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

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
 по дисциплине «Основы разработки САПР»

Разработка плагина «Этажерка для обуви» для САПР «Компас-3D» v19.

к.т.н., Доцент кафедры КСУП

\_\_\_\_\_\_\_\_\_\_ Калентьев А.А.

«21» апреля 2021 г.

Студент гр. 587-3

\_\_\_\_\_\_\_\_\_\_\_ Громов Н.В.

«21» апреля 2021 г.

Томск

2021

**Оглавление**

[**1 Описание САПР** 3](#_Toc69875427)

[**2 Описание API САПР** 5](#_Toc69875428)

[**3 Обзор аналогов плагина** 8](#_Toc69875429)

[**4 Описание объекта проектирования** 9](#_Toc69875430)

[**5 Диаграмма классов** 12](#_Toc69875431)

[**6 Макет пользовательского интерфейса** 15](#_Toc69875432)

[**7 Тестирование плагина** 17](#_Toc69875433)

[**7.1 Модульное тестирование** 17](#_Toc69875434)

[**7.2 Функциональное тестирование** 18](#_Toc69875435)

[**7.3 Нагрузочное тестирование** 23](#_Toc69875436)

[**8 Заключение** 26](#_Toc69875437)

[**9 Список литературы** 27](#_Toc69875438)

[**Приложение А** 28](#_Toc69875439)

**1 Описание САПР**

Программное обеспечение САПР для механического проектирования использует векторную графику в целях изображения объектов традиционного черчения или может также создавать растровую графику, отображающую общий вид проектируемых объектов. Тем не менее, это включает в себя больше, чем просто шаблонные формы. Как и при ручном создании технических и инженерных чертежей, выходные данные САПР должны передавать информацию, такую ​​как характеристики используемых материалов, процессы, размеры и допуски, в соответствии с соглашениями для конкретных приложений.

CAD может использоваться для проектирования кривых и фигур в двумерном (2D) пространстве; или кривых, поверхностей и твердых тел в трехмерном (3D) пространстве.

САПР является важным звеном в промышленном конструировании, широко используемым во многих отраслях, в том числе в автомобильной, судостроительной и аэрокосмической промышленности, промышленном и архитектурном проектировании, протезировании и многих других. САПР также широко используется в создании компьютерной анимации для спецэффектов в фильмах, рекламных и технических материалах, часто называемых цифровым контентом. Современное повсеместное распространение компьютеров означает, что даже флаконы для духов и диспенсеры для шампуней сегодня разрабатываются с использованием информационных технологий, невиданных инженерами 1960-х годов. Из-за своей огромной экономической важности, САПР стал основной движущей силой исследований в области вычислительной геометрии, компьютерной графики (как аппаратной, так и программной) и дискретной дифференциальной геометрии.[1]

КОМПАС-3D – это российская система трехмерного проектирования, ставшая стандартом для тысяч предприятий и десятков тысяч профессиональных пользователей. Система КОМПАС-3D широко используется для проектирования изделий основного и вспомогательного производств в таких отраслях промышленности, как машиностроение (транспортное, сельскохозяйственное, энергетическое, нефтегазовое, химическое и т.д.), приборостроение, авиастроение, судостроение, станкостроение, вагоностроение, металлургия, промышленно-гражданское строительство, товары народного потребления и т. д.

Актуальная версия продукта: КОМПАС-3D v19, на которой разработан плагин.[2]

**2 Описание API САПР**

Сегодня встречаются задачи, решение которых не реализованы в CAD-системах. Чаще всего это очень узкоспециализированные задачи, которые встречаются на каком-то конкретном предприятии или подотрасли. Для решения подобных задач возможно использовать КОМПАС-3D как платформу и на базе него создать свое приложение, которое позволит автоматизировать решение таких задач. Для создания таких приложений в КОМПАС-3D есть открытый API.[2]

При выполнение проекта будут использоваться две составляющих API ̶ Kompas6API5 и Kompas6Constants3D. Основные используемые интерфейсы: KompasObject, ksDocument3D, ksPart, ksEntity.

Таблица 2.1 ̶ Методы интерфейса KompasObject.

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| Document2D() |  | указатель на интерфейс графического документа ksDocument2D. | Позволяет получить указатель на интерфейс графического документа. |
| Document3D() |  | указатель на интерфейс документа трехмерной моде­ли ksDocument3D. | Позволяет получить указатель на интерфейс документа трехмерной модели. |
| ActiveDocument3D() |  | указатель на интерфейс документа трехмерной модели ksDocument3D. | Позволяет получить указатель на интерфейс текущего документа трехмерной модели |
| Visible |  |  | Позволяет получить и установить свойство видимости приложения КОМПАС-3D. |

Таблица 2.2 ̶ Методы интерфейса ksDocument3D.

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| Create(bool invisible, bool typeDoc) | |  |  | | --- | --- | | invisible | - признак режима редактирования доку­мента  (TRUE - невидимый режим,  FALSE -видимый режим), | | typeDoc | - тип документа  (TRUE - деталь,  FALSE - сборка). | | |  |  | | --- | --- | | TRUE | - в случае успешного за­вершения | | Позволяет создать документ-модель (деталь или сборку) |
| GetPart(int type) | |  |  | | --- | --- | | type | - тип компо­нента из пере­числения Типы компонентов. | | Указатель на интерфейс компонента ksPart или IPart. | Позволяет получить указатель на интерфейс компонента в соответствии с заданным типом |

Таблица 2.3 ̶ Методы интерфейса ksPart.

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| GetDefaultEntity(short objType) | |  |  | | --- | --- | | objType | - тип объекта. | | Указатель на интерфейс ksEntity или IEntity. | Позволяет получить указатель на интерфейс объекта, создаваемого системой по умолчанию |
| NewEntity(short objType) | |  |  | | --- | --- | | objType | - тип объекта. | | Указатель на интерфейс ksEntity или IEntity. | Позволяет создать новый интерфейс объекта и получить указатель на него |

Таблица 2.4 ̶ Методы интерфейса ksEntity.

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| GetDefinition() |  | Указатель на интерфейс IDispatch. | Позволяет получить указатель на интерфейс параметров объектов и элементов |
| Create() |  | |  |  | | --- | --- | | TRUE | - в случае успешного завершения | | Позволяет создать объект в модели |

**3 Обзор аналогов плагина**

1. Плагин Object3D для SketchUP.

SketchUP ̶ это ПО для работы с трехмерной графикой. Основной идеей SketchUp является простота интерфейса, что позволяет освоить работу с программой даже непрофессиональному пользователю. Программа реализует концепцию прямого моделирования геометрии, в рамках которой пользователь сначала стоит плоский контур из имеющихся примитивов, затем вытягивает его с целью создания или вычитания объема, после чего придает модели нужную форму посредством перетаскивания ее элементов (вершин, ребер и граней) с помощью указателя мыши.[4] Для данного ПО был создан плагин Shasoft Object3D на языке Ruby, который упрощает создание 3D-моделей мебели и позволяет с без глубоких знаний языки Ruby автоматизировать процесс создания моделей. Главным недостатком является отсутствие пользовательского интерфейса и необходимость наличия умений работы с языком Ruby.[5]

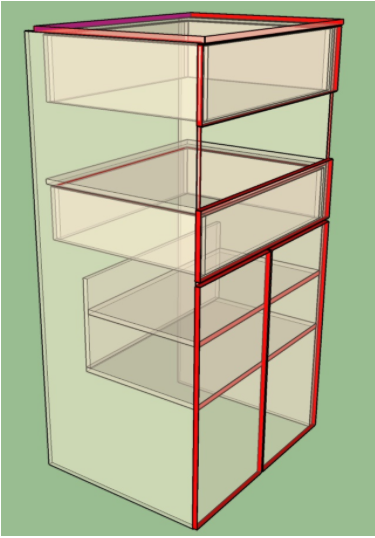


Рисунок 3.1 ̶ Пример работы плагина Object3D.

**4 Описание объекта проектирования**

Объект проектирования – этажерка для обуви.

Этажерка (или полка) для обуви ̶ это эстетичная замена привычному шкафу или тумбе. Предмет может использоваться не только дома, но и в офисах, позволяя избежать чрезмерной грязи на полу помещения в дождливую погоду. Каждая полка изготовлена из прочного пластика, который легко очищается от грязных следов обыкновенной тряпкой. Выдерживает несколько пар одновременно. Материал легко отмывается водой и чистящими средствами.[3]



Рисунок 4.1 ̶ 3D-модель этажерки для обуви

Все параметры объекта представлены на рисунке 4.2:

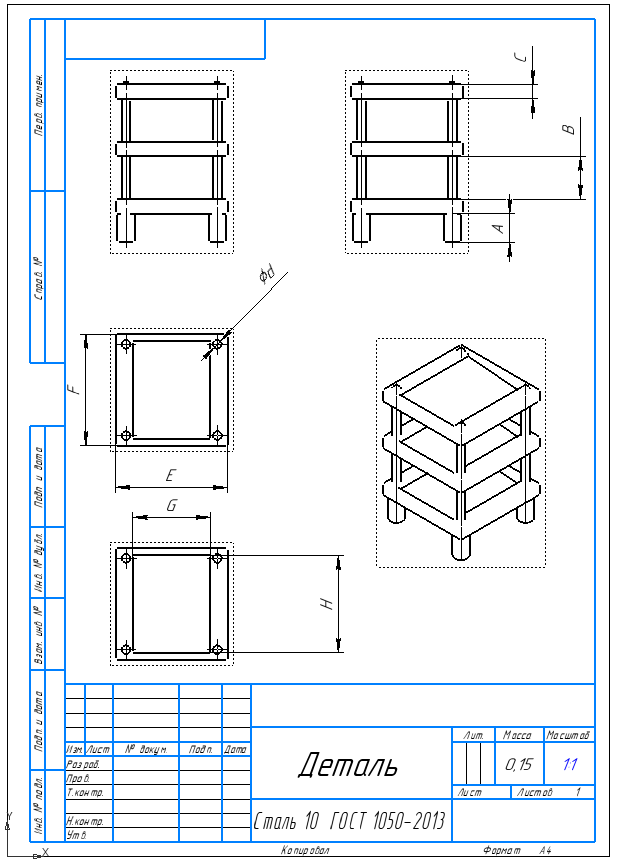


Рисунок 4.2 ̶ Чертеж этажерки

**Вводимые параметры этажерки**

1. Высота ножек этажерки(A). От 40мм до 70мм.

2. Высота креплений полок этажерки(B). От 160мм до 180мм.

3. Длина полки этажерки(E). От 420мм до 480мм.

4. Ширина полки этажерки(F). От 190мм до 220мм.

5. Высота (Толщина) полок этажерки(C). От 20мм до 40мм.

**Зависимые параметры этажерки**

1. Длина отделения для обуви этажерки(G).

Зависит от длины этажерки(E). G=0,7\*E.

2. Ширина отделения для обуви этажерки(H).

Зависит от ширины этажерки(F). H=0,85\*F.

**Константы построения этажерки**

1. Радиус основания ножек этажерки(D). D=20мм.

2. Радиус креплений полок этажерки(d). d=10мм.

3. Уклон отделения для обуви этажерки(N). N= 5°.

4. Общий радиус скруглений(r). r = 0.5 мм.

5. Отступ от края для креплений(J). J= 21.5 мм.

6. Глубина выдавливания отдела для обуви(K). K= 10 мм.

7. Высота верхнего крепления(L). L= 10мм

**5 Диаграмма классов**

Диаграмма классов является статической диаграммой. Он представляет статический вид приложения. Диаграмма описывает атрибуты и операции класса, а также ограничения, налагаемые на систему. Диаграммы классов широко используются при моделировании систем, ориентированных на объекты, потому что они являются единственными диаграммами UML, которые могут быть отображены непосредственно с помощью объектно-ориентированных языков.[6]

На рисунке 5.1 представлена диаграмма классов.

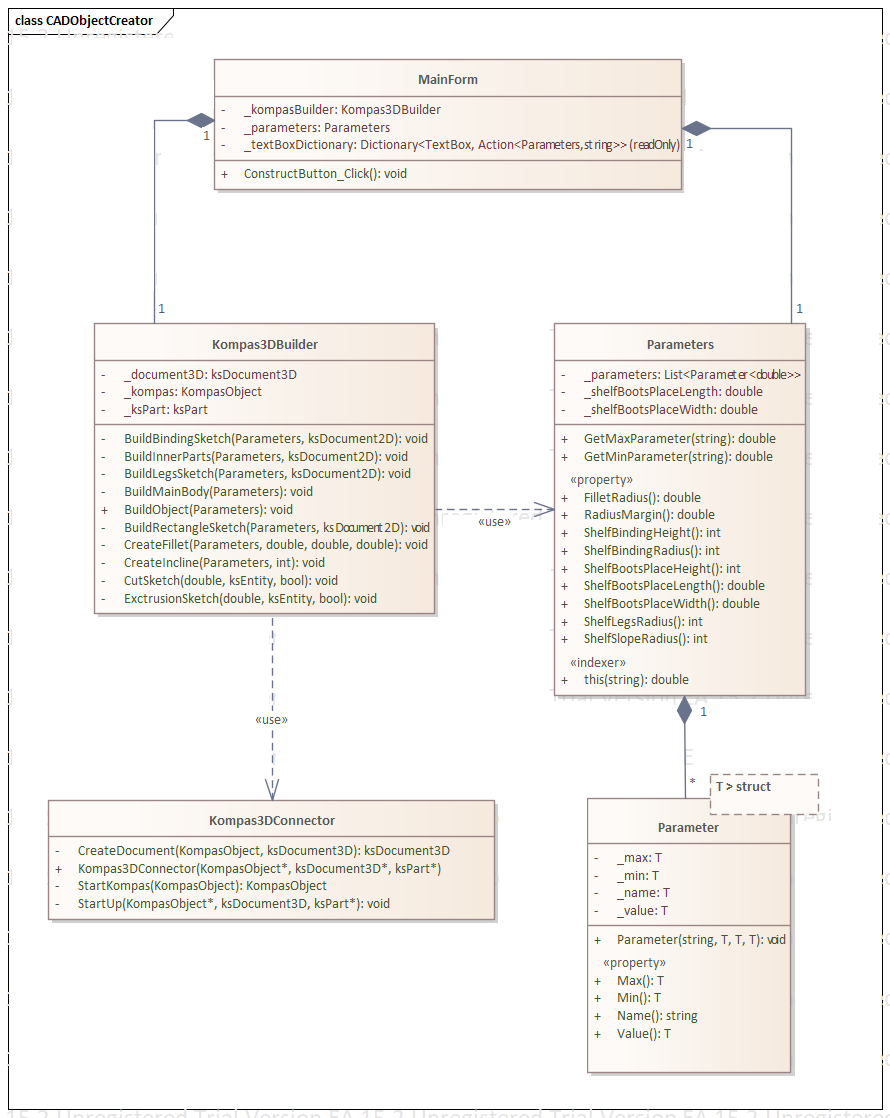
****

Рисунок 5.1 ̶ Диаграмма классов

В процессе разработки плагина были созданы следующие классы:

1. Parameter – базовый класс для задачи переменных параметров.

2. Parameters – класс, обеспечивающий работу со всеми параметрами

3. Kompas3DConnector – класс для работы с API Компас3D. Отвечает за открытие программы и создание документа детали.

4. Kompas3DBuilder – класс для работы с API Компас3D. Отвечает за построение эскизов, выдавливания, закругления, уклоны и другие построения этажерки.

5. MainForm – класс для отображения рабочего окна, с помощью которого пользователь будет взаимодействовать с программой.

На рисунке 5.2 изображена диаграмма классов приложения после завершения разработки дополнительного функционала и рефактронига.

После добавления новой функциональности и рефакторинга в структуре приложения произошли следующие изменения:

1. Было добавлено перечисление ParametersName для более быстрого обращения к существующим параметрам. Данный способ обращения также избавляет от многих ошибок связанных с обращением через индексатор.

2. В Kompas3DBuilder были добавлены методы BuildHookModel, BuildHookSketchLeft и BuildHookSketchRight для выполнения построения крючков с левой и правой стороны этажерки.

3. В Kompas3DConnector был проведен рефакторинг с целью избавления от ref и out в параметрах методов. Теперь есть методы которые возвращают необходимые для работы приложения значения.

4. В Kompas3DBuilder были добавлены многие методы, включающие в себя часто дублируемый код.

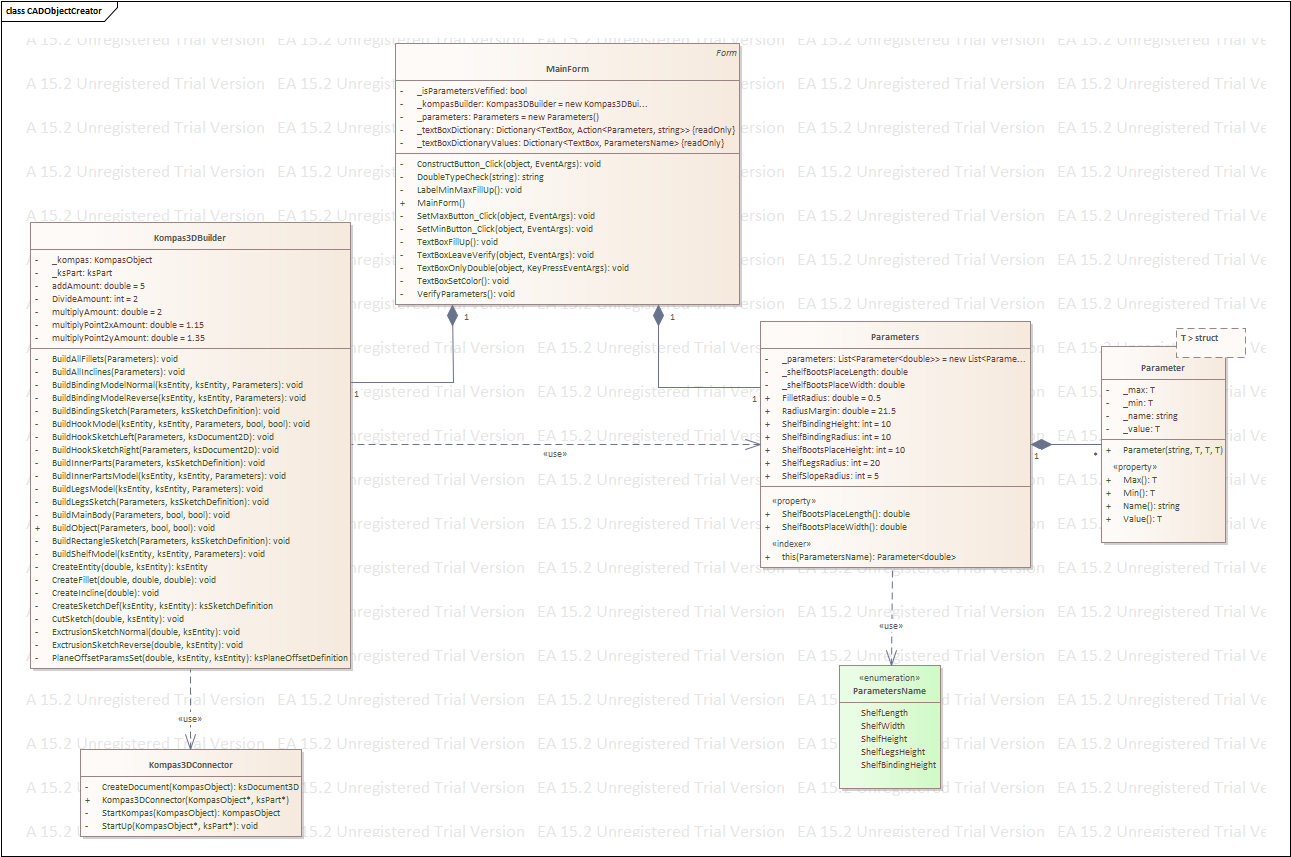


Рисунок 5.2 ̶ Диаграмма классов после добавления функциональности

**6 Макет пользовательского интерфейса**

Для ввода параметров был создан макет пользовательского интерфейса, который позволяет пользователю задать все необходимые параметры для построения этажерки. На рисунке 6.1 показан макет пользовательского интерфейса.

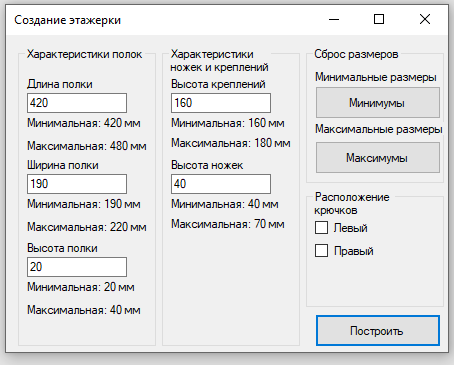


Рисунок 6.1 ̶ Макет пользовательского интерфейса.

Для реализации дополнительного функционала в макет была добавлена новая графа: Расположение крючков. С ее помощью можно выбрать какой из крючков необходимо построить на этажерке.

При вводе неправильных параметров поле с поменяет цвет на красный. Также будет выведено сообщения о превышенной границе ввода. Результат неправильного ввода изображен на рисунках 6.2 и 6.3.

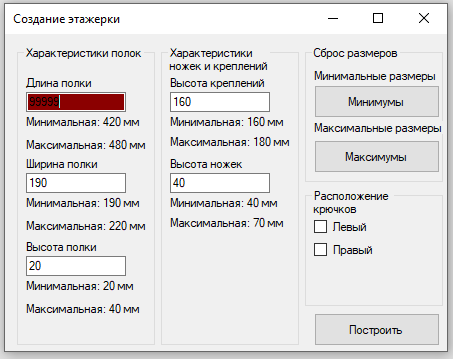


Рисунок 6.2 ̶ Ввод неправильных данных в поле формы

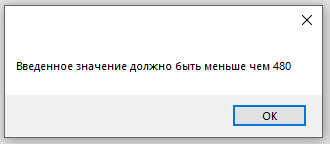


Рисунок 6.3 ̶ Вывод сообщения о неправильном вводе данных

# **7 Тестирование плагина**

## **7.1 Модульное тестирование**

Модульное тестирование ̶ это тестирование отдельного элемента изолированно от остальной системы. Такое тестирование предназначено для проверки правильности работы отдельно взятого класса. Чтобы исключить из результатов тестирования влияние потенциальных ошибок других классов, тестируемый класс должен быть максимально изолирован, т. е. не использовать объекты и методы других классов. Данное требование в итоге позволяет иначе взглянуть на взаимодействие классов и выполнить рефакторинг на уменьшение связности классов.[7]

Тестирование проводилось с помощью фреймворка модульного тестирования NUnit 3.13.1. Тестовые сценарии приведены в приложении А. На рисунке 7.1 показаны результаты тестов, а на рисунке 7.2 показана степень покрытия кода тестами. При проведении тестирования использовались следующие данные класса parameter: name:"Parameter", min:100, max:250, value:160.

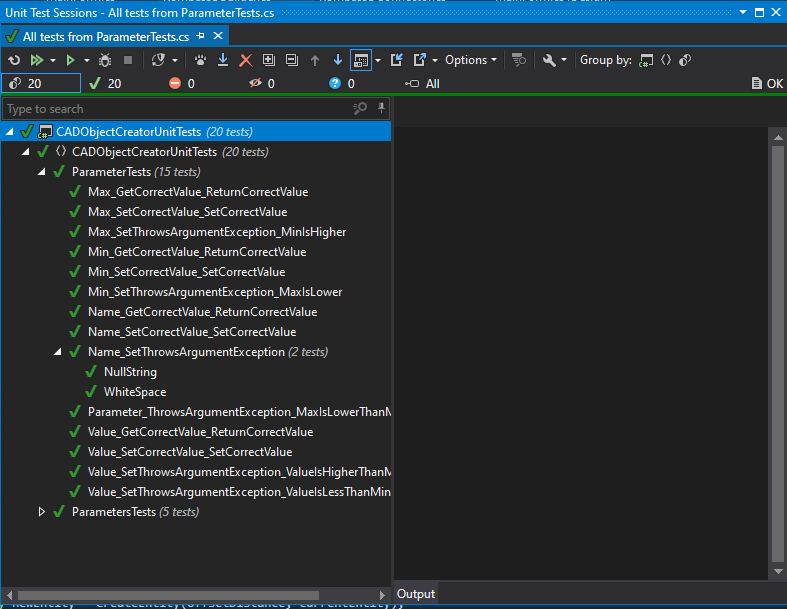


Рисунок 7.1 ̶ Результаты модульного тестирования

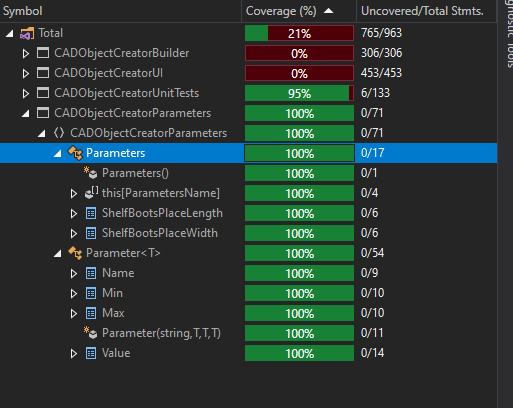


Рисунок 7.2 ̶ Результат проверки покрытия кода тестами

## **7.2 Функциональное тестирование**

Функциональное тестирование – это тестирование функциональности и поведения программы на соответствие требованиям функциональной спецификации. Функциональная спецификация определяет, что именно делает ПО, какие задачи оно решает. Для проведения функционального тестирования необходимо проверить правильность построения детали при минимальных и максимальных значениях входных параметров.

Минимальные входные параметры:

1. Высота ножек этажерки = 40мм.

2. Высота креплений полок этажерки = 160мм.

3. Длина полки этажерки = 420мм.

4. Ширина полки этажерки = 190мм.

5. Высота (Толщина) полок этажерки = 20мм.

Максимальные входные параметры:

1. Высота ножек этажерки = 70мм.

2. Высота креплений полок этажерки = 180мм.

3. Длина полки этажерки = 480мм.

4. Ширина полки этажерки = 220мм.

5. Высота (Толщина) полок этажерки =40мм.

Результаты построения при минимальных параметрах показаны на рисунке 7.3 а при максимальных параметрах на рисунке 7.4.

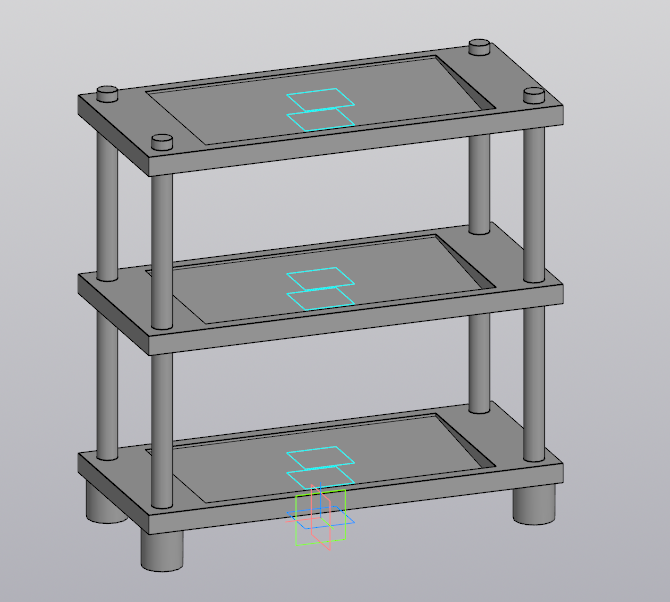


Рисунок 7.3 ̶ Построение этажерки при минимальных входных параметрах

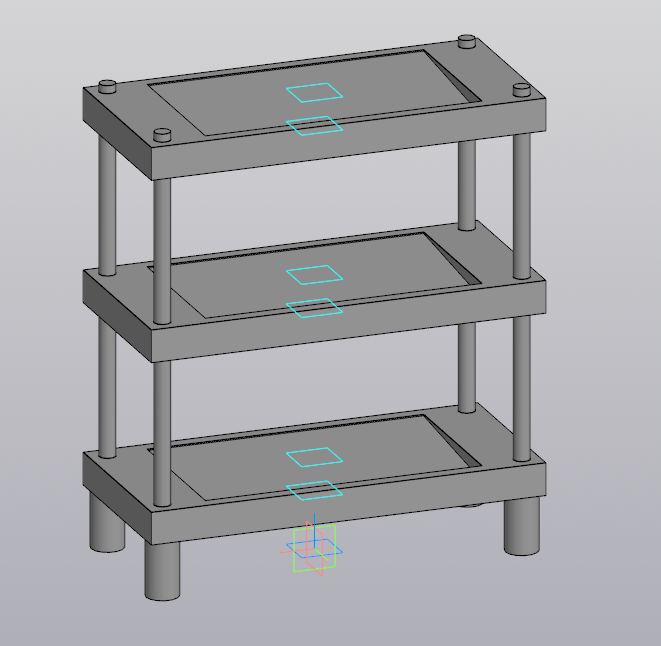


Рисунок 7.4 ̶ Построение этажерки при максимальных входных параметрах

На рисунке 7.5 показано построение этажерки с одним крючком слева, на рисунке 7.6 показано построение этажерки с одним крючком справа и на рисунке 7.7 показано построение этажерки с двумя крючками.

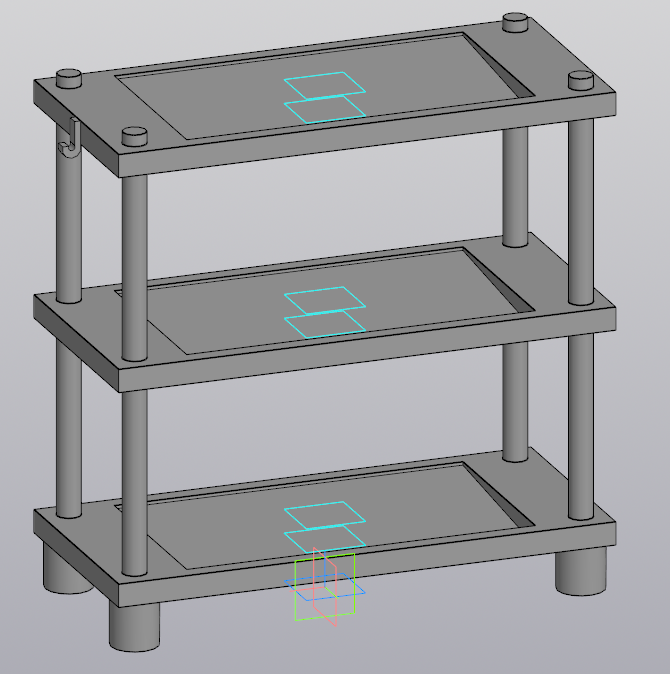


Рисунок 7.5 ̶ Построение этажерки с крючком слева

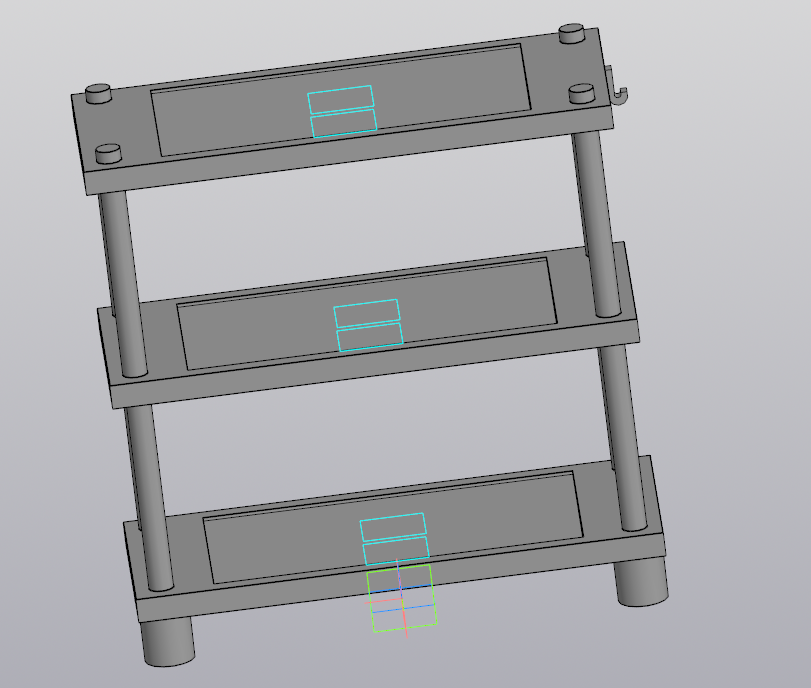


Рисунок 7.5 ̶ Построение этажерки с крючком справа

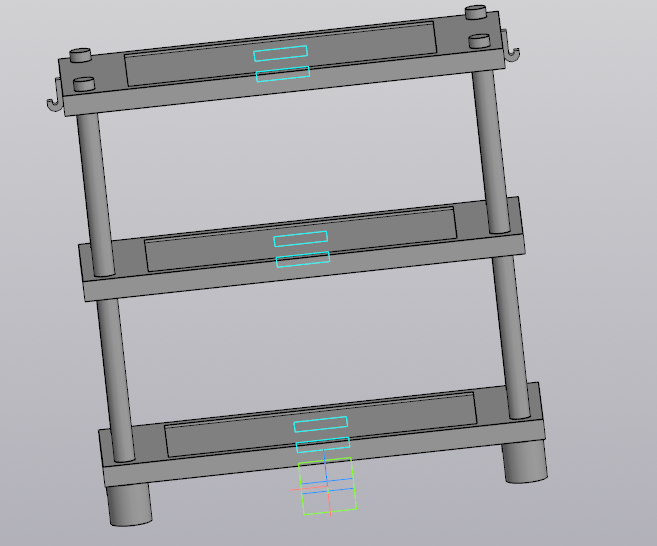


Рисунок 7.5 ̶ Построение этажерки с двумя крючками

## **7.3 Нагрузочное тестирование**

Нагрузочное тестирование  ̶ это автоматизированные испытания информационной системы (или приложения), имитирующие различные нагрузочные модели, с целью комплексной оценки производительности, проверки качественной и бесперебойной работы системы, а также проверки соответствия требованиям, предъявляемым к конкретному объекту тестирования.[8]

Тестирование проводилось на компьютере с следующими техническими параметрами:

Процессор: AMD Ryzen 5 3600 6-Core Processor 3.60 GHz

Оперативная память: 32Gb DDR4 DIMM 3000MHz

Видеоадаптер: NVIDIA GeForce GTX 1660

Операционная система: Windows 10 Pro 64-bit

На рисунке 7.6 представлено количество ОЗУ потребляемое при построение 200 деталей этажерки а на рисунке 7.7 затраченное время на построение этих деталей.

Рисунок 7.6 ̶ Затраченное ОЗУ на построение 200 деталей

Рисунок 7.7 ̶ Затраченное время(в секундах) на построение 200 деталей

На построение 200 деталей примерно было затрачено примерно 16 минут. Время построения не превышало 10 секунд и не занимало более чем 5500 ОЗУ единовременно. При достижении 100 построенных деталей началось увеличение затрачиваемого времени и снижение потребляемой памяти. Скорее всего это связано с очисткой памяти. В общем рост затрат ОЗУ линейный с небольшим скачком в районе 30 построенных деталей. В среднем до 100 деталей построение каждой занимало 3 секунды. Большое количество занимаемой памяти может быть связано с большим количеством выполняемых операций в момент построения скруглений и уклонов на детали.

# **8 Заключение**

В ходе курса ОРСАПР был изучен API Компас3D, основные этапы проектирования и составления проекта системы плагина, изучена предметная область объекта построения. Были составлены такие документы как техническое задание, проект системы и пояснительная записка. Были составлены UML диаграммы классов.

Результатом работы является плагин для САПР «КОМПАС-3D». Который выполняет построение этажерки для обуви с различными переменными параметрами.

**9 Список литературы**

1. САПР ̶ Википедия.[Электронный ресурс]. ̶ Режим доступа: <https://ru.wikipedia.org/wiki/Система_автоматизированного_проектирования> . (Дата обращения: 09.03.2021)

2. Официальный сайт Компас-3D.[Электронный ресурс]. ̶ Режим доступа: <https://kompas.ru/> .(Дата обращения: 09.03.2021)

3. Леруа-Мерлен. Этажерка для обуви.[Электронный ресурс.]. ̶ Режим доступа: <https://leroymerlin.ru/product/etazherka-dlya-obuvi-s-bokovoy-polkoy-korichnevyy-11656857/> .(Дата обращения: 09.03.2021)

4. Программа SketchUP.[Электронный ресурс]. ̶ Режим досутпа: <https://www.sketchup.com/ru> .(Дата обращения: 09.03.2021)

5. Плагин Object3D.[Электронный ресурс.]. ̶ Режим доступа: <https://habr.com/ru/post/442190/> . (Дата обращения: 09.03.2021)

6. UML- Диаграмма классов.[Электронный ресурс.]. ̶ Режим доступа: <https://ru.it-brain.online/tutorial/uml/uml_class_diagram/> . (Дата обращения: 19.03.2021)

7. Новые технологии в программировании : учебное пособие / А.А.Калентьев, Д.В.Гарайс, А.Е.Горяинов – Томск : Эль Контент, 2014.—176 с.

8. Нагрузочное тестирование.[Электронный ресурс]. ̶ Режим доступа: https://www.artwell.ru/services/nagruzochnoe-testirovanie/ . (Дата обращения: 20.04.2021)

# **Приложение А**

Таблица А.1 ̶ Тестовые сценарии

|  |  |  |
| --- | --- | --- |
| Метод | Входные параметры | Описание тестового случая |
| Name\_GetCorrectValue\_  ReturnCorrectValue() | “Новое Имя” | Позитивный тест гетера свойства Name класса Parameter |
| Name\_SetCorrectValue\_  SetCorrectValue() | “Новое имя” | Позитивный тест сетера свойства Name класса Parameter |
| Name\_SetThrowsArgumentException(string input) | “” | Присвоение свойством пустой строки с белым пробелом |
| null | Присвоение свойством null строки |
| Min\_GetCorrectValue\_  ReturnCorrectValue() | 50 | Позитивный тест гетера свойства Min класса Parameter |

Продолжение таблица А.1

|  |  |  |
| --- | --- | --- |
| Метод | Входные параметры | Описание тестового случая |
| Min\_SetCorrectValue\_  SetCorrectValue() | 15.33 | Позитивный тест сетера свойства Min класса Parameter |
| Min\_SetThrowsArgumentException\_  MaxIsLower() | 999.99 | Присвоению свойству Min значение большее максимального |
| Max\_GetCorrectValue\_  ReturnCorrectValue() | 300 | Позитивный тест гетера свойства Max класса Parameter |
| Max\_SetCorrectValue\_  SetCorrectValue() | 350.99 | Позитивный тест сетера свойства Min класса Parameter |
| Max\_SetThrowsArgumentException\_  MinIsHigher() | 10.5 | Присвоению свойству Max значение меньше минимального |
| Value\_GetCorrectValue\_  ReturnCorrectValue() | 150 | Позитивный тест гетера свойства Value класса Parameter |

Продолжение таблица А.1

|  |  |  |
| --- | --- | --- |
| Метод | Входные параметры | Описание тестового случая |
| Value\_SetCorrectValue\_  SetCorrectValue() | 200.99 | Позитивный тест сетера свойства Value класса Parameter |
| Value\_SetThrowsArgumentException\_  ValueIsLessThanMin() | 1.1 | Присвоению свойству Value значения меньше минимального |
| Value\_SetThrowsArgumentException\_  ValueIsHigherThanMax() | 9999999.99 | Присвоению свойству Value значения больше максимального |
| Parameter\_ThrowsArgumentException\_  MaxIsLowerThanMin() | "Parameter,", 100, 50, 160 | Инициализация конструктора с значением свойства Max меньшим чем свойства Min |
| ShelfBootsPlaceLength\_GetCorrectValue\_  ReturnCorrectValue() | parameters.ShelfBootsPlaceLength | Позитивный тест гетера свойства ShelfBootsPlaceLength класса Parameters |

Продолжение таблица А.1

|  |  |  |
| --- | --- | --- |
| Метод | Входные параметры | Описание тестового случая |
| ShelfBootsPlaceLength\_SetCorrectValue\_  SetCorrectValue() | 100 | Позитивный тест сетера свойства ShelfBootsPlaceLength класса Parameters |
| ShelfBootsPlaceWidth\_GetCorrectValue\_  ReturnCorrectValue() | parameters.ShelfBootsPlaceWidth | Позитивный тест гетера свойства ShelfBootsPlaceWidth класса Parameters |
| ShelfBootsPlaceWidth\_SetCorrectValue\_  SetCorrectValue() | 100 | Позитивный тест сетера свойства ShelfBootsPlaceWidth класса Parameters |
| Parameters\_SetCorrectValue\_  SetCorrectValue() | 430.5 | Позитивный тест сетера конструктора Parameters на задание верного значения свойству Value. |