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: Проектирование 3D модели улья используя Kompas3D API.

Подготовил:

студент гр.586-2

Яловский В. В.\_\_\_\_\_\_\_

Проверил:

К.т.н., доцент каф. КСУП

Калентьев А.А\_\_\_\_\_\_

Томск 2020

**Содержание**

1 Описание САПР 3

1.1 Описание программы 4

[1.2 Описание API 6](file:///D:\Proekt_sistemy_-_polny.docx#_Toc527183269)

[1.3 Обзор аналогов 9](file:///D:\Proekt_sistemy_-_polny.docx#_Toc527183270)

[2 Описание предмета проектирования 10](file:///D:\Proekt_sistemy_-_polny.docx#_Toc527183271)

[3 Проект программы 11](file:///D:\Proekt_sistemy_-_polny.docx#_Toc527183272)

[3.1 Диаграмма вариантов использования (Use Cases) 11](file:///D:\Proekt_sistemy_-_polny.docx#_Toc527183273)

[3.2 Диаграмма классов 13](file:///D:\Proekt_sistemy_-_polny.docx#_Toc527183274)

[3.3 Макет пользовательского интерфейса 15](file:///D:\Proekt_sistemy_-_polny.docx#_Toc527183275)

4 Тестирование……………………………………………………………………... 16

4.1 Функциональное тестирование……………………………………………...16

4.2 Модульное тестирование…………………………………………………… 18

4.3 Нагрузочное тестирование…………………………………………………..20

5 Заключение……………………………………………………………………….. 22

[Список литературы 23](file:///D:\Proekt_sistemy_-_polny.docx#_Toc527183276)

1. **Описание САПР**

Система автоматизированного проектирования — автоматизированная

система, реализующая информационную технологию выполнения функций проектирования. Представляет собой организационно-техническую систему, предназначенную для автоматизации процесса проектирования, состоящую из персонала и комплекса технических, программных и других средств автоматизации его деятельности.[1]

Цели создания САПР:

• сокращение трудоёмкости проектирования и планирования;

• сокращение сроков проектирования;

• сокращение себестоимости проектирования и изготовления, уменьшение затрат на эксплуатацию;

• повышение качества и технико-экономического уровня результатов проектирования;

• сокращение затрат на натурное моделирование и испытания.

Задачи САПР:

• автоматизация оформления документации;

• информационная поддержка и автоматизация процесса принятия решений;

• использование технологий параллельного проектирования;

• унификация проектных решений и процессов проектирования;

• повторное использование проектных решений, данных и наработок;

• стратегическое проектирование;

• замена натурных испытаний и макетирования математическим моделированием;

• повышение качества управления проектированием;

• применение методов вариантного проектирования и оптимизации.

**1.1 Описание программы**

КОМПАС-3D - система автоматизированного проектирования.

КОМПАС является системой автоматизированного проектирования проектной документации, а также ее оформления согласно стандартам единой системы конструкторской документации (ЕСКД). Разработан российской компанией «АСКОН».

Поставляющаяся в одном из двух вариантов КОМПАС-График и КОМПАС-3D, данная система предназначена соответственно для двухмерного черчения и пространственного проектирования.

КОМПАС-График может входить в состав КОМПАС-3D в качестве модуля для работы с эскизами и чертежами, а может использоваться самостоятельно, полностью выполняя задачи двухмерного проектирования и создания соответствующей документации.

Система полностью поддерживает стандарты ЕСКД, при этом все ассоциативные виды пространственных моделей (включая разрезы и сечения, а также различные виды – местные, с разрывом, по стрелке, и пр.) автоматически генерируются КОМПАС-График, а все изменения в модели автоматически изменяются на чертеже.

Шаблонные виды создаются в проекционной связи программой автоматически. Все данные чертежа (такие как геометрические размеры, наименование, масса и пр.) полностью синхронизированы с данными трехмерной модели.

Входящее в систему огромное количество библиотек объектов, автоматизирует выполнение всевозможных специализированных задач.

Базирующаяся на разработанном АСКОН оригинальном ядре, система КОМПАС-3D дает возможность:

- импортировать геометрию изделия во внешние расчетные пакеты,

- передавать геометрию в пакеты управляющих программ различного оборудования с ЧПУ,

- импортировать либо экспортировать созданные модели благодаря поддержке форматов SAT, IGES, XT, VRML, STEP.

Моделировать изделия в КОМПАС-3D можно используя уже готовые компоненты («снизу вверх») либо создавая компоненты в контексте конструкций («сверху вниз»), либо комбинируя эти два способа. Такой подход обеспечивает легкую модификацию получаемых моделей.[2]

Плюсы и минусы Компас 3D:

+ Система очень легка в освоении, причем даже для конструкторов, не имеющих опыт общения с 3D редакторами;

+ Система имеет большое количество библиотек элементов, стандартизированных по ГОСТ;

+ Данная система является продуктом отечественных разработчиков, а потому не существует никаких проблем с ее локализацией;

+ Хоть система и платная - она имеет вполне разумную стоимость;

+ Удобство оформления практически любых чертежей согласно нормам, установленных ЕСКД;

+ Программа имеет широкое распространение, кроме того, имеется бесплатная учебная версия;

+ Имеются встроенные средства для трассировки трубопроводов, электрических кабелей, жгутов;

+ Имеется встроенный модуль для создания электрических цепей;

+ Система обладает широкими возможностями для параметризации объектов;

+ Наличие превосходно продуманного 2D модуля для черчения;

+ Наличие широких возможностей для проектирования деталей, гнутых из листового металла;

+ Поддержка расчета упругих деталей;

+ Наличие встроенной системы обучения;

+ Несложный для обучения и довольно удобный интерфейс;

- Затрудненное переобучение на другие, особенно «тяжелые» аналогичные системы;

- Несмотря на то, что чертить довольно легко, проектировать значительно сложней;

- Отсутствие кинематического, прочностного, температурного и частотного анализа;

- Система спецификации до конца не продумана;

- Крайне медленное развитие системы;

- Нет возможности выполнять эргономические расчеты;

- Весьма скромные возможности для создания фотореалистичных изображений;

- Сложность и дороговизна модифицирования системы под собственные нужды;

- Слабая система поверхностного моделирования;

- Отсутствие инструментов для резервирования объемов;

- Некоторые проблемы при импортировании моделей из других CAD.[2]

**1.2 Описание API**

Главным интерфейсом API системы КОМПАС является KompasObject. Получить указатель на этот интерфейс (если быть точным, на интерфейс приложения API 5) можно с помощью экспортной функции CreateKompasObject(). Методы этого интерфейса, главные из которых представлены в табл. 1, 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 свойства и методы интерфейса KompasObject, которые были использованы при разработке плагина.

Таблица 1.1. Методы интерфейса KompasObject.

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возращаемое 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 |
| GetMathematic2D() |  | Указатель на интерфейс ksMathematic2D | Метод для получения указателя на интерфейс для работы с математическими функциями |
| GetParamStruct(short structType) | structType – тип интерфейса параметров | Указатель на интерфейс указаного типа из StructType2D | Метод для получения указателя на интерфейс графического документа (чертежа или фрагмента) |
| GetDynamicArray(long type) | ext – расширение имени файла,  filter – фильтр поиска (0 – фильтр формируется автоматически),  preview – признак подключения окна предварительного просмотра:  с полключением окна,  без подключения окна  typeDir – стартовая папка. | Строка с именем файла | Возвращает указательна на интерфейс динамического массива. |

Таблица 1.2 – Методы интерфейса IPart.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Возвращаемое значение | Описание |
| EntityCollection  (short objType) | |  |  | | --- | --- | | objType | - тип объектов, содержащихся в массиве. | | указатель на интерфейс [ksEntityCollection](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntityCollection.htm) или [IEntityCollection](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntityCollection.htm) | Формирует массив объектов и возвращает указатель на его интерфейс |
| 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). |   Типы объектов (objType):   |  |  |  | | --- | --- | --- | | o3d\_planeXOY | 1 | - плоскость XOY | | o3d\_planeXOZ | 2 | - плоскость XOZ | | o3d\_planeYOZ | 3 | - плоскость YOZ | | o3d\_pointCS | 4 | - точка начала системы координат | | o3d\_axisOX | 71 | - ось OX | | o3d\_axisOY | 72 | - ось OY | | o3d\_axisOZ | 73 | - ось OZ | | 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) | |  |  | | --- | --- | | objType | - [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) |
| UpdateDocumentParam() |  | TRUE – в случае успешного завершения. | Активизировать измененные параметры документа |

Окончание таблицы 1.3

|  |  |  |  |
| --- | --- | --- | --- |
| GetPart(int type) | type – тип компонента из перечисления Типы компонентов. |  | Получить указатель на интерфейс компонента в соответствии с заданным типом |

**1.3 Обзор аналогов**

В результате поиска не удалось найти прямого аналога для разрабатываемого плагина. Близлежащим аналогом является программа «Packer3D».

Программа позволяет проводить расчёты оптимальной укладки ящиков разного типа в набор транспортных средств типа контейнер, грузовик или вагон.

Стоимость программы составляет 85000 рублей. [4]

Внешний вид программы представлен на рисунке 1.1



Рисунок 1.1 – Внешний вид программы «Packer3D».

**2 Описание предмета проектирования**

Предметом проектирования является улей для пчёл. Улей — искусственное жилище, изготовленное человеком для содержания медоносных пчёл. На рисунке 2.1 изображена 3D модель улья.



Рисунок 2.1 – 3D модель улья.

Параметры проектируемой 3D модели:

•высота улья (не включая ножки) H (от 200 до 1800 мм);

•длина улья L (от 300 до 1800 мм);

•ширина улья W (от 300 до 1800 мм);

•диаметры входных отверстий для пчёл (от 10 до 75 мм);

•толщина(высота) крыши Rh (от 5 до 50 мм);

•длина ножек Ll (от 50 до 600 мм);

•ширина ножек Lw (от 50 до 600 мм);

•высота ножек Lh (от 50 до 1000 мм).

Зависимые параметры 3D модели:

•Расположение и размер ножек зависят от длины и ширины улья:

при увеличении длины (L) и ширины (W) на n мм происходит смещение и увеличение размера ножек на n мм, так же нельзя сделать размер ножек превышающий размер улья (Ll<= H/4), (Lw<=W/4), (Lh<=H/4);

•Количество этажей зависит от высоты улья. N = H/300,

N – количество этажей;

300 – высота одного этажа.

**3 Проект программы**

Унифицированный язык моделирования (UML) – это семейство графических нотаций, в основе которого лежит единая метамодель. Он помогает в описании и проектировании программных систем, в особенности систем, построенных с использованием объектно-ориентированных (ОО) технологий.[5]

**3.1 Диаграмма USE CASE**

Диаграммы вариантов использования представляют собой графическое представление взаимодействия пользователя и компьютерной системы. Каждый вариант использования охватывает некоторую очевидную для пользователей функцию системы и решает некоторую дискретную задачу пользователя. Список всех вариантов использования фактически определяет функциональные требования к системе, с помощью которых может быть сформулировано техническое задание [6]. Диаграмма вариантов использования изображена ниже на рисунке 3.1.

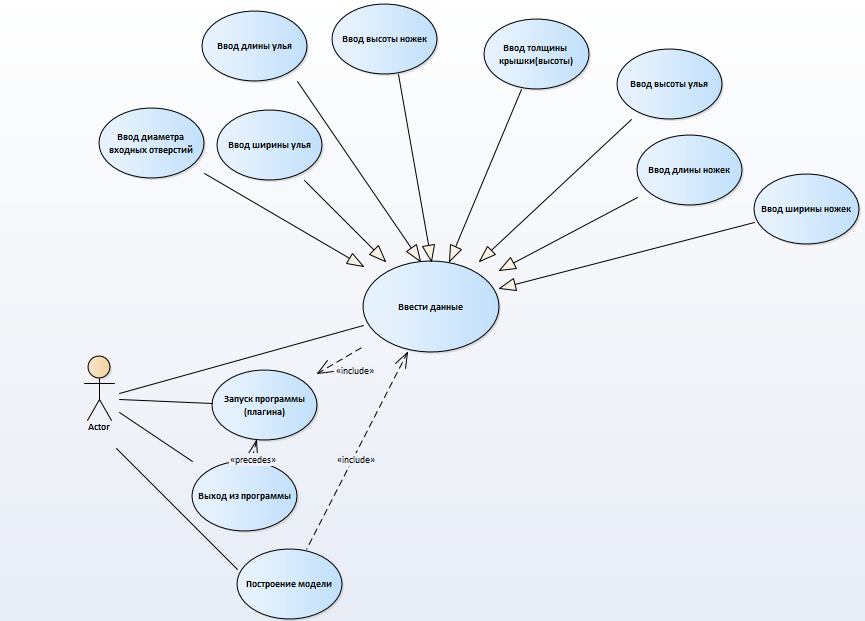


Рисунок 3.1 – Use Case диаграмма проекта.

После добавления дополнительной функциональности схема Use Case поменяла свой вид. Реализация дополнительной функциональности (построение доп. отверстий с других сторон) добавила три дополнительных параметра к варианту использования «Ввести данные». Схему можно увидеть на рисунке 3.2.

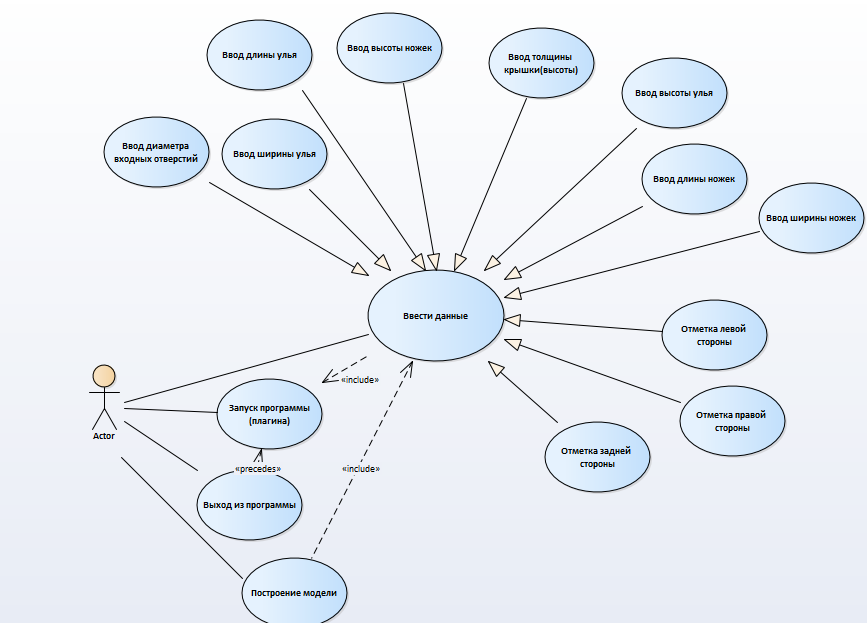


Рисунок 3.2 – Use Case диаграмма проекта после добавления дополнительной функциональности.

**3.2 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 [6]. Диаграмма связей проекта показана ниже на рисунке 3.3

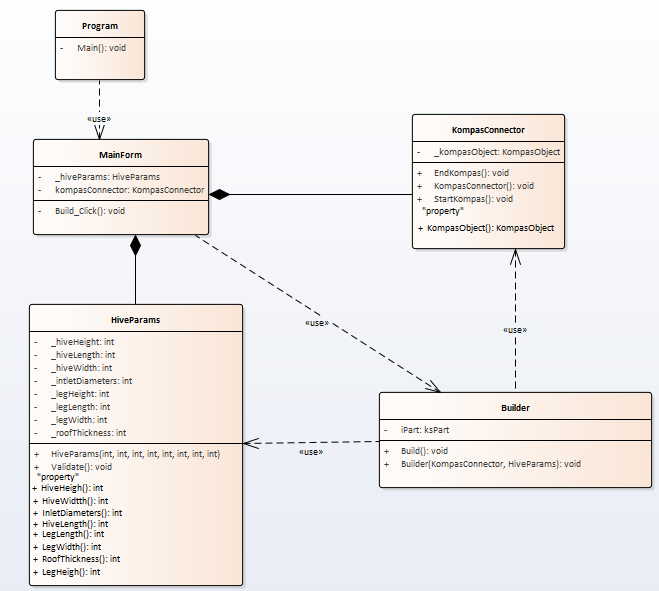


Рисунок 3.3 – Диаграмма классов.

Для реализации подсистемы были спроектированы следующие классы:

* MainForm – класс диалогового окна, обеспечивающий взаимодействие между пользователем и программой через форму;
* HiveParams − класс, хранящий в себе все параметры улья, осуществляет проверку зависимых параметров;
* KompasConnector – класс, отвечающий за работу с API КОМПАС 3D.
* Builder – класс, отвечающий за вызов методов API КОМПАС 3D, необходимых для постройки объекта проектирования.

Во время разработки проекта, диаграмма классов поменяла свой вид. К сущности «Main» были добавлены дополнительные методы, такие как:

«AutoFill» – заполнение полей стандартными значениями, обработчики событий «Textbox\_KeyPress» и «Textbox\_TextChanged», «ValidateAndSet\_Values» – проверка правильности введенных данных.

После реализации дополнительной функциональности, к сущности «Builder» добавились три метода, отвечающие за постройку отверстий с других сторон: «CreateLeftHoles», «CreateRightHoles», «CreateBackHoles».

В метод «Build» в сущности «Builder» добавились дополнительные параметры, отражающие выбор пользователя в создании отверстий.

Диаграмма, полученная после этапа реализации изображена на рисунке 3.4

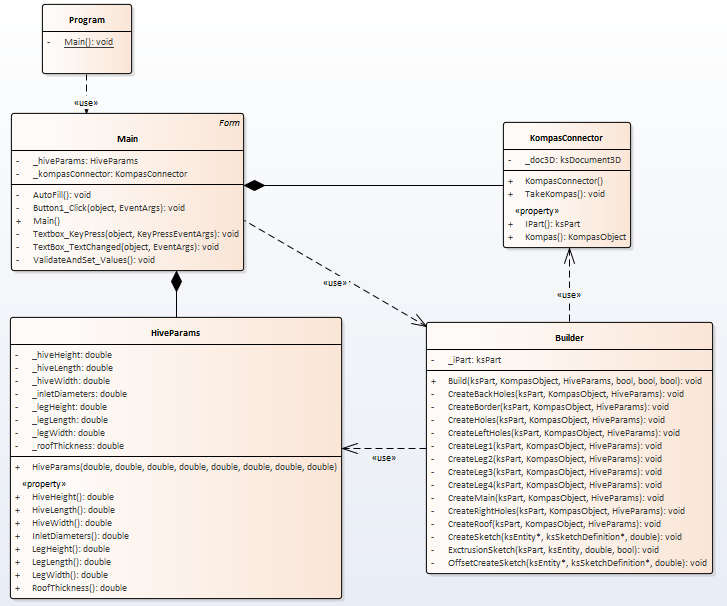
****

Рисунок 3.4 – Диаграмма классов после этапа реализации.

**3.3 Макеты пользовательского интерфейса**

Пользовательский интерфейс, или UI (User Interface) — это внешний вид продукта, способ общения между пользователем и программой. [6]

Большую часть макета интерфейса составляет блок ввода (обведен рамкой под номером 1). В блок ввода (отмечен рамкой под номером 1) пользователю необходимо ввести желаемые параметры для 3D модели улья.

Кнопка «Построить» (обведена рамкой под номером 4) при нажатии на которую произойдет построение 3D модели по параметрам, введённым в поля блока ввода.

Если данные в поля будут неправильно введены, эти поля будут подсвечивается красным цветом (отмечены знаком «–» в поле). Если же данные в поле введены верно, то оно будет подсвечиваться зелёным цветом (отмечены знаком «+» в поле).

При хотя бы одном неправильно заполненном поле, появится сообщение с просьбой проверки введенных данных (отмечено рамкой под номером 2).

Согласно дополнительному заданию размещен блок для отметки построения дополнительных отверстий (отмечено рамкой под номером 3)

Внешний вид пользовательского интерфейса изображен на рисунке 3.5.

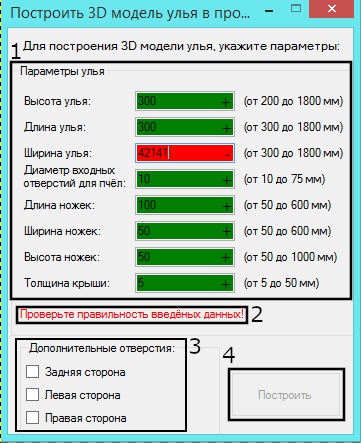


Рисунок 3.5 – Макет пользовательского интерфейса.

**4 Тестирование**

**4.1 Функциональное тестирование**

Функциональное тестирование – это тестирование функциональности и поведения нашей программы, для того чтобы убедится что поведение программы и ее функционал соответствуем требованиям функциональной спецификации. Обычно выполняется как тестирование черного ящика, подавая на вход какой-то набор данных и ожидая чего-то на выходе[7].

Выполним функциональное тестирование. Для этого построим модель с минимальными входными данными:

•высота улья – 200 мм;

•длина улья – 300 мм;

•ширина улья – 300 мм;

•диаметры входных отверстий для пчёл –10 мм;

•толщина(высота) крыши – 5 мм;

•длина ножек – 50 мм;

•ширина ножек – 50 мм;

•высота ножек – 50 мм.

Модель с минимальными входными параметрами показана на рисунке 4.1.

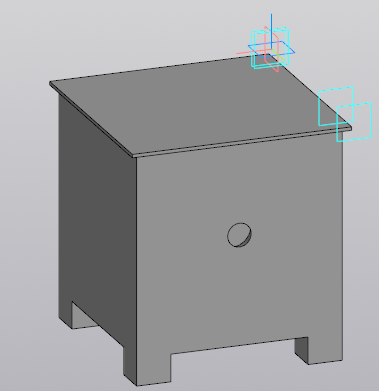
****

Рисунок 4.1 – Модель, построенная с минимальными параметрами.

Далее построим модель с максимально входными параметрами:

•высота улья – 1800 мм;

•длина улья – 1800 мм;

•ширина улья – 1800 мм;

•диаметры входных отверстий для пчёл –75 мм;

•толщина (высота) крыши – 50 мм;

•длина ножек – 600 мм;

•ширина ножек – 600 мм;

•высота ножек – 1000 мм.

Модель с максимальными входными параметрами показана на рисунке 4.2.

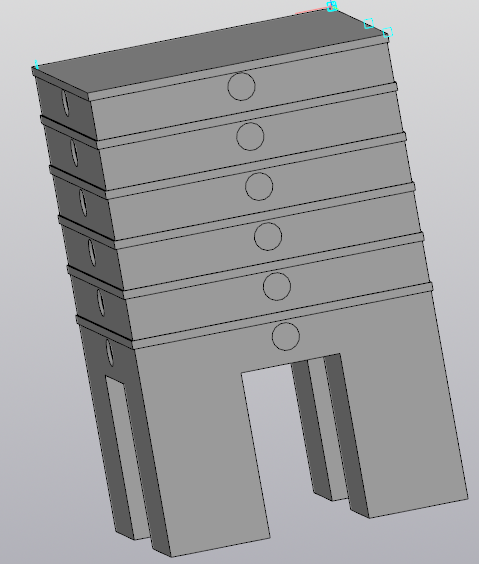
****

Рисунок 4.2 – Модель, построенная с максимальными параметрами.

При выходе за пределы допустимых значений появляется ошибка с просьбой проверить правильность введеных данных и поле, в котором неправильно введены данные подсветится красным цветом. Пример ошибки можно увидеть на рисунке 4.3.

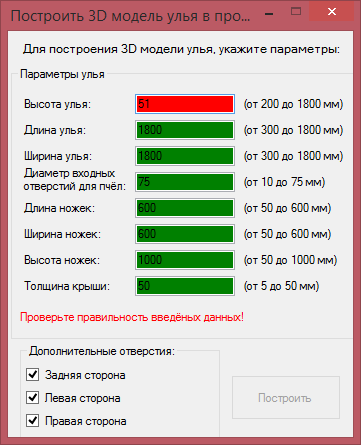


Рисунок 4.3 – Уведомление об ошибке при неправильно-введеных данных.

**4.2 Модульное тестирование**

Unit testing (юнит тестирование или модульное тестирование) — заключается в изолированной проверке каждого отдельного элемента путем запуска тестов в искусственной среде[8].

Список тестовых сценариев для модульного тестирования граничных значений входных параметров представлен в таблице 4.1

Таблица 4.1 – Список тестовых сценариев.

|  |  |
| --- | --- |
| Название тестового метода | Описание |
| TestHeightGet\_CorrectValue() | Позитивный тест проверки высоты улья. |
| TestLengthGet\_CorrectValue() | Позитивный тест проверки длины улья. |
| TestWidthGet\_CorrectValue() | Позитивный тест проверки ширины улья. |
| TestInLetDiametersGet\_CorrectValue() | Позитивный тест проверки диаметра отверстий улья. |
| TestLegHeightGet\_CorrectValue() | Позитивный тест проверки высоты ножек. |
| TestLegLengthGet\_CorrectValue() | Позитивный тест проверки длины ножек. |

Окончание таблицы 4.1

|  |  |
| --- | --- |
| TestLegWidthGet\_CorrectValue() | Позитивный тест проверки ширины ножек. |
| TestRoofThicknessGet\_CorrectValue() | Позитивный тест проверки толщины крыши. |
| TestBorderHeightSet\_ArgumentException(int wrongParam, string message) | Тестирование при вводе диапазона некорректных значений – высота улья. |
| TestBorderLengthSet\_ArgumentException(int wrongParam, string message) | Тестирование при вводе диапазона некорректных значений – длина улья. |
| TestBorderWidthSet\_ArgumentException(int wrongParam, string message) | Тестирование при вводе диапазона некорректных значений – ширина улья. |
| TestBorderWidthSet\_ArgumentException(int wrongParam, string message) | Тестирование при вводе диапазона некорректных значений – ширина улья. |
| TestBorderInLetSet\_ArgumentException(int wrongParam, string message) | Тестирование при вводе диапазона некорректных значений – диаметер входных отверстий. |
| TestBorderLegHeightSet\_ArgumentException(int wrongParam, string message) | Тестирование при вводе диапазона некорректных значений – высота ножки. |
| TestBorderLegLengthSet\_ArgumentException(int wrongParam, string message) | Тестирование при вводе диапазона некорректных значений – длина ножки. |
| TestBorderLegWidthSet\_ArgumentException(int wrongParam, string message) | Тестирование при вводе диапазона некорректных значений – ширина ножки. |
| TestBorderRoofThicknessSet\_ArgumentException(int wrongParam, string message) | Тестирование при вводе диапазона некорректных значений – толщина отверстия крыши. |

Результаты успешного прохождения всех модульных тестов приведен на рисунке 4.4.

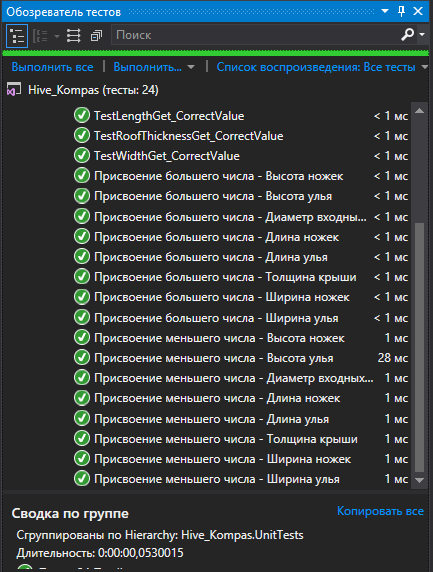
****

Рисунок 4.4 – Результаты модульного тестирования.

**4.3 Нагрузочное тестирование**

Нагрузочное тестирование позволяет оценить поведение системы при возрастающей нагрузке, целью нагрузочного тестирования является также определение максимальной нагрузки, которую может выдержать система.

Нагрузочное тестирование будет проводится на вычислительной машине со следующими характеристиками: процессор – Intel Core i7 3632qm 2.2ghz, 4GB RAM, операционная система Windows 8.1 x64.

После построения 193 деталей с минимальными параметрами программа завершилась со сбоем из-за нехватки оперативной памяти, поскольку начал работать файл подкачки, система потеряла отзывчивость. На момент сбоя количество потребляемой памяти было около 739 МБ. График зависимости показан на рисунке 4.5.

После построения 147 деталей с максимальными параметрами программа также завершилась со сбоем. Но момент сбоя количество потребляемой памяти было около 916 МБ. График зависимости, в котором синяя линия – максимальные параметры, зеленая линия – минимальные параметры, показан на рисунке 4.5.

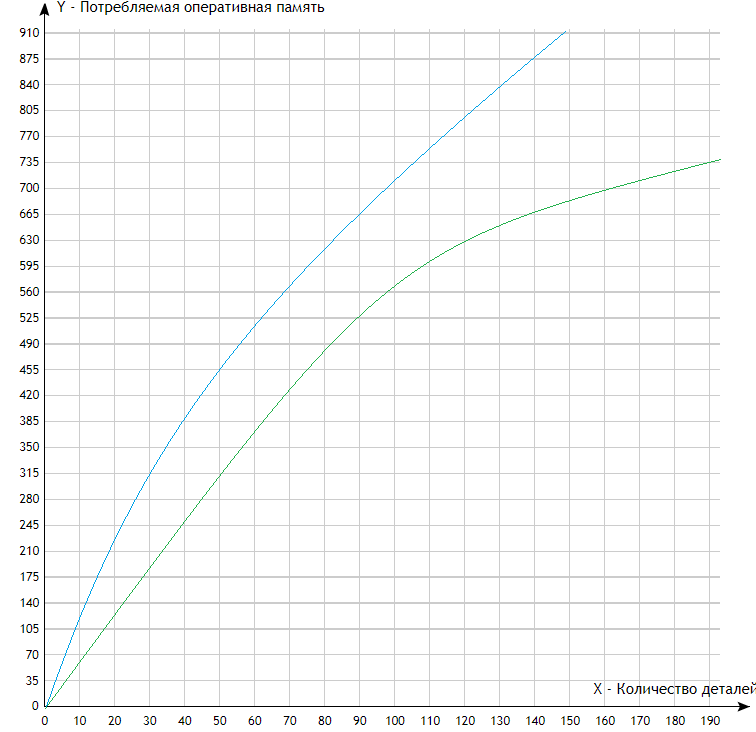
****

Рисунок 4.5 – График зависимости потребляемой памяти от количества построенных деталей.

По графикам можно определить, что при увеличении количества деталей, растёт потребление памяти.

В результате нагрузочного тестирования, построение останавливается при недостатке оперативной памяти. Файл подкачки решает проблему нехватки памяти, но по причине того, что файл подкачки располагается на медленной памяти в жёстком диске («HDD»), система начинает медленно обрабатывать запросы, а после заполнения и вовсе престаёт реагировать на действия пользователя. Улучшить результаты тестирования можно, если добавить системе больше оперативной памяти или перенести файл подкачки на твердотельный накопитель («SSD»).

**5 Заключение**

При выполнения лабораторных работ были изучены основные этапы проектирования программного продукта, проведена работа с API САПР

«КОМПАС-3D».

Было составлено техническое задание, разработан проект системы, составлены UML диаграммы классов и вариантов использования, разработан макет пользовательского интерфейса.

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

Над реализованным плагином были проведены функциональное, модульное и нагрузочное тестирования на платформе Windows 8.1 «x64».

**Список литературы**

1. САПР – System Enginering Thinking Wiki [Электронный ресурс]. – URL: <http://sewiki.ru/%D0%A1%D0%90%D0%9F%D0%A0> (дата обращения 25.01.2020)
2. Система трёхмерного моделирования компании Компас-3D [Электронный ресурс]. – URL: <https://ascon.ru/products/7/review/> (дата обращения 25.01.2020)
3. API – Википедия. [Электронный ресурс]. — URL: <https://ru.wikipedia.org/wiki/API> (дата обращения 25.01.2020)
4. Packer3d.ru – Программы и сервисы для оптимальной укладки грузов. [Электронный ресурс]. — URL: <http://www.packer3d.ru/catalog/20> (дата обращения 25.01.2020)
5. Этапы разработки пользовательского интерфейса [Электронный ресурс]. – URL: <https://vc.ru/design/58502-etapy-razrabotki-polzovatelskogo-interfeysa-kak-sdelat-tak-chtoby-ui-ne-lishil-vas-pribyli> (дата обращения 28.01.2020)
6. Мартин Фаулер «UML. Основы. Третье издание» / СПб: Питер, 2005. – 184 стр.
7. Тестирование программного обеспечения [Электронный ресурс]. –URL:<https://software-testing.org/testing/osnovnye-vidy-testirovaniya-programnogo-obespecheniya.html> (дата обращения 30.04.2020)
8. Море аналитической информации [Электронный ресурс]. – URL: <http://citforum.ru/SE/testing/unit_testing/> (дата обращения 30.04.2020)