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)

**Плагин для создания настольной лампы, для САПР КОМПАС-3D v18.1**

Пояснительная записка

по дисциплине «Основы разработки САПР»

Выполнил

Студент гр. 587-3

\_\_\_\_\_\_\_\_\_\_\_ Е.А. Краснов

\_\_\_\_\_\_\_\_\_

Проверил:

Доцент кафедры КСУП, к.т.н.

\_\_\_\_\_\_\_\_\_\_\_ А.А. Калентьев

\_\_\_\_\_\_\_\_\_\_

Томск 2021

**Оглавление**

1 Описание Компас-3D 3

2 Описание API 4

3 Обзор аналогов 7

4 Описание предмета проектирования 8

5 Диаграмма классов 11

6 Макет пользовательского интерфейса 13

7 Функциональное тестирование 15

8 Модульное тестирование 19

9 Нагрузочное тестирование 21

10 Заключение 25

Список литературы 26

Приложение А 27

**1 Описание Компас-3D**

«Компас» — семейство систем автоматизированного проектирования, универсальная система автоматизированного проектирования, позволяющая в оперативном режиме выпускать чертежи изделий, схемы, спецификации, таблицы, инструкции, расчетно-пояснительные записки, технические условия, текстовые и прочие документы. Изначально система ориентирована на оформления документации в соответствии с ЕСКД, ЕСТД, СПДС и международными стандартами, но этим возможности системы не ограничиваются.

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

Система «Компас-3D» включает следующие компоненты: система трёхмерного твердотельного моделирования, универсальная система автоматизированного проектирования «Компас-График» и модуль формирования спецификаций. Ключевой особенностью «Компас-3D» является использование собственного математического ядра и параметрических технологий. [1]

**2 Описание API**

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

В КОМПАС-3D существуют API двух версий: API 5 и API 7. Обе версии реализуют различные функции системы и дополняют друг друга. Обе версии программных интерфейсов в равной мере поддерживаются и развиваются с учетом самих изменений в системе. В основном, для создания полноценных подключаемых модулей достаточно методов и свойств интерфейсов API 5.

Главным интерфейсом API системы КОМПАС-3D является KompasObject. Получить указатель на этот интерфейс можно при работе под управлением внешнего приложения (контроллера) – после вызова стандартной системной функции. Методы этого интерфейса реализуют наиболее общие функции работы с документами системы, системными настройками, файлами, а также дают возможность получить указатели на другие интерфейсы [2].

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

Таблица 1.1. – Методы интерфейса KompasObject.

|  |  |  |
| --- | --- | --- |
| Метод | Возвращаемое значение | Описание |
| Document3D() | Указатель на интерфейс документа трёхмерной модели ksDocument3D | Даёт возможность получить указатель на интерфейс трёхмерного документа (детали или сборки) |
| Visible |  | Свойство видимости приложения |
| Quit |  | Метод для завершения программы Kompas-3D |
| ActivateControllerAPI |  | Метод для активации контроллера API |
| ksDocument2D |  | Интерфейс событий графического документа, события интерфейса позволяют контролировать состояние документа. |

Таблица 1.2 — Методы интерфейса IPart.

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| GetDefaultEntity  (short objType) | |  |  | | --- | --- | | objType | - тип объекта. | | |  | | --- | | Указатель на интерфейс [ksEntity](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntity.htm) или [IEntity](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntity.htm). | | Получить указатель на интерфейс объекта, создаваемого системой по умолчанию |
| GetPart(int type) | |  |  | | --- | --- | | type | - тип компонента. | | указатель на интерфейс компонента [ksPart](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksPart.htm) или [IPart](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksPart.htm). | Получить указатель на интерфейс компонента в соответствии с заданным типом |
| NewEntity(short objType) | |  |  | | --- | --- | | ob | jType- [тип объекта](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/Obj3dType_NewEntil_Part.htm). | | указатель на интерфейс [ksEntity](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntity.htm) или [IEntity](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntity.htm). | Создать новый интерфейс объекта и получить указатель на него |

Таблица 1.3 — 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. | Дает возможность создать пустой документ (деталь или сборку) |
| GetPart(int type) | type – тип компонента из перечисления Типы компонентов. |  | Получить указатель на интерфейс компонента в соответствии с заданным типом |

Таблица 1.4 – Методы интерфейса ksDocument2D

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| ksCircle (double xc, double yc, double rad, long style) | xc, yc - координаты центра окружности, rad - радиус окружности, style - стиль линии. | указатель на окружность - в случае удачного завершения, 0 - в случае неудачи. | Метод для создания окружности. |

Продолжение таблицы 1.4

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| ksCircle (double xc, double yc, double rad, long style) | xc, yc - координаты центра окружности, rad - радиус окружности, style - стиль линии. | указатель на окружность - в случае удачного завершения, 0 - в случае неудачи. | Метод для создания окружности. |
| long ksLineSeg (double x1, double y1, double x2, double y2, long style) | x1, y1 - координаты первой точки отрезка, x2, y2 - координаты второй точки отрезка, style - стиль линии. | указатель на отрезок - в случае удачного завершения, 0 - в случае неудачи. | Метод для создания отрезка. |
| ksArcByPoint (double xc, double yc, double rad, double f1, double f2, short direction, long style); | xc, yc - координаты центра окружности, rad - радиус окружности, x1, y1 - координаты начальной точки дуги, x2, y2 - координаты конечной точки дуги, direction - направление отрисовки дуги: 1 - против часовой стрелки, -1 - по часовой стрелке, style - стиль линии. | указатель на отрезок - в случае удачного завершения, 0 - в случае неудачи. | Метод создает дугу по центру и конечным точкам |

Таблица 1.5 —Методы интерфейса [ksEntity](mk:@MSITStore:C:\Program%20Files\ASCON\KOMPAS-3D%20v18%20Study\SDK\SDK.chm::/ksEntity_props.htm).

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| ksBossExtrusionDefinition(BOOL forward, short type, double depth, double draftValue, BOOL draftOutward); | Forward- - направление выдавливания: TRUE - прямое направление, FALSE - обратное направление, type - тип выдавливания, depth - глубина выдавливания, draftValue - угол уклона, draftOutward - направление уклона: FALSE - уклон наружу, TRUE - уклон внутрь. | TRUE – в случае успешного завершения. | Метод выдавливает эскиз в одном направлении |
| ksCutExtrusionDefinition (BOOL forward, short type, double depth, double draftValue, BOOL draftOutward); | Forward- - направление выдавливания: TRUE - прямое направление, FALSE - обратное направление, type - тип выдавливания, depth - глубина выдавливания, draftValue - угол уклона, draftOutward - направление уклона: FALSE - уклон наружу, TRUE - уклон внутрь. | TRUE – в случае успешного завершения. | Метод вырезания выдавливанием эскиз в одном направлении |

**3 Обзор аналогов**

Dialux —одна из самых функциональных компьютерных программ для выполнения светотехнических расчетов и инженерного проектирования внутреннего и внешнего освещения.

По изначально заданным условиям: количество светильников, их тип, расположение, - программа Dialux способна проводить разнообразные сложные светотехнические расчеты, при которых обязательно будут учтены все факторы, связанные с мебелью, различными предметами интерьера, геометрией помещения, цветом и текстурой всех поверхностей. Программа позволяет проводить расчеты для любых видов освещенности, КЕО, яркости, показателей блесткости, теней и дневного света. Утилита учитывает погодные условия, географическое расположение объекта, тени от окружающих объектов и зданий.

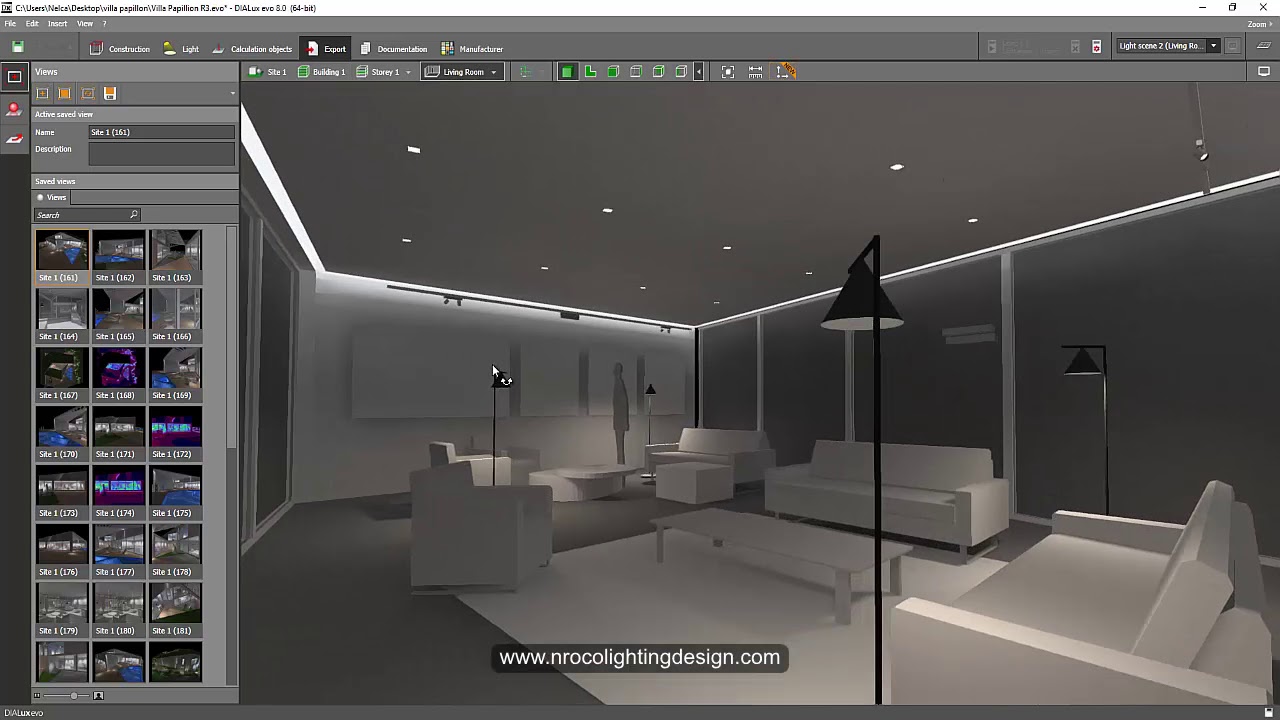
Для каждого участка можно смоделировать разнообразные световые эффекты с зеркальным отражением, прозрачностью, улучшенными текстурами по заданным параметрам. Модернизированное ядро программы делает просчеты сцен быстрыми, а цвета более естественными и насыщенными.

Программой полностью поддерживаются все современные национальные и международные стандарты и европейские единицы измерения. Можно экспортировать-импортировать объекты и данные в и из любых CAD-программ в форматах .dwg и .dxf. Наличие всплывающих подсказок и интуитивно понятное управление сильно облегчает работу с программой.

Программа DIALux значительно упрощает процесс расчета системы общего искусственного освещения помещений с трехмерной визуализацией проектных решений. Программа интерактивна: она позволяет пользователю перемещаться в освещенном, в соответствии с расчетом, интерьере.[3]

Интерфейс программы DIALux представлен на рисунке 3.1.

Рисунок 3.1 — Программа DIALux



**4 Описание предмета проектирования**

Настольные лампы служат для освещения определенной поверхности рабочего стола. Чаще всего их устанавливают для школьников, а также в рабочих кабинетах для создания качественного рабочего освещения в вечернее время, так как естественного света уже не хватает. Общего освещения комнаты также недостаточно. Кроме этого, настольные приборы освещения необходимы для вязания, вышивания и шитья. [4]

Плагин, предназначен для создания светильников заранее определенного дизайна (рисунок 4.1 – 4.3). Плагин должен:

1. Обеспечить изменение размера основания , .
2. Обеспечивать изменение размера у стойки , .
3. Обеспечивать изменение размера у площадки на которой закреплен патрон , .
4. Обеспечивать создание выреза под кнопку включения , , .
5. Обеспечивать создание выреза под электрический провод , , .
6. Обеспечивать создание отверстия под электрический провод в центре основания , .
7. Обеспечивать создание 2 отверстий под саморезы с расстоянием , и диаметром .

Рисунок 4.1 — Изображение настольной лампы

Рисунок 4.2 — Изображение настольной лампы

Рисунок 4.3 — Изображение настольной лампы

**5 Диаграмма классов**

Диаграмма классов описывает типы объектов системы и различного рода статические отношения, которые существуют между ними. На диаграммах классов отображаются также свойства классов, операции классов и ограничения, которые накладываются на связи между объектами.[5] Диаграмма классов приведена на рисунке 5.1.

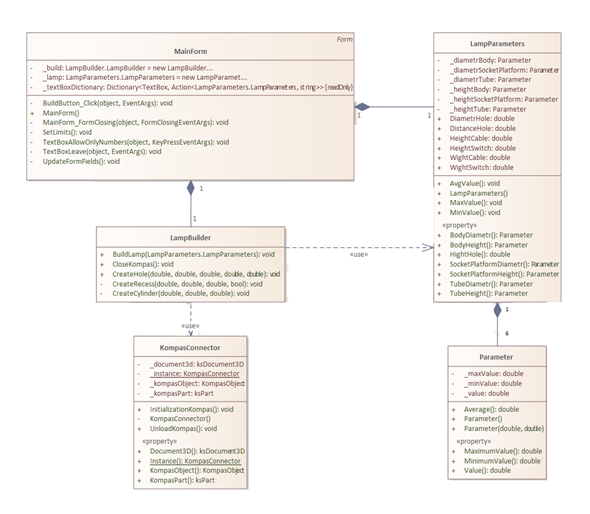


Рисунок 5.1 – Диаграмма классов

Для реализации был выбран следующий набор классов:

1. MainForm – класс диалогового окна, который обеспечивает взаимодействие между пользователем и программой;
2. Lamp − класс, хранящий в себе все параметры проектируемой 3D-модели;
3. Parameter – класс хранящий значение параметра, максимальное и минимальное ограничение.
4. KompasConnector – класс для работы с API КОМПАС 3D.
5. LampBuilder – класс, осуществляющий вызов методов API, необходимых для постройки 3D-модели.

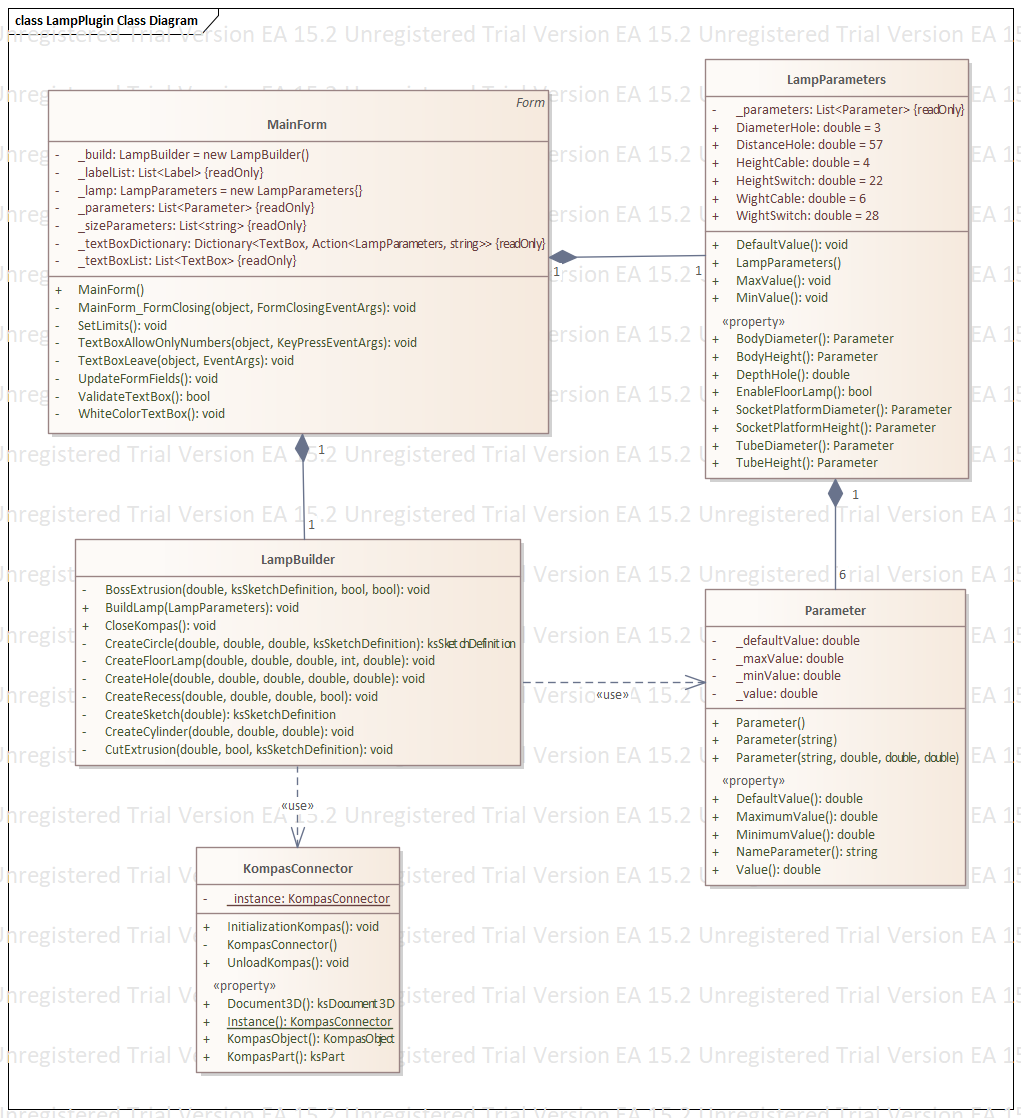
Во время работы над плагином были внесены изменения в MainForm, были добавлены листы Label, TextBox, варианты размера параметров, перечисления названий параметров, для корректной обработки данных. Добавлен метод ValidateTextBox, для сверки значений в TextBox и присвоенных параметров. Метод WhiteColorTextBox, добавлен для того, чтобы задать белый цвет TextBoxу.

В классе LampParameters, добавлено автосвойство для создания торшера.

В классе LampBuilder, добавлены методы CreateCircle, для создания окружности, CreateFloorLamp, для создания торшера, CreateSketch, для создания эскиза, BossExtrusion, для выдавливания эскиза, CutExtrusion, для вырезания выдавливанием.

В классе KompasConnector, поля заменены на автосвойства.

Диаграмма классов, после реализации добавлена на рисунке 5.2

Рисунок 5.2 – Диаграмма классов

**6 Макет пользовательского интерфейса**

Макет пользовательского интерфейса представляет собой форму для ввода параметров. При запуске программы в полях для ввода параметров находятся среднеарифметические значения. Пользователь может менять данные параметры (рисунок 6.1).

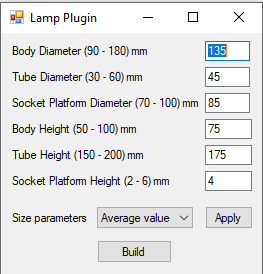


Рисунок 6.1 — Макет пользовательского интерфейса

При вводе некорректных данных всплывает окно с описанием ошибки (рисунок 6.2).

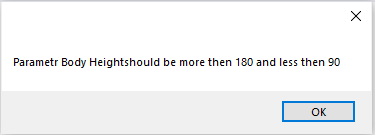


Рисунок 6.2 — Сообщение об ошибке

Поле, где было введено некорректное значение изменит цвет на красный (рисунок 6.3).

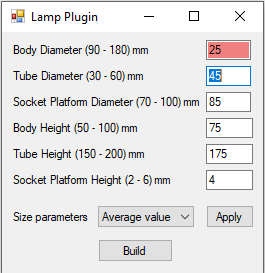


Рисунок 6.3 — Поле с некорректным параметром

Плагин предоставляет возможность для выбора предустановленных значений, а именно минимальных, средних, максимальных. В выпадающем списке можно выбирать предустановленные значения и при нажатии кнопки Apply происходит присваивание значений в поле для ввода значений (рисунок 6.4).

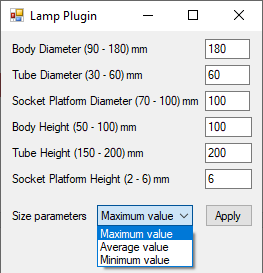


Рисунок 6.4 — Выбор предустановленных значений

**7 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. В данном случае будет проверяться правильность построения детали при различных входных параметрах

Построение модели с минимальными входными данными (рисунки 7.1, 7.2):

1. Высота основания ,
2. Диаметр основания .
3. Высота стойки ,
4. Диаметр стойки .
5. Высота площадки, на которой закреплен патрон .
6. Диаметр площадки, на которой закреплен патрон

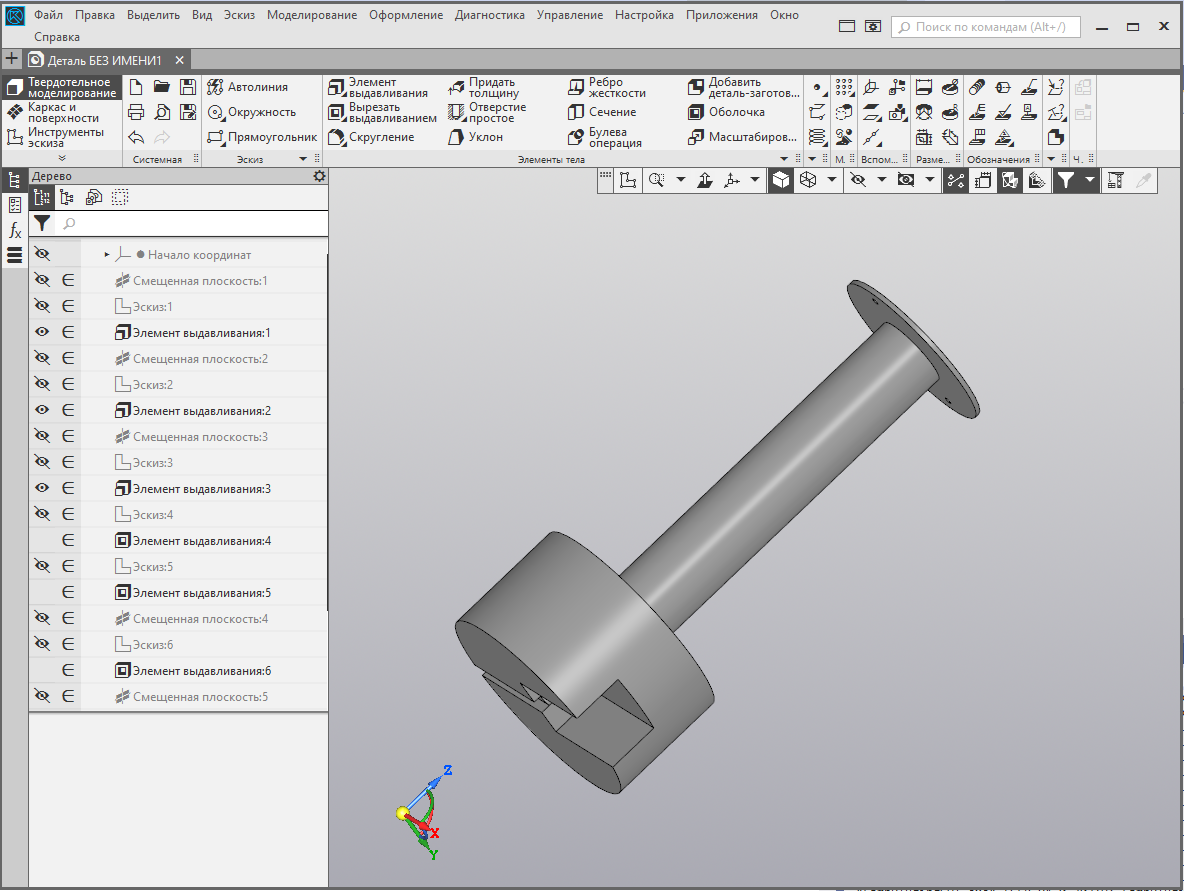
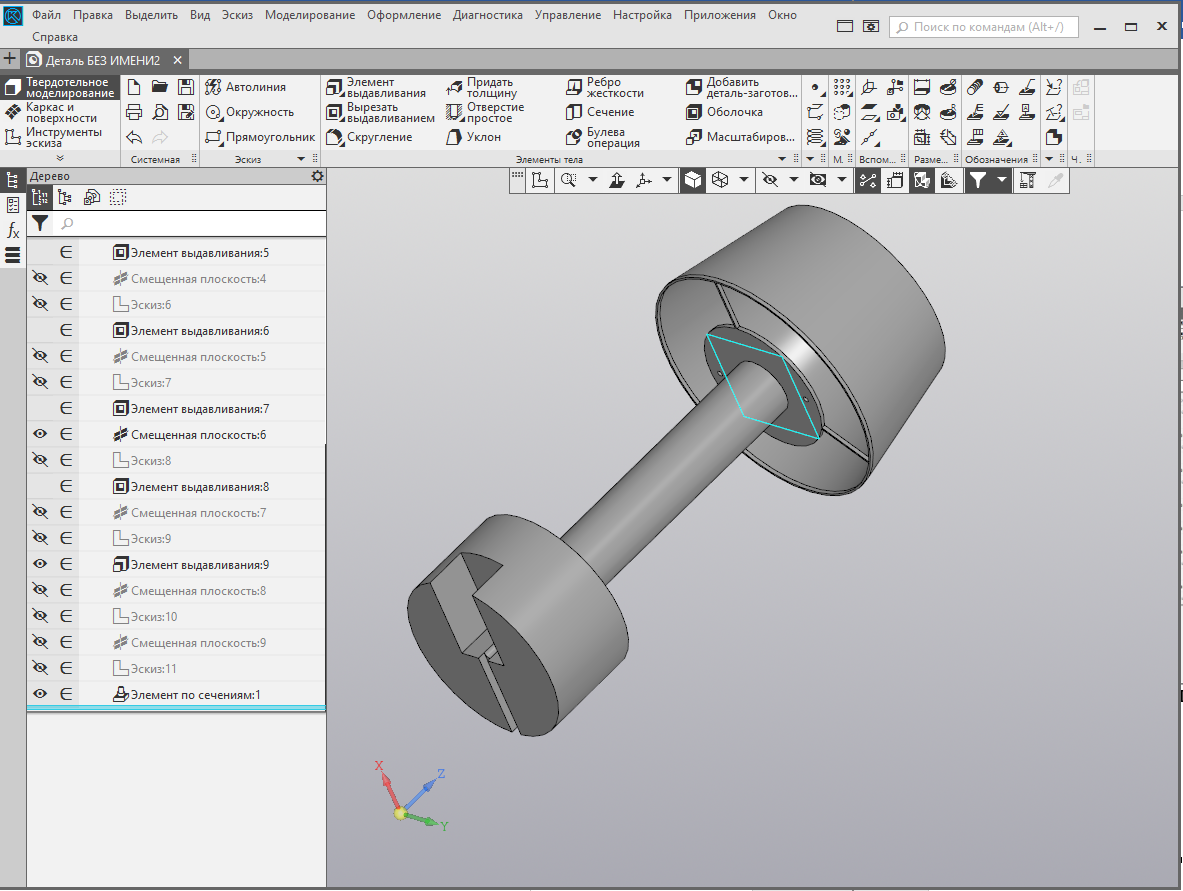
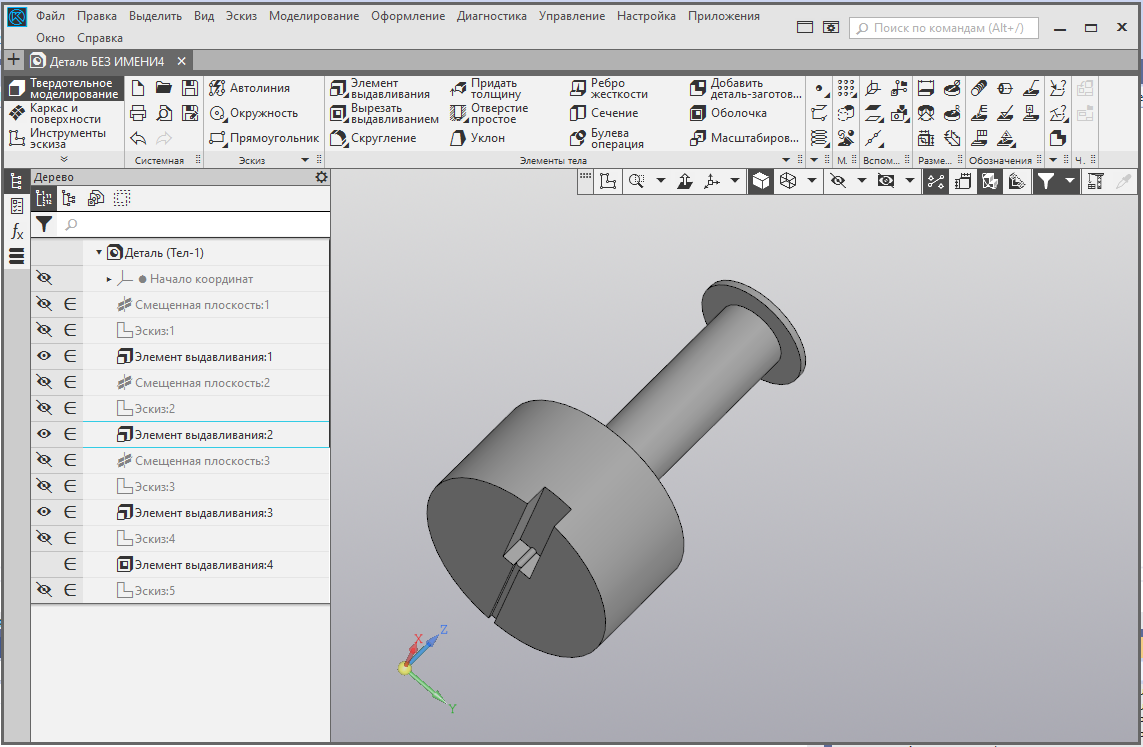
Рисунок 7.1 — Модель светильника, построенная с минимальными входными параметрами

Рисунок 7.2 — Модель светильника с торшером, построенная с минимальными входными параметрами

Построение модели с максимальными входными данными (рисунки 7.3, 7.4):

1. Высота основания ,
2. Диаметр основания .
3. Высота стойки ,
4. Диаметр стойки .
5. Высота площадки, на которой закреплен патрон .
6. Диаметр площадки, на которой закреплен патрон

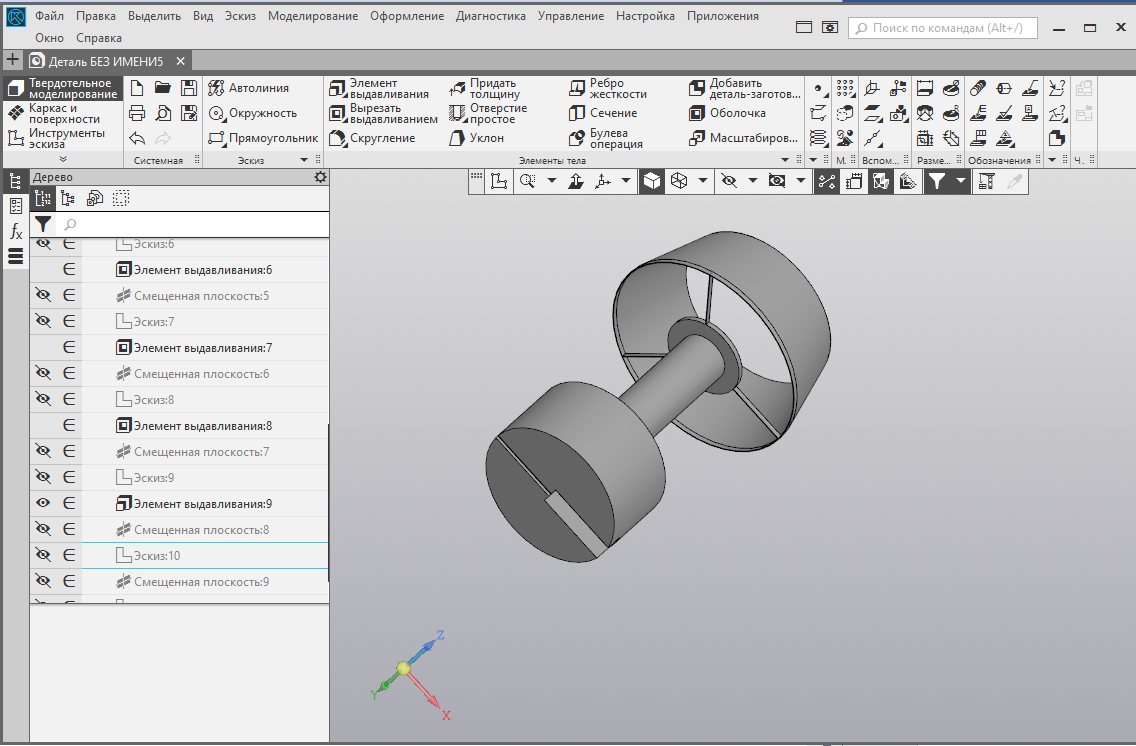
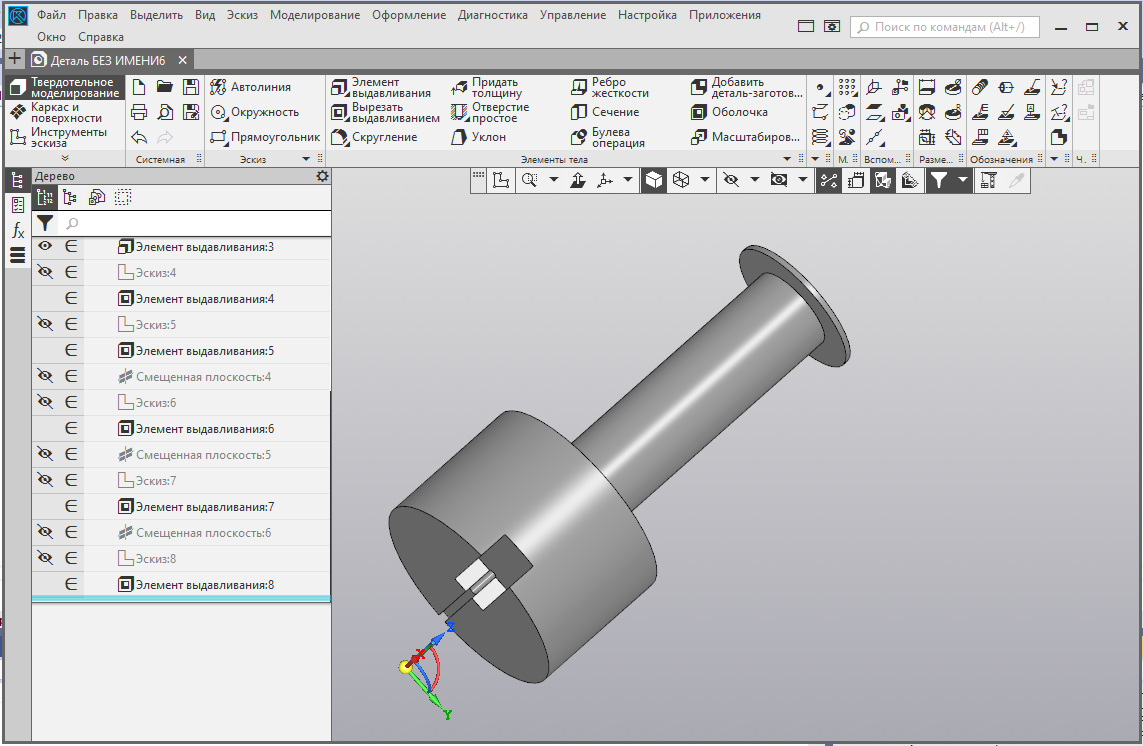
Рисунок 7.3 — Модель светильника, построенная с максимальными входными параметрами

Рисунок 7.4 — Модель светильника с торшером, построенная с максимальными входными параметрами

Построение модели с параметрами по умолчанию(рисунки 7.5, 7.6):

1. Высота основания ,
2. Диаметр основания .
3. Высота стойки ,
4. Диаметр стойки .
5. Высота площадки, на которой закреплен патрон .
6. Диаметр площадки, на которой закреплен патрон



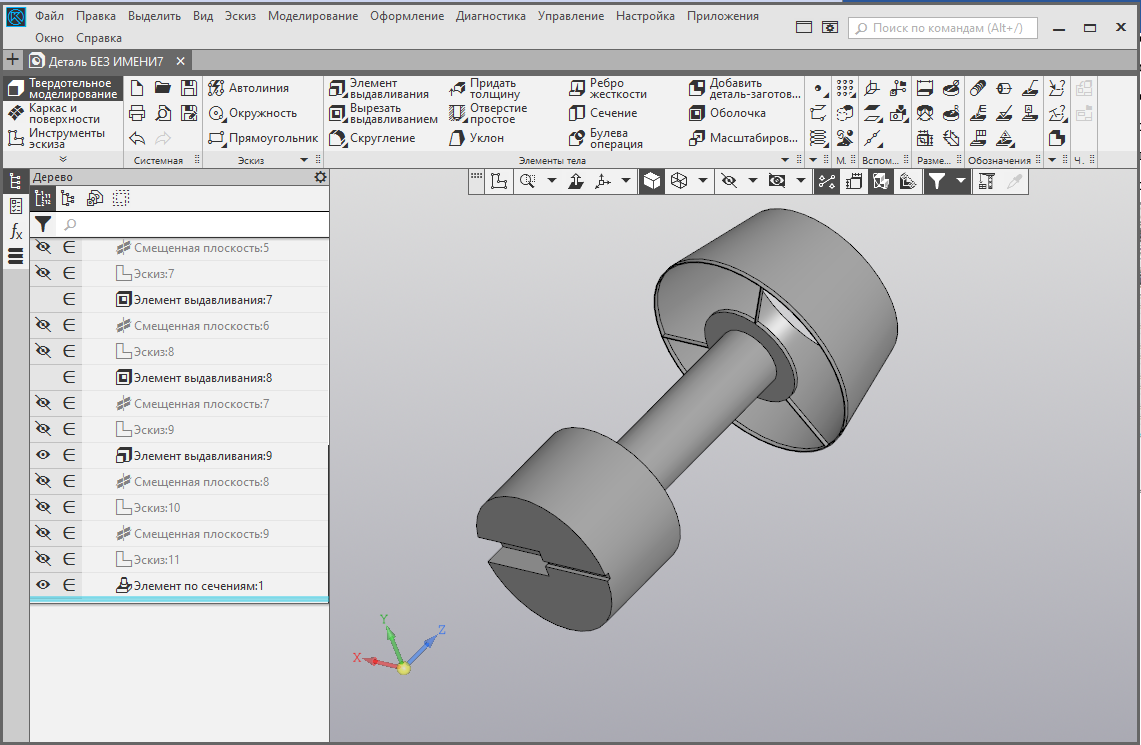
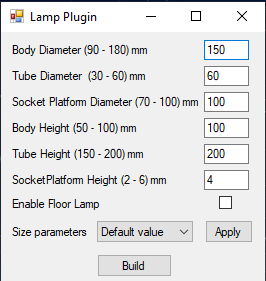
Рисунок 7.5 — Модель светильника, построенная с параметрами по умолчанию

Рисунок 7.6 — Модель светильника с торшером, построенная параметрами по умолчанию

При вводе заведомо некорректных данных программа сообщает об ошибках (рисунки 7.7, 7.8)

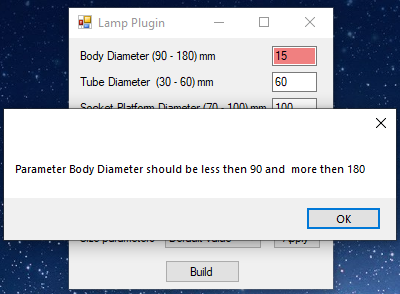
Рисунок 7.7 — Ввод некорректных параметров для основания диаметра

Рисунок 7.8 — Вывод сообщения о некорректном значении

**8 Модульное тестирование**

Юнит-тестирование (англ. «*unit*-*testing*», или модульное тестирование) — тестирование отдельного элемента изолированно от остальной системы. Относительно парадигмы объектно-ориентированного программирования системой является вся программа, а отдельным элементом — класс или его метод. Юнит-тестирование предназначено для проверки правильности работы отдельно взятого элемента. Чтобы исключить из результатов тестирования влияние потенциальных ошибок других элементов, тестируемый элемент должен быть максимально изолирован, то есть не использовать объекты и методы других классов [6].

На основе таблицы приведенных в приложении А тестовых сценариев (таблица А.1), проводилось тестирование корректности входных параметров 3D-модели.

Тестирование проводилось с помощью фреймворка модульного тестирования NUnit 3.13.1 для языков платформы .NET.

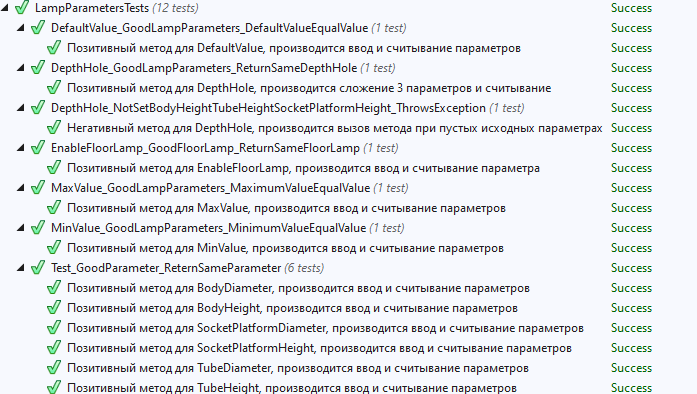
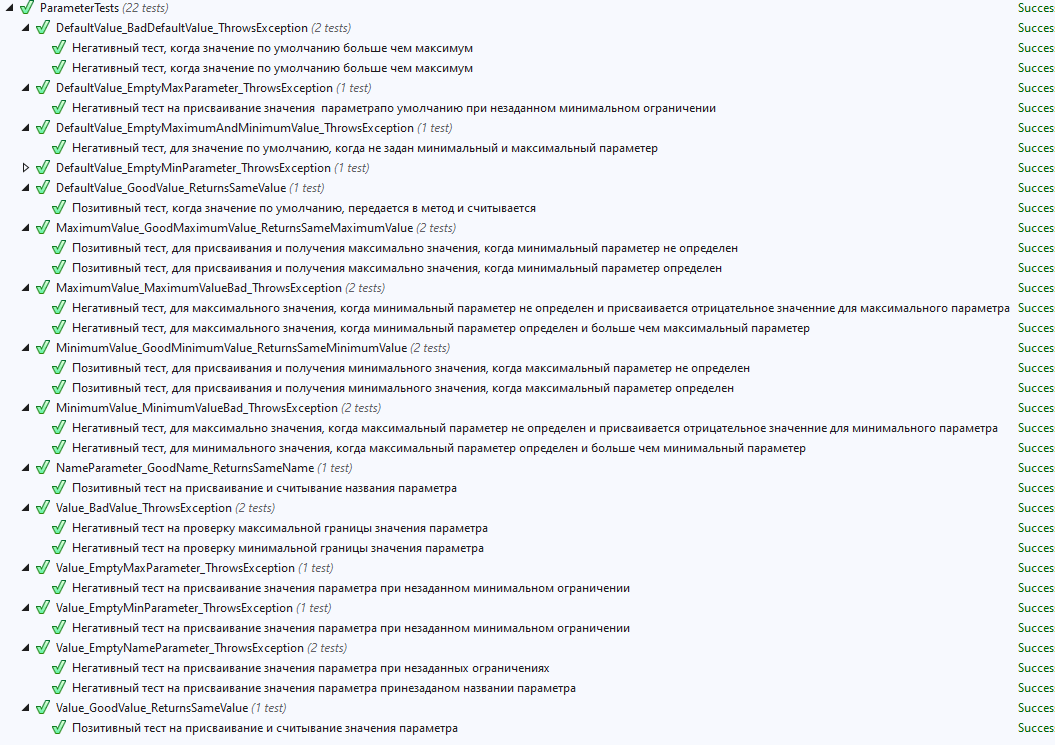
Результаты прохождения всех модульных тестов приведены на рисунках 8.1 и 8.2.

Рисунок 8.1 – Список пройденных юнит-тестов для класса LampParameters

Рисунок 8.2 – Список пройденных юнит-тестов для класса Parameters

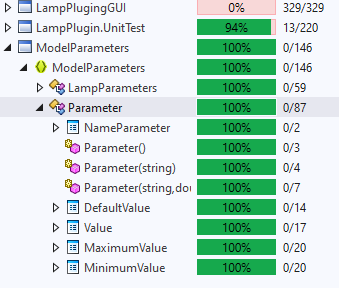
Результат покрытия моделей Parameter и LampParameters тестами приведен на рисунке 8.3. Цикломатическая сложность для класса LampParameters равна 20, а для класса Parameters 26.

Рисунок 8.3 – Результат покрытия тестами

**9 Нагрузочное тестирование**

Нагрузочное тестирование – тип тестирования, который позволяет оценить поведение системы при возрастающей нагрузке, целью нагрузочного тестирования является также определение максимальной нагрузки, которую может выдержать система [7].

Нагрузочное тестирование проводилось на компьютере со следующими характеристиками:

* Процессор: Intel Core I5-4690K 4.20 GHz;
* Оперативная память: 24558MB DDR3 1800 MHz;
* Графический процессор: NVIDIA GeForce 750 Ti;
* Операционная система: Windows 10 Pro 64-bit.

При задании максимального количества последовательно строящихся моделей равным 200, САПР «КОМПАС-3D» остановила дальнейшие построения после 197 модели, окно программы перестало реагировать на ввод. В диспетчере задач у процесса запущенной САПР отображался статус «Не работает». Исходя из этого, для выполнения нагрузочного тестирования, количество построенных деталей было принято равным 197.

Зависимость потребления оперативной памяти САПР «КОМПАС-3D» при построенных 197 моделях представлена на графике (рисунке 9.1).

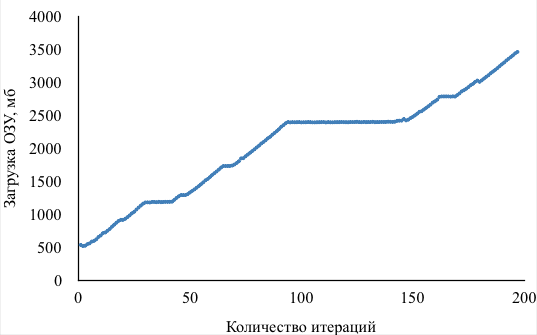
Рисунок 9.1 – Зависимость потребление ОЗУ от номера итерации

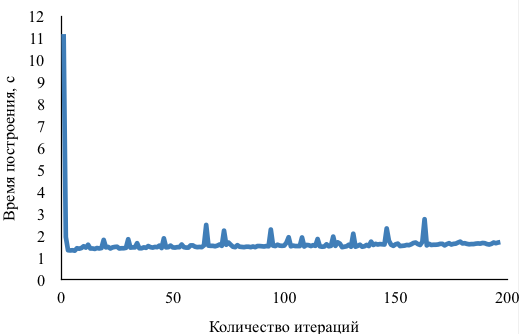
Потребление памяти резко возросло после построения 23 моделей. Пик потребления был достигнут на 29 построении и составил 810 МБ. При дальнейших построениях количество используемой ОЗУ уменьшается, при построении падает с 704 МБ до 554 MB, затем снова постепенно возрастает, не превышая 700 МБ.

График на рисунке 9.2 отображает зависимость застрачиваемого времени на каждое новое построение от количества уже построенных моделей. С начала время построения модели максимально, так как идет запуск САПР «КОМПАС-3D».

Рисунок 9.2 – Зависимость затрат времени на построение от номера итерации

Возможно, что такое потребление оперативной памяти связано с использованием 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, [жёстким диском](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))[8]. Для проверки данной гипотезы в Windows 10 был отключен файл подкачки и заново запущенно нагрузочное тестирование (рисунки 9.3 и 9.4).

Рисунок 9.3 – Зависимость потребление ОЗУ от номера итерации

Рисунок 9.4 – Зависимость затрат времени на построение от номера итерации

Из графика загрузки ОЗУ можно видеть, что он почти на протяжении всей длинны имеет линейны вид, а понижение потребления оперативной памяти, скорее всего, связано с переносом данных на внутренний накопитель данных. Время построения скорее всего имеет такие пики потребления из-за того-что, в процессе проведения тестирования были запущенны другие процессы, которые мешали работе Компаса.

**10 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 настольной лампы по задаваемым параметрам.

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

**Список литературы**

1. Компас (САПР) — Википедия. [Электронный ресурс] — Режим доступа: <https://ru.wikipedia.org/wiki/Компас_(САПР)> (дата обращения 25.03.2021)
2. Кидрук Максим. КОМПАС-3D V10 на 100% / М. Кидрук. – СПб.: Питер, 2009 – 560 с.
3. Программа Dialux для расчёта и проектирования освещения — Школа для электрика. [Электронный ресурс] — Режим доступа: http://electricalschool.info/main/lighting/1703-programma-dialux-dlja-raschjota-i.html (дата обращения 25.03.2021)
4. Настольные лампы. Виды и применение. Как выбрать и особенности — Электросам.Ру. [Электронный ресурс] — Режим доступа: https://electrosam.ru/glavnaja/jelektroobustrojstvo/osveshhenie/nastolnye-lampy/ (дата обращения 25.03.2021)
5. М. Фаулер. UML. Основы, 3-е издание. — Пер. с англ. — СПб: символ-Плюс, 2004– 192 с.
6. Новые технологии в программировании : учебное пособие / А.А.Калентьев, Д.В.Гарайс, А.Е.Горяинов – Томск : Эль Контент, 2014.—176 с.
7. Нагрузочное тестирование vs Тестирование производительности. [Электронный ресурс]. – https://performance-lab.ru/blog/load-testing/testirovanie-proizvoditelnosti (дата обращения 09.04.2021)
8. Виртуальная память — Википедия. [Электронный ресурс] — Режим доступа: [https://ru.wikipedia.org/wiki/Виртуальная память](https://ru.wikipedia.org/wiki/Виртуальная%20память) (дата обращения 17.04.2021)

# Приложение А

(справочное)

Таблица А.1 – Тестовые сценарии

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| NameParameter\_GoodName\_ReturnsSameName() |  | Позитивный тест на присваивание и считывание названия параметра |
| Value\_GoodValue\_ReturnsSameValue() |  | Позитивный тест на присваивание и считывание значения параметра |
| Value\_BadValue\_ThrowsException(double sourceValue) | 9.5 | Негативный тест на проверку минимальной границы значения параметра |
| 60.5 | Негативный тест на проверку максимальной границы значения параметра |
| Value\_EmptyNameParameter\_ThrowsException(string sourceName) | "" | "Негативный тест на присваивание значения параметра при незаданном названии параметра |
| "width" | Негативный тест на присваивание значения параметра при незаданных ограничениях |
| Value\_EmptyMaxParameter\_ThrowsException() |  | "Негативный тест на присваивание значения параметра при незаданном минимальном ограничении" |
| Value\_EmptyMinParameter\_ThrowsException() |  | Негативный тест на присваивание значения параметра при незаданном минимальном ограничении |
| MaximumValue\_GoodMaximumValue\_ReturnsSameMaximumValue(double sourceMinimumValue, double sourceMaximumValue) | 0, 50 | Позитивный тест, для присваивания и получения максимально значения, когда минимальный параметр не определен |
| 4, 50 | Позитивный тест, для присваивания и получения максимально значения, когда минимальный параметр определен" |

Продолжение таблицы А.1

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| MaximumValue\_MaximumValueBad\_ThrowsException(double sourceMinimumValue, double sourceMaximumValue) | 100, 50 | Негативный тест, для максимального значения, когда минимальный параметр определен и больше, чем максимальный параметр |
| 0, -1 | Негативный тест, для максимального значения, когда минимальный параметр не определен и присваивается отрицательное значение для максимального параметра" |
| MinimumValue\_GoodMinimumValue\_ReturnsSameMinimumValue(double sourceMinimumValue, double sourceMaximumValue) | 50, 0 | Позитивный тест, для присваивания и получения минимального значения, когда максимальный параметр не определен |
| 20, 50 | Позитивный тест, для присваивания и получения минимального значения, когда максимальный параметр определен" |
| MinimumValue\_MinimumValueBad\_ThrowsException(double sourceMinimumValue, double sourceMaximumValue) | 100, 50 | Негативный тест, для минимального значения, когда максимальный параметр определен и больше чем минимальный параметр |
| -1, 0 | Негативный тест, для максимально значения, когда максимальный параметр не определен и присваивается отрицательное значение для минимального параметра |
| DefaultValue\_GoodValue\_ReturnsSameValue() |  | Позитивный тест, когда значение по умолчанию, передается в метод и считывается |
| DefaultValue\_BadDefaultValue\_ThrowsException(double sourceDefaultValue) | 20 | Негативный тест, когда значение по умолчанию больше, чем максимум |
| 0.5 | Негативный тест, когда значение по умолчанию больше, чем максимум |
| DefaultValue\_EmptyMaximumAndMinimumValue\_ThrowsException() |  | Негативный тест для значения по умолчанию, когда не задан минимальный  и максимальный параметр |

Продолжение таблицы А.1

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| DefaultValue\_EmptyMaxParameter\_ThrowsException() |  | Негативный тест на присваивание значения параметра по умолчанию  при незаданном минимальном ограничении |
| DefaultValue\_EmptyMinParameter\_ThrowsException() |  | "Негативный тест на присваивание значения параметра по умолчанию" при незаданном минимальном ограничении" |
| Test\_GoodParameter\_ReternSameParameter(string nameParameter) | BodyDiameter | Позитивный метод для BodyDiameter, производится ввод и считывание параметров |
| BodyHeight | Позитивный метод для BodyHeight, производится ввод и считывание параметров |
| SocketPlatformDiameter | Позитивный метод для SocketPlatformDiameter производится ввод и считывание параметров |
| SocketPlatformHeight | Позитивный метод для SocketPlatformHeight, производится ввод и считывание параметров |
| TubeDiameter | Позитивный метод для TubeDiameter, производится ввод и считывание параметров |
| TubeHeight | Позитивный метод для TubeHeight, производится ввод и считывание параметров |
| EnableFloorLamp\_GoodFloorLamp\_ReturnSameFloorLamp() |  | Позитивный метод для EnableFloorLamp, производится ввод и считывание параметра |
| MaxValue\_GoodLampParameters\_MaximumValueEqualValue() |  | Позитивный метод для MaxValue, производится ввод и считывание параметров |
| MinValue\_GoodLampParameters\_MinimumValueEqualValue() |  | Позитивный метод для MinValue, производится ввод и считывание параметров |
| DefaultValue\_GoodLampParameters\_DefaultValueEqualValue() |  | Позитивный метод для DefaultValue, производится ввод и считывание параметров |

Продолжение таблицы А.1

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| DepthHole\_GoodLampParameters\_ReturnSameDepthHole() |  | Позитивный метод для DepthHole, производится сложение 3 параметров и считывание |
| DepthHole\_NotSetBodyHeightTubeHeightSocketPlatformHeight\_ThrowsException() |  | Негативный метод для DepthHole, производится вызов метода при пустых исходных параметрах |