Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Кафедра компьютерных систем в управлении и проектировании (КСУП)

**РАЗРАБОТКА ПЛАГИНА «СТАМЕСКА» ДЛЯ САПР «КОМПАС-3D»**

Пояснительная записка к лабораторной работе

По дисциплине «Основы разработки САПР» на тему

«Построение зубила в системе КОМПАС 3D»

Студент гр.589-2

\_\_\_\_\_\_Каминский В.М.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Доцент каф. КСУП

\_\_\_\_\_\_Калентьев А. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Томск 2022

СОДЕРЖАНИЕ

[**1.** **Введение** 4](#_Toc121032923)

[**2.** **Постановка и анализ задачи** 5](#_Toc121032924)

[**2.1.** **Описание САПР** 5](#_Toc121032925)

[**2.2.** **Выбор инструментов и средств реализации** 14](#_Toc121032926)

[**2.3.** **Назначение плагина** 16](#_Toc121032927)

[**2.4.** **Обзор аналогов** 17](#_Toc121032928)

[**3.** **Описание реализации** 18](#_Toc121032929)

[**3.1.** **UML** 18](#_Toc121032930)

[**3.2.** **Диаграмма классов** 19](#_Toc121032931)

[**4.** **Описание программы для пользователя** 27](#_Toc121032932)

[**5.** **Тестирование функционала** 29](#_Toc121032933)

[**5.1.** **Функциональное тестирование** 29](#_Toc121032934)

[**5.2.** **Модульное тестирование** 33](#_Toc121032935)

[**5.3.** **Нагрузочное тестирование** 35](#_Toc121032936)

[**Заключение** 37](#_Toc121032937)

[**Список использованных источников** 38](#_Toc121032938)

1. **Введение**

В настоящее время проектирование в своем понимании представляет собой автоматизированный процесс и в некотором роде программно-аппаратный. Проектировщику, который занимается разработкой сложного механизма, или устройства, требующего больших расчетов, математических вычислений при построении модели и высокой точности, подходят системы автоматизации проектных решений — САПР [1].

САПР позволяют уменьшить финансовые затраты на разработку макета (модели) проекта (объекта), а также сократить время, которое тратит проектировщик на создание модели объекта и составление проектной документации.

В каждой крупной САПР есть свой средства для разработки, которые предоставляются с целью дать возможность разработчикам расширить функционал данной системы под свои конкретные нужды. Данным средством является API — программируемый интерфейс приложения [2]. Это набор готовых средств: классов, процедур, функций, структур и т.д. API позволяет определить функциональность, которую предоставляет приложение, при этом абстрагируясь от того, как она реализована.

Расширение функционала в основном подразумевает разработку плагина или библиотеки на основе предоставленного API. В данном курсовом проекте стоит задача разработки плагина для построения 3D модели болта с гайкой в автоматизированном режиме. Плагин — независимо компилируемый программный модуль, динамически подключаемый к основной программе, предназначенный для расширения или использования ее возможностей [3].

В качестве системы, которая предоставляет API и для которой стоит задача разработать плагин, была взята САПР «КОМПАС-3D» версии 21.

1. **Постановка и анализ задачи**
   1. **Описание САПР**
      1. **Описание программы**

КОМПАС-3D — система трехмерного проектирования, ставшая стандартом для тысяч предприятий, благодаря сочетанию простоты освоения и легкости работы с мощными функциональными возможностями твердотельного и поверхностного моделирования. Ключевой особенностью продукта является использование собственного математического ядра С3D и параметрических технологий, разработанных специалистами АСКОН. КОМПАС-3D обеспечивает поддержку наиболее распространенных форматов 3D-моделей (STEP, ACIS, IGES, DWG, DXF), что позволяет организовывать эффективный обмен данными со смежными организациями и заказчиками, использующими любые CAD / CAM / CAE-системы в работе.

* + 1. **Описание API**

В КОМПАС на данный момент существуют API двух версий: API 5 и API 7. Обе версии реализуют различные функции системы и взаимно дополняют друг друга. Отсюда очевидно, что обе версии программных интерфейсов в равной мере поддерживаются и развиваются с учетом самих изменений в системе. В основном, для создания полноценных подключаемых модулей достаточно методов и свойств интерфейсов API 5.

Главным интерфейсом API системы КОМПАС является KompasObject. Получить указатель на этот интерфейс (если быть точным, на интерфейс приложения API 5) можно при работе под управлением внешнего приложения (контроллера) – после вызова стандартной системной функции. Методы этого интерфейса реализуют наиболее общие функции работы с документами системы, системными настройками, файлами, а также дают возможность получить указатели на другие интерфейсы (интерфейсы динамического массива, работы с математическими функциями, библиотек моделей или фрагментов и различных структур параметров определенного типа).

Ниже в таблице 2.1.1 представлены свойства и методы интерфейса KompasObject, которые были использованы при разработке плагина.

Таблица 2.1.1 – Методы и свойства интерфейса KompasObject, используемые при разработке плагина

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Document3D() | ksDocument | Метод для получения указателя на интерфейс трехмерного графического документа (детали или сборки) |
| GetParamStruct(short structType) | StructType2D | Метод для получения указателя на интерфейс графического документа (чертежа или фрагмента) |
| Visible | bool | Свойство видимости приложения |
| Quit() |  | Метод для закрытия активного окна приложения КОМПАС |

В таблице 2.1.2 представлены методы интерфейса ksEntity, которые были использованы при разработке плагина.

Таблица 2.2.2 – Методы интерфейса ksEntity, используемые при разработке плагина

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Create() | bool | Создать объект в модели |
| GetDefinition() | IUnkown | Получить указатель на интерфейс параметров объектов и элементов |
| Update() | bool | Изменить свойства объекта (используя ранее установленные свойства) |

В таблице 2.1.3 представлены свойства и методы интерфейса ksDocument2D, которые были использованы при разработке плагина.

Таблица 2.1.3 – Методы интерфейса ksDocument2D, используемые при разработке плагина

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ksLineSeg(double x1, double y1, double x2, double y2, int style) | int | Получить указатель на отрезок на двумерной плоскости либо 0 в случае ошибки |
| ksRegularPolygon(  ksRegularPolygonParam param, int style) | int | Получить указатель на многоугольник на двумерной плоскости либо 0 в случае ошибки |
| ksRectangle(  ksRectangleParam param, int style) | int | Получить указатель на прямоугольник на двумерной плоскости либо 0 в случае ошибки |
| ksCircle(  double xc, double yc, double rad, int style) | int | Получить указатель на окружность на двумерной плоскости либо 0 в случае ошибки |

В таблице 2.1.4 представлены свойства и методы интерфейса ksDocument3D, которые были использованы при разработке плагина.

Таблица 2.1.4 – Методы интерфейса ksDocument3D, используемые при разработке плагина

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Create (bool invisible, bool \_typeDoc) | bool | Создать документ-модель (деталь или сборку) |
| GetPart(int type) | ksPart | Получить указатель на интерфейс компонента в соответствии с заданным типом |

В таблице 2.1.5 представлены методы интерфейса ksPart, которые были использованы при разработке плагина.

Таблица 2.1.5 – Свойства и методы интерфейса ksPart, используемые при разработке плагина

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| EntityCollection(short objType) | ksEnintyCollection | Формирует массив объектов и возвращает указатель на его интерфейс |
| GetDefaultEntity(short objType) | ksEntity | Получить указатель на интерфейс объекта, создаваемого системой по умолчанию |
| GetPart(int type) | ksPart | Получить указатель на интерфейс компонента в соответствии с заданным типом |
| NewEntity(short objType) | ksEntity | Создать новый интерфейс объекта и получить указатель на него |

В таблице 2.1.6 представлены типы объектов документа-модели, которые были использованы при разработке плагина.

Таблица 2.1.6 – Некоторые типы объектов документа-модели

|  |  |  |
| --- | --- | --- |
| Идентификатор объекта | Название объекта | Интерфейс параметров |
| o3d\_unknown | Неизвестный (включает все объекты) |  |
| o3d\_planeXOZ | Плоскость XOZ | ksPlaneParam |
| o3d\_planeYOZ | Плоскость YOZ | ksPlaneParam |
| o3d\_planeXOY | Плоскость XOY | ksPlaneParam |
| o3d\_sketch | Эскиз | ksSketchDefinition |
| o3d\_face | Грань | ksFaceDefinition |
| o3d\_baseExtrusion | Базовая операция выдавливания | ksBaseExtrusionDefinition |
| o3d\_cutExtrusion | Вырезать выдавливанием | ksCutExtrusionDefinition |
| o3d\_baseLoft | Создание элемента по сечениям | ksBaseLoftDefinition |
| o3d\_baseEvolution | Создание кинематического элемента | ksBaseEvolutionDefinition |
| o3d\_cutEvolution | Вырезать кинематический элемент | ksCutEvolutionDefinition |

**Описание предмета проектирования**

Предметом проектирования является построение 3D модели стамески с прямым лезвием и уголковатым.

**Стамеска** — столярный инструмент, предназначенный: для выдалбливания неглубоких гнезд и отверстий; для снятия материала небольшой толщины (строгания); для подрезки плоскостей и выступов. Стамеска состоит из рукоятки и полотна.

Виды, выбранные для реализации плагина: угловатая и прямая.

**Угловатую** стамеску следует применять для создания выемок при художественной резьбе. При помощи инструмента удается сделать выпуклые или вогнутые засечки.

**Прямая** стамеска имеет прямой, плоский профиль кромки полотна. В среднем имеет ширину кромки полотна от 3 до 50 мм. Применяется такая стамеска для получения аккуратного, ровного среза. Стамеской можно делать углубление с прямым дном, или же убирать лишнюю поверхность древесины.

На рисунке 2.2.1 представлен вид на 3D модель прямой стамески

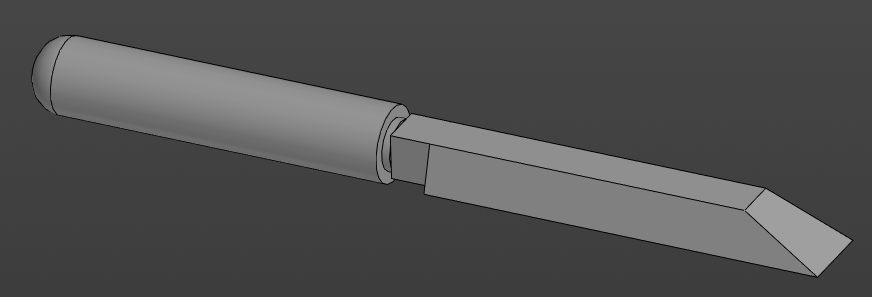


Рисунок 2.2.1 – Вид на 3D модель прямой стамески

На рисунке 2.2.2 представлен вид на 3D модель угловатой стамески

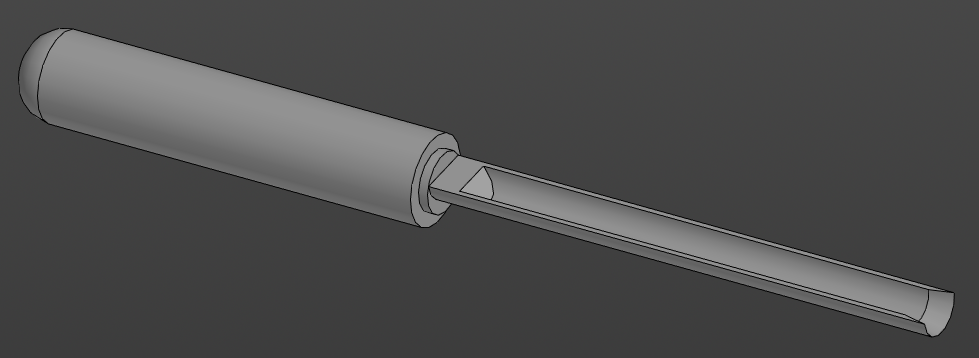


Рисунок 2.2. – Вид на 3D модель угловатой стамески

**Выбор инструментов и средств реализации**

В рамках лабораторной работы было предоставлено задание разработать плагин автоматического построения 3D модели детали. В качестве детали были выбраны модели двух видов зубил: столярного и зубила-пики.

Для реализации полноценного плагина необходимо было построить приложение, с интерфейсом для ввода параметров построения выбранной модели, выбрать систему трехмерного моделирования, наиболее подходящую для выбранного плагина, а также интерфейс программирования приложений(API) наиболее подходящий для выбранного плагина.

1. **Среда разработки**

В качестве среды разработки была выбрана Microsoft Visual Studio 2019.

Microsoft Visual Studio — линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом.

1. **Язык программирования**

В качестве языка программирования был выбран C#.

C# — объектно-ориентированный язык программирования.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

1. **Система трехмерного моделирования**

В качестве системы моделирования был выбран КОМПАС 3D.

Система «Компас-3D» предназначена для создания трёхмерных ассоциативных моделей отдельных деталей (в том числе, деталей, формируемых из листового материала путём его гибки) и сборочных единиц, содержащих как оригинальные, так и стандартизованные конструктивные элементы. Параметрическая технология позволяет быстро получать модели типовых изделий на основе проектированного ранее прототипа.

**Интерфейс программирования приложений**

В качестве API был выбран КОМПАС API.

API КОМПАС 3D это ориентированные на прикладного программиста инструментальные средства разработки приложений (библиотек конструктивов, прикладных САПР) на базе системы КОМПАС.

API КОМПАС 3D включает в свой состав 2D API и 3D API.

2D API обеспечивает доступ к системе КОМПАС для формирования и обработки двумерных графических документов.

* 1. **Назначение плагина**

Данный плагин представляет собой приложение с выбором параметров и модели стамески, с основной функцией – проектирование выбранного инструмента в соответствии заданным параметрам. Проектирование осуществляется с помощью САПР КОМПАС API. Плагин предназначен для упрощения проектировки и построения модели стамески, в приложении КОМПАС 3D.

Количество элементов построения и дерево действий над деталью показано на рисунке 2.4.

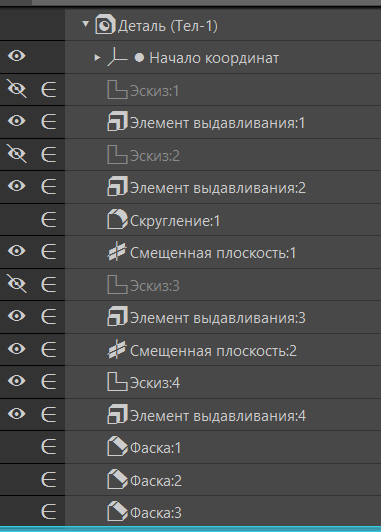


Рисунок 2.4 Элементы построения в дереве детали

Из рисунка видно, что количество произведенных действий над деталью достаточно большое, при этом у каждой имеется зависимость и допустимые значения. Поэтому вручную построение такой детали займет гораздо больше времени.

Время построения одной детали программой –1-7 секунд, в зависимости от порядкового номера детали.

Время построения 80 деталей последовательно – 3.5 минуты,

* 1. **Обзор аналогов**

**Плагин «3д модель стамески» для SolidWorks 2017**

SolidWorks – продукт компании SolidWorks Corporation (США). Программа Solid Works® – это система автоматизированного проектирования (САПР), использующая привычный графический интерфейс пользователя Microsoft Windows. Другими словами это легкое в освоении средство позволяет инженерам-проектировщикам быстро отображать свои идеи в эскизе, экспериментировать с элементами и размерами, а также создавать модели и подробные чертежи.

**Стамеска**, 6 конфигураций от 6 до 25 мм. Предназначена для обработки дерева (долбление). Обозначение уловное для удобства работы в проекте.

Состав: 3D сборка, детали Язык документа

Софт: SolidWorks 2017

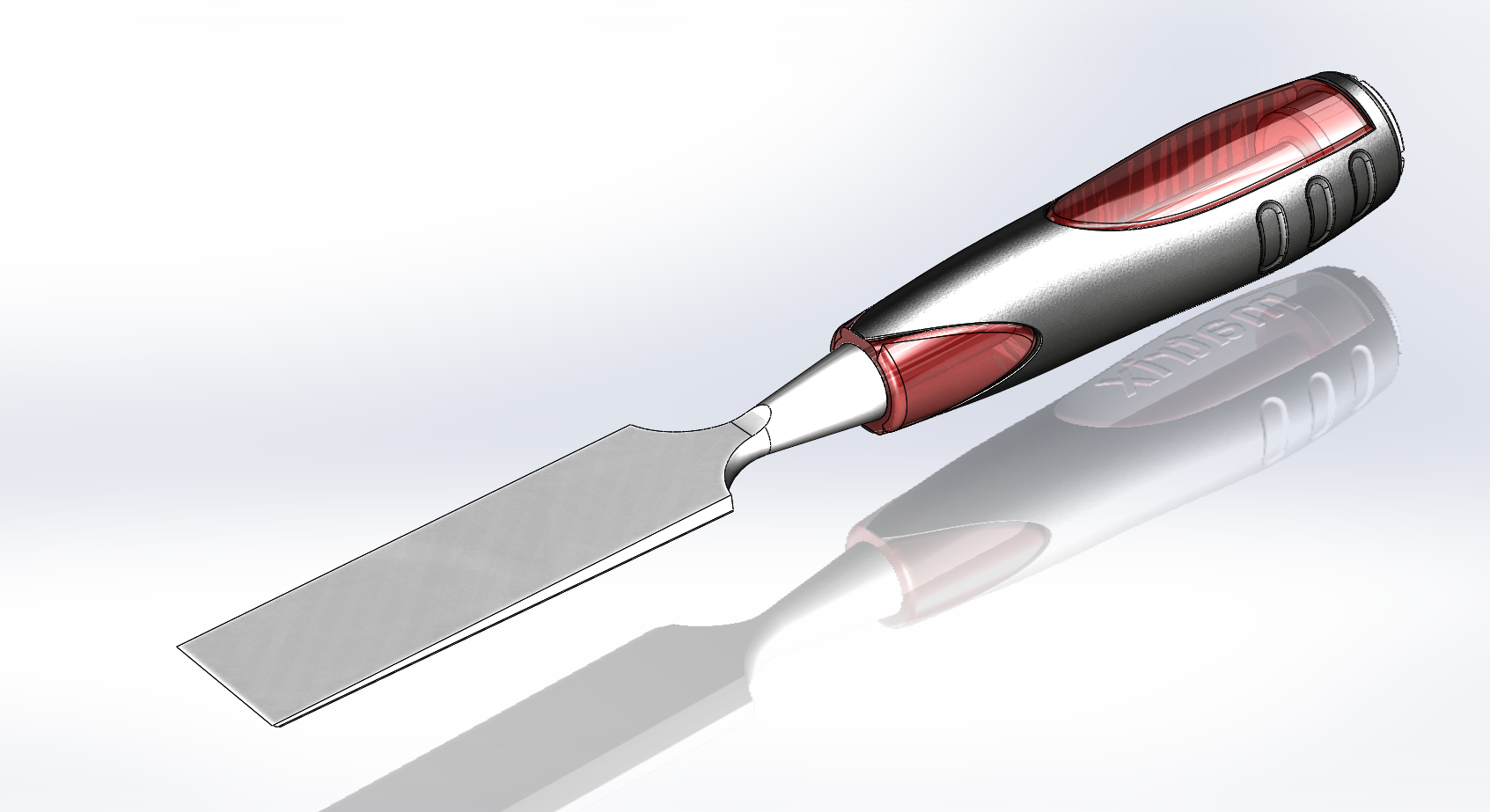


Рисунок 2.5 Моделируемые объект плагина

1. **Описание реализации**
   1. **UML**

Для графического описания абстрактной модели проекта, а также пользовательского взаимодействия (сценария действий) использован стандарт UML.

UML – это язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

При использовании UML были построены: диаграмма использования и диаграмма классов.

* 1. **Диаграмма классов**

Диаграмма классов для данного проекта представлена на рисунке 3.2.1

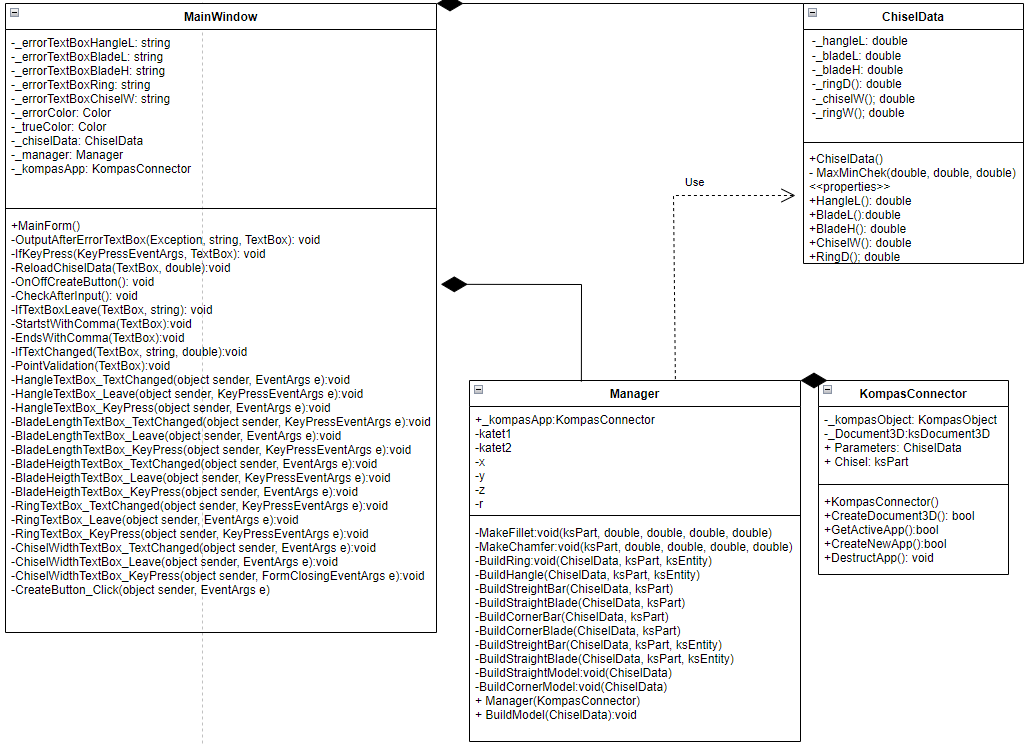


Рисунок 3.2.1 – UML-диаграмма классов

1) MainWindow – класс диалогового окна, который обеспечивает взаимодействие между пользователем и программой.

2) ChiselData − класс, хранящий в себе все параметры 3D-модели;

3) Manager − класс, осуществляющий инициализацию, создание модели и валидацию введенных данных;

4) KompasConnector – класс для работы с API КОМПАС 3D;

Класс MainWindow композирует классы ChiselData и Manager.

Класс Manager композирует класс CompasConnector и использует класс ChiselData.

В таблице 3.2.1 представлено описание полей и методов класса MainWindow.

Таблица 3.2.1 – Описание полей и методов класса MainWindow.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| \_errorTextBoxHangle | Переменная ошибки в поле рукояти |
| \_errorTextBoxBladeL | Переменная ошибки в поле длины лезвия |
| \_errorTextBoxBladeH | Переменная ошибки в поле высоты лезвия |
| \_errorTextBoxRing | Переменная ошибки в поле кольца |
| \_errorTextBoxChiselW | Переменная ошибки в поле ширины рукояти |
| -\_errorColor: Color | Переменная цвета ошибки |
| -\_trueColor: Color | Переменная цвета правильной валидации |

Продолжение таблицы 3.2.1

|  |  |
| --- | --- |
| -\_chiselData: ChiselData | Переменная класса параметров зубила |
| -\_manager:Manager | Переменная менеджера |
| -\_kompasApp: KompasConnector | Переменная компас коннектора |
| +MainForm() | Конструктор главного окна |
| -PointValidation(TextBox): void | Функция валидации запятых |
| -OutputAfterErrorTextBox(TextBox, Exception): void | Функция вывода ошибки |
| -StartsWithComma(TextBox textBox):void | Функция обработчик ввода запятой первым символом |
| -EndsWithComma(TextBox textBox): void | Функция обработчик ввода запятой последним символом |
| -IfKeyPress(KeyPressEventArgs, TextBox): void | Функция обработчика события нажатия на клавишу |
| -IfTextBoxTextChanged(TextBox, string, double): void | Функция обработчика события изменения текста текстбокса |
| -IfTextBoxLeave(TextBox):void | Функция обработчика события выхода из текстбокса |
| -ReloadChiselData(TextBox, double):void | Функция присвоения значения параметрам класса зубила |
| -CheckAffterInput():void | Функция валидации всех полей |
| -OnOffCreateButton():void | Функция включения выключения кнопки построения |

Продолжение таблицы 3.2.1

|  |  |
| --- | --- |
| -RadioButtonModel\_Click(object, EventArgs): void | Обработчик события клика при выборе модели стамески |
| -TextBox\_TextChanged(object, EventArgs):void | Обработчик события изменения текста в текстбоксах |
| - HandleTextBox\_KeyPress(object, KeyPressEventArgs):void | Обработчик события нажатия на клавишу в текстбоксе длины рукояти |
| - HandleTextBox\_Leave(object, EventArgs):void | Обработчик события выхода из текстбокса длины рукояти |
| -BladeLengthTextBox\_KeyPress(object, KeyPressEventArgs : void | Обработчик события нажатия на клавишу в текстбоксе длины лезвия |
| -BladeLenghtTextBox\_Leave(object, EventArgs):void | Обработчик события выхода из текстбокса длины лезвия |
| - BladeHeightTextBox\_KeyPress(object, KeyPressEventArgs):void | Обработчик события нажатия на клавишу в текстбоксе высоты лезвия |
| - BladeHeightTextBox\_Leave(object, EventArgs):void | Обработчик события выхода из текстбокса высоты лезвия |
| -RingTextBox\_KeyPress(object ,KeyPressEventArgs):void | Обработчик события нажатия на клавишу в текстбоксе диаметра кольца |

Продолжение таблицы 3.2.1

|  |  |
| --- | --- |
| - RingTextBox\_Leave(object, EventArgs):void | Обработчик события выхода из текстбокса диаметра кольца |
| -ChiselWidthTextBox\_KeyPress(object , KeyPressEventArgs):void | Обработчик события нажатия на клавишу в текстбоксе ширины стамески |
| - ChiselWidthTextBox\_Leave(object, EventArgs):void | Обработчик события выхода из текстбокса длины ширины стамески |
| -CreateButton\_Click(object, EventArgs):void | Обработчик события нажатия на кнопку построения детали |

В таблице 3.2.2 представлено описание свойств и методов класса ChiselData.

Таблица 3.2.2 – Описание свойств и методов класса ChiselData.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| -minl: double | Минимальное значение длины рукояти |
| -maxl: double | Максимальное значение длины рукояти |
| -minbl: double | Минимальное значение длины лезвия |
| -maxbl: double | Максимальное значение длины лезвия |
| -minbh: double | Минимальное значение ширины рукояти |
| -maxbh: double | Максимальное значение ширины рукояти |
| -minrd: double | Минимальное значение диаметра кольца |
| -maxrd: double | Максимальное значение диаметра кольца |
| -minc: double | Минимальное значение ширины стамески |
| -maxc: double | Максимальное значение ширины стамески |
| -MaxMinCheck(double, double, double) | Вывод ошибки от максимальной и минимальной длины |
| +HandleL: double | Длина рукояти |
| +BladeL: double | Длина лезвия |
| +BladeH: double | Ширина лезвия |
| +RingD: double | Диаметр кольца |
| +ChiselW: double | Ширина стамески |
| +ChiselData() | Объявление класса |

В таблице 3.2.3 представлено описание полей и методов класса Manager.

Таблица 3.2.3 – Описание полей и методов класса Manager.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| +\_kompasConnector:KompasConnector | Подключение к компас 3D |
| +Manager(KompasConnector) | Конструктор |
| +BuilderStraightModel (ChiselData):void | Функция сборки модели прямой стамески |
| +BuilderCornerModel (ChiselData):void | Функция сборки модели прямой стамески |
| -MakeChamfer(ksPart, bool, double, double,double, double, double): void | Функция построения фаски |
| -MakeFillet(ksPart, bool, double, double,double, double, double): void | Функция построения скругления |
| -BuildRing(ChiselData, ksPart, ksEntity):void | Функция построения кольца |
| -BuildHandle(ChiselData, ksPart, ksEntity):void | Функция построения рукояти |
| -BuildStraightBar(ChiselData, ksPart):void | Функция построения прямого бруска |
| -BuildCornerBar(ChiselData, ksPart):void | Функция построения угловатого бруска |
| -BuildStraightBlade(ChiselData, ksPart):void | Функция построения прямого лезвия |
| -BuildCornerBlade(ChiselData, ksPart):void | Функция построения уголковатого лезвия |

В таблице 3.2.4 представлено описание полей и методов класса KompasConnector.

Таблица 3.2.4 – Описание свойств и методов класса KompasConnector.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| +KompasConnector() | Конструктор |
| -KompasObject: KompasObject | Интерфейс API-системы КОМПАС |
| -Document3D:ksDocument3D | Интерфейс создания документа |
| +Chisel:ksPart | Интерфейс создания детали |
| +CreateDocument3D:bool | Функция создания документа |
| -GetActiveApp():bool | Функция активации приложения |
| -CreateNewApp():bool | Функция создания приложения |
| +DestructApp():void | Функция удаления приложения |

1. **Описание программы для пользователя**

При запуске программы пользователя встречает окно параметров зубила (рис.4.1a), где в верхней части окна находится изображение моделируемого объекта, а в нижней – поля для ввода параметров.

Сразу под изображением находится элемент выбора модели, с помощью которой можно выбрать либо построение прямого либо уголковатого зубила.

Изначально пункт выбора модели находится в положении «Straight blade» - прямой стамески, при переключении режима на «Corner blade» окно примет вид показанный на рисунке 4.1.б.

.2.1 Макеты пользовательского интерфейса

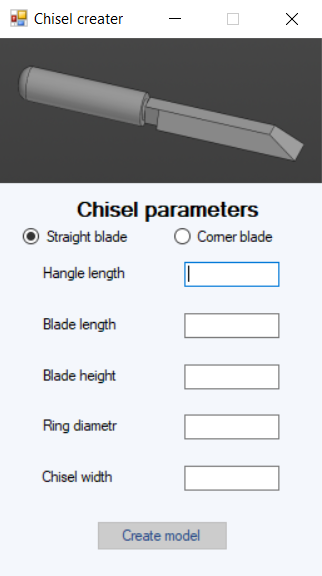


Рисунок 4.2.1a – начальный интерфейс программы

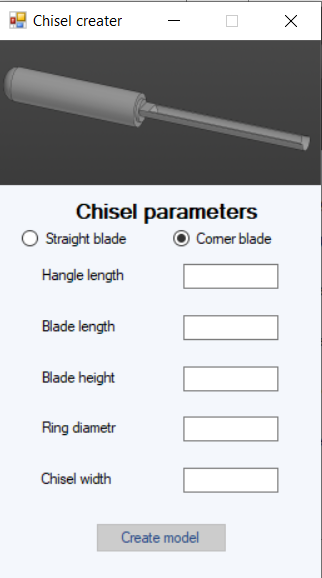


Рисунок 4.2.1б – вид окна с переключением вида стамески

1) В поле невозможно ввести буквы и символы, в тоже время комбинации клавиш доступны (ctrl c/v/x);

2) В поле невозможно ввести больше одной запятой;

3) При вводе нескольких нулей в начале либо удалит все нули если нет запятой после них, либо оставит только 1 ноль, если есть запятая 2м символом;

4) При вводе запятой первым символом, подставит ноль в начало;

5) При выходе из поля удалит незначащие нули, а также запятую, если она является последним символом.

1. **Тестирование функционала**
   1. **Функциональное тестирование**

Функциональное тестирование — это тестирование в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения (ПО) в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

При запуске плагина открывается окно с параметрами и изображением модели стамески, после корректного введения всех параметров становится доступна кнопка построения, при нажатии на которую будет запущено приложение КОМПАС 3D, если оно не было запущено вручную.

При вводе некорректных данных, поле параметра изменяет цвет фона на светло красный и высвечивается подсказка о возможном решении ошибки. О корректно введенных значениях программа уведомит пользователя зеленым фоном всех полей и активацией кнопки «Build».

На рисунке 4.1.1 показана разница между формой с корректно и некорректно введенными значениями соответственно.

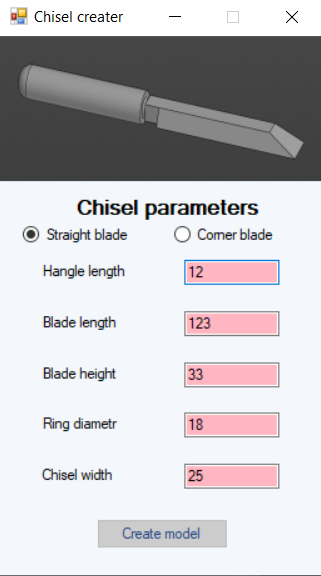
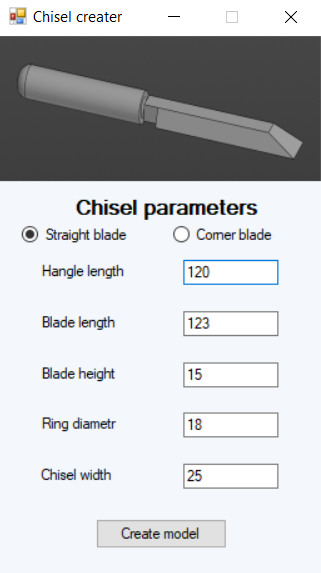


Рисунок 5.1.1 Ввод корректных и некорректных значений

На рисунке 5.1.2 представлен ввод минимально допустимых значений.

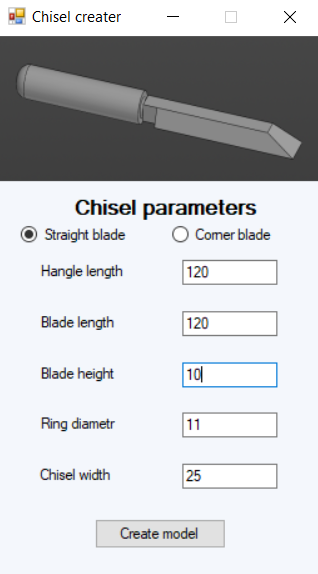


Рисунок 5.1.2 Ввод минимально допустимых значений

На рисунке 5.1.3 представлен результат ввода минимально допустимых значений.

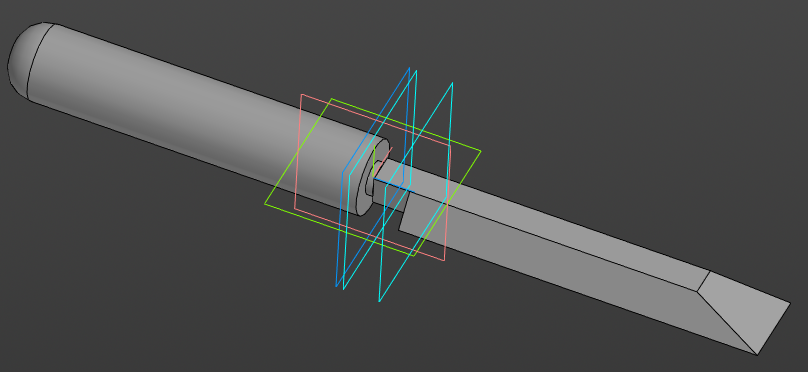


Рисунок 5.1.3 Результат ввода минимально допустимых значений

На рисунке 5.1.4 представлен ввод максимально допустимых значений.

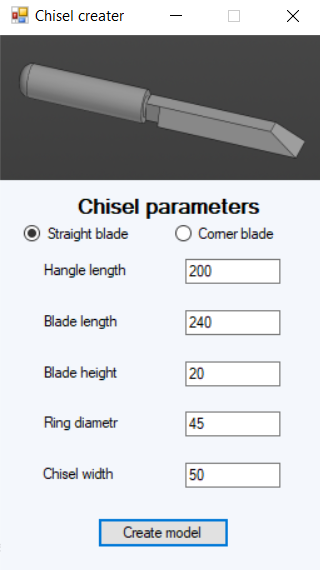


Рисунок 5.1.4 Ввод максимально допустимых значений

На рисунке 5.1.5 представлен результат ввода максимально допустимых значений.

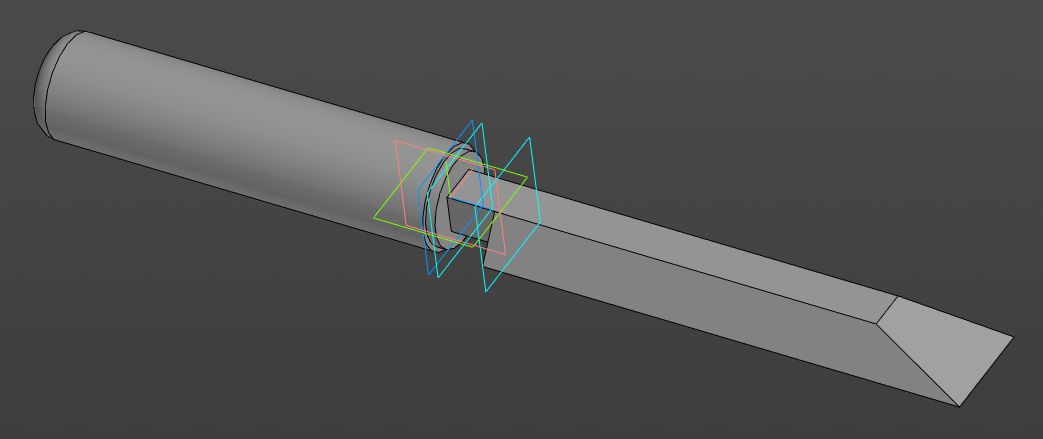


Рисунок 5.1.5 Результат ввода максимально допустимых значений

* 1. **Модульное тестирование**

Список тестовых сценариев для модульного тестирования представлен в таблице 5.2.

Таблица 5.2 – Список тестовых сценариев

|  |  |
| --- | --- |
| Название тестового метода | Назначение |
| TestValidation(bool, double, double, double, double, double, double):void | Тестирование параметров детали. Присвоение корректных и некорректных значений. |
| HandleL\_BladeL\_validator (bool, double, double):void | Тестирование зависимости длины рукояти и длины лезвия |
| BladeH\_RingD\_ChiselW\_Validator (bool, double, double, double):void | Тестирование зависимости параметров высоты лезвия, диаметра кольца и ширины стамески |

Результаты успешного прохождения всех модульных тестов приведены на рисунке 5.2

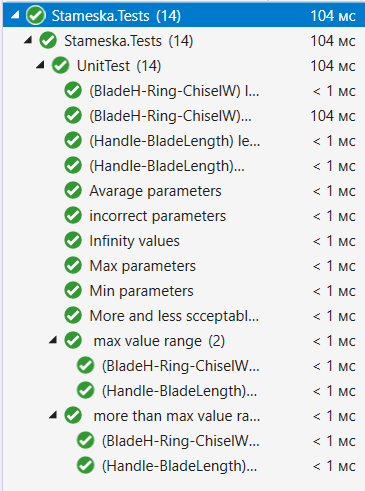
.

Рисунок 5.2 – Результаты модульных тестов

* 1. **Нагрузочное тестирование**

В целях проверки производительности работы плагина, было проведено нагрузочное тестирование. Тестирование производилось на ПК со следующей конфигурацией:

– AMD Ryzen 3400U with Radeon Graphics;

– 8 ГБ ОЗУ;

– графический процессор объемом памяти 4 ГБ.

Для нагрузочного тестирования был создан проект консольного приложения с бесконечным циклом построения детали. Для измерения времени был использован класс Stopwatch.

На рисунке 5.3.1 представлен график зависимости времени от количества построенных деталей.

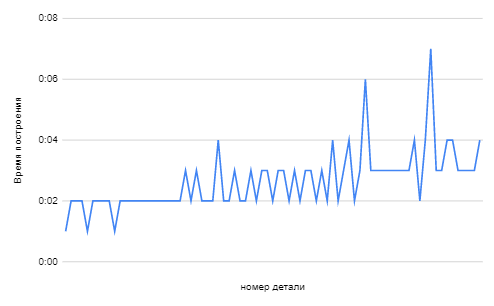


Рисунок 5.3.1 – График зависимости времени построения

от количества деталей

На рисунке 5.3.2 представлен график зависимости загруженности памяти от количества построенных деталей

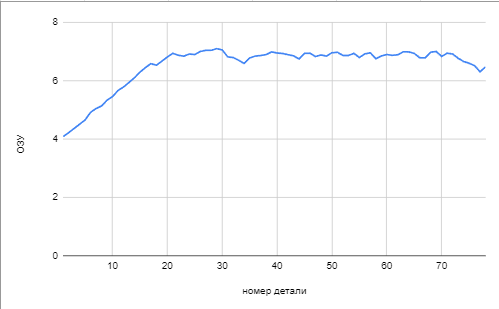


Рисунок 5.3.2 – График зависимости загруженности памяти

от количества построенных деталей

Тестирование длилось 3.5 минутs, было построено 78 моделей стамески.

Исходя из графика, представленного на рисунке 5.3.1 можно выделить линейную зависимость времени построения детали от ее номера. Из графика 5.3.2 видно как ОЗУ доходит до своего максимального значения и перестает расти.

**Заключение**

В ходе выполнения лабораторной работы были изучены основные этапы проектирования программного продукта, изучена предметная область предмета проектирования, также было изучено API системы автоматизированного проектирования «КОМПАС-3D».

В результате полученные знания были применены для реализации плагина для автоматизации построения модели объекта «Стамеска» в рабочей плоскости программы «КОМПАС-3D».

Для реализованного плагина были проведены функциональное, нагрузочное и модульное тестирования.

**Список использованных источников**

1. Kompas Invisible Руководство пользователя – Режим доступа: <https://kompas.ru/solutions/developers/kompas-invisible/> (Дата обращения 10.10.2022)
2. КОМПАС 3D: О программе. Официальный сайт САПР КОМПАС [Электронный ресурс] – Режим доступа: : <http://kompas.ru/kompas-3d/about/> (Дата обращения 12.11.2022)
3. API – Википедия [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/API> (Дата обращения 12.10.2022)
4. UML. [Электронный ресурс]. – Режим доступа: [http://www.uml.org/](http://www.uml.org/%20) (дата обращения 12.10.2022)
5. NUnit Documentation – Github [Электронный ресурс] – Режим доступа: <https://github.com/nunit/docs/wiki/NUnit-Documentation> (дата обращения 21.11.2022)
6. Функциональное тестирование – Википедия. [Электронный ресурс]. – Режим доступа: <http://ru.wikipedia.org/wiki/> Функциональное тестирование (дата обращения 21.12.2016)