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)

Разработка плагина «Киянка» для САПР «КОМПАС-3D» v.18

Пояснительная записка по дисциплине

«Основы разработки САПР» (ОрСАПР)

Выполнил:

Студент гр.588-3

\_\_\_\_\_\_\_Абдеев Т.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Руководитель:

к.т.н., доцент

каф. КСУП

\_\_\_\_\_\_\_Калентьев А.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**РЕФЕРАТ**

Лабораторная работа 32 с., 16 рис., 11 источников.

Ключевые слова: КОМПАС-3D, VISUAL STUDIO, C#, КОМПАС API, ПЛАГИН, САПР.

Целью данной работы является разработка плагина для автоматизации проектирования киянки.

Результатом работы является плагин, осуществляющий построение киянки по заданным пользователям параметрам.

Пояснительная записка выполнена в текстовом редакторе Microsoft Word 2019.

Содержание

[1 Введение 4](#_Toc100568040)

[2 Описание API 5](#_Toc100568041)

[3 Обзор аналогов 11](#_Toc100568042)

[4 Описание предмета проектирования 12](#_Toc100568043)

[4.1 Выбор инструментов и средств реализации 15](#_Toc100568044)

[4.2 Назначение плагина 15](#_Toc100568045)

[5 Описание реализации 16](#_Toc100568046)

[5.1 Диаграмма классов 16](#_Toc100568047)

[6 Описание программы для пользователя 19](#_Toc100568048)

[7 Тестирование плагина 22](#_Toc100568049)

[7.1 Функциональное тестирование 22](#_Toc100568050)

[7.2 Модульное тестирование 25](#_Toc100568051)

[7.3 Нагрузочное тестирование 27](#_Toc100568052)

[Заключение 31](#_Toc100568053)

[Список использованных источников 32](#_Toc100568054)

[Приложение А 33](#_Toc100568055)

[Приложение Б 39](#_Toc100568056)

# Введение

Автоматизация моделирования имеет огромное значение для развития науки, техники и производства в современном обществе. В настоящее время автоматизация – основной способ повышения производительности и эффективности труда инженерно-технических работников, занимающихся моделированием сложных устройств. Использование автоматизации в проектировании позволяет создавать все более сложные технические объекты и гибко реагировать на появление новых решений и технологий в той или иной области техники. Она позволяет значительно повысить точность расчетов, выбрать наилучшие варианты для реализации на основе строгого математического анализа всех или большинства вариантов проекта с оценкой технических, технологических и экономических характеристик производства и эксплуатации проектируемого объекта, значительно повысить качество конструкторской документации, существенно сократить сроки проектирования и передачи конструкторской документации в производство, эффективнее использовать технологическое оборудование с программным управлением [1].

Таким образом, целью данной работы является разработка плагина, автоматизирующего построение модели «Киянка» для системы автоматизированного проектирования КОМПАС-3D с помощью интегрированной среды разработки Visual Studio 2019 Сommunity. [2]

Интегрированная среда разработки Visual Studio – это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

# Описание API

API (англ. Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой.

В «КОМПАС-3D» на данный момент существуют API двух версий: API 5 и API 7. Обе версии реализуют различные функции системы и взаимно дополняют друг друга. Обе версии программных интерфейсов в равной мере поддерживаются и развиваются с учетом самих изменений в системе.

Главным интерфейсом API системы КОМПАС является KompasObject. За получение указателя на этот интерфейс (фактически, на интерфейс приложения API 5) отвечает экспортная функция CreateKompasObject(). 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). [3]

Ниже в таблице 1.1 представлены основные свойства и методы интерфейса KompasObject.

Таблица 1.1 – Методы и свойства интерфейса KompasObject

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Тип возвращаемых данных | Описание |
| Document3D() |  | ksDocument | Метод для получения указателя на интерфейс трехмерного графического документа (детали или сборки) |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Тип возвращаемых данных | Описание |
| GetParamStruct(short structType) | structType – тип интерфейса параметров | StructType2D | Метод для получения указателя на интерфейс графического документа (чертежа или фрагмента) |
| Visible |  | bool | Свойство видимости приложения |
| Quit() |  |  | Метод для закрытия активного окна приложения |

В таблице 1.2 представлены методы интерфейса ksEntity, необходимые для разработки плагина.

Таблица 1.2 – Методы интерфейса ksEntity

|  |  |  |
| --- | --- | --- |
| Название | Тип возвращаемых данных | 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) |
| NewEntity (short objType) | указатель на интерфейс ksEntity или IEntity. | Реальный объект создается в модели после вызова метода [Create](mk:@MSITStore:D:\KOMPAS\SDK\SDK.chm::/ksEntity_Create.htm). |

В таблице 1.3 представлены свойства и методы интерфейса ksDocument2D, необходимые для разработки плагина.

Таблица 1.3 – Методы интерфейса ksDocument2D

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Тип возвращаемых данных | Описание |
| ksRectangle(  ksRectangleParam param, int style) | param – параметры прямоугольника.  style – стиль линии. | int | Получить указатель на прямоугольник на двумерной плоскости либо 0 в случае ошибки |
| ksCircle(  double xc, double yc, double rad, int style) | xc, yc – координаты центра окружности.  rad – радиус окружности.  style – стиль линии. | int | Получить указатель на окружность на двумерной плоскости либо 0 в случае ошибки |

В таблице 1.4 представлены свойства и методы интерфейса ksDocument3D, необходимые для разработки плагина.

Таблица 1.4 – Методы интерфейса ksDocument3D

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Входные параметры | Тип возвращаемых данных | Описание | |
| Create (bool invisible, bool \_typeDoc) | invisible – признак режима редактирования документа (true – невидимый режим, false –  видимый режим), typeDoc – тип документа (true – деталь, false – сборка). | bool | Создать документ-модель (деталь или сборку) | |
|  | |  |
| GetPart(int type) | type – тип компонента из перечисления Типы компонентов. | ksPart | | Получить указатель на интерфейс компонента в соответствии с заданным типом |

В таблице 1.5 представлены методы интерфейса ksPart, необходимые для разработки плагина.

Таблица 1.5 – Свойства и методы интерфейса ksPart.

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

В таблице 1.6 представлены типы объектов документа-модели, необходимые для разработки плагина.

Таблица 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\_bossExtrusion | Приклеивание выдавливанием | ksBossExtrusionDefinition |
| o3d\_cutExtrusion | Вырезать выдавливанием | ksCutExtrusionDefinition |
| o3d\_fillet | Операция "скругление" | ksFilletDefinition |
| o3d\_edge | Ребро | ksEdgeDefinition |
| o3d\_planeOffset | Смещённая плоскость | ksPlaneOffsetDefinition |

# Обзор аналогов

**Библиотека «Инструмент» для Компас-3D**

Сторонняя библиотека «Инструмент» обеспечивает быстрое вычерчивание различных инструментов в системе Компас-3D в соответствии с их точными размерами, согласно ГОСТ, ОСТ или ТУ и работает как отдельный модуль. Основные геометрические параметры инструментов хранятся в базе данных инструментов и полностью соответствуют ГОСТ. Для того чтобы можно было автоматически начертить инструмент любого типоразмера в графической системе, для каждого его вида разработана своя математическая модель. С помощью математических моделей и происходит расчет всех недостающих размеров инструмента, необходимых для его вычерчивания. [4]

На рисунке 3.1 показаны различные инструменты, построенные с помощью бибилиотеки.



Рисунок 3.1 — Интерфейс библиотеки «Инструмент»

# Описание предмета проектирования

Предметом проектирования является киянка.

Киянка представляет собой столярный молоток из твердых пород древесины или резины. [5]

Есть виды работ, в которых нужен более мягкий подход и не требуется большой силы. Распрямление пластин из металла, работа с долотами и стамесками, рукоятки которых имеют обжимное кольцо, сборка мебели – в таких видах работ и пригодится киянка. Пользование киянкой предохраняет рукоятки режущих инструментов от повреждений.

Для различных видов работ могут потребоваться киянки различных размеров, поэтому при необходимости можно изменять следующие параметры:

1. Ширина бойка киянки A (от 40 мм до 80 мм);
2. Длина бойка киянки B (от 100 мм до 200 мм);
3. Высота бойка киянки C (от 40 мм до B/2 мм);
4. Диаметр ручки киянки D (от 20 мм до A-10 мм);
5. Высота ручки киянки H (от 150 мм до 400 мм).
6. Радиус фаски на бойке F (от 1 до 4 мм).

Плагин имеет следующие зависимости:

1. Высота бойка киянки C должна не превышать половины длины бойка киянки B.

2. Максимальный диаметр ручки киянки D всегда должен быть на минимум на 10 мм меньше ширины бойка киянки A.

На рисунке 4.1 представлен чертеж модели киянки с указанием изменяемых параметров.

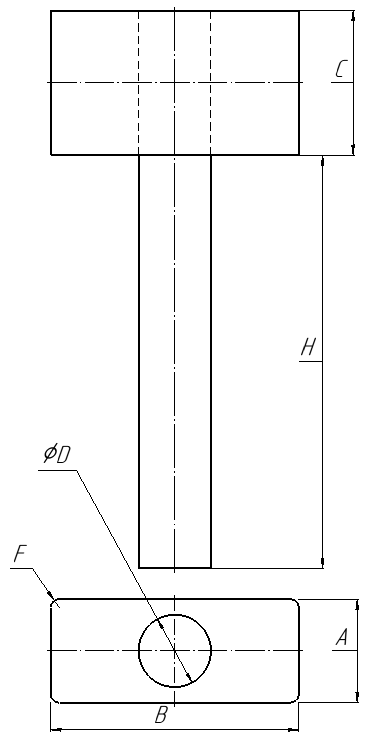


Рисунок 4.1 – Чертеж модели киянки

На рисунке 4.2 представлен общий вид модели киянки.

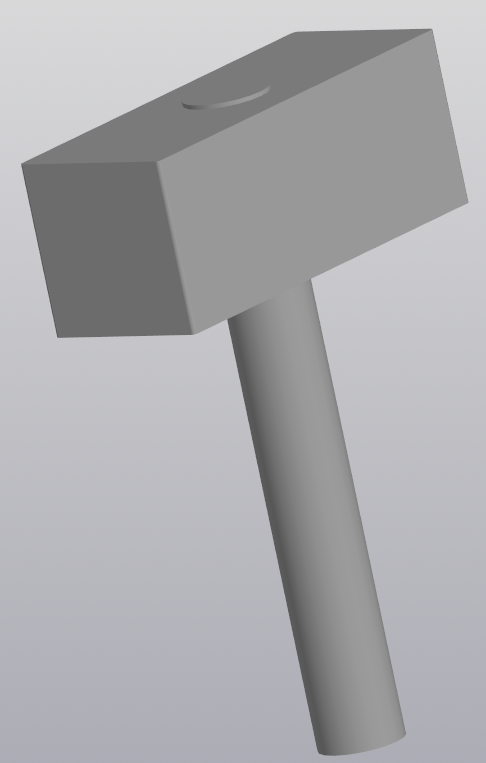


Рисунок 4.2 – Общий вид модели киянки

## Выбор инструментов и средств реализации

На основе требований к техническому заданию программа выполнена на языке программирования C# в среде Microsoft Visual Studio 2019 с использованием .NET Framework 4.7.2 [2], библиотека «Kompas6API5» [6] для основных операций в САПР КОМПАС-3D.

Инструментом тестирования и создания модульных тестов был выбран тестовый фреймворк NUnit [7] версии 3.13.2.

Технология разработки графического интерфейса: Windows Forms [8].

## Назначение плагина

Назначение разрабатываемого плагина обусловлено быстрым моделированием киянки с разными размерами бойка и ручки. Благодаря данному расширению, проектировщики могут наглядно рассмотреть спроектированную модель, при необходимости перестроить под необходимые им параметры.

# Описание реализации

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

UML 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 возможна генерация кода и наоборот. [9]

При использовании UML были простроена диаграмма классов.

## Диаграмма классов

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними.

На рисунке 5.1 представлена изначальная диаграмма классов.

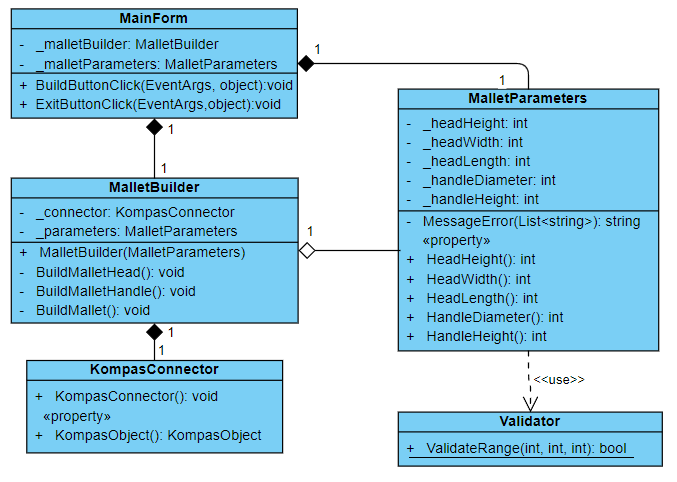


Рисунок 5.1 – Изначальная диаграмма классов

* KompasConnector – обеспечивает связь с API «КОМПАС-3D»;
* MainForm – содержит пользовательский интерфейс плагина;
* MalletParameters – содержит параметры 3D-модели киянки;
* MalletBuilder – осуществляет вызов методов API, необходимых для постройки 3D-модели;
* Validator – содержит методы проверки диапазона вводимых значений.

В итоговом проекте созданы следующие классы и методы, которые отображены на итоговой диаграмме классов (рисунок 5.2).

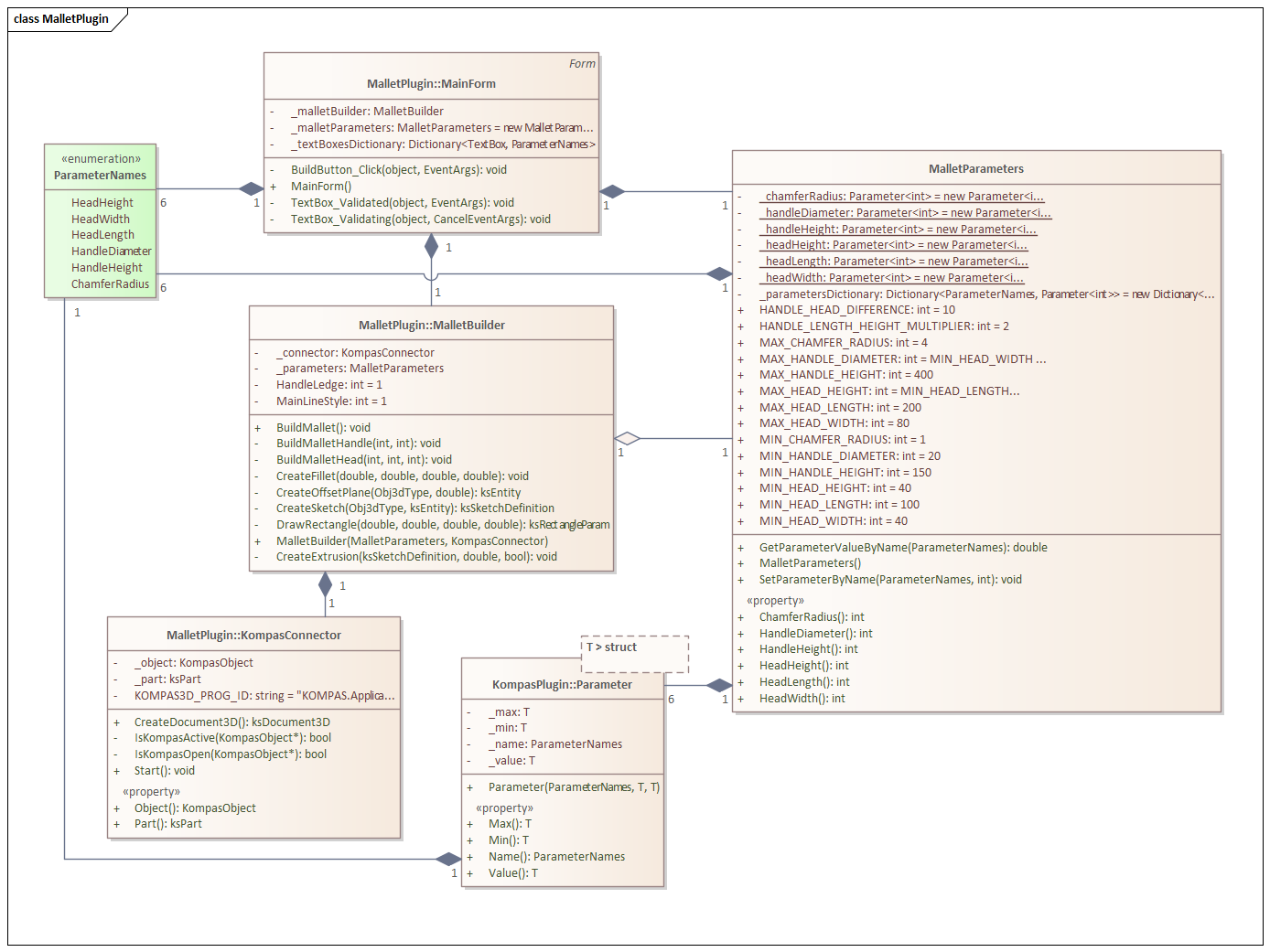


Рисунок 5.2 – Итоговая диаграмма классов

В процессе разработки была переосмыслена изначальная диаграмма классов, в результате чего:

* Константы минимальных и максимальных значений параметров были занесены в поля класса MalletParameters для удобства и доступности их значений в пользовательском коде, так же в нескольких классах был использован словарь для уменьшения дублирования кода и упрощения передачи значений параметров;
* Было принято решение не создавать отдельный класс Validator, вместо этого проверку параметров в заданном диапазоне осуществляет шаблонны класс Parameter, проверка зависимых параметров происходит в свойствах каждого параметра;
* Был разработан шаблонный класс Parameter для удобства проверки и задания граничных значений каждого параметра. Также данный класс осуществляет валидацию значения, заносимого в параметр, и формирует сообщения об ошибках ввода;
* Для предотвращения ошибок при поиске и занесении значения в параметры было решено отказаться от задания имени в строковом формате и было создано перечисление ParameterNames.

Описания полей, свойств и методов разработанных классов представлены в приложении А.

# Описание программы для пользователя

Пользовательский интерфейс плагина представляет собой пользовательскую форму с полями для ввода изменяемых параметров. Левее полей располагается описание вводимых в них параметров и их допустимые размеры. Ниже располагается кнопка для запуска построения «Build mallet». Если все данные введены верно, при нажатии на эту кнопку будет происходить построение модели. Справа находится изображение, демонстрирующее размеры модели для лучшего ориентирования пользователя.

На рисунке 6.1 представлен макет интерфейса программы.

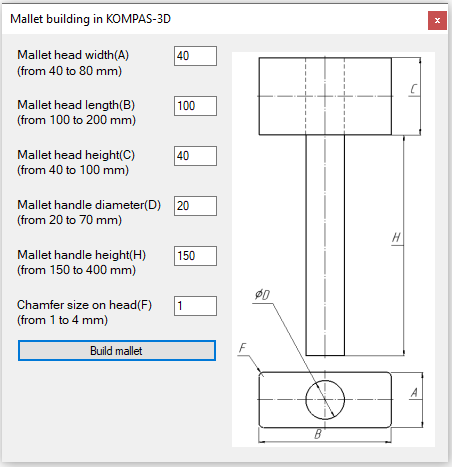


Рисунок 6.1 – Пользовательский интерфейс

При введении неверных параметров, кнопка «Build mallet» будет недоступна, а поля с неправильными данными будут выделены красным цветом, при наведении на них курсором мыши будет появляться сообщение с описанием ошибки. На рисунке 6.2 представлен макет интерфейса с некорректно введенными данными.

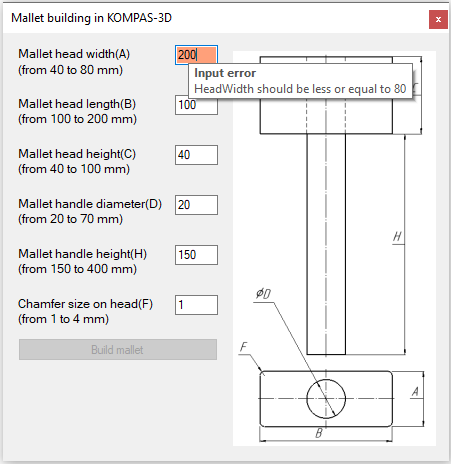


Рисунок 6.2 – Макет интерфейса с неверно введенными данными

После ввода корректных параметров кнопка «Build mallet» будет снова доступна и при ее нажатии будет построена модель киянки.

Киянка, построенная по параметрам, заданным по умолчанию, представлена на рисунке 6.3

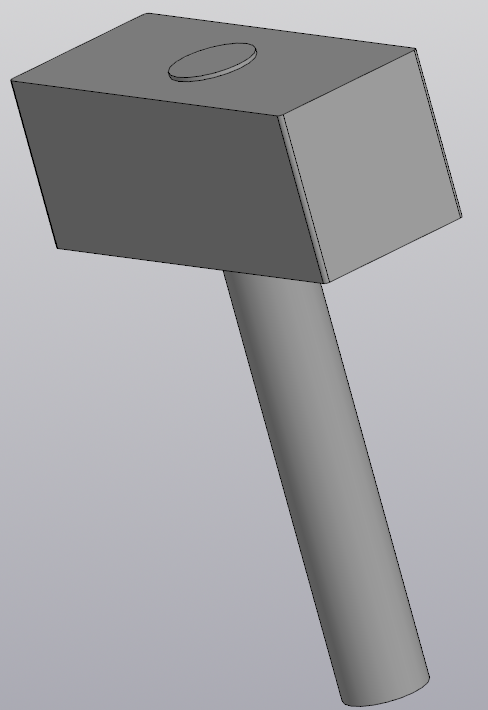


Рисунок 6.3 – Модель киянки, построенная

по параметрам по умолчанию

# Тестирование плагина

Тестирование позволяет убедиться в работоспособности программы, выявлять ошибки при изменении какого-либо функционала.

## Функциональное тестирование

При функциональном тестировании проверялось корректность работы плагина, а именно, соответствие полученного результата в виде трехмерной модели, с входными параметрами. [10]

Проведено тестирование максимальных и минимальных параметров модели.

Минимальные значения:

1. Ширина бойка киянки = 40 мм;
2. Длина бойка киянки = 100 мм;
3. Высота бойка киянки= 40 мм;
4. Диаметр ручки киянки = 20 мм;
5. Высота ручки киянки = 150 мм;
6. Размер фасок на бойке = 1 мм;

Максимальные значения:

1. Ширина бойка киянки = 80 мм;
2. Длина бойка киянки = 200 мм;
3. Высота бойка киянки= 100 мм;
4. Диаметр ручки киянки = 70 мм;
5. Высота ручки киянки = 400 мм;
6. Размер фасок на бойке = 4 мм;

На рисунке 7.1 представлена модель киянки с минимально введенными параметрами.

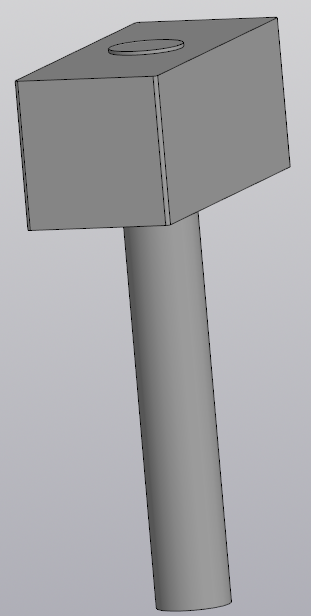


Рисунок 7.1 – Модель киянки с минимальными введенными параметрами

На рисунке 7.2 представлена модель киянки с максимальными введенными параметрами.

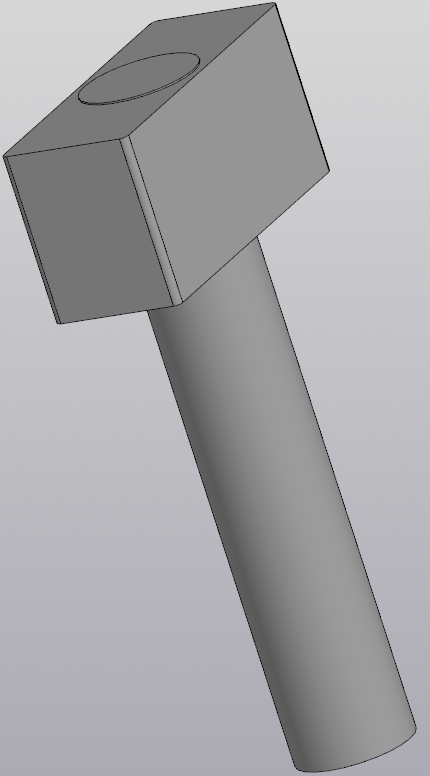


Рисунок 7.2 – Модель киянки с максимальными параметрами

## Модульное тестирование

В целях проверки корректности работы методов и свойств классов при помощи тестового фреймворка NUnit версии 3.13 проведено модульное тестирование, проверялись открытые поля, свойства и методы. Были протестированы классы модели: Parameter и MalletParameters. На рисунке 7.4 представлена информация о модульном тестировании программы. Покрытие модели тестами составило сто процентов, что показано на рисунке 7.5. Описания тестов представлены в приложении Б.

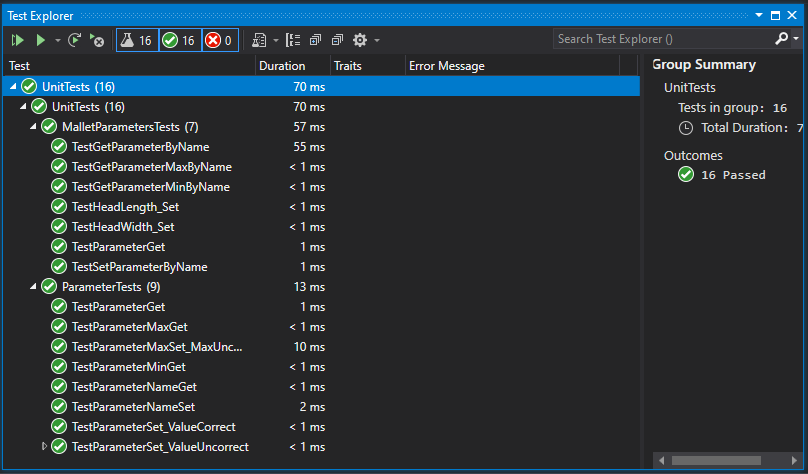


Рисунок 7.4 – Модульное тестирование плагина

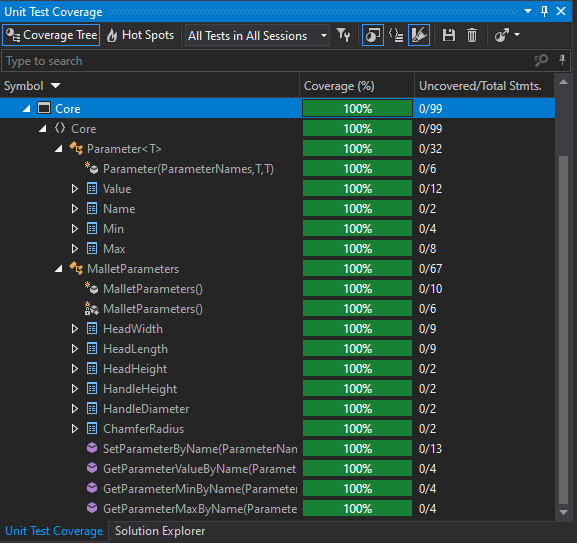


Рисунок 7.5 – Покрытие кода тестами

## Нагрузочное тестирование

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

– ЦП ntel(R) Core(TM) i5-3230M CPU 2.60GHz;

– 8 ГБ ОЗУ;

– графический процессор объемом памяти 2 ГБ.

Для нагрузочного тестирования был задан бесконечный цикл построения детали. Для измерения времени был использован класс Stopwatch. Тестирование заключалось в построении киянки с минимальными параметрами. На рисунке 7.6 представлен график зависимости времени от количества построенных деталей.

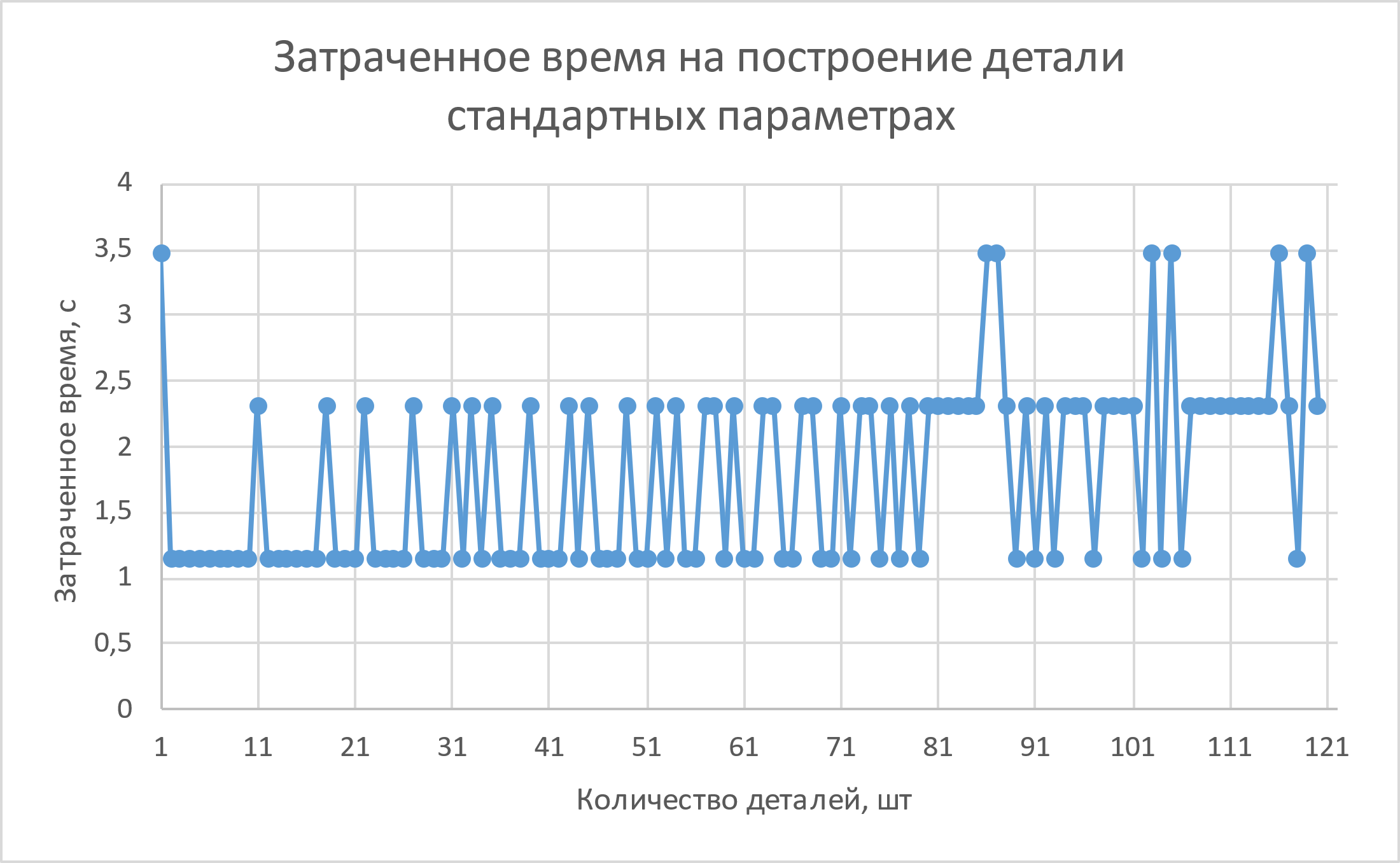


Рисунок 7.6 – График зависимости времени построения

от количества построенных киянок

На рисунке 7.7 представлен график зависимости загруженности памяти от количества построенных деталей



Рисунок 7.7 – График зависимости загруженности памяти

от количества построенных киянок

Тестирование длилось три с половиной минуты, было построено сто двадцать моделей киянки со стандартными параметрами, работа плагина завершилась сообщением об ошибки памяти в Visual Studio. Исходя из графика, представленного на рисунке 7.6 можно увидеть, что построения детали в основном занимает менее трёх секунд, но после построения восемьдесят пятой детали, время построения некоторых деталей стало возрастать, данную тенденцию можно соотнести с самым большим изменением занимаемой памяти показанное на рисунке 7.7. Резкие уменьшения потребления памяти скорее всего обусловлены тем, что КОМПАС-3D подключает алгоритмы оптимизации при достижении максимальной занимаемой оперативной памяти компьютера.

Важным фактором является использование файла подкачки, а точнее переход в виртуальную память. Виртуальная память – метод [управления памятью](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C%D1%8E) [компьютера](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), позволяющий выполнять программы, требующие больше [оперативной памяти](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C), чем имеется в компьютере, путём автоматического перемещения частей программы между основной памятью и вторичным хранилищем (например, [твердотельным](https://ru.wikipedia.org/wiki/%D0%96%D1%91%D1%81%D1%82%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA) накопителем) [11].

Также было решено провести второе нагрузочное тестирование, в процессе которого будут создаваться киянки с максимальными параметрами, в процессе построения, которых используется больше операций.

На рисунках 7.8 и 7.9 показан результат данного тестирования:

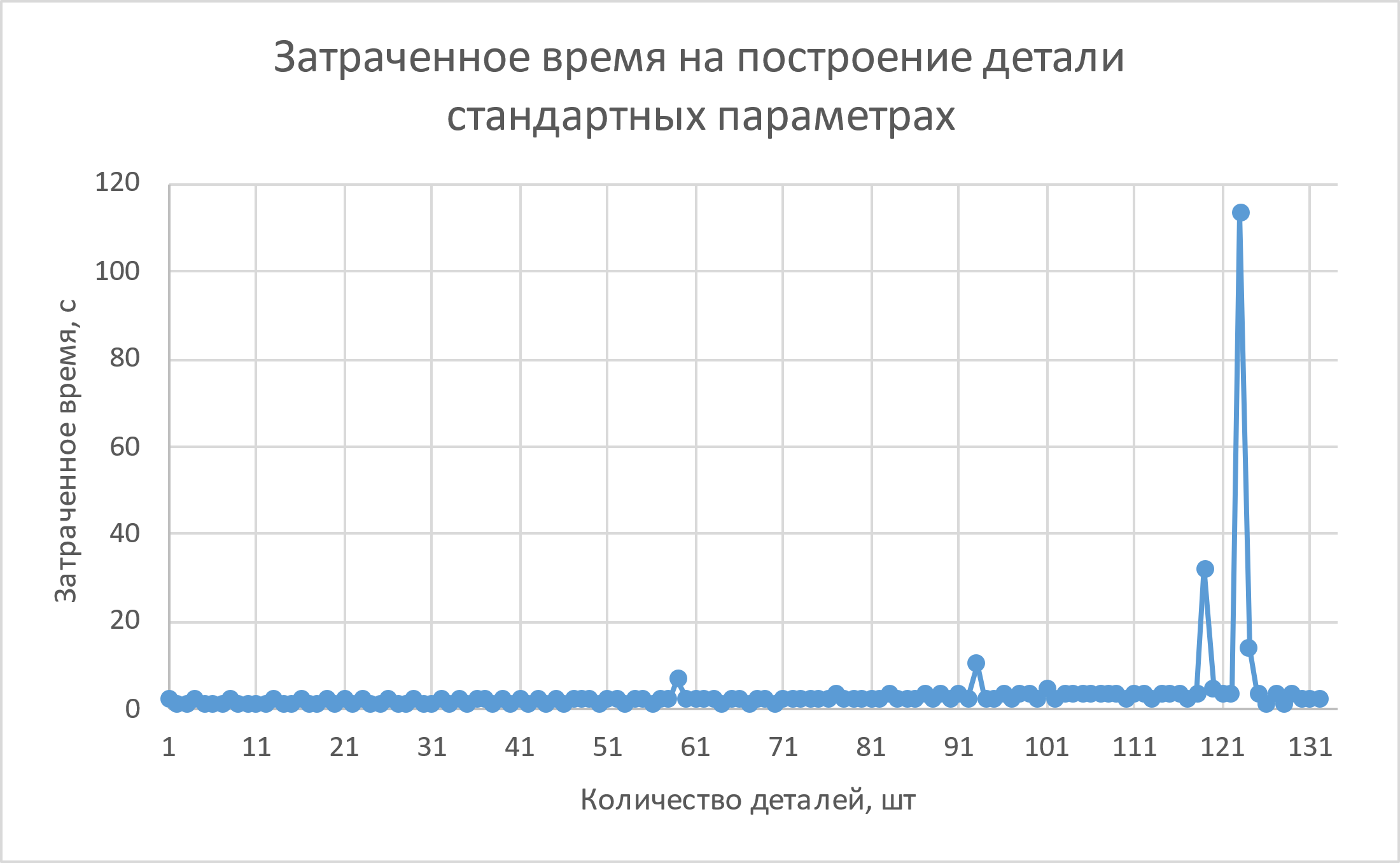


Рисунок 7.8 – График зависимости времени от

количества построенных изогнутых киянок

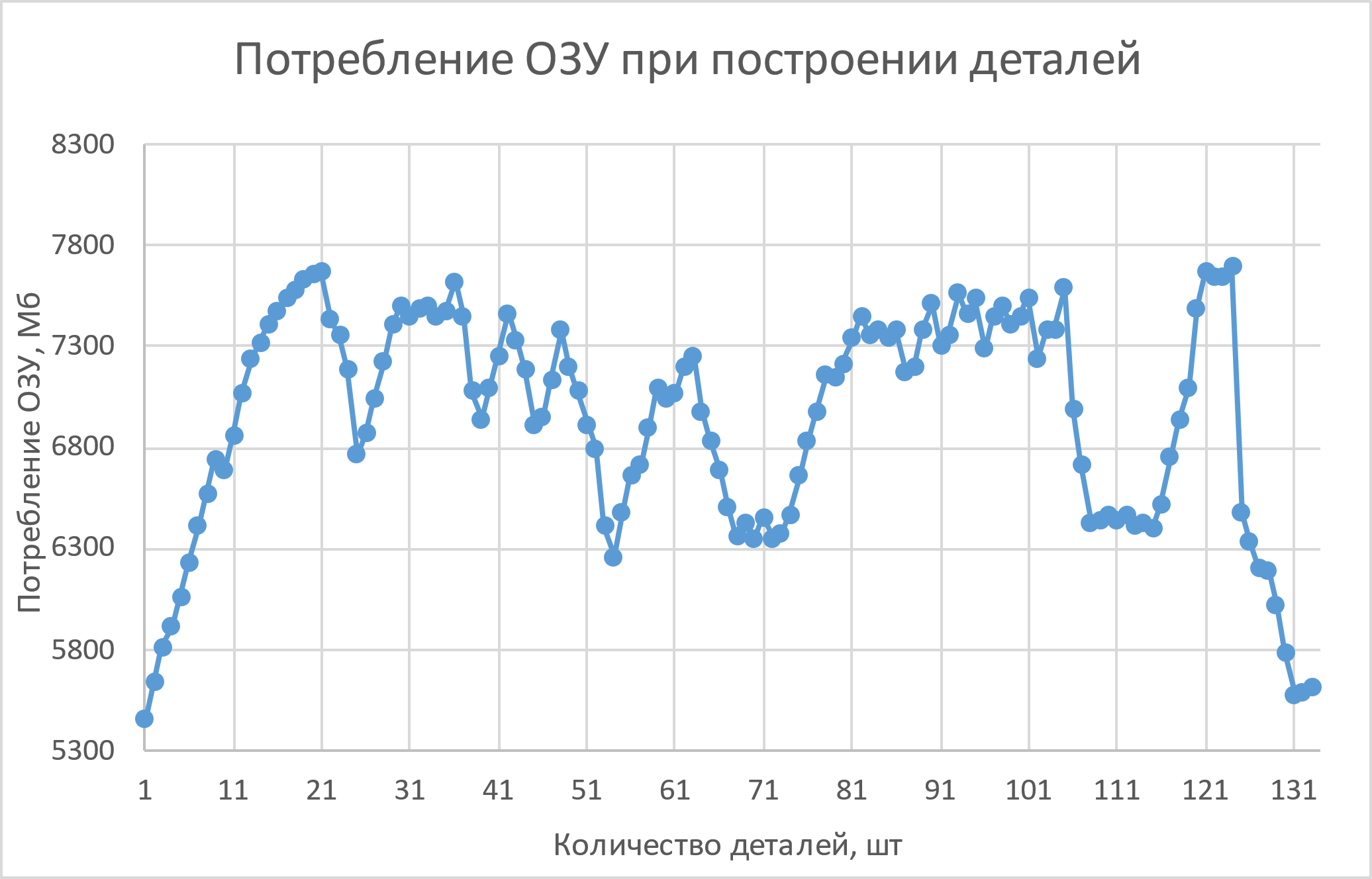


Рисунок 7.9 – График зависимости загруженности памяти

от количества построенных изогнутый киянок

Нагрузочное тестирование с построением изогнутых киянок завершилось через десять минут предварительной остановкой работы ПК с сообщением о критическом сбое работы ОС Windows.

Исходя из графиков тестирования 7.8 и 7.9, можно выделить линейное увеличение затрачиваемой памяти на построение деталей до достижения максимального занимаемого объема, после которого скорее всего началась использование файлов подкачки и работа алгоритмов оптимизации. Время построения каждой детали оставалось в диапазоне от двух до трёх с половиной секунд, однако по мере выполнения происходили скачки времени построения –первый составил около семи секунд, второй – десять, третий – тридцать две и четвертый – сто тринадцать секунд. После последнего скачка потребление памяти резко снизилось и вскоре программа завершила свою работу.

# Заключение

При выполнения лабораторных работ были изучены основные этапы проектирования программного продукта и его реализации, предметная область объекта проектирования, SDK «КОМПАС-3D». Было составлено техническое задание, разработан проект системы, составлены UML диаграммы классов, разработан макет пользовательского интерфейса.

В результате работы был разработан и реализован плагин для САПР «КОМПАС-3D», выполняющий построение 3D-модели киянки по заданным параметрам.

Над реализованным плагином были проведены функциональное, модульное и нагрузочное тестирование на платформе Windows 10.

# Список использованных источников

1. Автоматизация вычислительных процедур в прикладных задачах инженерного проектирования [Электронный ресурс]. – URL: https://scienceforum.ru/2014/article/2014000201 (дата обращения: 17.03.2022).
2. Visual Studio [Электронный ресурс]. – URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 17.03.2022).
3. . Базовые интерфейсы API системы КОМПАС [электронный ресурс]. – URL: <https://it.wikireading.ru/23741> (дата обращения 17.03.2022).
4. Библиотека «Инструмент» для «Компас-3D» [электронный ресурс]. – URL: <http://www.insoftmach.ru/Instrument.html> (дата обращения 17.03.2022).
5. Киянка [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Киянка/, свободный (дата обращения: 17.03.2022).
6. КОМПАС-3D для разработчиков. [Электронный ресурс]. – URL: https://kompas.ru/solutions/developers/ (дата обращения: 17.03.2022).
7. NUnit [Электронный ресурс]. – URL: https://nunit.org/ (дата обращения: 17.03.2022).
8. Введение в Windows Forms [Электронный ресурс]. – URL: https://metanit.com/sharp/windowsforms/1.1.php (дата обращения: 17.03.2022).
9. UML. [Электронный ресурс]. – Режим доступа: http://www.uml.org/ (дата обращения 17.03.2022).
10. Функциональное тестирование [Электронный ресурс]. – URL: https://www.software-testing.ru/library/testing/functional-testing/68--model-based- (дата обращения: 17.03.2022).
11. Виртуальная память – Википедия. [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Виртуальная память (дата обращения 17.03.2022)

# Приложение А

Таблица А.1 – Класс MainForm

|  |  |  |
| --- | --- | --- |
| Название | Тип  возвращаемых данных | Описание |
| \_malletBuilder | Mallet  Builder | Закрытое поле. Объект класса построителя |
| \_malletParameters | Mallet  Parameters | Закрытое поле. Объект класса с параметрами |
| \_textBoxesDictonary | <TextBox, ParameterNames> | Словарь хранящий пару объекта класса TextBox и имя параметра соответствующего данному TextBox |
| BuildButton(object sender, EventArgs e) |  | Обработчик нажатия кнопки “Построить” |
| TextBox\_Validated(object sender, EventArgs e) |  | Устанавливает стиль для проверенного значения |
| TextBox\_Validating(object sender, EventArgs e) |  | Общий метод проверки значения в текстбоксе |
| MainForm() | MainForm | Конструктор главной формы с необходимой инициализацей |

Таблица А.2 – Класс Parameter

|  |  |  |
| --- | --- | --- |
| Название | Тип  возвращаемых данных | Описание |
| \_min | Заданное при инициализации | Закрытое поле. Минимальное значение параметра. |
| \_max | Заданное при инициализации | Закрытое поле. Максимальное значение параметра. |
| \_value | Заданное при инициализации | Закрытое поле. Значение параметра. |
| \_name | ParameterNames | Закрытое поле. Имя параметра. |
| Name() | ParameterNames | Возвращает и устанавливает имя параметра. |
| Value() | Заданное при инициализации | Возвращает и устанавливает значение параметра, проверяется входит ли в диапазон между минимальным и максимальным. |
| Min() | Заданное при инициализации | Возвращает и устанавливает минимальное значение. |
| Max() | Заданное при инициализации | Возвращает и устанавливает максимальное значение, проверяется чтобы было больше минимального. |
| Parameter(Parameter  Names,  double,double,double) | Parameter | Конструктор параметра |

Таблица А.3 – Класс MalletParameters

|  |  |  |
| --- | --- | --- |
| Название | Тип  возвращаемых данных | Описание |
| \_chamferRadius | Parameter<int> | Закрытое поле. Параметр фаски на бойке |
| \_handleDiameter | Parameter<int> | Закрытое поле. Параметр диаметра |
| \_handleHeight | Parameter<int> | Закрытое поле. Параметр высоты ручки |
| \_headHeight | Parameter<int> | Закрытое поле. Параметр высоты бойка |
| \_headLength | Parameter<int> | Закрытое поле. Параметр длины бойка |
| \_headWidth | Parameter<int> | Закрытое поле. Параметр ширины бойка |
| \_parametersDictionary | <ParameterNames, Parameter<int>> | Словарь хранящий пару имени параметра и указатель на параметр с данным именем |
| SetValueByName(  ParameterNames,  doublу) |  | Устанавливает значение параметра по имени |
| GetValueByName (ParameterNames) | int | Возвращает значение параметра. |
| MalletParameters() |  | Конструктор параметров модели |
| ChamferRadius | int | Задаёт/возвращает фаску бойка |
| HandleDiameter | int | Задаёт/возвращает диаметр ручки |
| HandleHeight | int | Задаёт/возвращает высоту ручки |
| HeadHeight | int | Задаёт/возвращает высоту бойка |
| HeadLength | int | Задаёт/возвращает длина бойка |
| HeadWidth | int | Задаёт/возвращает ширину бойка |
| GetParameterMinByName(ParameterNames) | int | Метод возвращающий минимальное значение параметра по имени |
| GetParameterMAxByName(ParameterNames) | int | Метод возвращающий максимальное значение параметра по имени |

Продолжение Таблицы А.3

|  |  |  |
| --- | --- | --- |
| HANDLE\_HEAD\_DIFFERENCE | int = 10 | Константа разницы между диаметром ручки и шириной бойка |
| HANDLE\_LENGTH\_HEIGHT\_MULTIPLIER | int = 2 | Константа отношения длины бойка к высоте бойка |
| MAX\_CHAMFER\_RADIUS | int = 4 | Константа максимального значения фаски |
| MIN\_CHAMFER\_RADIUS | int = 1 | Константа минимального значения фаски |
| MAX\_HANDLE\_DIAMETER | int = MIN\_HEAD\_WIDTH - HANDLE\_HEAD\_DIFFERENCE | Константа максимального значения диаметра ручки |
| MIN\_HANDLE\_DIAMETER | int = 20 | Константа минимального значения диаметра ручки |
| MAX\_HANDLE\_HEIGHT | int = 400 | Константа максимального значения высоты ручки |
| MIN\_HANDLE\_HEIGHT | int = 150 | Константа минимального значения высоты ручки |
| MAX\_HEAD\_HEIGHT | int = MIN\_HEAD\_LENGTH \* HANDLE\_LENGTH\_HEIGHT\_MULTIPLIER | Константа максимального значения высоты бойка |
| MIN\_HEAD\_HEIGHT | int = 40 | Константа минимального значения высоты бойка |
| MAX\_HEAD\_LENGTH | int = 200 | Константа максимального значения длины бойка |
| MIN\_HEAD\_LENGTH | int = 100 | Константа минимального значения длины бойка |
| MAX\_HEAD\_WIDTH | int = 80 | Константа максимального значения ширины бойка |
| MIN\_HEAD\_WIDTH | int = 40 | Константа минимального значения ширины бойка |

Таблица А.4 – Класс MalletBuilder

|  |  |  |
| --- | --- | --- |
| Название | Тип  возвращаемых данных | Описание |
| \_parameters | CardanForkParameters | Закрытое поле. Хранит объект параметров киянки |
| \_connector | KompasConnector | Закрытое поле. Объект класса конектора для связи с КОМПАС-3D |
| HandleLedge | int = 1 | Константа зазора между ручкой и бойком |
| MainLineStyle | int = 1 | Константа основного стиля линии |
| BuildMallet() |  | Создаёт модель киянки, вызывает BuildMalletHandle() и BuildMalletHead() |
| BuildMalletHandle(int, int) |  | Метод строящий ручку киянки, вызывается BuildMallet() |
| BuildMalletHead(int, int, int) |  | Метод стоящий боёк киянки, вызывается в BuildMallet() |
| MalletBuilder(Mallet  Parameters, KompasConnector) | MalletBuilder | Конструктор MalletBuilder |
| DrawRectangle(double,  double, double,  double) | ksRectangleParam | Метод рисующий прямоугольник по заданным парамтерам |
| CreateExtrusion  (ksSketchDefinition,  depth, bool) |  | Метод осуществляющий выдавливание |
| CreateСгеExtrusion  (ksSketchDefinition,  depth, bool) |  | Метод осуществляющий вырезание |
| CreateSketch(Obj3dType,  double) | ksSketch  Definition | Создаёт новый эскиз в плоскости |
| CreateFillet(double,  double, double) |  | Метод создающий фаску на ребре |

Таблица А.5 – Класс KompasWrapper

|  |  |  |
| --- | --- | --- |
| Название | Тип  возвращаемых данных | Описание |
| \_kompasObject | KompasObject | Закрытое поле. Объект API КОМПАС-3D |
| \_part | ksPart | Закрытое поле. Компонент сборки |
| Object() | KompasObject | Свойство. Возвращает объект интерфейса API |
| Part() | ksPart | Свойство. Возвращает компонент сборки |
| CreareDocument3D() | ksDocument3d | Запускает окно создания 3D-модели |
| IsKompasActive  (KompasObject\*) | bool | Метод смотрит, есть ли КОМПАС-3D в таблице запущенных объектов |
| IsKompasOpen  (KompasObject\*) | bool | Метод запускающий КОМПАС-3D |
| Start() |  | Метод начала работы с КОМПАС-3D, вызывает сначала IsKompasActive() и если тот возвращает false вызывает IsKompasOpen |

# Приложение Б

Таблица Б.1 – Описание тестов MalletParametersTests.

|  |  |
| --- | --- |
| Тестовый метод | Описание тестового случая |
| TestSetParameterByName() | Тест метода передающий значение в сеттер параметра по его имени |
| TestGetParameterByName() | Тест на геттер значения параметра по имени |
| TestParameterGet() | Позитивный тест на геттеры параметров |
| TestGetParameterMinByName() | Тест на геттер минимума параметра по имени |
| TestHeadLength\_Set() | Тест на сеттер длины бойка |
| TestHeadWidth\_Set() | Тест на сеттер ширины бойка |
| TestGetParameterMaxByName() | Тест на геттер максимума параметра по имени |

Таблица Б.2 – Описание тестов ParameterTests.

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| TestParameterNameSet |  | Позитивный тест на сеттер имени параметра |
| TestParameterNameGet |  | Позитивный тест на геттер имени параметра |
| TestParameterSet\_ValueUncorrect  (double) | -1000, 1000 | Негативный тест на сеттер параметра |
| TestParameterSet\_ValueСorrect() |  | Позитивный тест на сеттер параметра |
| TestParameterGet() |  | Позитивный тест на геттер параметра |
| TestParameterMaxSet\_MaxUncorrect  (double) | -1 | Негативный тест на сеттер максимума |
| TestParameterMaxGet() |  | Позитивный тест на геттер максимума |
| TestParameterMinGet() |  | Позитивный тест на геттер максимума |