. Из-за того, что обеспечение бесплатное, это положительно сказывается на затраты производства, но официальная поддержка, как у более крупных компаний, может долго отсутствовать,

Bobcad-Cam – простое и дешевое программное обеспечение, которое выполняет все нужные функции для написания G-code. Удобный и понятный интерфейс делает его хорошим выбором для новичков, но его функциональность очень урезана, по сравнению с другими представителями рынка.

Edgcam – достаточно мощный и эффективный инструмент, имеет

огромный спектр предустановок для обработки различных материалов, при этом очень производительно и точно. К минусам можно отнести очень большую стоимость лицензии, вследствие чего придется прибегнуть к помощи более крупных компаний.

2.4 Определение требований к программному обеспечению

К технологическим приспособлениям для станков с ЧПУ предъявляется ряд специфических требований, обусловленных особенностью эксплуатации этих станков, несоблюдение которых значительно снижает эффективность их применения [3].

Точность. Приспособления должны иметь повышенную размерную точность. Погрешности базирования и закрепления, возникающие при установке заготовок в приспособлениях, должны быть сведены к минимуму.

Жесткость. Для возможности использования полной мощности станка на черновых операциях приспособления должны иметь повышенную жесткость. В то же время, конструкция приспособления должна обеспечить получение высокой точности на чистовых операциях.

Полное базирование заготовки и приспособления на станке. Относительное перемещение заготовки и инструмента на станках с ЧПУ осуществляется в системе заданных координат. Следовательно, для обеспечения автоматической ориентации опор относительно начала координат станка приспособления должны иметь полное базирование на станке, обеспечивающее строго определенное их положение относительно нулевой точки станка.

Обеспечение свободного доступа инструмента к заготовке. Станки с ЧПУ обеспечивают возможность обработки до 4–5 поверхностей с одной установки заготовки. Для этой цели приспособления должны обеспечивать возможность подхода инструмента ко всем обрабатываемым поверхностям.

Автоматизация операций закрепления, совмещение зажима –разжима заготовки с обработкой. Одним из путей для существенного сокращения времени простоев станков с ЧПУ является уменьшение времени зажима-разжима заготовок. Возможность обработки на станках с ЧПУ максимального числа поверхностей заготовки с одной ее установки резко увеличивает цикл обработки заготовки на одном станке, что обуславливает возможность смены заготовки вне рабочей зоны станка или вне станка во время обработки на станке другой заготовки [6].

Универсальность, переналаживаемость приспособлений. Станки с ЧПУ, в отличие от станков-автоматов, обладают высокой гибкостью, так как переналадка их может заключаться лишь в смене УП. Наибольшая часть подготовительно-заключительного времени затрачивается не на переналадку станка, а на смену или переналадку оснастки – приспособлений и инструмента. Поэтому, для сокращения простоя станков приспособления должны обеспечивать возможность их быстрой переналадки или смены. На станках с ЧПУ наиболее эффективно использовать системы переналаживаемых приспособлений, обеспечивающих возможность обработки широкой номенклатуры заготовок благодаря перекомпоновке, смене или регулированию установочных и зажимных элементов.

Многоместность. Приспособления, применяемые в серийном производстве при обработке малогабаритных деталей, должны быть многоместными, так как при этом возможна обработка поверхностей во всех заготовках последовательно одним и тем же инструментом. Производительность обработки увеличивается за счет сокращения времени, затрачиваемого на смену инструмента. Кроме того, многоместные приспособления обеспечивают возможность смены заготовок во время работы станка и многостаночное обслуживание [6].

Расширение технологических возможностей станка. Экономическая эффективность обработки деталей зависит от степени использования

технологической оснастки. Оснащение станков приспособлениями и оснасткой, расширяющими их технологические возможности, обеспечивает большой эффект в условиях индивидуального и серийного производств, компенсируя отсутствие тех или иных видов оборудования, устраняя дополнительную транспортировку деталей от станка к станку, ликвидируя недогрузку отдельных видов станков.

2.5 Разработка архитектуры системы

Архитектура системы ЧПУ является модульной и каждый модульный блок определяется задачами, которые в нем решаются (Рисунок 10).

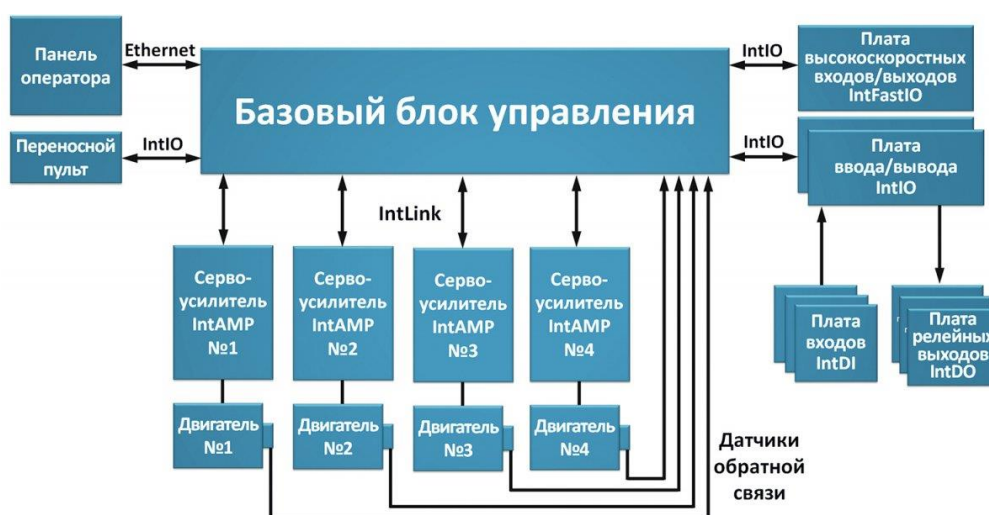


Рисунок 10 – Модульная архитектура ЧПУ

Задачи системы:

- геометрическая. Она состоит из нескольких модулей. Основные: интерпретатор управляющей программы, интерполятор. Интерпретатор передает информацию интерполятору из управляющей программы в особом виде. Особый вид передаваемой программы представлен именно так, как он был бы понятен интерполятору. То есть основная задача интерпретатора – «перевести» информацию и передать ее интерполятору. Также интерпретатор

выполняет некоторые расчеты, преобразовывает координатные системы и системы измерения. На этом этапе происходит и разделение всех видов задач. Задачи интерполятора – создание траектории движения инструмента на станке, с помощью которого выполняется обработка детали. Производится расчет скорости обработки, точности, с которой в конце изготовления должна быть выполнена деталь [9];

– логическая. Решение данной задачи может происходить по-разному. Первый вариант – программно в пределах самой системы ЧПУ. Второй – использование программируемого контроллера, который применяется чаще. При данном способе решения логической задачи необходим дополнительный персональный компьютер (ПК) и терминал. В таком случае мощность возрастает из-за того, что ПК объединяет терминал и сам программируемый контроллер. Получается целостная система, у которой присутствует еще один модуль как ввода, так и вывода информации. Это необходимо для реализации циклов электроавтоматизации станка. То есть это система, работающая на принципе автоматического управления механизмами, содержащимися в станке, и группами этих механизмов. Операции, которые должны быть выполнены этими механизмами, инициируются электрическими сигналами – унитарным кодом, исходящий от объекта управления [8].

2.6 Выбор технологий и инструментов разработки

Выбор подходящих технологий и инструментов для разработки системы управления станком с ЧПУ является сложной задачей, требующей учета множества факторов, таких как требования проекта, доступные ресурсы, опыт команды разработчиков и совместимость с существующими системами.

В данном проекте был выбран язык программирования C# благодаря его простоте, гибкости и обширной библиотечной экосистеме. Это решение обеспечивает создание высокопроизводительного и простого в обслуживании

программного обеспечения для ЧПУ. Рассмотрим подробнее преимущества этого выбора:

- поддерживает проверку типов, объектно-ориентированное программирование, обработку исключений и асинхронное программирование;
- обладает множеством библиотек и инструментов для создания приложений, включая управление файлами, сетями и базами данных, что особенно важно для разработки программного обеспечения ЧПУ;
- обеспечивает надежные инструменты для разработки, тестирования и отладки приложений. Интегрируется с инструментами разработки программного обеспечения для ЧПУ, включая симуляторы траектории движения инструмента и инструменты визуализации;
- C# имеет большое сообщество разработчиков, множество сторонних ресурсов, библиотек и инструментов, что помогает оптимизировать процесс разработки и предоставляет поддержку на всех этапах;

Для разработки пользовательского интерфейса была выбрана среда Windows Forms (Рисунок 11), поскольку она проста в использовании, функциональна и легко реализуема.

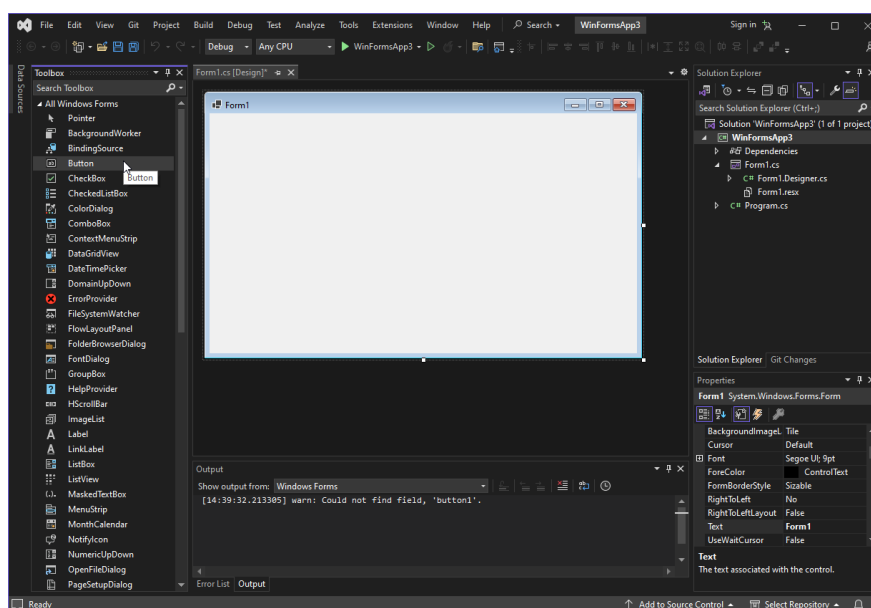


Рисунок 11 – Windows Forms

Перейдем к преимуществам:

- Windows Forms предлагает богатый выбор готовых элементов управления, таких как кнопки, текстовые поля, списки и таблицы, что упрощает создание интерфейсов;
- простота модели программирования, основанной на событиях, делает её идеальной для быстрой разработки графических пользовательских интерфейсов;
- обеспечивает удобный интерфейс, высокую производительность и простоту обработки данных;
- комплексная документация и активное сообщество разработчиков помогают в проектировании графических интерфейсов.

Visual Studio (Рисунок 12), как среда разработки, выбрана благодаря мощным функциям и набору инструментов, упрощающих разработку на C#:

- включает интеллектуальное завершение кода, возможности отладки;
- интеграцию с системами контроля версий, такими как Git, SVN и Mercurial;
- функции анализа кода, рефакторинга и автоматического форматирования повышают качество кода и производительность разработчиков.

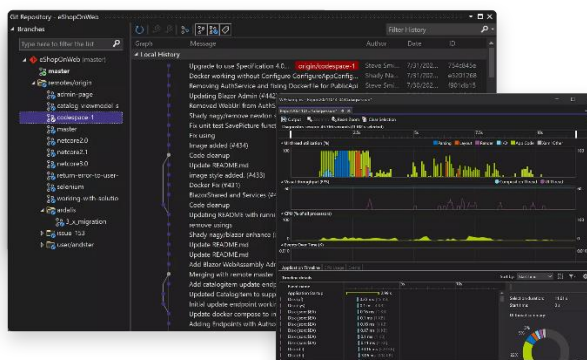


Рисунок 12 – Visual Studio

3 Практическая реализация программного обеспечения

3.1 Реализация пользовательского интерфейса

Не мало важная часть проекта заключается в создании удобной рабочей среды для программного обеспечения. Интерфейс – это способ сделать систему проще в использовании и сделать ее более эффективной, он должен быть удобным, простым в использовании и функциональным для достижения этих целей.

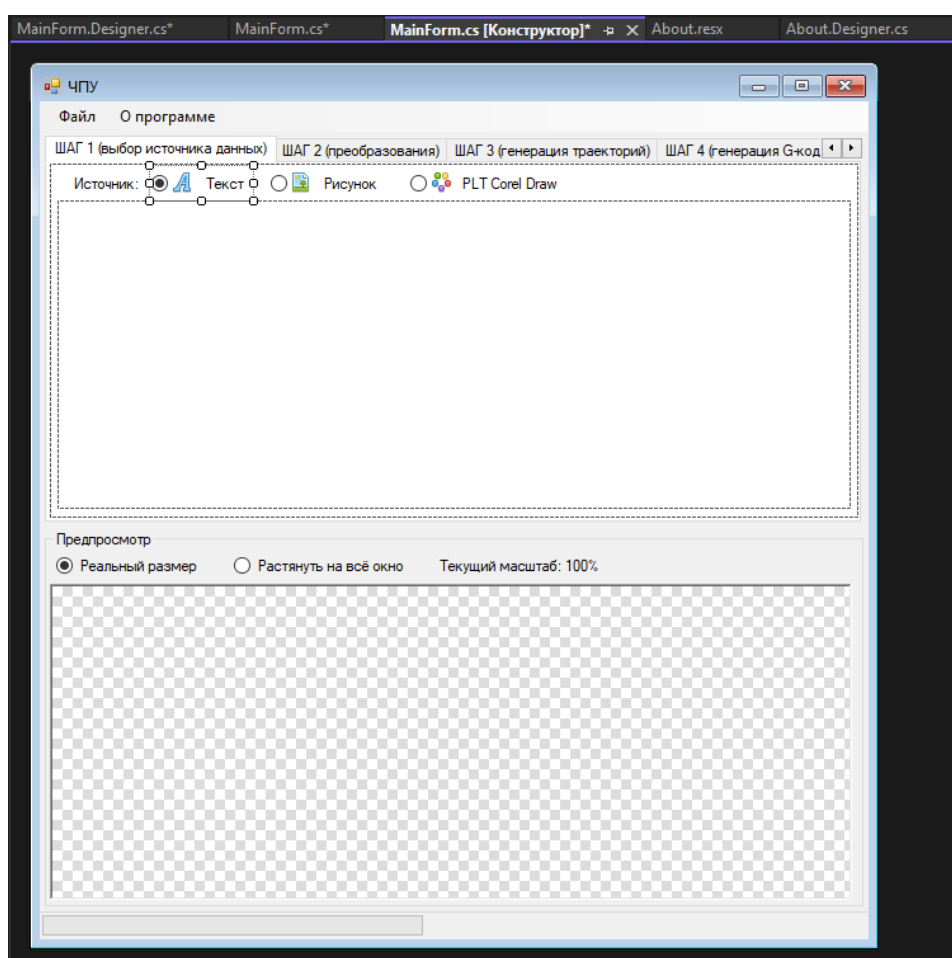


Рисунок 13 – Выбор источника данных

На рисунке 13 реализован интерфейс загрузки данных. Пользователь может выбрать источник. Также, здесь есть окно предпросмотра, которое будет видно независимо от того, отображается вкладка приложения или нет.

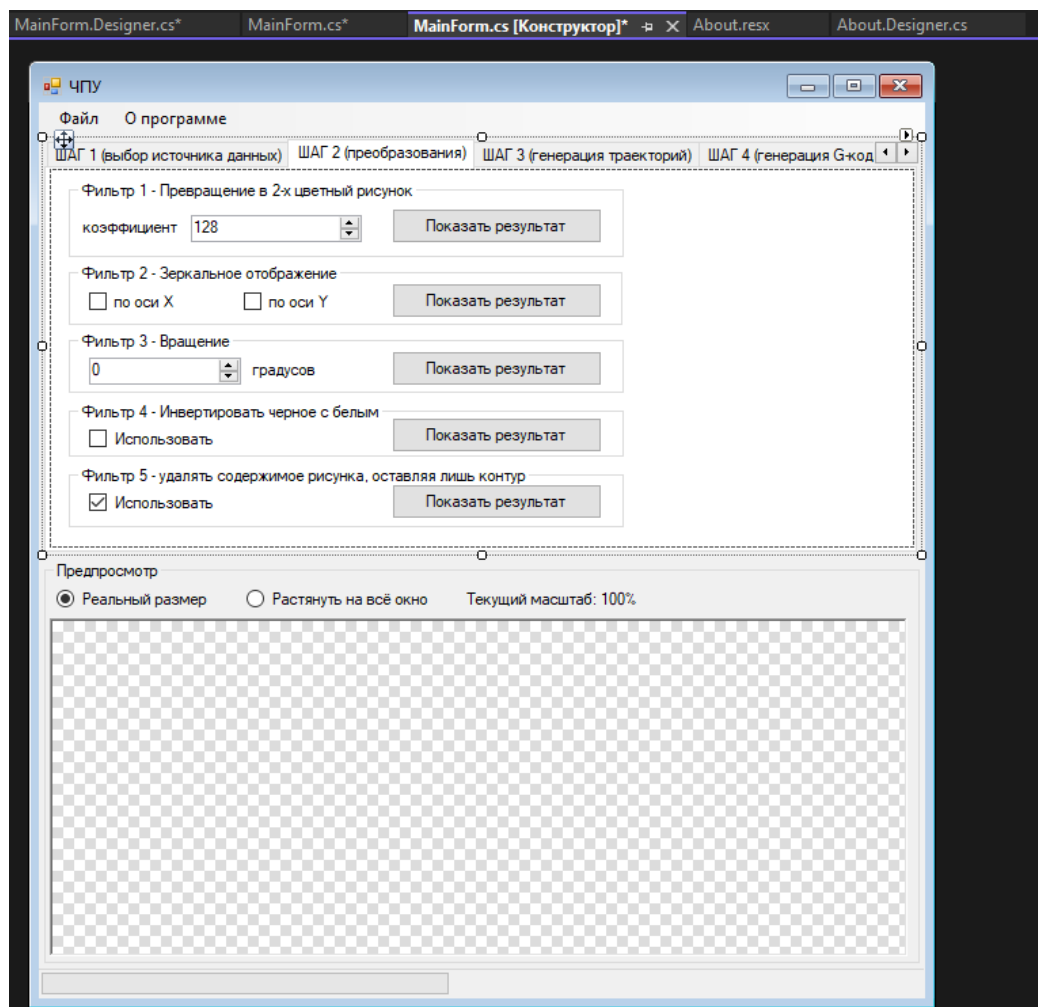


Рисунок 14 – Окно преобразований

На рисунке 14 изображении показано окно преобразования изображения или текста из исходного состояния в двухцветный рисунок, который программа может использовать для инициализации и реализации разработанных алгоритмов. Параметры, которые можно изменить или переключить, имеют решающее значение для отображения изменяемого объекта:

- зеркальное отображение;
- вращение изображений;
- инверсия черного и белого.

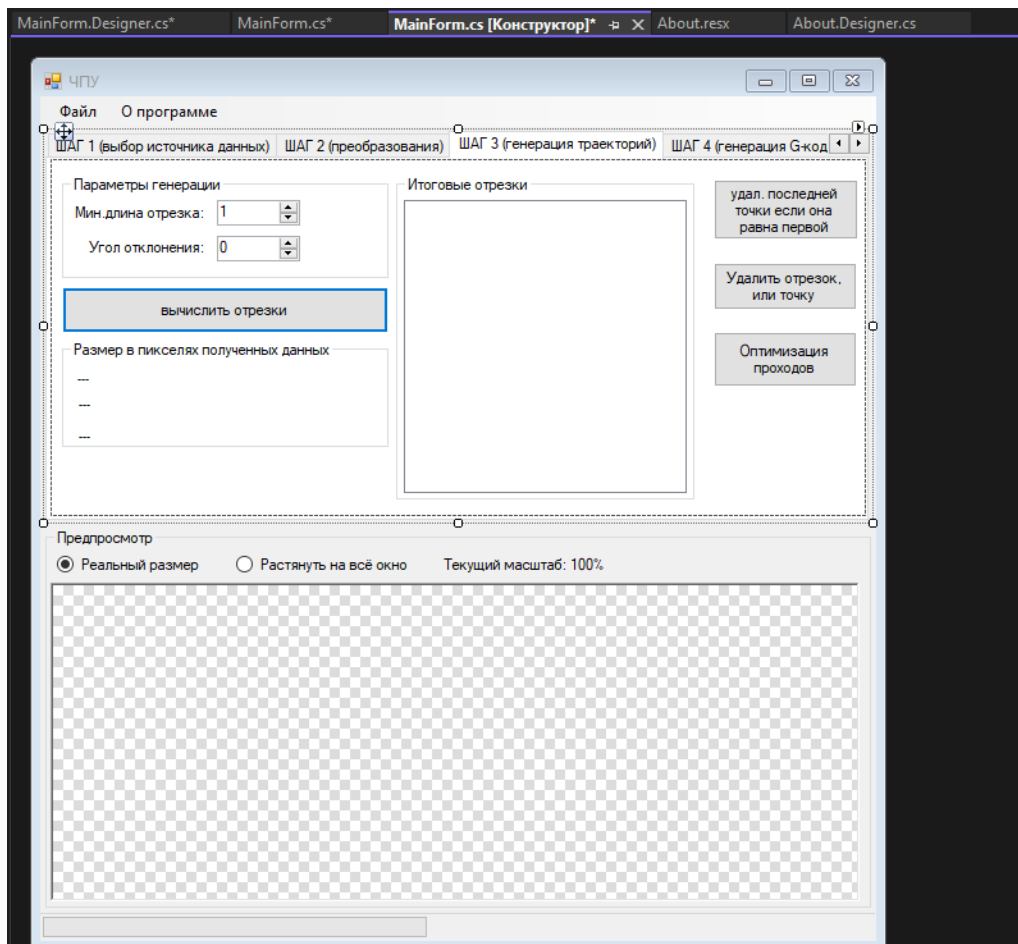


Рисунок 15 – Окно генерации траекторий

На рисунке 15 показано окно генерации траектории. Изображение или текст будут разделены на разделы и сгенерированы с путями в форме точек x и y вместе с их положениями.

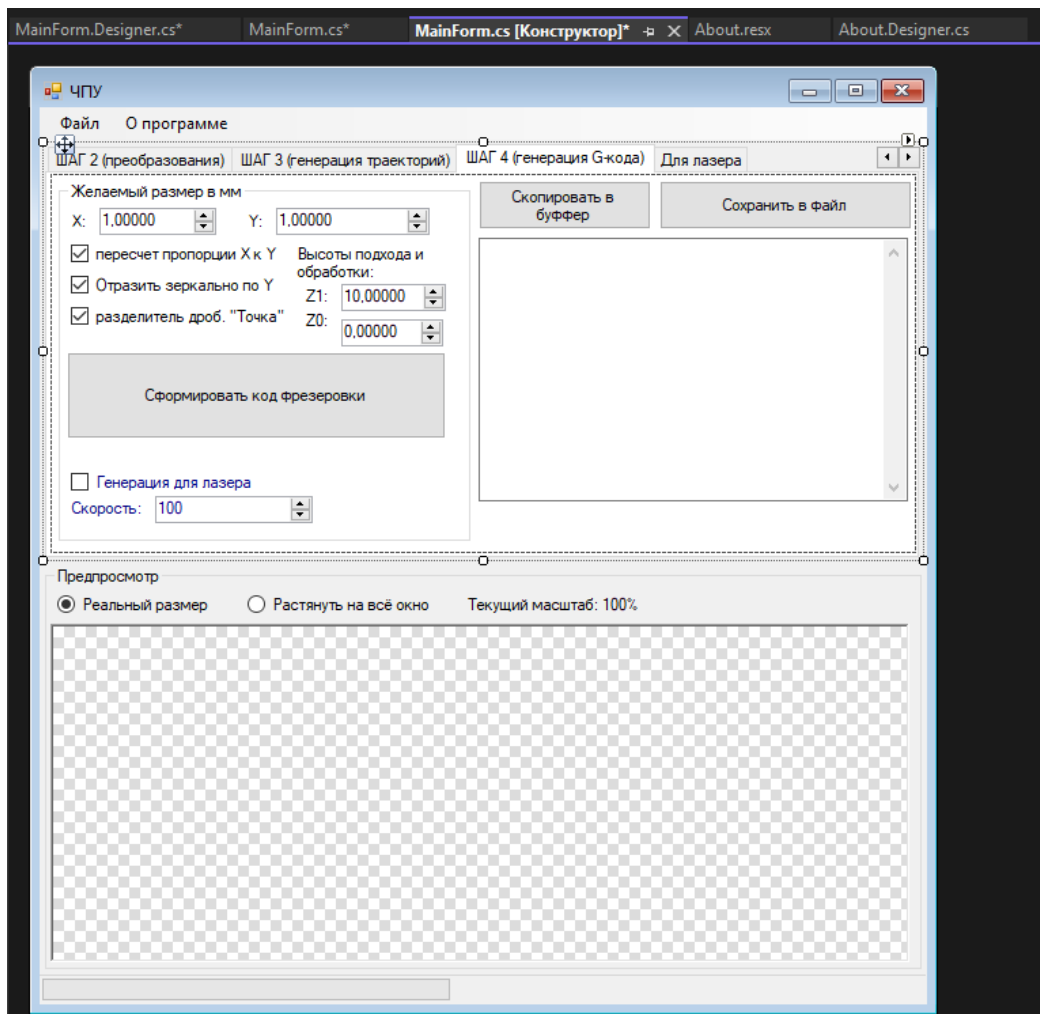


Рисунок 16 – Окно генерации G-кода

На рисунке 16 реализовано окно, в котором дальнейшем мы и получим итоговый G-код для станка с ЧПУ. На этом шаге реализованы дополнительные функции, которые нужны для конечного кода:

- размер изделия в миллиметрах;
- зеркальное отражение изделия по оси Y;
- задаваемые значения высоты подхода и обработки.

Эти параметры помогут получить именно тот результат, который нужен на производстве.

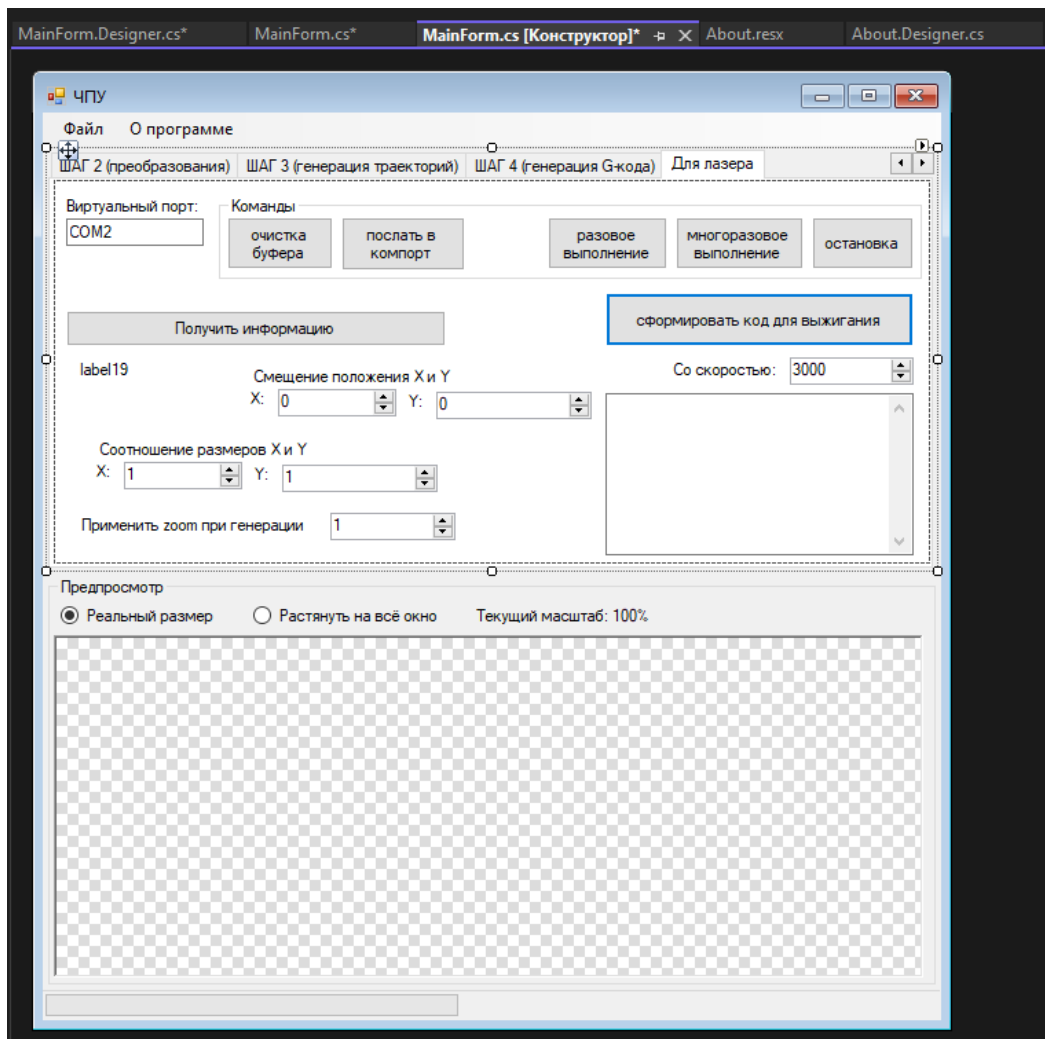


Рисунок 17 – Генерация кода для лазерного станка

Также была реализована возможность генерации кода для лазерного станка. Интерфейс для этого представлен на рисунке 17.

3.2 Реализация основных функциональных блоков

В этом разделе рассмотрена реализация основных блоков программного обеспечения для системы ЧПУ (числового программного управления) оборудования. ЧПУ обеспечивает управление процессами обработки материалов на станках с ЧПУ, позволяя автоматизировать и оптимизировать производственные процессы. Разберем главные функции разработанного программного обеспечения.

Одной из самых главных функций является функция вычисления отрезков. Она отвечает за разбиение исходных данных на отрезки. Реализация функции вычисления отрезков представлена на рисунке 18.

Данный фрагмент кода представляет собой обработчик событий, который проверяет выбор радиокнопки и вызывает один из трех методов: `GetVectorFromText`, `GetVectorFromImage` и `GetVectorFromPLT`.

```
//вычисление отрезков
Ссылка 1
private void buttonCalculateVectors_Click(object sender, EventArgs e)
{
    if (radioButtonTypeSourceText.Checked)
    {
        if (_selFont.UseVectorFont)
        {
            if (_selFont.UseSystemFont)
            {
                //используем системный шрифт
                _lines = GetVectorFromText(_selFont.textString.Text, _selFont.comboBoxFont.Text, (float)_selFont.textSize.Value);
            }
            else
            {
                //используем внешний файл шрифта
                _lines = GetVectorFromText(_selFont.textString.Text, _selFont.comboBoxFont.Text, (float)_selFont.textSize.Value, _selFont.nameFontFile.Text);
            }
        }
        else
        {
            Bitmap tmp = GetImageFromStep(6);
            _lines = GetVectorFromImage(tmp);
        }
    }
    else if (radioButtonTypeSourcePLT.Checked)
    {
        _lines = GetVectorFromPLT(_selPLT.textBoxFileName.Text);
    }
    else
    {
        Bitmap tmp = GetImageFromStep(6);
        _lines = GetVectorFromImage(tmp);
    }

    RefreshTree();
    //pictureBoxPreview.Image = GetTrajectory();
}
```

Рисунок 18 – Функция вычисления отрезков

На рисунках 19 и 20 описаны функции `GetVectorFromText`, которая отвечает за расчет самих точек, в случае если исходные данные для генерации представлены в виде текста.

```

private List<List<PointF>> GetVectorFromText(string text, string fontName, float fontSize, string extFileFont = "")
{
    PointF[] pts = null; //список точек
    byte[] ptsType = null; //информация о начале/окончании отрезка

    using (GraphicsPath path = new GraphicsPath())
    {
        if (extFileFont != "")
        {
            PrivateFontCollection customFontFromFile;
            customFontFromFile = new PrivateFontCollection();
            customFontFromFile.AddFontFile(extFileFont);
            if (customFontFromFile.Families.Length > 0)
            {
                Font font = new Font(customFontFromFile.Families[0], (int)fontSize);
                path.AddString(text, font.FontFamily, (int)FontStyle.Regular, fontSize, new PointF(0f, 0f), StringFormat.GenericDefault);
            }
            else
            {
                MessageBox.Show(@"Ошибка загрузки шрифта из файла!!!");
            }
        }
        else
        {
            path.AddString(text, new FontFamily(fontName), (int)FontStyle.Regular, fontSize, new PointF(0f, 0f), StringFormat.GenericDefault);
        }

        path.Flatten();

        if (path.PointCount == 0) //нет отрезков
        {
            return new List<List<PointF>>();
        }

        pts = path.PathPoints;
        ptsType = path.PathTypes;
    }
}

```

Рисунок 19 – Функция GetVectorFromText

```

//для сбора информации о точках у отрезка
List<PointF> ListPoints = new List<PointF>();
// для сбора информации об отрезках
List<List<PointF>> Lines = new List<List<PointF>>();

int index = 0;
foreach (PointF value in pts)
{
    byte ptypePoint = ptsType[index];

    // это первая точка
    if (ptypePoint == 0)
    {
        ListPoints.Add(value);
    }

    //а это продолжение
    if (ptypePoint == 1)
    {
        ListPoints.Add(value);
    }

    //окончание
    if (ptypePoint == 129)
    {
        ListPoints.Add(value);
        ListPoints.Add(ListPoints[0]); //иногда не нужна
        Lines.Add(ListPoints);
        ListPoints = new List<PointF>();
    }

    if (ptypePoint == 161)
    {
        ListPoints.Add(value);
        //ListPoints.Add(ListPoints[0]);
        Lines.Add(ListPoints);
        ListPoints = new List<PointF>();
    }

    index++;
}

return Lines;
}

```

Рисунок 20 – Функция GetVectorFromText

На рисунке 21 представлена функция, которая проверяет наличие черных точек вокруг той, в которой находится программа.

```
//Возвращает истину если есть черные точки по соседству
Ссылка 1
private bool CheckBlackPoint(ref Bitmap tmp,int pX,int pY)
{
    bool Fined = false;

    Color compareColor = Color.Black;

    Color Color1 = tmp.GetPixel(pX+1, pY);
    Color Color2 = tmp.GetPixel(pX+1, pY+1);
    Color Color3 = tmp.GetPixel(pX, pY+1);
    Color Color4 = tmp.GetPixel(pX-1, pY+1);
    Color Color5 = tmp.GetPixel(pX-1, pY);
    Color Color6 = tmp.GetPixel(pX-1, pY-1);
    Color Color7 = tmp.GetPixel(pX, pY-1);
    Color Color8 = tmp.GetPixel(pX+1, pY-1);

    Fined = (Color1.ToArgb() == compareColor.ToArgb() || Color2.ToArgb() == compareColor.ToArgb() || Color3.ToArgb() == compareColor.ToArgb() || Color4.ToArgb() == compareColor.ToArgb() || Color5.ToArgb() == compareColor.ToArgb() || Color6.ToArgb() == compareColor.ToArgb() || Color7.ToArgb() == compareColor.ToArgb() || Color8.ToArgb() == compareColor.ToArgb());

    return Fined;
}
```

Рисунок 21 – Функция ChekBlackPoint

Не менее важной функцией является RotatePic. Благодаря ей изображение можно будет поворачивать как удобно пользователю (Рисунки 22, 23).

```
private Bitmap RotatePic(Bitmap bmpBU, float w, bool keepWholeImg)
{
    //keepWholeImg - если истина, то изображение ужимается, при повороте, что-бы вписаться в рамки изображения
    Bitmap bmp = null;
    Graphics g = null;
    try
    {
        //Modus
        if (!keepWholeImg)
        {
            bmp = new Bitmap(bmpBU.Width, bmpBU.Height, PixelFormat.Format24bppRgb);

            g = Graphics.FromImage(bmp);
            float hw = bmp.Width / 2f;
            float hh = bmp.Height / 2f;
            g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
            g.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;

            //translate center
            g.TranslateTransform(hw, hh);
            //rotate
            g.RotateTransform(w);
            //re-translate
            g.TranslateTransform(-hw, -hh);
            g.DrawImage(bmpBU, 0, 0);
            g.Dispose();
        }
        else
        {
            //get the new size and create the blank bitmap
            float rad = (float)(w / 180.0 * Math.PI);
            double fw = Math.Abs(Math.Cos(rad) * bmpBU.Width) + Math.Abs(Math.Sin(rad) * bmpBU.Height);
            double fh = Math.Abs(Math.Sin(rad) * bmpBU.Width) + Math.Abs(Math.Cos(rad) * bmpBU.Height);

            bmp = new Bitmap((int)Math.Ceiling(fw), (int)Math.Ceiling(fh));

            g = Graphics.FromImage(bmp);
        }
    }
}
```

Рисунок 22 – Функция RotatePic

```

g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
g.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;

float hw = bmp.Width / 2f;
float hh = bmp.Height / 2f;

System.Drawing.Drawing2D.Matrix m = g.Transform;

//here we do not need to translate, we rotate at the specified point
m.RotateAt(w, new PointF((float)(bmp.Width / 2), (float)(bmp.Height / 2)), System.Drawing.Drawing2D.MatrixOrder.Append);

g.Transform = m;

//draw the rotated image
g.DrawImage(bmpBU, new PointF((float)((bmp.Width - bmpBU.Width) / 2), (float)((bmp.Height - bmpBU.Height) / 2)));
g.Dispose();
}
}
catch
{
if ((bmp != null))
{
bmp.Dispose();
bmp = null;
}

if ((g != null))
{
g.Dispose();
}

// MessageBox.Show("Fehler.");
}

return bmp;
}

```

Рисунок 23 – Функция RotatePic

Для получения картинки из текста нужно высчитать высоту и ширину текста, преобразовать объект в картинку уже с нужными параметрами высоты и ширины и цветом фона. Функция, которая прodelывает все эти действия представлена на рисунке 24.

```

// Создаем объект Graphics для вычисления высоты и ширины текста.
Graphics graphics = Graphics.FromImage(bitmap);

// Определение размеров изображения.
width = (int)graphics.MeasureString(text, font).Width;
height = (int)graphics.MeasureString(text, font).Height;

// Пересоздаем объект Bitmap с откорректированными размерами под текст и шрифт.
bitmap = new Bitmap(bitmap, new Size(width, height));

// Пересоздаем объект Graphics
graphics = Graphics.FromImage(bitmap);

// Задаем цвет фона.
graphics.Clear(Color.White);
// Задаем параметры анти-алиасинга
graphics.SmoothingMode = SmoothingMode.None;
graphics.TextRenderingHint = TextRenderingHint.SingleBitPerPixel;
// Пишем (рисуем) текст
graphics.DrawString(text, font, new SolidBrush(Color.Black), 0, 0);

graphics.Flush();

return (bitmap);
}

```

Рисунок 24 – Функция преобразования картинки из текста

На рисунках 25, 26, 27, 28 описана функции `GetVectorFromImage`, которая отвечает за расчет самих точек, в случае если исходные данные для генерации представлены в виде изображения.

```
private List<List<PointF>> GetVectorFromImage(Bitmap image)
{
    // для сбора информации о точках у отрезка
    List<PointF> ListPoints = new List<PointF>();
    // для сбора информации об отрезках
    List<List<PointF>> Lines = new List<List<PointF>>();

    Bitmap tmp = GetImageFromStep(0);

    for (int pY = 1; pY < tmp.Height - 1; pY++)
    {
        for (int pX = 1; pX < tmp.Width - 1; pX++)
        {
            Color compareColor = tmp.GetPixel(pX, pY);

            int tmpPalitra = ((int)compareColor.R + (int)compareColor.G + (int)compareColor.B) / 3;

            if (tmpPalitra != 0) continue; // очень светлая точка, такое нас не интересует

            //начинаем обход

            //временные координаты, для обхода
            int tx = pX;
            int ty = pY;

            bool bypass = true; //для определения того что занимаемся обходом по точкам
            int direction = 1; //начальное направление вправо, по часовой стрелке.

            bool firstPoint = true;
            float firstX = 0;
            float firstY = 0;

            //начинаем обход траектории
            while (bypass)
            {
                //1) добавляем текущую точку
                // points.Add(new mPoint(tx, ty));
                ListPoints.Add(new PointF(tx, ty));

                if (firstPoint)
                {
                    firstPoint = false;
                    firstX = tx;
                    firstY = ty;
                    //и стираем текущую черную точку
                    tmp.SetPixel(tx, ty, Color.FromArgb(255, 0, 255, 255));
                }
                else
                {
                    //и стираем текущую черную точку
                    tmp.SetPixel(tx, ty, Color.FromArgb(255, 255, 255, 255));
                }
            }
        }
    }
}
```

Рисунок 25 – Функция `GetVectorFromImage`


```

bool tst = CheckBlackPoint(ref tmp, tx, ty);
if (tst == false)
{
    bypass = false;
    continue;
}

//2) определяем направление с которого просканируем окружающую остановку
direction += 5;

//типа "переполнение"
if (direction > 8) direction -= 8;

//3) по очереди по всем направлениям проверим наличие черной точки
int test = 7;

while (test > 0)
{
    int oX = tx;
    int oY = ty;

    switch (direction)
    {
        case 1:
            oX++;
            break;
        case 2:
            oX++; oY++;
            break;
        case 3:
            oY++;
            break;
        case 4:
            oX--; oY++;
            break;
        case 5:
            oX--;
            break;
        case 6:
            oX--; oY--;
            break;
        case 7:
            oY--;
            break;
        case 8:
            oX++; oY--;
            break;
        default:
            break;
    }
}

```

Рисунок 26 – Функция GetVectorFromImage

```

//если цвет в нужном направлении черный хорошо, иначе продолжим дальше
Color testColor = tmp.GetPixel(oX, oY);
int testPalitra = ((int)testColor.R + (int)testColor.G + (int)testColor.B) / 3;

if (testPalitra == 0)
{
    //наша точка
    test = 0;
    tx = oX;
    ty = oY;

    continue;
}

if (direction < 8) direction++;
else direction = 1;

test--;
} //while (test>0)

} //while (bypass)

// В связи с тем что мы закрашиваем пройденные точки, то необходимо проверить, последняя точка замыкается с начальной, или нет
PointF pp = ListPoints[ListPoints.Count - 1];
float distance = (firstX - pp.X) + (firstY - pp.Y);
if (distance < 2 && distance > -2) ListPoints.Add(new PointF(firstX,firstY));

List<PointF> TMPpoints = new List<PointF>();

////добавим начальную точку в любом случае
TMPpoints.Add(ListPoints[0]);

```

Рисунок 27 – Функция GetVectorFromImage

```

for (int i = 1; i < ListPoints.Count - 1; i++)
{
    float angl = GetAngleFrom3Points(ListPoints[i - 1], ListPoints[i], ListPoints[i + 1]);
    float deltaAngle = (float)numericUpDownAngle.Value;

    bool bAddPoint = true;

    // проверка угла
    if ((angl >= (180 - deltaAngle) && angl <= (180 + deltaAngle))) bAddPoint = false;

    // проверка длины
    //int lenghAB = 1;
    if (numericUpDownMunimumLenght.Value > 1 && TMPpoints.Count > 0)
    {
        float xa = (TMPpoints[TMPpoints.Count - 1].X - ListPoints[i].X);
        float ya = (TMPpoints[TMPpoints.Count - 1].Y - ListPoints[i].Y);
        double dlenght = Math.Sqrt(xa * xa + ya * ya);

        if (dlenght < (double)numericUpDownMunimumLenght.Value) bAddPoint = false;
    }

    if (bAddPoint) TMPpoints.Add(ListPoints[i]);
}

////и конечную точку в любом случае
TMPpoints.Add(ListPoints[ListPoints.Count - 1]);

bool needAddOtrezok = true;

if (TMPpoints.Count == 2)
{
    if (TMPpoints[0].X == TMPpoints[1].X &&
        TMPpoints[0].Y == TMPpoints[1].Y)
    {
        needAddOtrezok = false;
    }
}

//обход по контуру завершен
//treeView1.Nodes.Add(tn);
if (needAddOtrezok) Lines.Add(TMPpoints);

////очистим
ListPoints = new List<PointF>();
}
}

return Lines;
}

```

Рисунок 28 – Функция GetVectorFromImage

Далее рассмотрим функцию `GenerateGkode_Click`, которая на основе полученных отрезков сформирует готовую программу для фрезерного станка с ЧПУ. Она представлена на рисунках 29, 30, 31.

```
private void GenerateGkode_Click(object sender, EventArgs e)
{
    // применим дельты, для смещения в начало координат
    float deltaX = _maxX - (_maxX - _minX);
    float deltaY = _maxY - (_maxY - _minY);

    float koefX = (_maxX - _minX) / (float)numericUpDownCalcX.Value;
    float koefY = (_maxY - _minY) / (float)numericUpDownCalcY.Value;

    decimal dZ0 = numericUpDownZ0.Value;
    decimal dZ1 = numericUpDownZ1.Value;

    if (checkBoxForLaser.Checked)
    {
        dZ0 = 0;
        dZ1 = 0;
    }

    string sZ0 = dZ0.ToString("#0.###");
    string sZ1 = dZ1.ToString("#0.###");

    string Gkode = "" + Environment.NewLine;

    if (checkBoxForLaser.Checked)
    {
        Gkode += "M5" + Environment.NewLine;

        sZ0 = (0).ToString("#0.###");
        sZ1 = (0).ToString("#0.###");

        Gkode += "G0 F" + numericUpDownSpeedLaser.Value.ToString("00000") + Environment.NewLine;
        Gkode += "G1 F" + numericUpDownSpeedLaser.Value.ToString("00000") + Environment.NewLine;
    }
    else
    {
        Gkode += "M3" + Environment.NewLine;
    }

    Gkode += "g0 x0 y0 z" + sZ1 + Environment.NewLine;
}
```

Рисунок 29 – Функция `GenerateGkode_Click`

```

foreach (List<PointF> line in _lines)
{
    double x = (line[0].X - deltaX);
    double y = (line[0].Y - deltaY);

    if (checkBoxMirrorY.Checked)
    {
        y = -y + (_maxY - _minY);
    }

    x = x / koefX;
    y = y / koefY;

    if (!checkBoxForLaser.Checked)
    {
        //опускание с высоты подхода до высоты фрезеровки
        Gkode += "g0 x" + x.ToString("#0.###") + " y" + y.ToString("#0.###") + " z" + sZ1 + Environment.NewLine;
    }

    bool firstPoint = true;

    foreach (PointF point in line)
    {
        x = (point.X - deltaX);
        y = (point.Y - deltaY);

        if (checkBoxMirrorY.Checked)
        {
            y = -y + (_maxY - _minY);
        }

        x = x / koefX;
        y = y / koefY;

        Gkode += "g1 x" + x.ToString("#0.###") + " y" + y.ToString("#0.###") + " z" + sZ0 + Environment.NewLine;

        if (firstPoint)
        {
            //перед началом линии включим лазер
            if (checkBoxForLaser.Checked)
            {
                Gkode += "M3" + Environment.NewLine;
            }
            firstPoint = false;
        }
    }
}

```

Рисунок 30 – Функция GenerateGkode_Click

```

//после завершения линии выключим лазер
if (checkBoxForLaser.Checked)
{
    Gkode += "M5" + Environment.NewLine;
}

x = (line[line.Count - 1].X - deltaX);
y = (line[line.Count - 1].Y - deltaY);

if (checkBoxMirrorY.Checked)
{
    y = -y + (_maxY - _minY);
}

x = x / koefX;
y = y / koefY;

if (!checkBoxForLaser.Checked)
{
    // поднятие на безопасную высоту
    Gkode += "g0 x" + x.ToString("#0.###") + " y" + y.ToString("#0.###") + " z" + sZ1 + Environment.NewLine;
}

if (checkBoxUsePoint.Checked)
{
    Gkode = Gkode.Replace(", ", ".");
}

Gkode += "M5" + Environment.NewLine;

textBoxGkod.Text = Gkode;
}

```

Рисунок 31 – Функция GenerateGkode_Click

Также опишем функцию getLaserCode для получения кода для лазерного станка с ЧПУ (Рисунок 32).

```
Ссылка 2
private string getLaserCode()
{
    // применим дельты, для смещения в начало координат
    //int deltaX = (int)((float)numericUpDownLazerX.Value / _maxX);
    //int deltaY = (int)((float)numericUpDownLazerY.Value / _maxY);

    double koeffX = (_maxX - _minX) / (double)numericUpDownCalcX.Value;
    double koeffY = (_maxY - _minY) / (double)numericUpDownCalcY.Value;

    //string sZ0 = numericUpDownZ0.Value.ToString("#0.###");
    //string sZ1 = numericUpDownZ1.Value.ToString("#0.###");

    string gkode = "";

    foreach (List<PointF> line in _lines)
    {
        int X1 = (int)line[0].X * (int)numericUpDownzoom.Value;
        int Y1 = (int)line[0].Y * (int)numericUpDownzoom.Value;

        X1 = (int)numericUpDownLazerX.Value * X1;
        Y1 = (int)numericUpDownLazerY.Value * Y1;

        X1 += (int)numericUpDownMoveLazerX.Value;
        Y1 += (int)numericUpDownMoveLazerY.Value;

        //gkode += "M5 " + "X" + X1.ToString("00000") + " Y" + Y1.ToString("00000") + " F" + numericUpDownLaserSpeed.Value.ToString("00000") + "\r\n";
        gkode += "M5 " + "X" + X1.ToString("00000") + " Y" + Y1.ToString("00000") + " F00010" + "\r\n";

        foreach (PointF point in line)
        {
            //int X2 = deltaX * pnt.X;
            //int Y2 = deltaY * pnt.Y;

            int X2 = (int)point.X * (int)numericUpDownzoom.Value;
            int Y2 = (int)point.Y * (int)numericUpDownzoom.Value;

            X2 += (int)numericUpDownMoveLazerX.Value;
            Y2 += (int)numericUpDownMoveLazerY.Value;

            gkode += "M3 " + "X" + X2.ToString("00000") + " Y" + Y2.ToString("00000") + " F" + numericUpDownLaserSpeed.Value.ToString("00000") + "\r\n";
        }
    }
    gkode += "M5 " + "\r\n";

    return gkode;
}
```

Рисунок 32 – Функция getLaserCode

3.3 Отладка и тестирование программы на примере текстовых данных

Отладка и тестирование программного обеспечения для ЧПУ оборудования являются критическими этапами разработки. Эти процессы не только помогают обнаружить и исправить ошибки, но и гарантируют, что программа работает стабильно и эффективно.

Проведем полный цикл от ввода данных до получения готового G-кода. Первым шагом выбираем источник входных данных, в нашем случае это

текст. После этого выбираем шрифт и размер. В окне предпросмотра появляется введенный текст и, если нас всё устраивает переходим ко второму шагу. Все этапы первого шага представлены на рисунке 33.

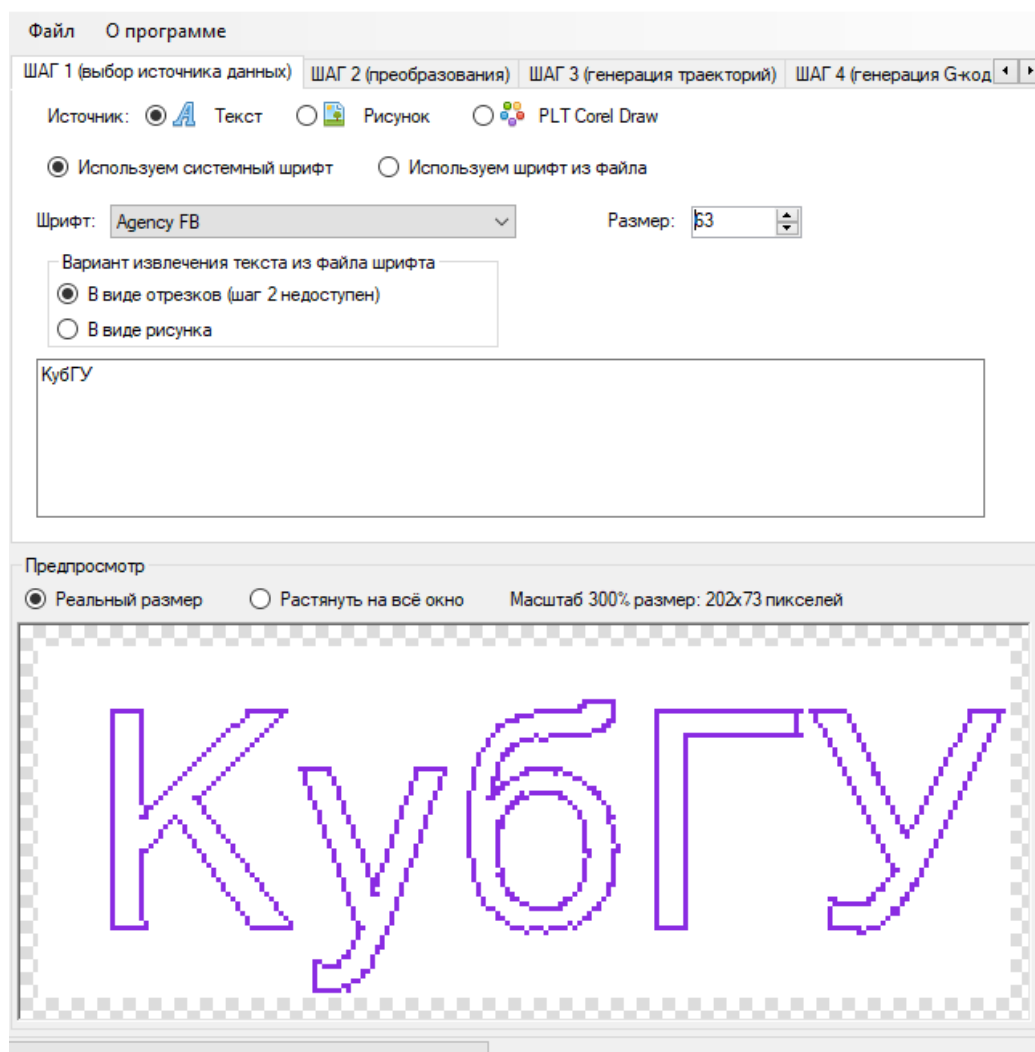


Рисунок 33 – Реализация первого шага

Вкладка программы шаг 2 будет заблокирована, так как он нужен только для того, чтобы преобразовывать изображения. Во вводных данных указан текст, поэтому в нашем случае это не требуется и переходим к шагу 3. Реализацию этой ситуации можно наблюдать на рисунке 34.

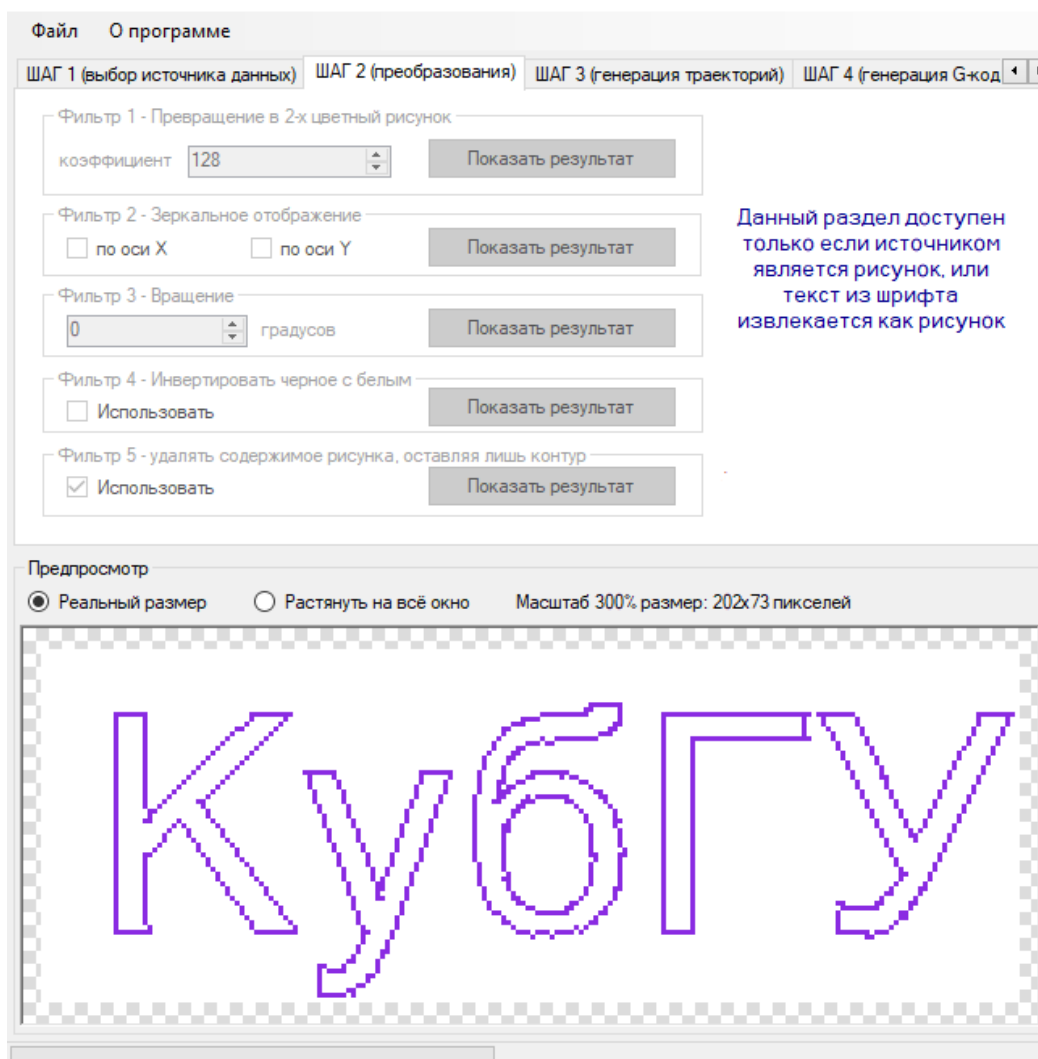


Рисунок 34 – Заблокированное окно преобразования

На 3 шаге необходимо указать минимальный размер отрезка и угол отклонения, если он потребуется. И нажать на кнопку «Вычислить отрезки». После этого в окне итоговые отрезки отобразятся все вычисленные отрезки и их подсветка при нажатии на каждый из них. На рисунках 35 и 36 показан пример работы.

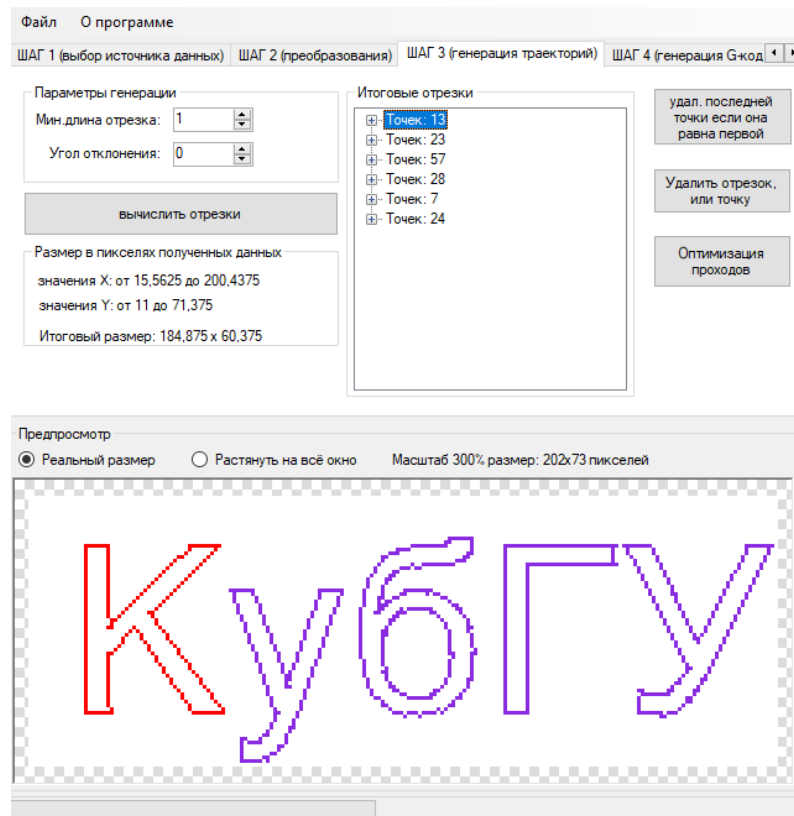


Рисунок 35 – Визуализация 3 шага

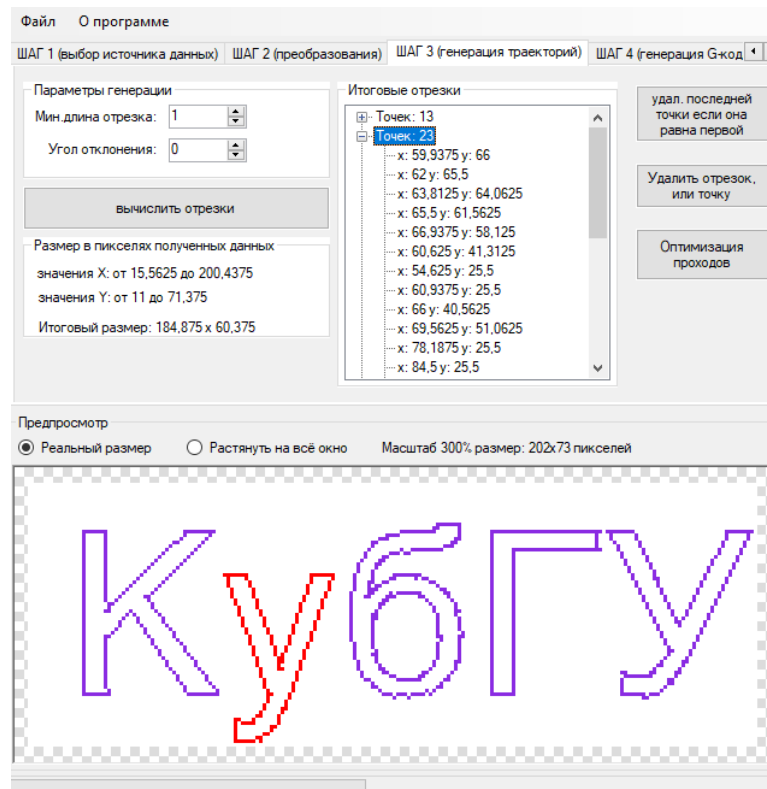


Рисунок 36 – Визуализация 3 шага

На финальном шаге указываем в окне «Желаемый размер в мм» все необходимые параметры и нажимаем на кнопку «Сформировать код фрезеровки». После проделанных действий в окне вывода получим готовый код для ЧПУ фрезерного оборудования с возможностью сохранить или скопировать. Итог работы программы можно посмотреть на рисунке 37.

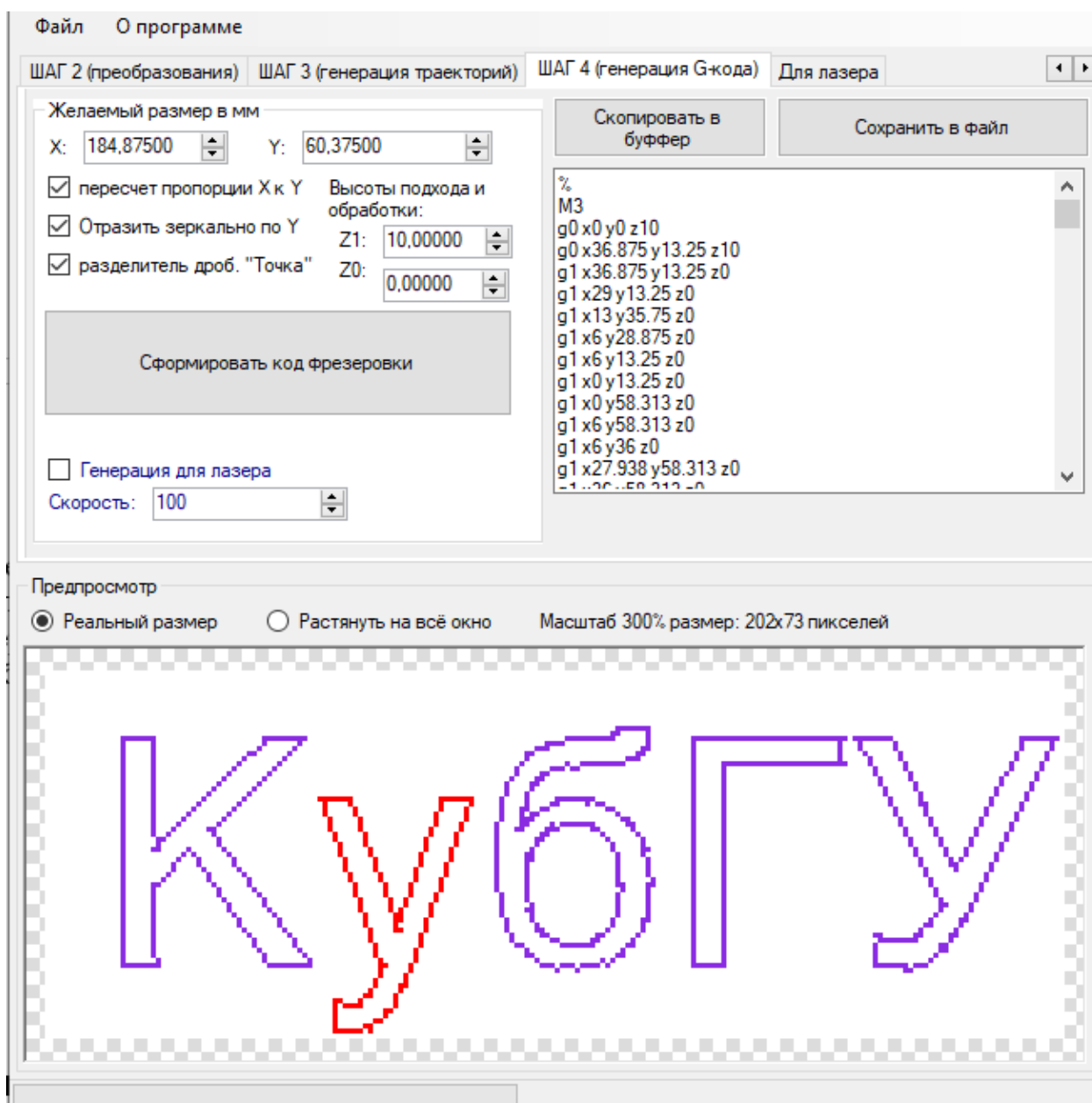


Рисунок 37 – Итог работы программы

После этого шага была получена готовая программа для ЧПУ оборудования, которую можно применить в производственном процессе.

3.4 Отладка и тестирование программы на примере изображения

Далее, наглядно рассмотрим цикл выполнения программы с выбранным типом исходных данных – изображение. Для того чтобы выбрать изображение, в окне выбора источника данных нужно поставить маркер возле рисунка, после чего открывается окно выбора файла. Оно представлено на рисунке 38.

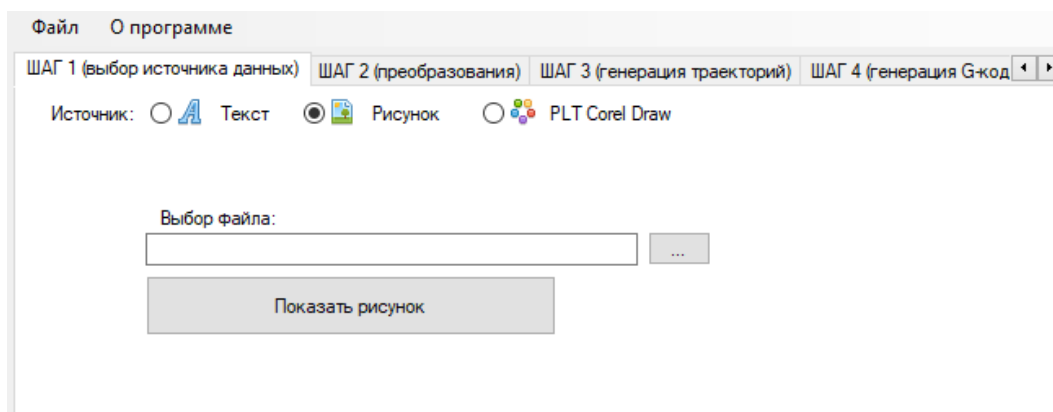


Рисунок 38 – Окно выбора файла

После выбора файла в формате .jpg и нажатия кнопки «Показать рисунок» в окне предпросмотра появится загруженное изображение. Следующим этапом станет доступен шаг 2. На нем необходимо преобразовать цветное изображение в черно-белое и выполнить необходимые преобразования. Как это выглядит можно посмотреть на рисунке 39.

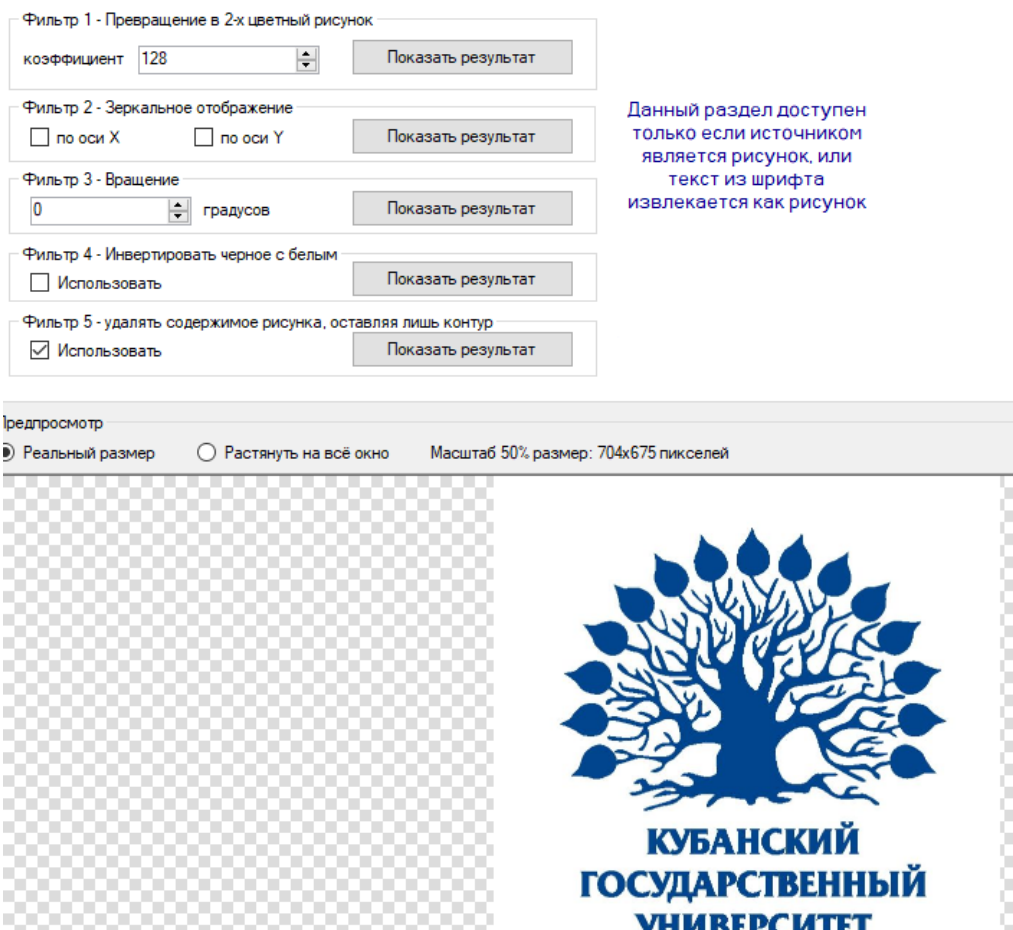


Рисунок 39 – Шаг 2

Проведем следующие настройки изображения:

- отразим изображение по оси X. Это может пригодиться если нам нужно фрезеровать содержимое изображение с обратной стороны материала;
- повернем изображение на 90 градусов.

Результатом этих преобразований является изображение, представленное на рисунке 40.

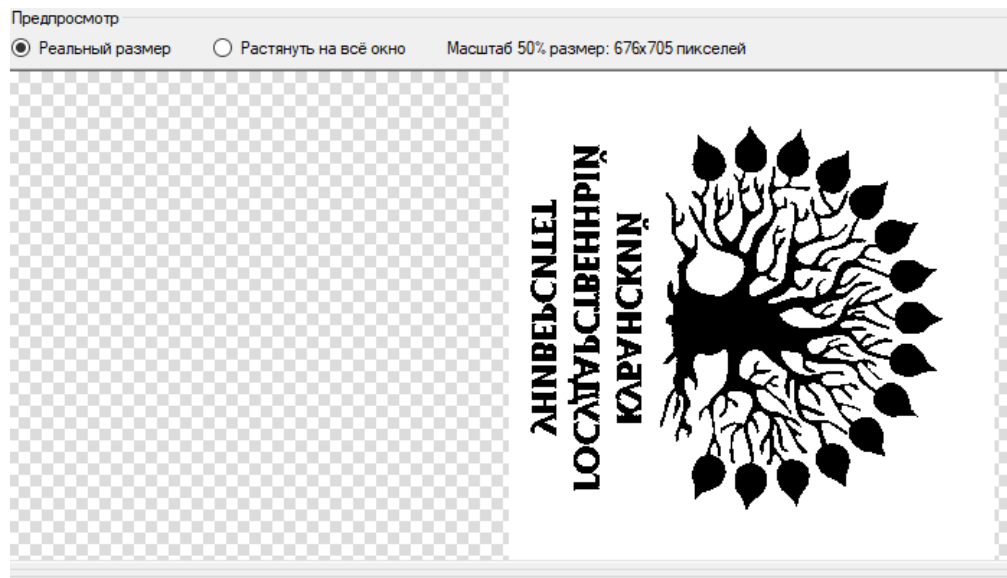


Рисунок 40 – Новое изображение

В окне 3 шага нажимаем на кнопку «вычислить отрезки» и получаем изображение, разбитое на группы отрезков (Рисунок 41).

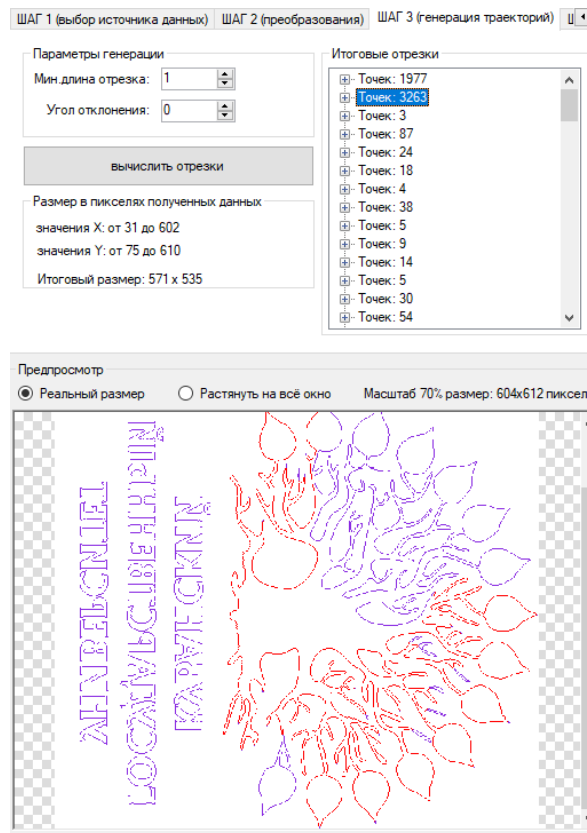


Рисунок 41 – Изображение, разбитое на отрезки

Так как изображение намного сложнее текста из-за своего разнообразия форм, то и отрезков понадобилось намного больше.

И последним шагом осталось преобразовать готовые сектора в G-code. Сделать это можно путем нажатия кнопки «Сформировать код фрезеровки» во вкладке генерации G-кода (Рисунок 42).

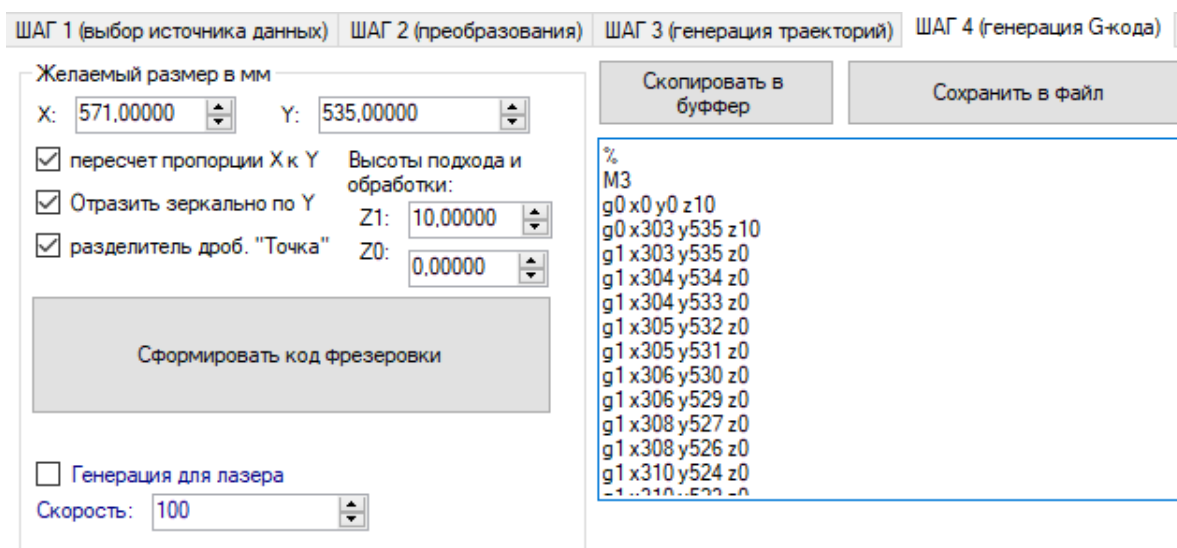


Рисунок 42 – Итог выполнения программы

Результатом выполнения программы стал G-code отредактированного растрового изображения для дальнейшей загрузки в систему фрезерного станка с ЧПУ.

3.5 Описание разработанного алгоритма

Для реализации этого программного обеспечения был разработан и применен собственный алгоритм преобразования изображения в G-code. Разберем его подробно.

Первым шагом для получения G-code будет выбор начального изображения, которое представлено на рисунке 43.



Рисунок 43 – Начальное изображение

Так как для фрезерного станка совершенно не важен цвет изображаемых объектов, требуется обесцветить картинку, сделав ее черно-белой для упрощения работы (Рисунок 44).



Рисунок 44 – Черно-белое изображение

Следующим шагом в получении готового кода будет разбиение исходного изображения на логически разбитые отрезки для дальнейшей оптимизации программного обеспечения.

Отрезки разбиваются либо ограничением на количество точек в отрезке, либо по завершенным пересекающимся линиям. Пример двух этих случаев представлен на рисунках 45 и 46.

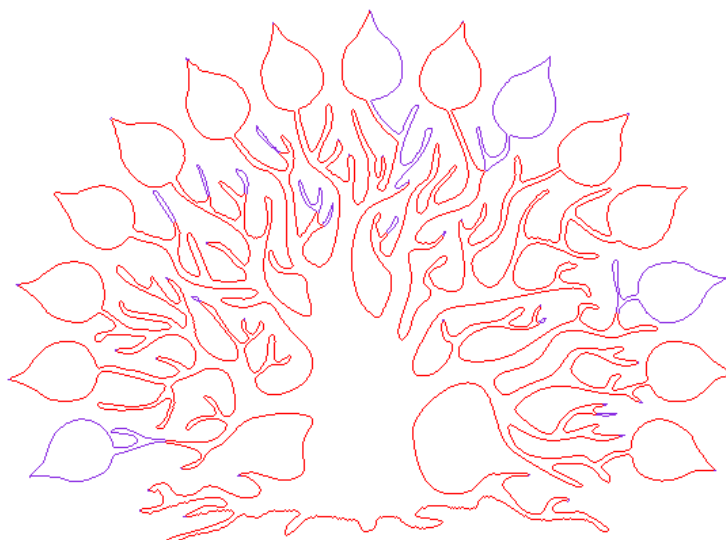


Рисунок 45 – Отрезок, ограниченный количеством

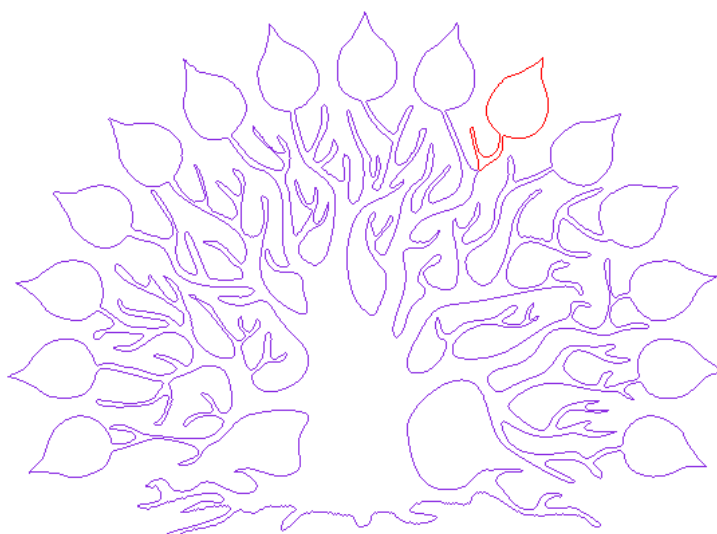


Рисунок 46 – Отрезок, ограниченный пересечением

Идея данной процедуры заключалась в рациональном использовании оперативной памяти и вычислительной мощности процессора персонального компьютера, на котором генерируется код.

Для сравнения понаблюдаем за нагрузкой процессора в момент генерации кода для полного изображения и для отдельно разбитых отрезков. Полученные показания представлены на рисунках 47 и 48.

12%	47%	0%	0%	2%
ЦП	Память	Диск	Сеть	GPU

Рисунок 47 – Параметры системы при разбиении

24%	48%	12%	31%	16%
ЦП	Память	Диск	Сеть	GPU

Рисунок 48 – Параметры системы без разбиения

Как можно наблюдать, итоговая нагрузка на процессор снизилась в 2 раза. Дальнейшей оптимизации не требуется, так как остальные вычислительные работы не зависят от грамотности распределения отрезков.

Последним шагом является расчет самих точек по формулам векторной алгебры. Суть реализованных в коде программного обеспечения функций по вычитыванию самих точек заключается в поочередном обходе всех точек каждого отрезка и смещении координат к каждой новой точке.

ЗАКЛЮЧЕНИЕ

ЧПУ остается очень важным и актуальным в различных областях, таких как производство, робототехника, авиация и медицина. Оно обеспечивает точное и эффективное управление механизмами и устройствами, повышая производительность и качество работы. С развитием технологий ЧПУ становится еще более гибким и доступным, что расширяет его применение и делает его востребованным во многих отраслях.

В рамках данной выпускной квалификационной работы мной была достигнута поставленная цель – разработка и оптимизация программного обеспечения для ЧПУ оборудования в производственных предприятиях. В процессе реализации данной цели были решены следующие задачи:

- применены основы разработки для программного обеспечения ЧПУ;
- проведён анализ и сравнение, представленных на рынке, программных продуктов для разработки обеспечения ЧПУ;
- разработан и применен алгоритм получения G-code;
- выявлены преимущества и недостатки готовых программных решений;
- сформулированы требования к программному обеспечению;
- реализовано готовое приложение, с помощью которого можно написать G-code для ЧПУ оборудования.

Реализованное программное обеспечение позволяет создавать код для ЧПУ станков. Оно преобразует текстовые данные и растровые изображения в векторы и высчитывает траектории движения. Результатом программы является готовый G-code, который в последствии можно импортировать, дополнить и загрузить в станок.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аверченков, А. В. Станки с ЧПУ: устройство, программирование, инструментальное обеспечение и основастка / А. В. Аверченков – Москва, 2014. – ISBN 978-5-9765-1830-8.
2. Колошкина, И. Е. Основы программирования для станков с ЧПУ / И. Е. Колошкина – Москва, 2019. – ISBN 978-5-534-12512-2.
3. Турчин, Д. Е. Программирование обработки на станках с ЧПУ / Д. Е. Турчин – Москва, 2022. – ISBN 978-5-9729-0867-7.
4. Колошкина, И. Е. Основы программирования для станков с ЧПУ в САМ-системе / И. Е. Колошкина – Москва, 2022. – ISBN 978-5-9729-0949-0.
5. Классификация станков с ЧПУ, их виды и возможности: сайт – URL: <https://top3dshop.ru/blog/klassifikatsija-stankov-s-chpu.html> (Дата обращения 20.04.2024)
6. Программирование станков с ЧПУ: сайт URL: <https://top3dshop.ru/blog/cnc-machines-programming.html> (Дата обращения 20.04.2024)
7. Системы ЧПУ. Поколения систем ЧПУ. Термины и понятия систем ЧПУ: сайт URL: https://stanki-katalog.ru/st_53.htm (Дата обращения 01.06.2024)
8. Принцип системы ЧПУ: сайт URL: <https://infofrezer.ru/stati/printsip-sistemy-chpu-chislovogo-programmnogo-upravleniya-freznykh-stankov/> (Дата обращения 04.06.2024)
9. Назначение и архитектура современных систем с ЧПУ: сайт URL: <https://scienceforum.ru/2022/article/2018028898> (Дата обращения 04.06.2024)
10. Что такое G-code для 3D-печати: сайт URL: <https://3d-m.ru/chto-takoe-g-code-dlya-3d-pechati/> (Дата обращения 06.06.2024)