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

на тему «Разработка библиотеки «Табурет»

для САПР «Компас-3D»

по дисциплине «Основы разработки САПР»

Выполнил:

студент гр. 589-1

\_\_\_\_\_\_\_А.А. Жданова

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

Руководитель:

к.т.н, доцент каф. КСУП

\_\_\_\_\_\_\_ А.А. Калентьев

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

Томск 2022

**Реферат**

Учебная работа 33 страница, 13 таблиц, 15 рисунков, 13 источников.

Ключевые слова: КОМПАС-3D, ТАБУРЕТ, СИСТЕМА АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ, БИБЛИОТЕКА.

Целью данной работы является разработка библиотеки табурета для системы автоматизированного проектирования Компас-3D.

В процессе работы должны были пройдены все стадии разработки программного обеспечения: выбор темы, составление технического задания, составление проекта системы, создание первой реализации и её тестирование, создание конечной библиотеки.

Отчет по учебной работе выполнен в текстовом редакторе Microsoft Word 2016.

**Оглавление**

[Введение 4](#_Toc120874339)

[1 ПОСТАНОВКА И АНАЛИЗ ЗАДАЧИ 4](#_Toc120874340)

[1.1 Описание предмета проектирования 5](#_Toc120874341)

[1.2 Выбор инструментов и средств реализации 7](#_Toc120874342)

[1.3 Назначение плагина 7](#_Toc120874343)

[2 ОБЗОР АНАЛОГОВ 8](#_Toc120874344)

[2.1 SketchUp: Плагин для моделирования мебели 9](#_Toc120874345)

[3 ОПИСАНИЕ РЕАЛИЗАЦИИ 10](#_Toc120874346)

[4 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПОЛЬЗОВАТЕЛЯ 10](#_Toc120874347)

[5 ТЕСТИРОВАНИЕ ПРОГРАММЫ 22](#_Toc120874348)

[5.1 Функциональное тестирование 22](#_Toc120874349)

[5.2 Модульное тестирование 24](#_Toc120874350)

[5.3 Нагрузочное тестирование 26](#_Toc120874351)

[Заключение 26](#_Toc120874352)

[Список использованных источников 31](#_Toc120874353)

# **Введение**

В настоящее время, большинство предприятий стремятся проектировать в трехмерном пространстве. Трехмерные CAD-системы предоставляют проектировщику большой простор для творчества и при этом позволяют значительно ускорить процесс выпуска проектно-сметной документации. Наряду со скоростью, такие системы позволяют повысить точность проектирования: становится проще отследить спорные моменты в конструкции.

КОМПАС-3D, как универсальная система трехмерного проектирования, находит своё применение при решении различных задач, в том числе и архитектурно-строительного и технологического проектирования.

Наиболее широкое применение система получила в решении задач проектирования металлических конструкций - стальных сооружений, фасадных и купольных конструкций из алюминиевого профиля и т.п.

Основные компоненты КОМПАС-3D — собственно система трехмерного твердотельного моделирования, универсальная система автоматизированного проектирования КОМПАС-График и модуль проектирования спецификаций. Все они легки в освоении, имеют русскоязычные интерфейс и справочную систему.

Система КОМПАС-3D предназначена для создания трехмерных ассоциативных моделей отдельных деталей и сборочных единиц, содержащих как оригинальные, так и стандартизованные конструктивные элементы. Параметрическая технология позволяет быстро получать модели типовых изделий на основе однажды спроектированного прототипа. Многочисленные сервисные функции облегчают решение вспомогательных задач проектирования и обслуживания производства [1].

Так, целью данной работы является разработка библиотеки табурета для системы автоматизированного проектирования Компас-3D.

# **1 ПОСТАНОВКА И АНАЛИЗ ЗАДАЧИ**

В рамках учебной дисциплины «Основы разработки САПР» требовалось разработать библиотеку в соответствии с техническим заданием. На основе заданных параметров библиотека, взаимодействуя с САПР «Компас-3D», должна строить трёхмерную модель табурета. Также библиотека должна позволять изменять входные параметры модели.

## **1.1 Описание предмета проектирования**

Предметом проектирования является табурет.

Измеряемые параметры:

* W – ширина сиденья (300 – 400 мм);
* S – расстояние между ножками (190 – 230 мм);
* H – высота сиденья (10 – 50 мм);
* w1 – толщина ножек (30 – 50 мм);
* h1 – высота ножек (300 – 500 мм);
* h2 – высота царги (20 – 90 мм).

Ширина сиденья должна быть больше расстояния между ножками, но не более чем на 190 мм.

Изображение предмета проектирования с обозначенными параметрами приведено на рисунке 1.1.

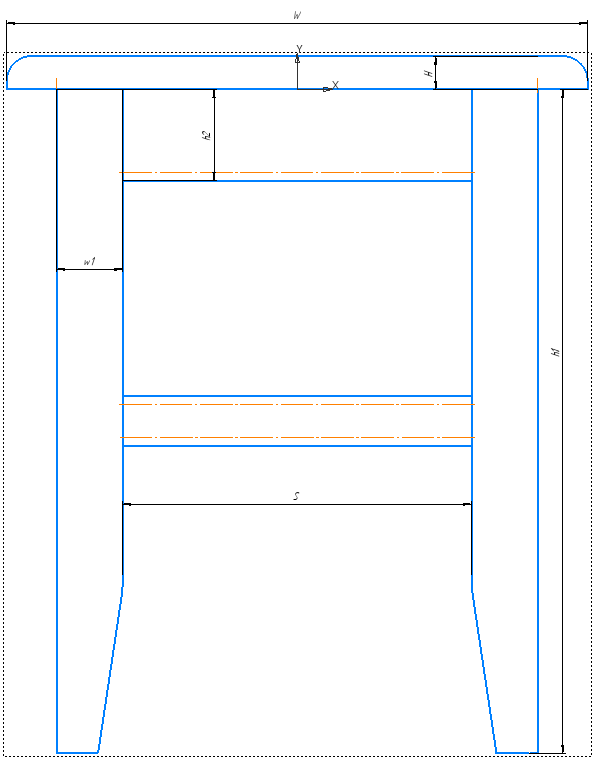


Рисунок 1.1 – Чертеж табурета

На рисунке 1.2 представлена 3D-модель табурета.



Рисунок 1.2 – 3D-модель табурета

Также для предмета существует ГОСТ 12029-93 – Мебель. Стулья и табуреты. Определение прочности и долговечности [2].

# **1.2 Выбор инструментов и средств реализации**

# Для создания библиотеки используется интегрированная среда разработки Visual Studio 2019 [3]. Библиотека написана с пользовательским интерфейсом на WindowsForms с использованием .NET Framework 4.7.2 [4]. Инструментом для тестирования является библиотеки NUnit 3.13.3 и NUnit3TestAdapter 4.2.1 [5]. В качестве системы автоматизированного проектирования выбран «Компас-3D» v.20 [1].

# **1.3 Назначение плагина**

Программа предназначена для автоматизации моделирования детали «Табурет».

Плагин позволяет пользователю ввести вышеперечисленные значения через графический интерфейс. В программе предусмотрена проверка корректности введенных данных.

При запуске моделирования с некорректными значениями программа выводит сообщение об ошибке и отменяет построение модели. При правильно введенных значениях результатом работы программы будет созданная по ним модель табурета.

2 ОБЗОР АНАЛОГОВ

# **2.1 SketchUp: Плагин для моделирования мебели**

Craftreport – плагин для SketchUp решение для частных мастерских. Многократно упрощает процесс формирования заявок на:

1. расскрой/кромление листовых материалов;

2. заказ фасадов;

3. заказ фурнитуры и крепежa.

Абонемент на Craft-kincten.ru дает возможность его владельцу иметь доступ к актуальным 3D моделям кухонных гарнитуров разных стилей с высокой деталировкой (петли, ящики, крепеж и пр.). Представленные на сайте модели кухонь — это воплощённые в жизнь проекты частной мастерской. Средства SketchUp позволяют максимально комфортно редактировать данные модели под свои задачи [6].

Плагин с открытым исходным кодом. В нем можно указать направление рисунка текстуры, задать кромку для каждой стороны детали и экспортировать список деталей в таблицу для распила.

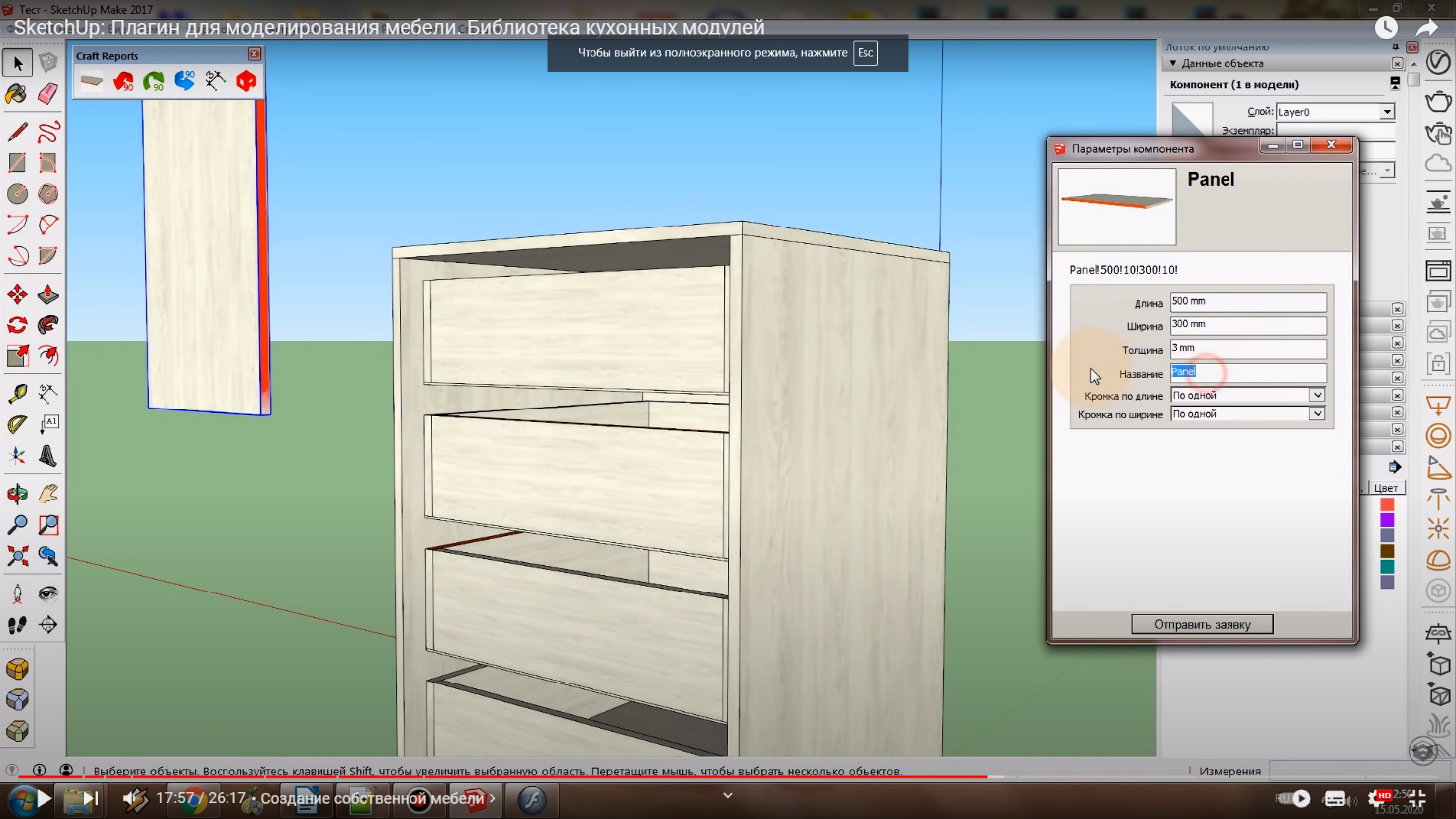


Рисунок 2.1 – Пользовательский интерфейс плагина Craftreport для моделирования мебели

# **3 ОПИСАНИЕ РЕАЛИЗАЦИИ**

Диаграмма классов – это UML-диаграмма, которая описывает систему, визуализируя различные типы объектов внутри системы и виды статических связей, которые существуют между ними. Он также иллюстрирует операции и атрибуты классов [7].

Enterprise Architect (EA) – CASE-инструмент для проектирования и конструирования программного обеспечения. EA поддерживает спецификацию UML2.0+, описывающую визуальный язык, которым могут быть определены модели проекта.

Некоторые из ключевых функций ЕА:

* создание элементов UML-моделей широкого круга назначения;
* размещение этих элементов в диаграммах и пакетах;
* создание коннекторов между элементами;
* документирование созданных элементов;
* генерация кода для конструируемого ПО;
* реверс-инжиниринг имеющегося кода на некоторых языках [8].

На рисунке 3.1 представлена примерная диаграмма классов разрабатываемой библиотеки до реализации.

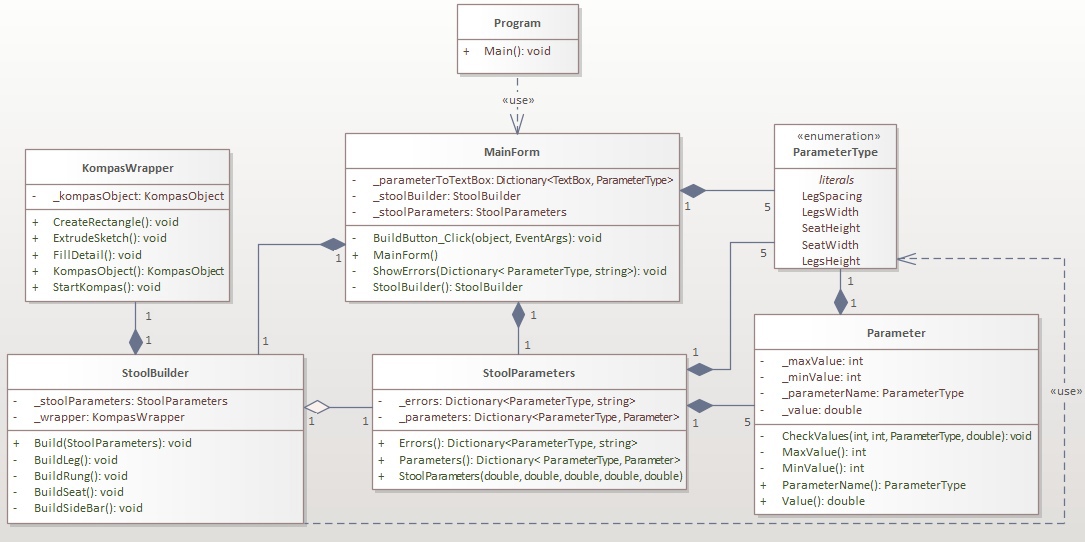


Рисунок 3.1 – UML-диаграмма классов до реализации

Класс «Program», использует «MainForm» для обработки действий в графическом интерфейсе. Класс «KompasWrapper» предназначен для взаимодействия с САПР «Компас 3D». «StoolBuilder» содержит в себе методы создания 3D модели, класс «StoolParameters» введенные значения в графическом интерфейсе. Перечисление «ParameterType» используется для минимизации шанса ошибки при вводе параметров в коде. Класс «Parameter» нужен для обозначения области допустимых значений параметра и проверки, находится ли параметр в диапазоне.

Ошибка будет происходить в тех случаях, когда будет произведена попытка постройки модели, значения параметров которой:

* не входят в допустимый диапазон значений;
* включают символы отличные от цифр;
* являются отрицательными.

В таблицах 3.1-3.6 представлены перечисление Parameters и основные классы MainForm, KompasWrapper, StoolBuilder, Parameter и StoolParameters их поля, методы и свойства до реализации.

Таблица 3.1 – Класс MainForm

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название | Тип | Описание |
| Поля | \_stoolParameters | StoolParameters | Объект параметров табурета |
| \_stoolBuilder | StoolBuilder | Объект класса построения детали |
| \_parameterToTextBox | Dictionary <  ParameterType, \_parameterToTextBox> | Словарь для введенных параметров |
| Свойства | StoolBuilder | StoolBuilder | Возвращает объект построения детали |
| Методы | MainForm | Конструктор | Конструктор основной формы |
| ShowErrors(Dictionary<ParameterType, string>) | void | Показ ошибок |
| BuildButton\_Click(object, EventArgs) | void | Построение при нажатии на кнопку |

Таблица 3.2 – Класс KompasWrapper

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название | Тип | Описание |
| Поля | \_kompasObject | KompasObject | Объект Компас API |
|  | KompasObject | KompasObject | Возвращает объект Компас API |
| Методы | СreateRectangle | void | Построение прямоугольника |
| ExtrudeSketch | void | Выдавливание эскиза |
| FillDetail | void | Закрасить деталь |
| StartKompas | void | Запуск Компас-3D |

Таблица 3.3 – Класс StoolBuilder

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название | Тип | Описание |
| Поля | \_wrapper | KompasWrapper | Связь с Компас-3D |
| \_stoolParameters | StoolParameters | Объект параметров табурета |
| Методы | Build(StoolParameters) | void | Построение детали по заданным параметрам |
| BuildSeat | void | Построение сидения |
| BuildLeg | void | Построение ножек |
| BuildRung | void | Построение проножки |
| BuildSideBar | void | Построение царги |

Таблица 3.4 – Класс Parameter

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название | Тип | Описание |
| Поля | \_minValue | int | Минимальная величина |
| \_maxValue | int | Максимальная величина |
| \_value | double | Введенная величина |
| \_parameterType | ParameterType | Тип параметра |
| Свойства | Value | double | Возвращает и устанавливает значение параметра |
| MinValue | int | Возвращает и устанавливает значение минимальной величины |
| MaxValue | int | Возвращает и устанавливает значение максимальной величины |
| ParameterType | ParameterType | Возвращает и устанавливает значение типа параметра |
| Методы | CheckValues(int, int, ParameterType, double) | void | Проверка введенных значений параметров |

Таблица 3.5 – Перечисление ParameterType

|  |  |
| --- | --- |
| Название элемента | Описание |
| SeatWidth | Длина сиденья |
| SeatHeight | Ширина сиденья |
| LegsWidth | Ширина ножек |
| LegsHeight | Длина ножек |
| LegSpacing | Расстояние между ножками |

Таблица 3.6 – Класс StoolParameters

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название | Тип | Описание |
| Поля | \_parameters | Dictionary <  ParameterType, Parameter> | Словарь "тип параметра - параметр" |
| \_errors | Dictionary< ParameterType, string> | Словарь ошибок |

Окончание таблицы 3.6.

|  |  |  |  |
| --- | --- | --- | --- |
| Свойства | Parameters | Dictionary <  ParameterType, Parameter> | Возвращает и устанавливает значение словаря "тип параметра - параметр" |
| Errors | Dictionary< ParameterType, string> | Возвращает и устанавливает значение словаря ошибок |
| Методы | StoolParameters(double, double, double, double, double) | Конструктор | Создает объект класса табурета |

В процессе реализации программы классы претерпели сильные изменения, и диаграмма стала выглядеть иначе. Перечисление «ParameterType» теперь содержит 6 элементов, в том числе высоту царги. Другие связанные с этим изменения методов были добавлены и в остальные классы. Связь между «StoolParameters» и «StoolBuilder» изменена на использование.

Класс «Parameter» теперь имеет элементы для вывода ошибок при неверном вводе значений для параметров, конструктор, проверку на равенство объектов класса для unit-тестирования. Также удалены лишние свойства. В классе «StoolParameters» исправлено отображение свойств, добавлена проверка взаимосвязи параметров между собой и функция создания объекта построения для проверки на ошибки и изменения значения величин, принятых из формы. «StoolBuilder» больше не содержит объект \_stoolParameters. В класс «KompasWrapper» добавлены поля, свойства и методы, связанные с построением детали в Компас-3D, в том числе расширены функции для построения линии, дуги и прочего. В «MainForm» добавлены обозначения для цвета ошибки и верного цвета, а также проверки на правильный ввод и наличие ошибок.

На рисунке 3.2 представлена итоговая диаграмма классов разрабатываемой библиотеки после реализации.

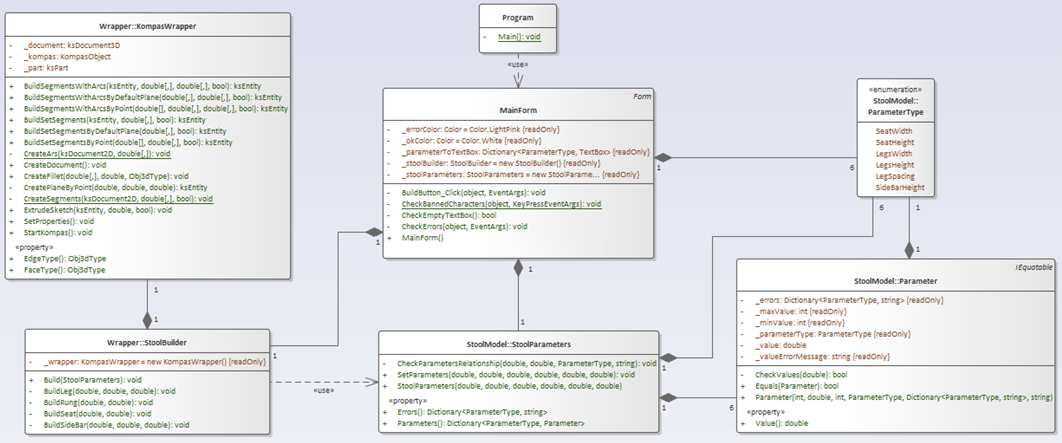
****

Рисунок 3.2 – UML-диаграмма классов после реализации

В таблицах 3.7-3.12 представлены перечисление Parameters и основные классы MainForm, KompasWrapper, StoolBuilder, Parameter и StoolParameters их поля, методы и свойства после реализации.

Таблица 3.7 – Класс MainForm

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название | Тип | Описание |
| Поля | \_stoolParameters | StoolParameters | Объект параметров табурета |
| \_stoolBuilder | StoolBuilder | Объект класса построения детали |
| \_parameterToTextBox | Dictionary <  ParameterType, \_parameterToTextBox> | Словарь для введенных параметров |
| \_errorColor | Color | Цвет поля при ошибке |
| \_okColor | Color | Цвет поля при корректных данных |
| Методы | MainForm | Конструктор | Конструктор основной формы |
| CheckErrors(object, EventArgs) | void | Проверка введенных значений |
| CheckEmptyTextBox | bool | Поиск пустых полей |
| CheckBannedCharacters(object,KeyPressEventArgs) | void | Запрет ввода символов и больше одной точки в число |
| BuildButton\_Click(object, EventArgs) | void | Построение при нажатии на кнопку |

Таблица 3.8 – Класс KompasWrapper

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название | Тип | Описание |
| Поля | \_kompas | KompasObject | Объект Компас API |
| \_part | ksPart | Деталь |
| \_document | ksDocument3D | Документ-модель |
| Свойства | FaceType | Obj3dType | Возвращает тип перечисления грани |
| EdgeType | Obj3dType | Возвращает тип перечисления ребра |
| Методы | StartKompas | void | Запуск Компас-3D |
| CreateDocument | void | Создание файла в Компас-3D |
| SetProperties | void | Установка свойств модели |

Окончание таблицы 3.8.

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | CreateSegments(ksDocument2D, double[,]) | void | Создание на эскизе отрезков |
| CreateArs(ksDocument2D, double[,]) | void | Создание на эскизе дуг |
| CreatePlaneByPoint(double, double, double) | ksEntity | Создание плоскости построения по точке |
| BuildSetSegments(kEntity, double[,], bool) | ksEntity | Построение набора отрезков |
| BuildSetSegmentsByDefaultPlane(double[,], bool) | ksEntity | Создание эскиза набора отрезков по базовой плоскости |
| BuildSetSegmentsByPoint(double[,],double[,], bool) | ksEntity | Создание эскиза набора отрезков по точке |
| BuildSegmentsWithArcs(ksEntity, double[,], double[,], bool) | ksEntity | Создание эскиза из отрезков и дуг |
| BuildSegmentsWithArcsByDefaultPlane(double[,], double[,], bool) | ksEntity | Создание эскиза отрезков и дуг по базовой плоскости |
| BuildSegmentsWithArcsByPoint(double[,], double[,], double[,], bool) | ksEntity | Создание эскиза отрезков и дуг по точке |
| ExtrudeSketch(ksEntity, double, bool) | void | Выдавливание эскиза |
| CreateFillet(double[,],double, Obj3dType) | void | Создания скругления |

Таблица 3.9 – Класс StoolBuilder

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название | Тип | Описание |
| Поля | \_wrapper | KompasWrapper | Связь с Компас-3D |
| Методы | Build | void | Построение детали по заданным параметрам |
| BuildSeat | void | Построение сидения |
| BuildLeg | void | Построение ножек |
| BuildRung | void | Построение проножки |
| BuildSideBar | void | Построение царги |

Таблица 3.10 – Класс Parameter

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название | Тип | Описание |
| Поля | \_minValue | int | Минимальная величина |
| \_maxValue | int | Максимальная величина |
| \_value | double | Введенная величина |
| \_parameterType | ParameterType | Тип параметра |
| \_valueErrorMessage | string | Сообщение об ошибке |
| \_errors | Dictionary<ParameterType, string> | Словарь ошибок для значений параметров |
| Свойства | Value | double | Возвращает и устанавливает значение параметра |
| Методы | Parameter(int, double, int, ParameterType, Dictionary<ParameterType, string>, string) | Конструктор | Создание объекта класса Parameter |
| CheckValues(double) | bool | Проверка введенных значений параметров |
| Equals(Parameter) | bool | Проверка на равенство объектов класса. |

Таблица 3.11 – Перечисление ParameterType

|  |  |
| --- | --- |
| Название элемента | Описание |
| SeatWidth | Длина сиденья |
| SeatHeight | Ширина сиденья |
| LegsWidth | Ширина ножек |
| LegsHeight | Длина ножек |
| LegSpacing | Расстояние между ножками |
| SideBarHeight | Высота царги |

Таблица 3.12 – Класс StoolParameters

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название | Тип | Описание |
| Свойства | Parameters | Dictionary <  ParameterType, Parameter> | Словарь "тип параметра - параметр" |
| Errors | Dictionary< ParameterType, string> | Словарь ошибок |

Окончание таблицы 3.12.

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | StoolParameters(double, double, double, double, double, double) | Конструктор | Создает объект класса табурета |
| SetParameters(double, double, double, double, double, double) | void | Создает объект для построения |
| CheckParametersRelationship(double, double, ParameterType, string) | void | Проверка взаимосвязи параметров между собой |

4 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПОЛЬЗОВАТЕЛЯ

Пользовательский интерфейс (user interface или сокращенно UI) – это интерфейс, с помощью которого человек может управлять программным обеспечением или аппаратным оснащением [9].

В данном случае пользовательский интерфейс представляет собой форму для ввода параметров. После введения всех необходимых параметров, а следом нажатия на кнопку «Построить» строится 3D-модель табурета.

Для данной формы осуществляется проверка введенных значений: пользователь может ввести только положительные численные значения. При попытке построить модель с некорректных введенными значениями текстовое поле окрашивается в светло-красный, а также на экране появляется окно с выводом всех ошибок.

При запуске в форму автоматически вбиваются стандартные параметры.

Ширина сиденья должна быть больше расстояния между ножками не более чем на 190 мм.

На рисунке 3.2 представлены макет пользовательского интерфейса.

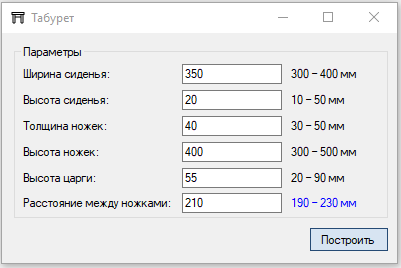


Рисунок 4.1 – Макет пользовательского интерфейса

На рисунке 4.2 представлена неверно заполненная форма. На рисунке 4.3 представлено окно с выводом всех ошибок.

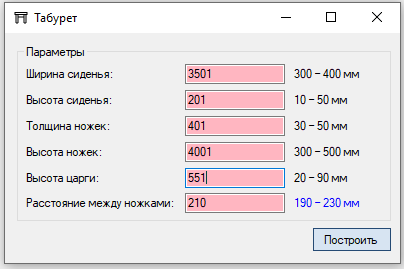


Рисунок 4.2 – Неверно заполненная форма

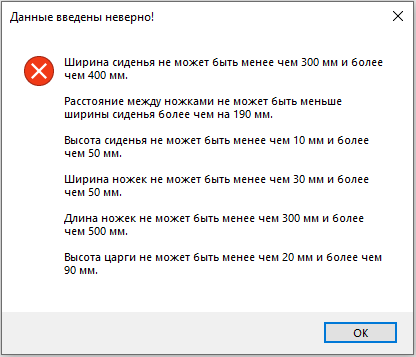


Рисунок 4.3 – Окно с выводом всех ошибок

При вводе верных значений и последующем нажатии кнопки происходит построение модели. 3D-модель табурета, построенная в САПР «Компас 3D», представлена на рисунке 4.4.



Рисунок 4.4 – 3D-модель табурета

5 ТЕСТИРОВАНИЕ ПРОГРАММЫ

5.1 Функциональное тестирование

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

В зависимости от сложности приложения, тестирование может проводиться на различных уровнях:

* Компонентное (модульное) – тестирование отдельных компонентов программного продукта, сфокусированное на их специфике, назначении и функциональных особенностях;
* Интеграционное тестирование – проводится после компонентного тестирования и направлено на выявление дефектов взаимодействия различных подсистем на уровне потоков управления и обмена данными.

Компоненты системы могут рассматриваться, как отдельные подсистемы. Внутри каждой подсистемы могут быть выделены отдельные компоненты, для которых проводится компонентное и интеграционное тестирование. Для сложных программных продуктов образуется иерархическая структура процесса тестирования, на каждом уровне которой объектом тестирования является определенная часть программного комплекса. [10].

В рамках функционального тестирования были проведены тесты библиотеки при введённых минимальных, средних и максимальных параметрах.

Минимальные параметры:

* Ширина сиденья - 300 мм;
* Расстояние между ножками - 190 мм;
* Высота сиденья - 10 мм;
* Толщина ножек - 30 мм;
* Высота ножек - 300 мм;
* Высота царги - 20 мм.

На рисунке 5.1 представлен результат построения при минимальных параметрах.



Рисунок 5.1 - Построение при минимальных параметрах

Средние параметры:

* Ширина сиденья - 350 мм;
* Расстояние между ножками - 210 мм;
* Высота сиденья - 30 мм;
* Толщина ножек - 40 мм;
* Высота ножек - 400 мм;
* Высота царги - 55 мм.

На рисунке 5.2 представлен результат построения при средних параметрах.

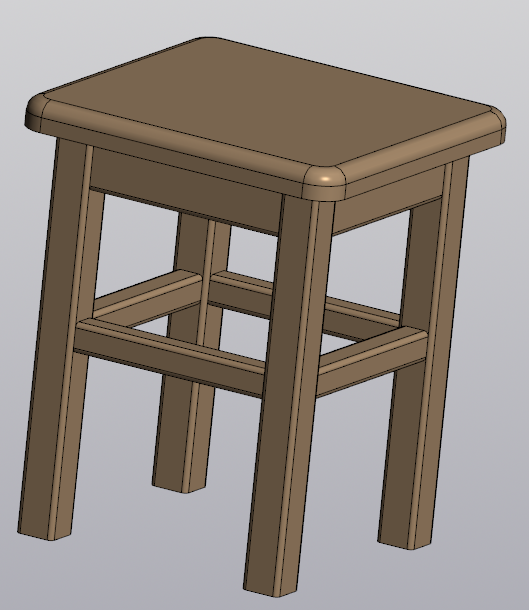


Рисунок 5.2 - Построение при максимальных параметрах

Максимальные параметры:

* Ширина сиденья - 400 мм;
* Расстояние между ножками - 280 мм;
* Высота сиденья - 50 мм;
* Толщина ножек - 50 мм;
* Высота ножек - 500 мм;
* Высота царги - 90 мм.

На рисунке 5.3 представлен результат построения при максимальных параметрах.

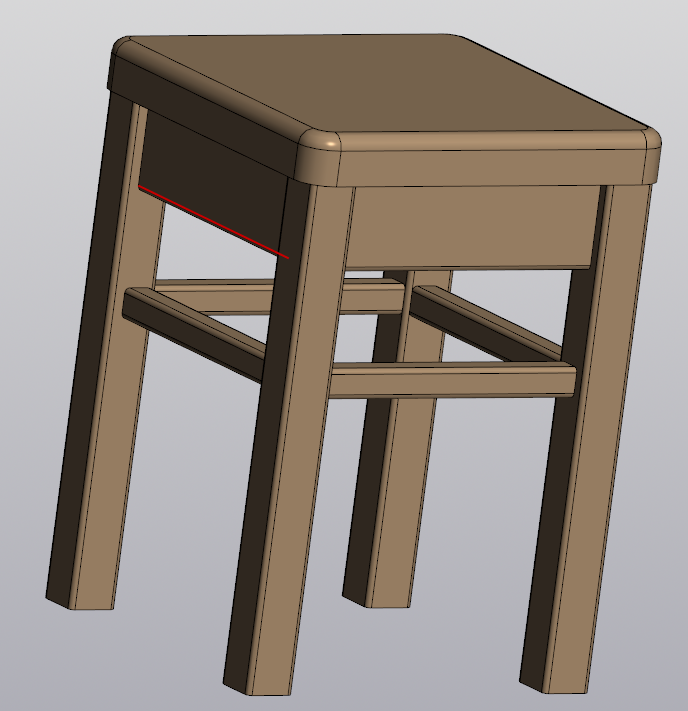


Рисунок 5.3 - Построение при максимальных параметрах

5.2 Модульное тестирование

Модульное тестирование (Unit Testing) – это тип тестирования программного обеспечения, при котором тестируются отдельные модули или компоненты программного обеспечения. Его цель заключается в том, чтобы проверить, что каждая единица программного кода работает должным образом. Данный вид тестирование выполняется разработчиками на этапе кодирования приложения. Модульные тесты изолируют часть кода и проверяют его работоспособность. Единицей для измерения может служить отдельная функция, метод, процедура, модуль или объект [11].

Для тестирования были использованы библиотеки NUnit 3.13.3 и NUnit3TestAdapter 4.2.1 [5]. Покрытие тестами логики программы – 100%. Всего написано 8 тестов. На рисунке 5.4 представлено тестирование логики.

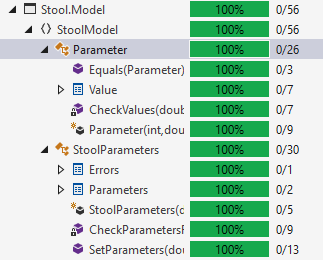


Рисунок 5.4 – Модульное тестирование

5.3 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 [11].

Нагрузочное тестирование проводилось на персональном компьютере со следующей конфигурацией: Конфигурация компьютера, на котором проводилось нагрузочное тестирование представлена в таблице 5.1.

Таблица 5.1 – Конфигурация компьютера

|  |  |
| --- | --- |
| Процессор | Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz 3.60 GHz |
| Оперативная память | 16,0 ГБ |
| Тип системы | 64-разрядная операционная система, процессор x64 |
| Видеокарта | Видеокарты серии GeForce GTX 1070 объемом памяти 8 ГБ |

Для тестирования было использовано зацикленное построение модели с параметрами по умолчанию.

Было произведено 5 нагрузочных тестов. В среднем за один цикл тестирования было построено 132 модели. По полученным измерениям был рассчитан средний результат, а затем построены графики. На рисунке 5.4 показана зависимость используемой оперативной памяти в гигабайтах от количества построенных моделей.



Рисунок 5.4 – Зависимость оперативной памяти от количества моделей

На графике можем выделить 3 этапа. Изначально относительно прямая линия растет, затем начинает расти быстрее, а затем скорость ее возрастания существенно сокращается, и появляются скачки. Первое изменение скорости может быть результатом нескольких причин. К таким причинам относятся:

* Снижение нагрузки на ОЗУ материнской платой, если текущая модель имеет такую функцию;
* Ограничение оперативной памяти для построения другими работающими программами, например, антивирусом;
* Снятие уже неактивных проектов из ОЗУ самим Компасом-3D.

Чтобы не потерять содержимое памяти, заряд конденсаторов периодически восстанавливается («регенерируется») через определённое время, называемое циклом регенерации (обычно 2 мс). Так как для регенерации памяти периодически приостанавливается обращение к памяти, это снижает среднюю скорость обмена [12]. Отсюда понижение нагрузки и так называемые скачки на графике.

Когда свободной памяти недостаточно для одновременного запуска ресурсоёмких программ задействуется механизм виртуальной памяти. С её помощью продолжается обмен оперативными данными, которые требуются приложениям прямо сейчас, а старые и долго неиспользуемые данные перемещаются на медленное файловое хранилище. Файл подкачки повышает производительность всей системы, расширяет степень многозадачности, помогает без затрат на апгрейд оперативной памяти расширить возможности компьютера [13]. Поэтому в конце графика мы видим снижение нагрузки на оперативную память.

На рисунке 5.5 показана зависимость потраченного времени от количества построенных моделей.



Рисунок 5.5 – Зависимость времени от количества построенных моделей

Из графика видно, что зависимость необходимого времени от числа деталей является практически линейной.

Заключение

В результате выполнения данной учебной работы были пройдены все стадии разработки программного обеспечения: выбор темы, составление технического задания, составление проекта системы, создание первой реализации и её тестирование, создание конечной библиотеки. Был изучен API приложения Компас-3D. Также были найдены аналоги разрабатываемого плагина. Были спроектированы UML диаграммы классов и проведено функциональное, модульное и нагрузочное тестирования созданной библиотеки.

Список использованных источников

1. Проектирование в системе Компас-3D [Электронный ресурс] – Режим доступа: <https://works.doklad.ru/view/cdxwB2HPqPk.html> (дата обращения: 02.12.2022)
2. Межгосударственный стандарт. Мебель. Стулья и табуреты [Электронный ресурс] – Режим доступа: <https://docs.cntd.ru/document/1200017702> (дата обращения: 02.12.2022)
3. Добро пожаловать в интегрированную среду разработки Visual Studio [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022> (дата обращения: 02.12.2022)
4. Автономный установщик Microsoft платформа .NET Framework 4.7.2 для Windows [Электронный ресурс] – Режим доступа: <https://support.microsoft.com/ru-ru/topic/автономный-установщик-microsoft-платформа-net-framework-4-7-2-для-windows-05a72734-2127-a15d-50cf-daf56d5faec2> (дата обращения: 02.12.2022)
5. NUnit [Электронный ресурс] – Режим доступа: <https://www.nuget.org/packages/NUnit/> (дата обращения: 02.12.2022)
6. CraftReports плагин SketchUp для моделирования и отчетов [Электронный ресурс] – Режим доступа: <https://craft-kitchen.ru/> (дата обращения: 02.12.2022)
7. Новые технологии в программировании: учебное пособие / А. А. Калентьев, Д. В. Гарайс, А. Е. Горяинов — Томск: Эль Контент, 2014. — 176 с.
8. Enterprise Architect [Электронный ресурс] – Режим доступа: <https://sparxsystems.com/> (дата обращения: 02.12.2022)
9. Что такое интерфейс пользователя? [Электронный ресурс] – Режим доступа: <https://www.internet-technologies.ru/articles/newbie/polzovatelskiy-interfeys.html> (дата обращения: 02.12.2022)
10. Функциональное тестирование: пробы с физическими нагрузками: учебно-методическое пособие. – Томск: Издательство Томского государственного университета, 2021. – 38 с.
11. Тестирование, оценка программного обеспечения. / М. М. Меженная, Т. В. Гордейчук, М. М. Борисик, О. С. Медведев, И.Ф. Киринович. – Минск: БГУИР, 2016. – 64 с.
12. RAM (Random Access Memory) [Электронный ресурс] – Режим доступа: <https://ru.bmstu.wiki/index.php?title=RAM_(Random_Access_Memory)&mobileaction=toggle_view_desktop> (дата обращения: 13.12.2022)
13. Файл подкачки — что это такое, зачем нужен, что делает [Электронный ресурс] – Режим доступа: <https://www.zeluslugi.ru/info-czentr/it-glossary/term-file-podkachki> (дата обращения: 13.12.2022)