Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждения высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

**РАЗРАБОТКА ПЛАГИНА «ДВЕРЬ»**

**ДЛЯ «КОМПАС-3D»**

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по дисциплине

«Основы разработки САПР» (ОРСАПР)

Выполнил:

студент гр. 580-3

\_\_\_\_\_\_\_\_\_\_ Пищулин М.Д.

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Руководитель:

к.т.н., доцент каф. КСУП \_\_\_\_\_\_\_\_\_\_ Калентьев А.А. «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Томск, 2023

Оглавление

[1 ВВЕДЕНИЕ 3](#_Toc153469116)

[2 ПОСТАНОВКА И АНАЛИЗ ЗАДАЧИ 4](#_Toc153469117)

[3 ОПИСАНИЕ ПРЕДМЕТА ПРОЕКТИРОВАНИЯ 5](#_Toc153469118)

[4 ВЫБОР ИНСТРУМЕНТОВ И СРЕДСТВ РЕАЛИЗАЦИИ 6](#_Toc153469119)

[5 НАЗНАЧЕНИЕ ПЛАГИНА 7](#_Toc153469120)

[6 ОБЗОР АНАЛОГОВ 8](#_Toc153469121)

[7 ОПИСАНИЕ РЕАЛИЗАЦИИ 9](#_Toc153469122)

[7.1 Диаграмма классов после проектирования 9](#_Toc153469123)

[7.2 Диаграмма классов после реализации 9](#_Toc153469124)

[8 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПОЛЬЗОВАТЕЛЯ 16](#_Toc153469125)

[9 ТЕСТИРОВАНИЕ ПЛАГИНА 18](#_Toc153469126)

[9.1 Функциональное тестирование 18](#_Toc153469127)

[9.2 Модульное тестирование 20](#_Toc153469128)

[9.3 Нагрузочное тестирование 22](#_Toc153469129)

[10 ЗАКЛЮЧЕНИЕ 25](#_Toc153469130)

[11 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc153469131)

# **1 ВВЕДЕНИЕ**

Современная индустрия требует эффективных решений для оптимизации процессов проектирования и производства. В этом контексте, разработка автоматизированных систем играет ключевую роль, обеспечивая ускоренный и точный процесс создания продукции. В рамках данного проекта реализовывается приложение для автоматизированного построения двери в среде КОМПАС-3D, используя язык программирования C#.

Проектирование двери является сложным процессом, который включает в себя учет различных параметров и требований. Автоматизация этого процесса позволит значительно упростить и ускорить разработку дверей.

Целью данного проекта является создание приложения, которое позволит инженерам и дизайнерам легко создавать и модифицировать двери и управлять их параметрами.

# **2 ПОСТАНОВКА И АНАЛИЗ ЗАДАЧИ**

Определение требований проекта (1-7 октября): согласование требований к проекту, определение параметров детали, построение детали вручную в среде КОМПАС-3Д.

Анализ существующих решений (8-14 октября): поиск прямых и косвенных аналогов.

Проектирование архитектуры (15-21 октября): продумывание возможной архитектуры приложения. Подводные камни: архитектура может поменяться в ходе разработки, так как можно не все предусмотреть во время проектирования.

Разработка пользовательского интерфейса (1-5 ноября): продумывание удобного пользовательского интерфейса, взаимодействие параметров между собой.

Написание основной логики (6-20 ноября): написание функциональности. Подводные камни: продуманная архитектура может не сработать.

Тестирование (21-25 ноября): покрытие Model тестами.

Интеграция среды КОМПАС-3Д (26-30 ноября): работа с API КОМПАС-3Д [3].

Разработка дополнительной функциональности (1-14 декабря): доработка приложения после изменения требований заказчиком.

Написание пояснительной записки (15-22 декабря): написание описания проекта.

# **3 ОПИСАНИЕ ПРЕДМЕТА ПРОЕКТИРОВАНИЯ**

Дверь – элемент стеновой конструкции, предназначенный для заполнения дверных проемов и состоящий из дверного блока, крепления дверного блока к проему, монтажных швов, системы уплотнений и облицовки и обеспечивающий при закрытом положении дверного полотна защиту от климатических, шумовых и других воздействий, а также от несанкционированного прохода.[2]

Чертёж показан на рис. 3.1.

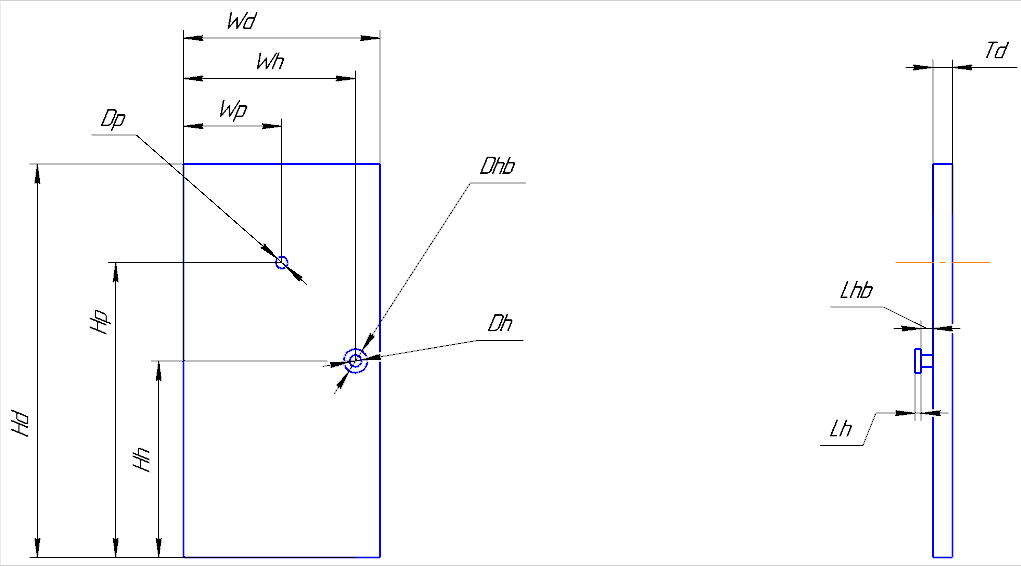


Рисунок 3.1 – Модель двери с размерами

# **4 ВЫБОР ИНСТРУМЕНТОВ И СРЕДСТВ РЕАЛИЗАЦИИ**

Для разработки плагина "Дверь" для КОМПАС-3Д был выбран язык программирования C# с использованием .NET и WinForms.

Также, были выбраны следующие библиотеки:

Для поддержания качества кода: ReSharper, XAML Styler, Editor Guidelines, Spell Checker.

Для тестирования: Fine Code Coverage, NUnit, NUnit3TestAdapter.

# **5 НАЗНАЧЕНИЕ ПЛАГИНА**

Плагин "Двери" разработан с целью автоматизации процесса проектирования дверей в среде КОМПАС-3D. Он предназначен для облегчения и ускорения разработки дверей, предоставляя инженерам и дизайнерам эффективный инструмент для создания и модификации дверных конструкций. Плагин позволяет учитывать различные параметры и требования, обеспечивая точность и высокую производительность в процессе создания продукции.

# **6 ОБЗОР АНАЛОГОВ**

Прямым аналогом плагина по построению дверей является скрипт для 3dsMax – Glass Door Generator v2.0[1]:

Как видно из названия, скрипт строит стеклянные двери с возможностью визуализации сразу в программе. На рисунке 6.1 представлен интерфейс генератора:

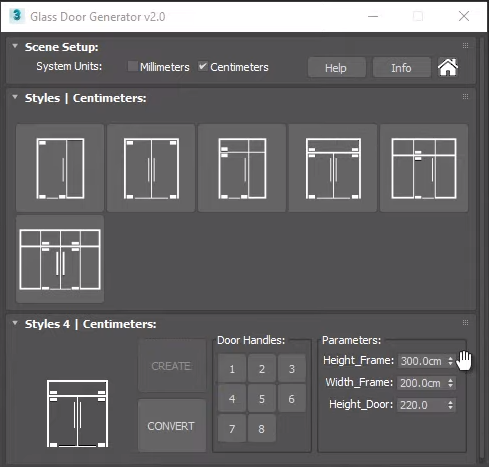


Рисунок 6.1 – Интерфейс плагина по построению дверей

Результатом работы плагина является полноценная дверь (рисунок 6.2), которую можно открывать и закрывать:

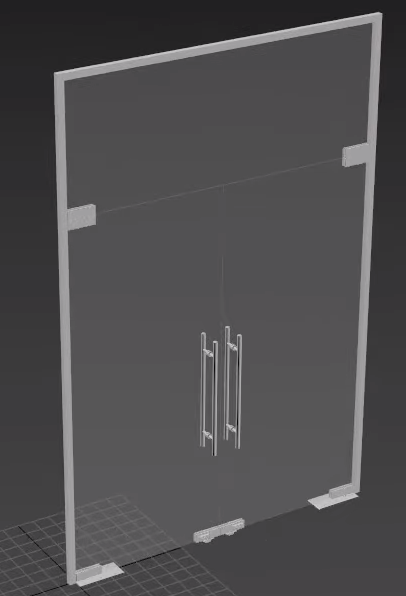


Рисунок 6.2 – Результат работы плагина

# **7 ОПИСАНИЕ РЕАЛИЗАЦИИ**

* 1. **Диаграмма классов после проектирования**

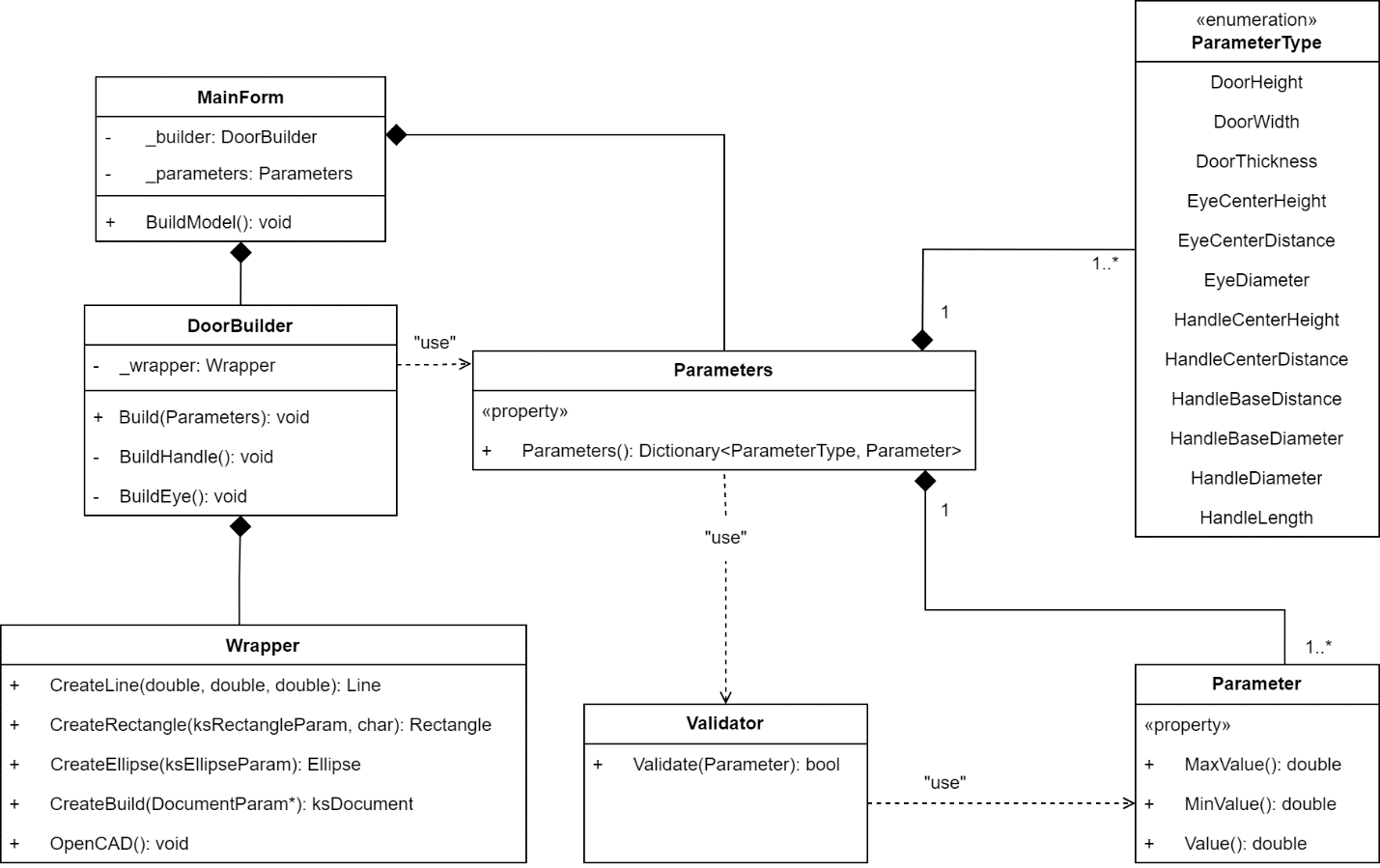


Рисунок 7.1 – Архитектура плагина как отдельного приложения

Основные классы проекта:

* **MainForm** – является главным окном приложения. Хранит в себе параметры (**Parameters**) и объект класса строителя модели (**DoorBuilder**);
* **Parameters** – класс, хранящий в себе параметры модели;
* **DoorBuilder** – класс строитель модели;
* **Wrapper** – класс обёртка API КОМПАС. В нем находятся все нужные методы создания примитивов и документов, которые пригодятся для построения модели.
  1. **Диаграмма классов после реализации**

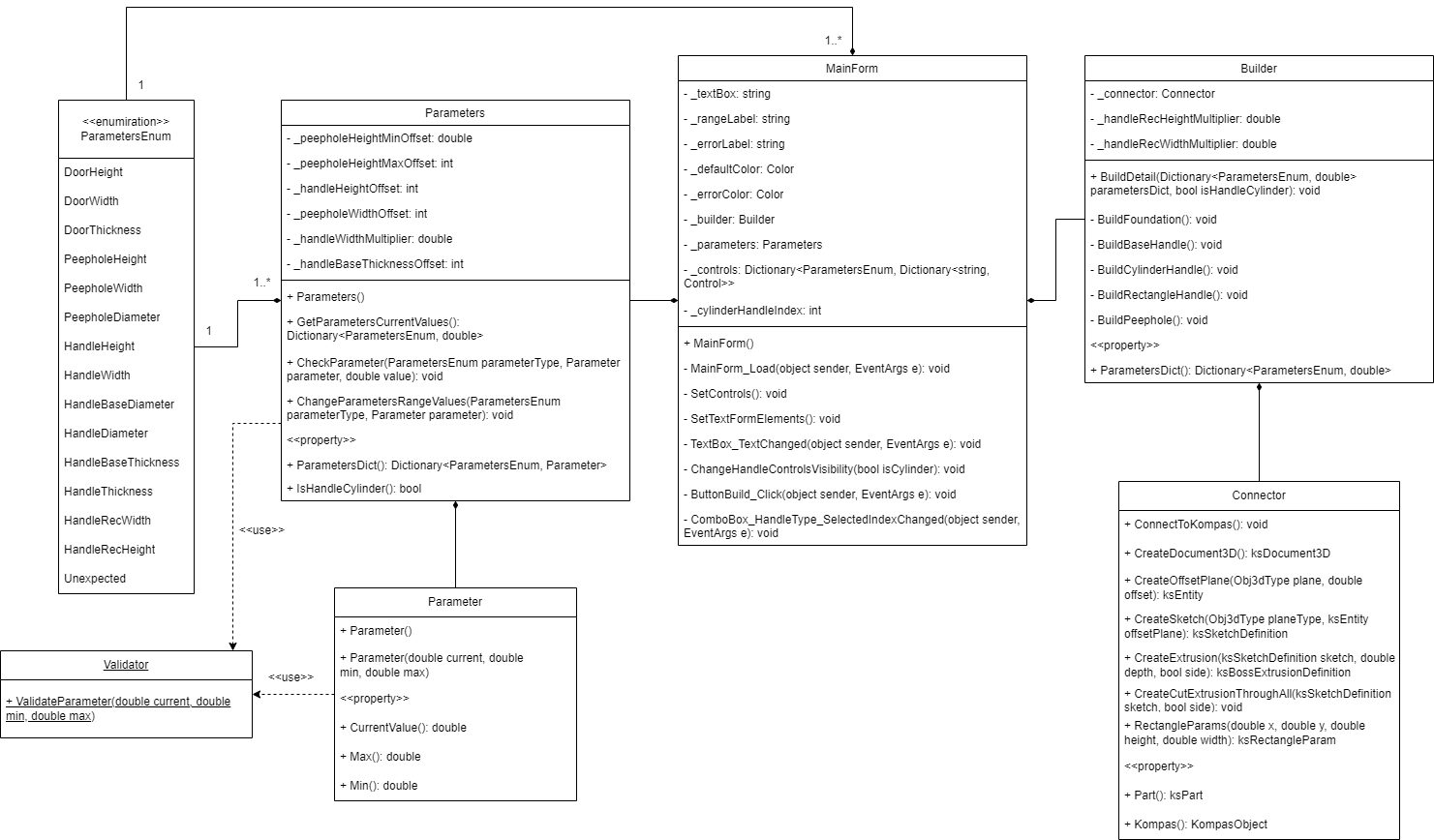


Рисунок 7.2 – Архитектура плагина после реализации.

Таблица 7.1 – Поля класса MainForm.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| \_textBox | string | Строка обозначающая textBox. |
| \_rangeLabel | string | Строка обозначающая rangeLabel. |
| \_errorLabel | string | Строка обозначающая errorLabel. |
| \_defaultColor | Color | Цвет по умолчанию. |
| \_errorColor | Color | Цвет для обозначения ошибок. |
| \_builder | Builder | Экземпляр строителя. |
| \_parameters | Parameters | Экземпляр класса параметров. |
| \_controls | Dictionary<ParametersEnum, Dictionary<string, Control>> | Словарь элементов формы необходимые для отображения параметров. |
| \_cylinderHandleIndex | int | Индекс типа цилиндрообразной ручки. |

Таблица 7.2 – Методы класса MainForm.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Аргументы | Тип данных | Описание |
| MainForm() | - |  | Инициализирует новый экземпляр класса MainForm. |
| MainForm\_Load() | object sender, EventArgs e | void | Обработчик события загрузки формы. |
| SetControls() | - | void | Задает элементы управления формы для каждого параметра. |
| SetTextFormElements() | - | void | Устанавливает текст в контролы на основе текущих параметров. |
| TextBox\_TextChanged() | object sender, EventArgs e | void | Обработчик изменения текста в textBox. |
| ChangeHandleControlsVisibility() | bool isCylinder | void | Меняет видимость контролов типа ручки. |
| ButtonBuild\_Click() | object sender, EventArgs e | void | Обработчик нажатия кнопки "Построить". |
| ComboBox\_HandleType\_SelectedIndexChanged() | object sender, EventArgs e | void | Обработчик комбобокса выбора типа ручки. |

Добавлены методы для работы с элементами формы, обработчики событий текстбоксов и комбобокса, Созданы константы для цветов и обозначения элементов форм.

Таблица 7.3 – Методы класса Parameter.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Аргументы | Тип данных | Описание |
| CurrentValue() | - | double | Получает или задает текущее значение параметра. |
| Max() | - | double | Получает или задает максимальное значение параметра. |
| Min() | - | double | Получает или задает минимальное значение параметра. |
| Parameter() | - |  | Конструктор по умолчанию. |
| Parameter() | double current,double min,double max |  | Конструктор с вводимыми значениями. |

Добавлены конструкторы.

Таблица 7.4 – Поля класса Parameters.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| \_peepholeHeightMinOffset | double | Минимальный отступ глазка от высоты. |
| \_peepholeHeightMaxOffset | int | Максимальный отступ глазка от высоты. |
| \_handleHeightOffset | int | Отступ рукоятки. |
| \_peepholeWidthOffset | int | Отступ глазка по горизонтали. |
| \_handleWidthMultiplier | double | Множитель ширины рукоятки. |
| \_handleBaseThicknessOffset | int | Выступ рукоятки. |

Таблица 7.5 – Методы класса Parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Аргументы | Тип данных | Описание |
| ParametersDict() | - | Dictionary<ParametersEnum, Parameter> | Словарь, тип параметра - параметр. |
| IsHandleCylinder() | - | bool | Является ли тип ручки цилиндром. |
| Parameters() | - |  | Инициализирует новый экземпляр класса Parameters со значениями по умолчанию. |

Продолжение таблицы 7.5 – Методы класса Parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Аргументы | Тип данных | Описание |
| GetParametersCurrentValues() | - | Dictionary<ParametersEnum, double> | Получает словарь текущих значений параметров. |
| CheckParameter() | ParametersEnum parameterType,Parameter parameter,double value | void | Проверяет параметр. |
| ChangeParametersRangeValues() | ParametersEnum parameterType,Parameter parameter | void | Изменяет граничные значения параметров. |

Добавлены методы для проверки и расчета параметров и их граничных значений. Также, добавлены константы для отступов и множителей.

Таблица 7.6 – Элементы перечисления ParametersEnum.

|  |  |
| --- | --- |
| Название | Описание |
| DoorHeight | Высота двери. |
| DoorWidth | Ширина двери. |
| DoorThickness | Толщина двери. |
| PeepholeHeight | Высота глазка. |
| PeepholeWidth | Расстояние от левого края двери до центра глазка. |
| PeepholeDiameter | Диаметр глазка. |
| HandleHeight | Высота центра ручки. |
| HandleWidth | Расстояние от левого края двери до центра ручки. |
| HandleBaseDiameter | Диаметр основания ручки. |
| HandleDiameter | Диаметр ручки. |
| HandleBaseThickness | Длина основания ручки. |
| HandleThickness | Длина ручки. |
| HandleRecWidth | Ширина ручки. |
| HandleRecHeight | Длина ручки. |
| Unexpected | Некорректный параметр. |

Добавлены параметры для дополнительной функциональности и тип некорректного параметра.

Таблица 7.7 – Методы класса Validator.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Аргументы | Тип данных | Описание |
| ValidateParameter() | double current, double min, double max | void | Проверяет параметр на правильность. |

Изменена логика валидатора.

Таблица 7.8 – Поля класса Builder.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| \_connector | Connector | Экземпляр коннектора. |
| \_handleRecHeightMultiplier | double | Множитель заступа длины ручки на основание. |
| \_handleRecWidthMultiplier | double | Множитель заступа ширины ручки на основание. |

Таблица 7.9 – Методы класса Builder.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Аргументы | Тип данных | Описание |
| ParametersDict() | - | Dictionary<ParametersEnum, double> | Словарь текущих значений всех параметров. |
| BuildDetail() | Dictionary<ParametersEnum, double> parametersDict,bool isHandleCylinder | void | Строит деталь в Компасе. |
| BuildFoundation() | - | void | Строит основу двери. |
| BuildBaseHandle() | - | void | Строит основание рукоятки. |
| BuildCylinderHandle() | - | void | Строит рукоятку. |
| BuildRectangleHandle() | - | void | Строит прямоугольную рукоятку. |
| BuildPeephole() | - | void | Строит глазок. |

Добавились поля для множителей отступов. Декомпозирован метод BuildDetail(), добавлено условие выбора типа ручки.

Таблица 7.10 – Методы класса Builder.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Аргументы | Тип данных | Описание |
| Part() | - | ksPart | Компонент исполнения. |
| Kompas() | - | KompasObject | Получает объект Компаса. |
| ConnectToKompas() | - | void | Подключается к активной сессии Компаса. |
| CreateDocument3D() | - | ksDocument3D | Создает 3D-документ Компаса. |
| CreateOffsetPlane() | Obj3dType plane, double offset | ksEntity | Создает смещенную плоскость относительно другой плоскости. |
| CreateSketch() | Obj3dType planeType,ksEntity offsetPlane | ksSketchDefinition | Создает эскиз на заданной плоскости. |
| CreateExtrusion() | ksSketchDefinition sketch,double depth,bool side | ksBossExtrusionDefinition | Создает выдавливание на основе эскиза. |
| СreateCutExtrusionThroughAll() | ksSketchDefinition sketch, bool side | void | Метод, осуществляющий вырезание через все поверхности. |
| RectangleParams() | double x, double y, double height, double width | ksRectangleParam | Метод рисования прямоугольника. |

Перенесены методы построения в Connector, добавлен метод для построения прямоугольника.

1. **ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПОЛЬЗОВАТЕЛЯ**

Интерфейс программы интуитивно понятен. Необходимо ввести параметры двери и тип ручки или оставить их по умолчанию (рис. 8.1 – 8.2).

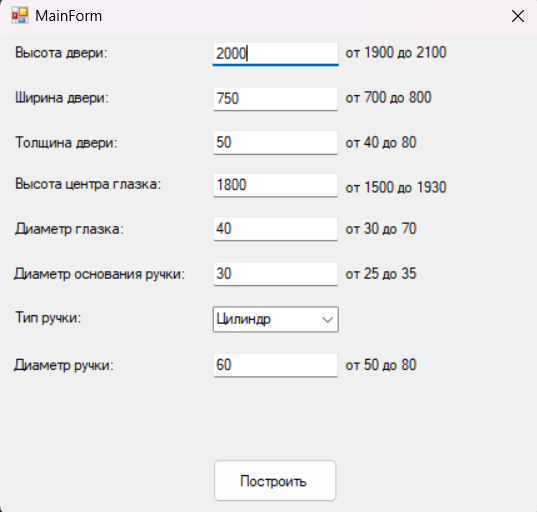


Рисунок 8.1 – UI программы по умолчанию.

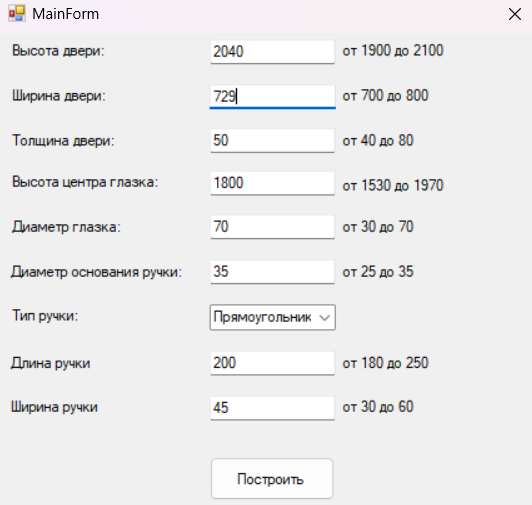


Рисунок 8.2 – UI программы с введенными параметрами.

При возникновении каких-либо ошибок, кнопка “Построить” будет заблокирована (рис. 8.3 – 8.4).

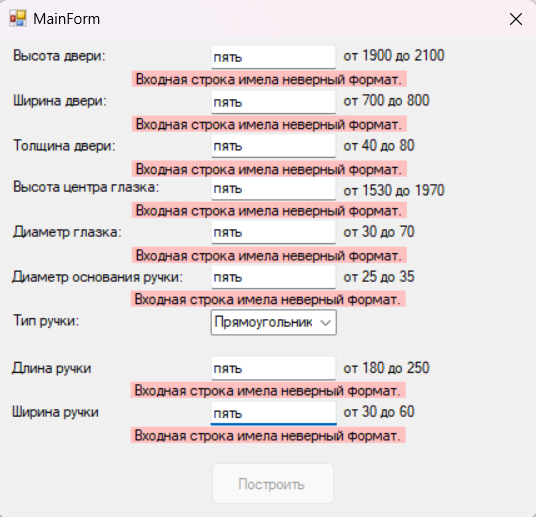


Рисунок 8.3 – UI программы с некорректно введенными параметрами.

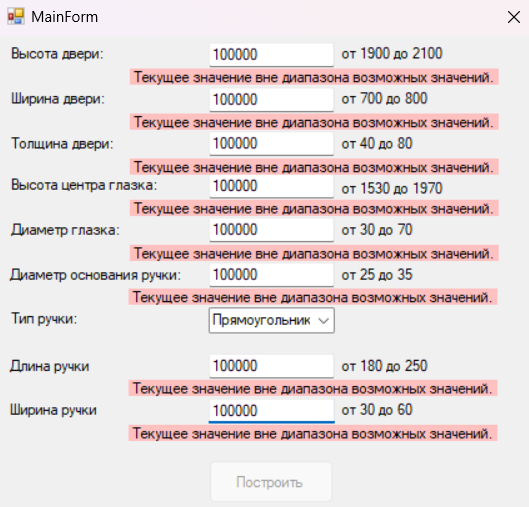


Рисунок 8.4 – UI программы с некорректно параметрами, не попадающими в диапазон.

# **9 ТЕСТИРОВАНИЕ ПЛАГИНА**

## **9.1 Функциональное тестирование**

Стандартные параметры, заданы по умолчанию (рис. 9.1).

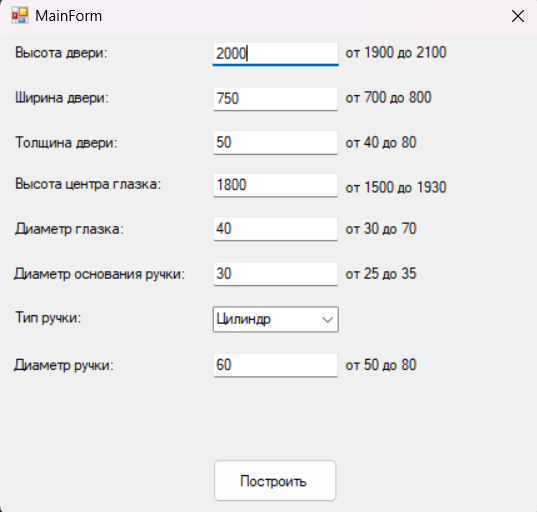


Рисунок 9.1 – Параметры по умолчанию.

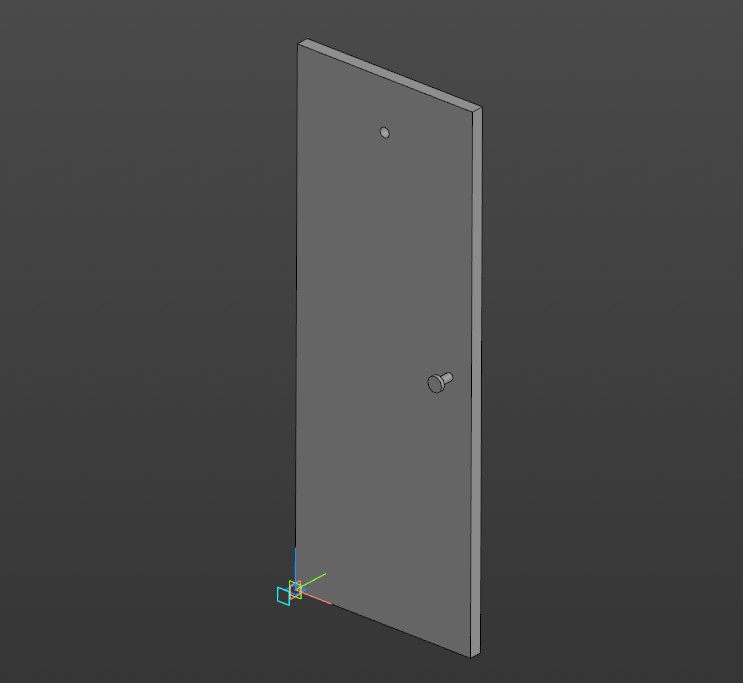


Рисунок 9.2 – Деталь с заданными стандартными параметрами.

Введем максимальные параметры.

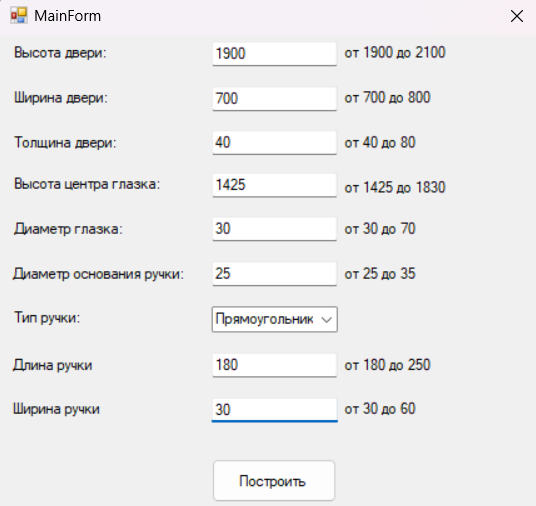


Рисунок 9.3 – Минимальные параметры.

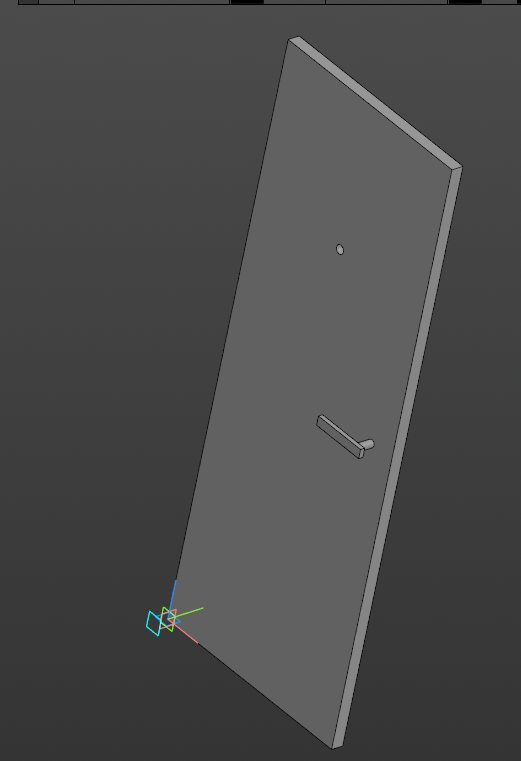


Рисунок 9.4 – Деталь с заданными минимальными параметрами.

Введем максимальные параметры.

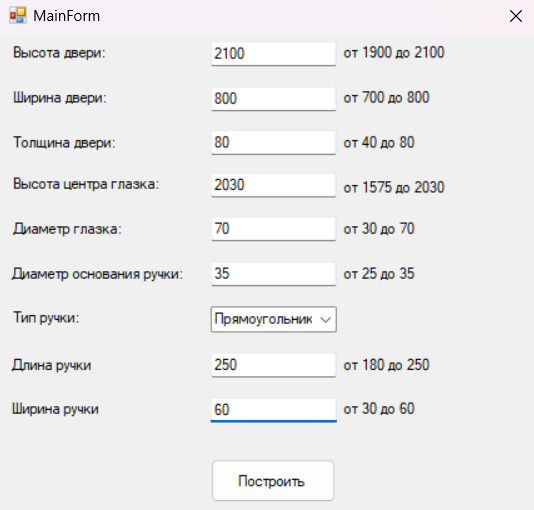


Рисунок 9.5 – Максимальные параметры.

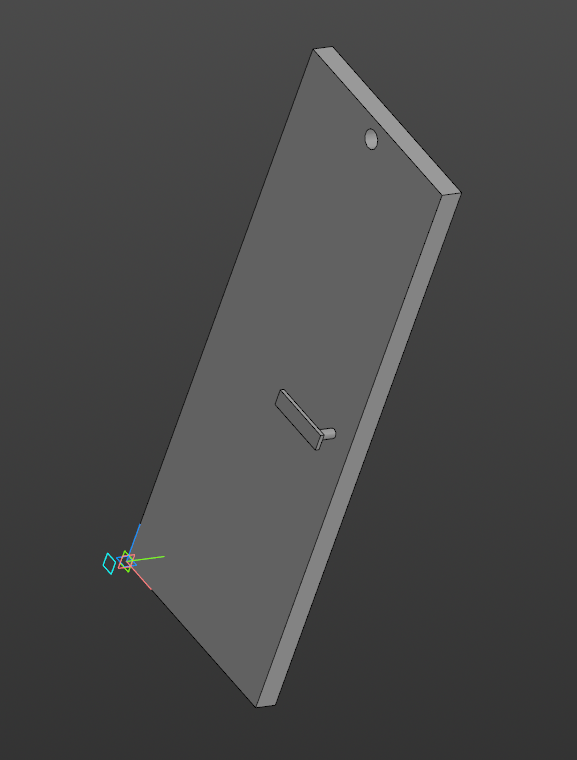


Рисунок 9.6 – Деталь с заданными максимальными параметрами.

## **9.2 Модульное тестирование**

Всего написано 39 тестов.

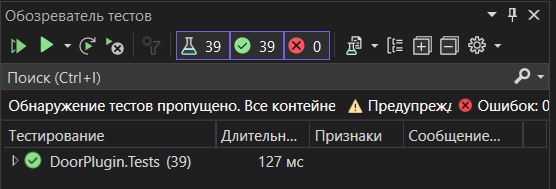


Рисунок 9.7 – Количество тестов.

Model покрыта на 100%.

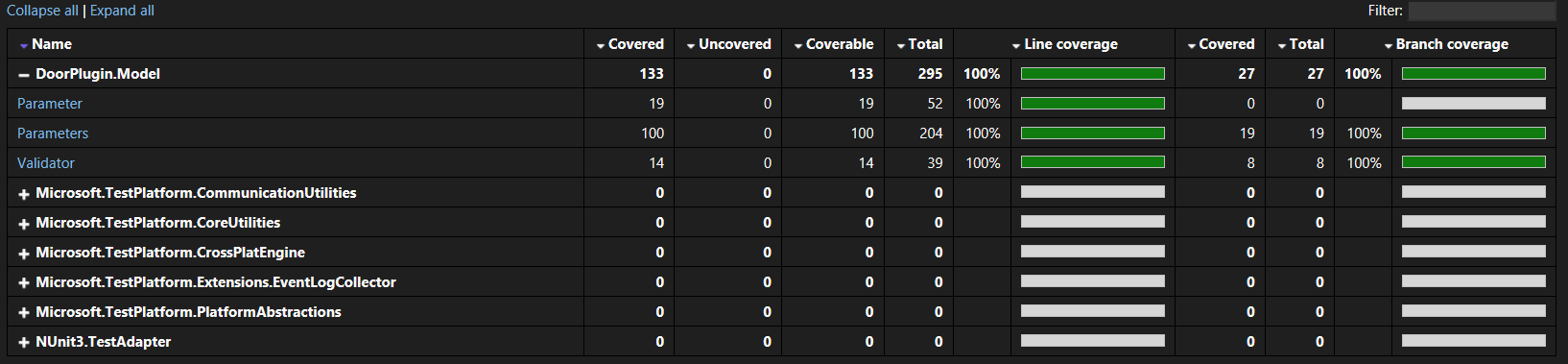


Рисунок 9.8 – Покрытие кода тестами.

Таблицы с описанием тестов.

Таблица 9.1 – Методы класса ParametersTests.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Аргументы | Тип данных | Описание |
| AssertParameter\_Parameter\_UpdateParameter() | ParametersEnum parameterType, double currentValue, double maxValue, double minValue, double value | void | Положительный тест присвоения нового значения параметру. |
| AssertParameter\_WrongParameterValue\_ThrowException() | ParametersEnum parameterType, double currentValue, double maxValue, double minValue, double value | void | Отрицательный тест присвоения нового значения параметру. |
| IsHandleCylinder\_SetGetCorrectValue | bool isHandleCylinder | void | Положительный тест IsHandleCylinder. |
| GetParametersCurrentValues\_CurrentValuesAreEqual() | ParametersEnum[] parametersTypes, double[] currentValues | void | Положительный тест проверки текущих значений параметров. |
| ChangeParametersRangeValues\_Parameter\_UpdateRangeValues() | ParametersEnum parameterType, double currentValue, double maxValue, double minValue, ParametersEnum[] parametersTypes, double[] expectedParameters | void | Положительный тест метода расчетов с параметрами, которые влияют на другие параметры. |
| ChangeParametersRangeValues\_Parameter\_NothingHappens() | ParametersEnum parameterType, double currentValue, double maxValue, double minValue | void | Положительный тест метода расчетов с параметрами, которые не влияют на другие параметры. |
| ChangeParametersRangeValues\_Parameter\_ThrowException() | ParametersEnum parameterType, double currentValue, double maxValue, double minValue | void | Отрицательный тест метода расчетов с некорректными параметрами |

Таблица 9.2 – Методы класса ParameterTests.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Аргументы | Тип данных | Описание |
| Parameter\_Initialization\_SetCorrectly | double currentValue, double maxValue, double minValue | void | Положительный тест конструктора Parameter. |
| Parameter\_SetProperties\_UpdateValues | double currentValue, double maxValue, double minValue | void | Положительный тест задания значений Parameter. |
| Parameter\_Initialization\_IncorrectValues\_ThrowArgumentException | double currentValue, double maxValue, double minValue | void | Отрицательный тест задания значений Parameter. |

Таблица 9.3 – Методы класса ValidatorTests.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Аргументы | Тип данных | Описание |
| ValidateRange\_ShouldNotThrowException | double value, double minValue, double maxValue | void | Положительный тест валидатора. |
| ValidateRange\_ShouldThrowArgumentException | double value, double minValue, double maxValue | void | Отрицательный тест валидатора. |

### 9.3 Нагрузочное тестирование

Тестирование проводилось на системе со следующими характеристиками:

* Оперативная память – 8,00 ГБ;
* Процессор – HexaCore Intel Xeon E5-2620 v3, 3200 MHz
* Компьютер с ACPI на базе x64
* Видеокарта – NVIDIA GeForce GTX 1660 6 Гб.

Проведем нагрузочное тестирование с обычными параметрами. Компас завис на 59 построении.

Рисунок 9.9 – График зависимости памяти ОЗУ (ГБ) от построения модели.

Из графика 1 видно, что память изменяется линейно, но с некоторыми скачками вниз. Это связано с тем, что для регенерации памяти происходит остановка обмена данными, что снижает нагрузку на оперативную память. В конце график не сильно идет вверх, это связано с тем, что, когда на оперативную память идет сильная нагрузка, ОС начинает использовать файл подкачки для разгрузки оперативной памяти.

Рисунок 9.10 – График зависимость времени от построения модели.

Проанализировав график 2, можем сделать вывод, что с увеличением количества построенных моделей время, затрачиваемое на построение моделей, не увеличивается. Это может быть связано с тем, что модель двери является достаточно простой, из-за чего даже при значительных количествах моделей их нагрузки на оперативную память и процессор недостаточно для замедления построения моделей.

# **10 ЗАКЛЮЧЕНИЕ**

Реализован плагин для построения дверей для САПР “КОМПАС-3Д”. Создано приложение, которое позволит инженерам и дизайнерам легко создавать и модифицировать двери и управлять параметрами.

Приложение реализовано на C# (.NET, WinForms), использует API КОМПАС-3Д. Также, во время разработки были использованы сторонние библиотеки для C#: для поддержания качества кода (ReSharper, XAML Styler, Editor Guidelines, Spell Checker); для тестирования (Fine Code Coverage, NUnit, NUnit3TestAdapter).

Найден прямой аналог для 3dsMax – Glass Door Generator v2.0 [1], который позволяет генерировать стеклянные двери с различными пресетами.

Model программы протестирована на 100%. Также, проводилось нагрузочное тестирование. Оно показало, что память заканчивается на 59 построении детали на системе ниже среднего (по современным меркам), необходимо больше оперативной памяти.

В ходе реализации проекта была изучена документация API [3] Компаса. Также, разобран функционал сторонних библиотек для C# таких как ReSharper, Fine Code Coverage и других.

# **11 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Glass Door Generator v2.0 | 3dsMax [Электронный ресурс]. URL: <https://archviztools.gumroad.com/l/Glass_Door_Generator>;
2. Дверь [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/Дверь>;
3. Документация к API KOMPAS-3D [Электронный ресурс] // ASCON [Официальный сайт] URL: <https://help.ascon.ru/KOMPAS_SDK/22/ru-RU/index.html> (дата обращения: 25.10.2023).