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

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

Пояснительная записка по лабораторному проекту

«Разработка плагина моделирования кости домино для системы   
Компас - 3D»  
по дисциплине «Основы разработки САПР»

Выполнил:

Студент гр. 588-1

\_\_\_\_\_\_\_\_\_ К.В.Рыжнев

« » 2022 г.

Проверил:

к.т.н., доцент каф. КСУП

\_\_\_\_\_\_\_\_\_ А.А.Калентьев  
« » 2022 г.

Томск 2022

**Реферат**

Пояснительная записка 29 с., 16 рис., 18 таблиц, 7 источников.

Ключевые слова: КОМПАС-3D, ПЛАГИН, КОСТЬ, ДОМИНО, САПР.

Целью данной работы является разработка плагина для создания трехмерной модели кости домино, согласно заданным параметрам, для системы автоматизированного проектирования «КОМПАС-3D v.19»

В результате разработан плагин, строящий трехмерную модель кости домино в КОМПАС-3D.

Отчет по пояснительной записке выполнен в текстовом редакторе Microsoft Word 2016.

Оглавление

[1. Описание САПР 4](#_Toc94046628)

[1.1 Описание программы «Компас - 3D» 4](#_Toc94046629)

[1.2 Описание API САПР КОМПАС-3D 5](#_Toc94046630)

[1.3 Обзор аналогов 14](#_Toc94046631)

[1.3.1 MechaniCS 14](#_Toc94046632)

[2 Описание предмета проектирования 16](#_Toc94046633)

[3 Стек технологий и системные требования 17](#_Toc94046634)

[4. Описание плагина 18](#_Toc94046635)

[4.1 Диаграмма классов 18](#_Toc94046636)

[4.2 Макет пользовательского интерфейса 20](#_Toc94046637)

[5 Тестирование программы 22](#_Toc94046638)

[5.1 Функциональное тестирование 22](#_Toc94046639)

[5.2 Модульное тестирование 24](#_Toc94046640)

[5.3 Нагрузочное тестирование 25](#_Toc94046641)

[Заключение 28](#_Toc94046642)

[Список используемых источников 29](#_Toc94046643)

[Приложение А 30](#_Toc94046643)

# 1. Описание САПР

## 1.1 Описание программы «Компас - 3D»

КОМПАС-3D — система трехмерного проектирования, ставшая стандартом для тысяч предприятий, благодаря сочетанию простоты освоения и легкости работы с мощными функциональными возможностями твердотельного и поверхностного моделирования. Ключевой особенностью продукта является использование собственного математического ядра C3D и параметрических технологий, разработанных специалистами АСКОН. КОМПАС-3D обеспечивает поддержку наиболее распространенных форматов 3D-моделей (STEP, ACIS, IGES, DWG, DXF), что позволяет организовывать эффективный обмен данными со смежными организациями и заказчиками, использующими любые CAD / CAM / CAE-системы в работе [1].

КОМПАС-3D учебная версия – версия, предназначенная для учебы, полностью бесплатна для образовательных учреждений. Студенческая версия не отличается по функциональности, она направленна не на коммерческую цель, а на образовательную.

КОМПАС-3D имеет несколько прикладных приложений:

* КОМПАС-3D LT – версия для ознакомления в ПО, моделированием, черчением. Можно создавать несложные модели деталей, с помощью чертежного редактора можно делать выкройки для одежды и т.п.
* КОМПАС-3D HOME – версия для моделирования моделей техники и т.п. Система поддерживает 3D-принтеры, тем самым можно легко получить реальный объект из трехмерной модели в реальной жизни [1].

## 1.2 Описание API САПР КОМПАС-3D

В КОМПАС 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 [2].

Главным интерфейсом API системы КОМПАС является KompasObject, получить указатель на этот интерфейс (на интерфейс приложения API 5) можно с помощью экспортной функции CreateKompasObject() [2].

Методы этого интерфейса (Таблица 1.1) 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) [2].

Таблица 1.1 – Методы интерфейса KompasObject().

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Тип** | **Описание** |
| Document3D() | ksDocument | Метод для получения указателя на интерфейс трехмерного документа (детали или сборки) |
| ActivateControllerAPI() | Bool | Метод для активации API Компас 3D |
| Visible | Bool | Свойство видимости приложения |

В таблицах 1.2 – 1.3 представлены методы и параметры интерфейса ksDocument3D для сборки модели в программе КОМПАС–3D.

Таблица 1.2 – Методы интерфейса ksDocument3D.

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Тип** | **Описание** |
| Create(bool invisible, bool typeDoc) | Bool | Метод для создания пустого документа (деталь или сборку) |
| GetPart(int type) | ksPart | Метод, возвращающий указатель на интерфейс детали или компонента сборки |
| Filename | String | Свойство, определяющее имя файла, из которого вставлен компонент |
| Visible | Bool | Свойство видимости приложения |

Таблица 1.3 – Входные параметры, используемых методов интерфейса ksDocument3D.

|  |  |  |
| --- | --- | --- |
| **Метод** | **Входной параметр** | **Описание** |
| Create(bool invisible, bool typeDoc) | Invisible | Признак ре­жима редакти­рования доку­мента (TRUE - неви­димый режим, FALSE - види­мый режим) |

Продолжение таблицы 1.3

|  |  |  |
| --- | --- | --- |
| **Метод** | **Входной параметр** | **Описание** |
| Create(bool invisible, bool typeDoc) | typeDoc | Тип докумен­та (TRUE - де­таль, FALSE - сбор­ка) |
| GetPart(int type) | Type | Тип компонента из перечисления.  Типы компонентов:  pInPlace\_Par – компонент, редактируемый на месте;  pNew\_Part – новый компонент;  pEdit\_Part – редактируемый компонент;  pTop\_Part – главный компонент, в составе которо­го находится новый или редактируе­мый или указанный компонент |

В таблице 1.4 представлены методы, которые соответствуют свойствам трехмерных элементов модели.

Таблица 1.4 – Методы интерфейса ksEntity.

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Тип** | **Описание** |
| Create() | Bool | Метод для создания объекта в модели |
| GetDefinition() | IUnkown | Метод для получения указателя на интерфейс параметров объектов и элементов |
| Create() | Bool | Метод для создания объекта в модели |

В таблицах 1.5 – 1.6 представлены методы и параметры интерфейса ksPart для создания объекта и получение на него указателя.

Таблица 1.5 – Методы интерфейса ksPart.

|  |  |  |
| --- | --- | --- |
| **Название метода или свойства** | **Тип** | **Описание** |
| GetDefaultEntity (short objType) | ksEntity | Метод для получения указателя на интерфейс объекта, создаваемого системой в трехмерном документе по умолчанию |
| NewEntity (short objType) | ksEntity | Метод, создающий интерфейс нового трехмерного объекта и возвращающий указатель на него |

Таблица 1.6 – Входные параметры, используемых методов интерфейса интерфейса ksPart.

|  |  |  |
| --- | --- | --- |
| **Метод** | **Входной параметр** | **Описание** |
| GetDefaultEntity (short objType) | objType | Тип объектов, содержащихся в массиве |
| NewEntity (short objType) | objType | Тип объектов, содержащихся в массиве |

В таблицах 1.7 – 1.8 представлены методы и параметры интерфейса ksDocument2D для создания изображения эскизов для трехмерной модели.

Таблица 1.7 – Используемые методы интерфейса ksDocument2D.

|  |  |  |
| --- | --- | --- |
| **Название** | **Возвращаемое значение** | **Описание** |
| ksCircle (double xc, double yc, double rad, long style) | Указатель на окружность – в случае удачного завершения,  0 – в случае неудачи | Создать окружность |
| ksEllipse (LPDISPATCH param) | Указатель на эллипс – в случае удачного завершения,  0 – в случае неудачи | Создать эллипс |

Таблица 1.8 – Описание входных параметров, используемых методов интерфейса ksDocument2D.

|  |  |  |
| --- | --- | --- |
| **Метод** | **Входной параметр** | **Описание входного параметра** |
| ksCircle (double xc, double yc, double rad, long style) | xc, yc | Координаты центра окружности |
| Rad | Радиус окружности |
| style | Стиль линии |
| ksEllipse (LPDISPATCH param) | param | Указатель на интерфейс [ksEllipseParam](mk:@MSITStore:D:\Program%20Files\Компас\SDK\SDK.chm::/ksEllipseParam.htm) |

В таблице 1.9 представлены методы интерфейса ksSketchDefinition для работы с эскизами.

Таблица 1.9 – Используемые методы интерфейса ksSketchDefinition.

|  |  |  |
| --- | --- | --- |
| **Название** | **Возвращаемый тип** | **Описание** |
| BeginEdit() | Bool | Войти в режим редактирования эскиза (ksDocument2D) |

Продолжение таблицы 1.9.

|  |  |  |
| --- | --- | --- |
| EndEdit() | Bool | Выйти из режима редактирования эскиза |

В таблице 1.10 представлены типы объектов документа-модели, используемые при разработке плагина.

Таблица 1.10 – Типы объектов документа-модели.

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Тип объекта** | **Название объекта** | **Интерфейс параметров** |
| GetDefault Entity (short objType) | o3d\_planeXOZ | Плоскость XOZ | ksPlaneParam |
| o3d\_axisOY | Ось OY |  |
| NewEntity (short objType) | o3d\_sketch | Эскиз трехмерной операции | ksSketchDefinition |
| o3d\_cutExtrusion | Операция вырезания выдавливанием | ksCutExtrusionDefinition |
| o3d\_circularCopy | Копирование по концентрической сетке | ksCircularCopyDefinition |
| o3d\_bossExtrusion | Операция выдавливания | ksBossExtrusionDefinition |

В таблицах 1.11 – 1.13 представлены используемые методы интерфейсов для типов объектов.

Таблица 1.11 – Методы интерфейса ksCutExtrusionDefinition

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| SetSideParam (bool forward, short type, double depth, double draftValue, bool draftOutward) | Bool | Установить параметры выдавливания в одном направлении |

Таблица 1.12 – Методы интерфейса ksCircularCopyDefinition

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| SetAxis(LPDISPATCH axis); | Bool | Установить ось копирования |
| SetCopyParamAlongDir (long count, double step, bool factor, bool dir) | Bool | Установить параметры копирования |

Таблица 1.13 – Описание входных параметров интерфейсов.

|  |  |  |
| --- | --- | --- |
| **Метод** | **Входной параметр** | **Описание** |
| SetSideParam (bool forward, short type, double depth, double draftValue, bool draftOutward) | Forward | Направление выдавливания: TRUE – прямое направление, FALSE – обратное направление |

Продолжение таблицы 1.13

|  |  |  |
| --- | --- | --- |
| **Метод** | **Входной параметр** | **Описание** |
|  | Type | Тип выдавливания.  Виды:  etBlind – строго на глубину;  etThroughAll – через всю деталь;  etUpToVertexTo – на расстояние до вершины;  etUpToVertexFrom – на расстояние за вершину;  etUpToSurfaceTo – на расстояние до поверхности;  etUpToSurfaceFrom – на расстояние за поверхность;  etUpToNearSurface – до ближайшей поверхности. |
| Depth | Глубина выдавливания |
| draftValue | Угол уклона |
| draftOutward | Направление уклона: FALSE - уклон наружу, TRUE - уклон внутрь |

Продолжение таблицы 1.13

|  |  |  |
| --- | --- | --- |
| **Метод** | **Входной параметр** | **Описание** |
| SetAxis(LPDISPATCH axis); | Axis | Указатель на интерфейс оси [ksEntity](mk:@MSITStore:C:\Program%20Files\ASCON\KOMPAS-3D%20V16\SDK\SDK.chm::/ksEntity.htm) |
| SetCopyParamAlongDir (long count, double step, bool factor, bool dir) | Count | Количество копий |
| Step | Шаг |
| Factor | Признак полного шага |
| Dir | Направление копирования |
| SetSketch (LPDISPATCH sketch) | Sketch | Указатель на интерфейс эскиза [ksEntity](mk:@MSITStore:C:\Program%20Files\ASCON\KOMPAS-3D%20V16\SDK\SDK.chm::/ksEntity.htm) |

В таблице 1.14 представлено описание входных параметров этих методов.

Таблица 1.14 – Методы интерфейса ksBossExtrusionDefinition.

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| SetSideParam (bool forward, short type, double depth, double draftValue, bool draftOutward) | Bool | Установить параметры выдавливания в одном направлении |

## 1.3 Обзор аналогов

### 1.3.1 MechaniCS

**MechaniCS** — Приложение MechaniCS располагает стандартным набором инструментов для оформления чертежей по ЕСКД. В нем удобно создавать форматы и штампы чертежей, проставлять размеры и шероховатость поверхностей, допуски размеров, формы и расположения, делать выноски и различные специальные обозначения. При помощи маркеров пользователи имеют возможность создавать собственные элементы оформления чертежей. Есть инструменты создания спецификаций. Специальные средства разработаны для проектирования типовых механических соединений. Кроме того, в приложение включены методики расчета различных механических характеристик деталей машин. База элементов содержит стандартные и унифицированные врезаемые элементы, детали и сборочные единицы, которые можно использовать при проектировании [3].

Все детали общей конструкторско-технологической базы обладают интеллектом и являются объектно-зависимыми. При изменении параметров одной детали все связанные с ней объектно-зависимые детали изменятся автоматически, причем в соответствии с их параметрами в базе. Такая технология — мощный инструмент многовариантного проектирования, залог повышения качества выпускаемых проектов. Важно, что этот подход одинаково доступен пользователям AutoCAD и Autodesk Inventor [3].

На Рисунке 1.1 представлено приложение MechaniCS.



Рисунок 1.1 – Скриншот приложения MechaniCS.

# 2 Описание предмета проектирования

Кость домино – это пластинка определенного размера, которая традиционно изготавливается из слоновой кости, но преимущественно в современном мире из дерева и пластика. Данная пластинка применяется для игры в Домино, каждая кость имеет в своем арсенале определенное количество круглых выемок, именно они влияют на ценность кости.

Игра домино применяется в соревновательных и развлекательных целях, существуют спортивные состязания по игре в домино. Как таковых нормативных размеров кости не существует, и они не прописаны в государственных документах, поэтому производители ориентируются исключительно на предпочтения покупателей.

Значение геометрических параметров может быть зависимо от других

1. Высота кости = 60 мм ≤А≤120 мм;
2. Толщина кости = 10 мм ≤B≤30 мм;
3. Ширина кости = 30 мм ≤С≤1/2\*A мм;
4. Диаметр выемки значения = 8 мм ≤D≤15 мм;
5. Ширина каёмки = 3 мм≤ E≤1/5\*A мм.

Изображение кости: A(высота кости), B(толщина кости), C(ширина кости), D(диаметр выемки), E(ширина каёмки) приведено на рисунке 2.1.

****

Рисунок 2.1 – Изображение кости

# 3 Стек технологий и системные требования

Язык программирования: C#;

Среда разработки: Visual Studio 2019 версия 16.8.5 с применением Microsoft.NET Framework 4.8.04084;

Тестирование с помощью библиотек: NUnit версия 3.12.0, NUnit3TestAdapter версия 3.16.1.;

Система КОМПАС 3D V19.

Для реализации пользовательского интерфейса использовался WinForm.

Взаимодействие плагина с системой КОМПАС (с функциями моделирования, математическими функциями ядра системы и пр.) осуществляется посредством программных интерфейсов, называемых API. В КОМПАС на данный момент существуют API двух версий: API 5 и API 7. Обе версии реализуют различные функции системы и взаимно дополняют друг друга [4].

В основном, для создания полноценных подключаемых модулей, достаточно методов и свойств интерфейсов API 5.

# 4. Описание плагина

## 4.1 Диаграмма классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. [5] Целью создания диаграммы классов является графическое представление статической структуры элементов системы.

Изначальная диаграмма классов плагина представлена на рисунке 4.1.

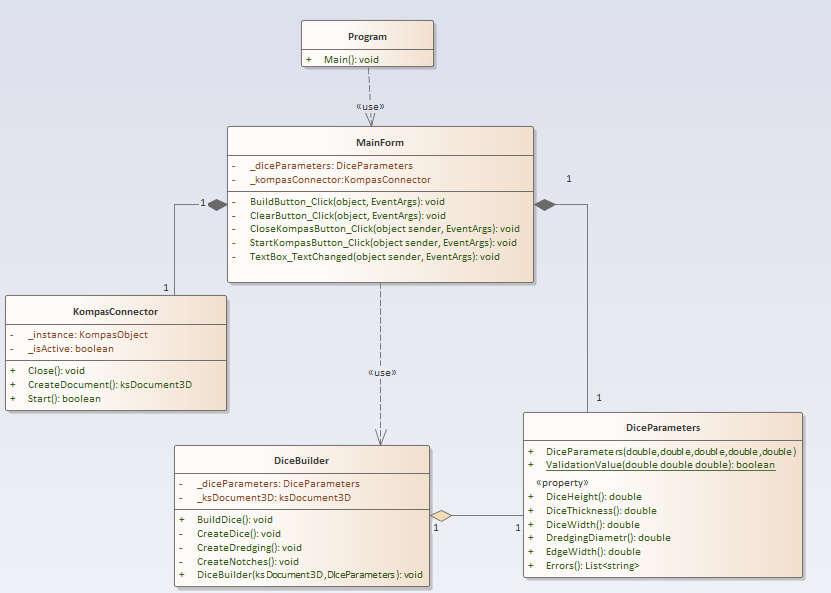


Рисунок 4.1 – Диаграмма классов плагина «Кость домино»

Класс «Program», использует «MainForm» для обработки действий в графическом интерфейсе.

Класс «DiceBuilder» содержит в себе методы создания 3D модели в «Компас 3D»,

Класс «DiceParameters» включает в себя введенные значения в графическом интерфейсе.

Класс «KompasConnector» производит запуск программы «КОМПАС-3D» и строит объект в этой программы.

Диаграмма классов после добавления дополнительной функциональности представлена на рисунке 4.2.

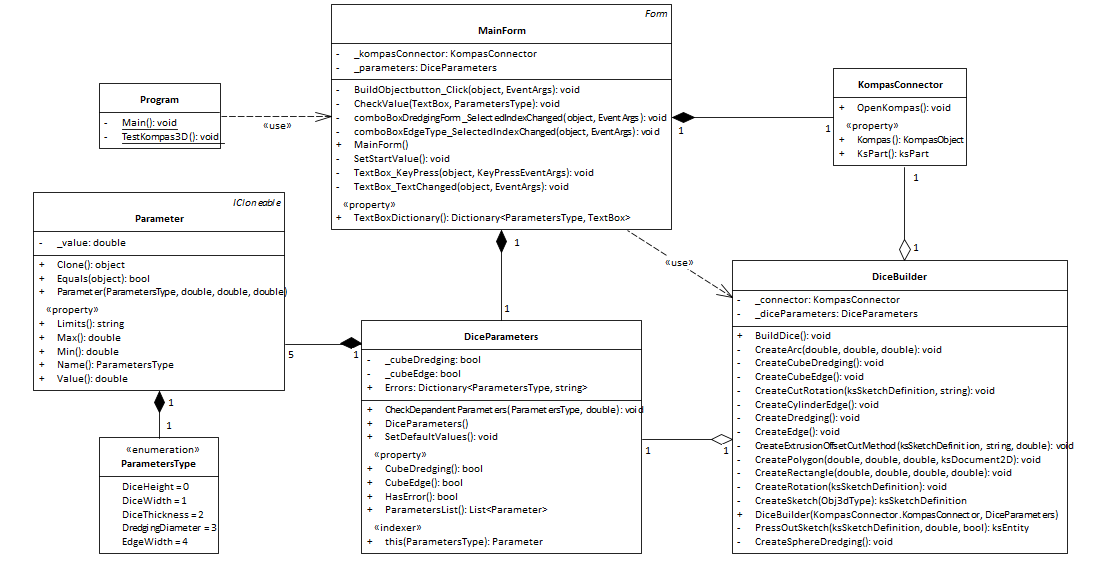


Рисунок 4.2 – Диаграмма классов с дополнительной функциональностью

Опишем некоторые изменения, возникшие после добавления дополнительной функциональности и переработки кода в процессе разработки плагина.

Добавлены новые функции в MainForm: изменение формы, реализация обработки TextBox, внесение параметров при нажатии кнопки и другие небольшие изменения.

Возникла связь-агрегация DiceBuilder и KompasConnector, теперь DiceBuider хранит в себе объект KompasConnector, который позволяет обратиться к нему проще из других функциях в DiceBuilder.

Был создан класс параметров ParametersEnum для упрощения работы с формой и устранения дублирования кода.

В класс DiceBuilder добавлены методы для реализации дополнительной функциональности.

## 4.2 Макет пользовательского интерфейса

Макет пользовательского интерфейса создан с помощью Windows Form. На рисунке 4.3 представлен макет пользовательского интерфейса для ввода параметров модели. Так же на рисунке 4.3 представлено, как будет отображаться не правильный ввод параметров (поле окрашивается в красный цвет), если значения введены правильно, то окно не изменяет цвет.

Перед пользователем представлены 7 полей, предназначенный для ввода параметров (в мм) детали. Помимо этого, присутствует кнопка для построения модели, в дальнейшем при нажатии на которую будет загружаться Компас-3D и начинаться построение модели кости, и кнопка для построения модели по базовым параметрам, чтобы облегчить пользователю представление модели. Также в правой части формы пользователь видит схему будущей детали с разметкой всех параметров.

Модель пользовательского интерфейса представлена на рисунке 4.3

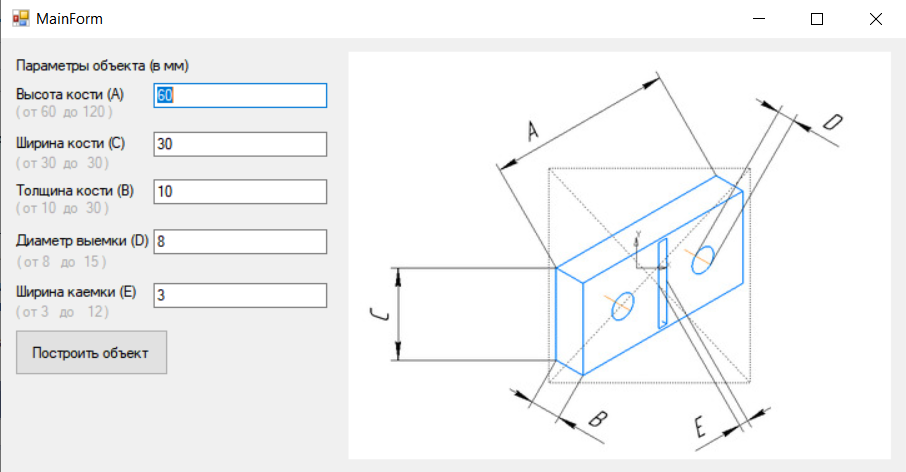


Рисунок 4.3 – Макет пользовательского интерфейса.

Если пользователь вводит неверные данные, то программа выдает предупреждение (Рисунок 4.4).

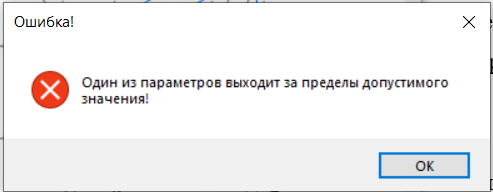


Рисунок 4.4 – Пример сообщения об ошибке.

На рисунке 4.5 изображено диалоговое окно документа САПР «КОМПАС – 3D» после нажатия кнопки «Построить» и построения модели.

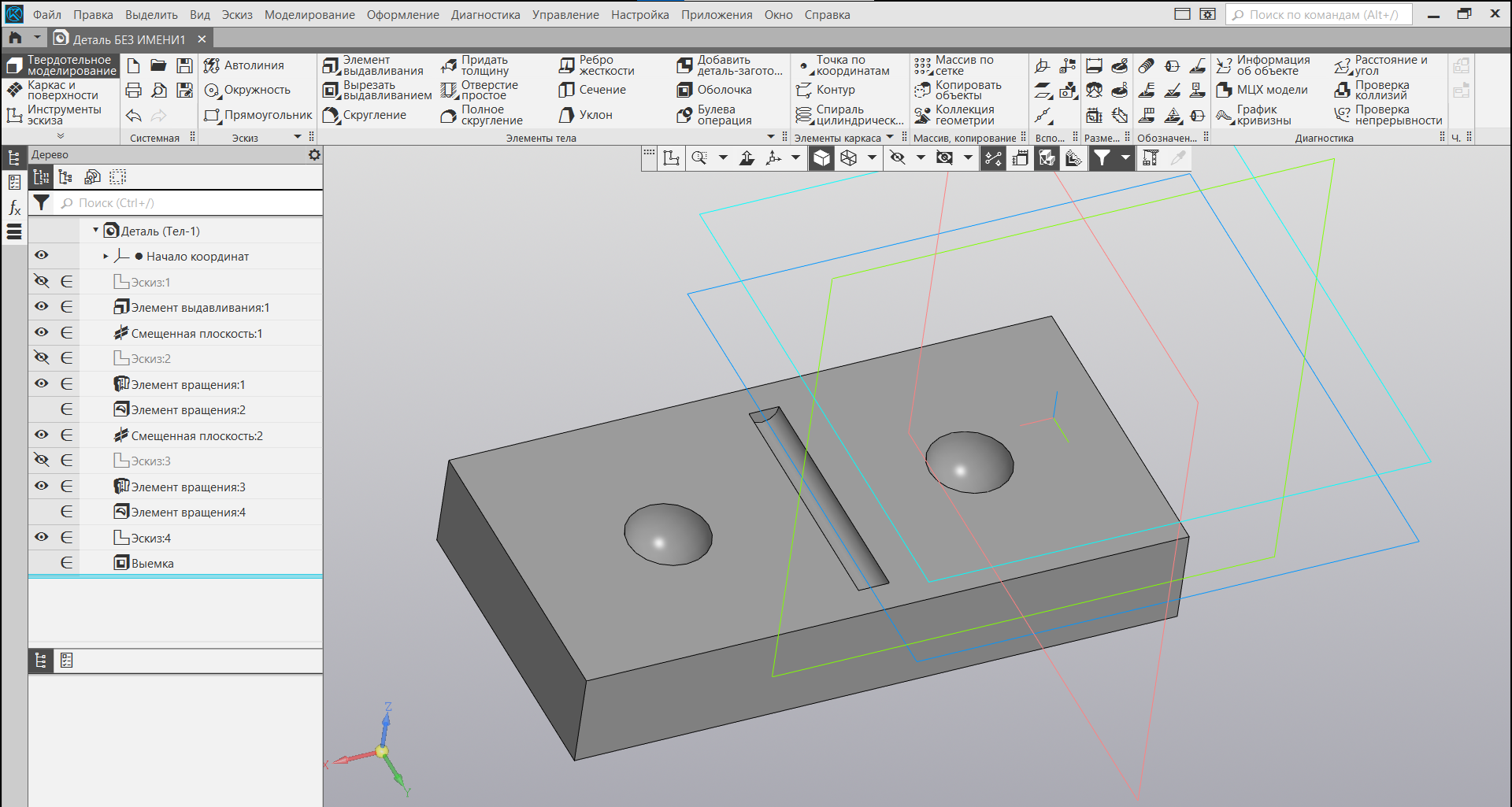


Рисунок 4.5 – Диалоговое окно документа САПР «КОМПАС – 3D»

Макет пользовательского интерфейса после добавления дополнительной функциональности представлен на рисунке 4.6.

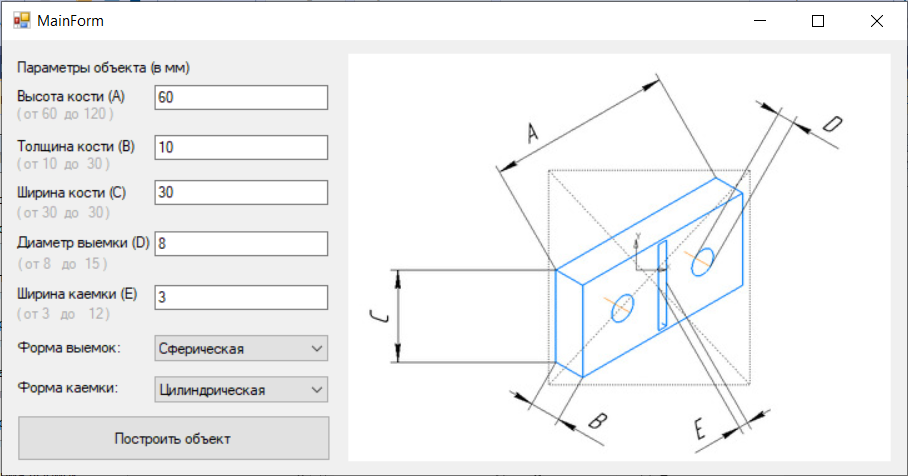


Рисунок 4.6 – Макет пользовательского интерфейса после добавления дополнительной функциональности.

# 5 Тестирование программы

Тестирование — это проверка соответствия объекта желаемым и требуемым критериям [7]. Несоответствие критериям называется ошибкой.

В проекте будет применятся три вида тестирования: функциональное тестирование, модульное тестирование, нагрузочное тестирование.

## 5.1 Функциональное тестирование

При функциональном тестировании кости домино проверялось соответствие полученной трёхмерной модели с введенными параметрами.

Проведено тестирование с максимальными, минимальными (по умолчанию) данными.

На рисунке 5.1 представлена модель с минимальными (по умолчанию) данными.

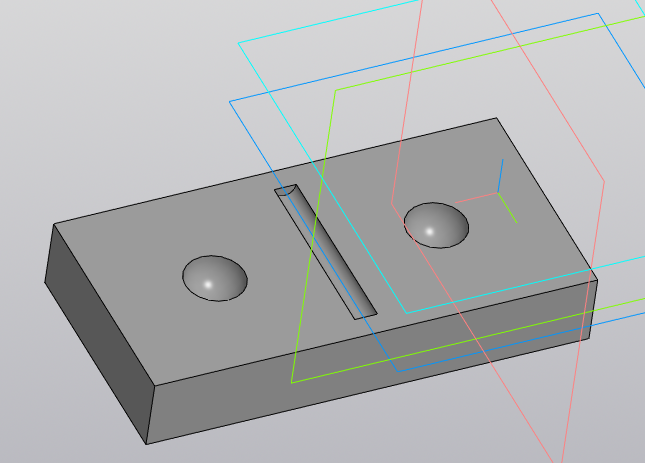
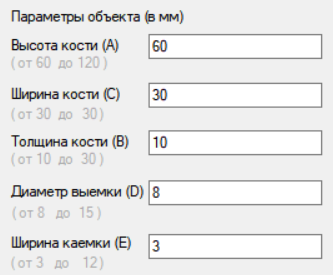


Рисунок 5.1 – Модель с минимальными (по умолчанию) данными.

На рисунке 5.2 представлена модель с максимальными значениями параметров.



Рисунок 5.2 – Модель кости с максимальными значениями.

На рисунке 5.3 представлена модель с квадратными выемками.

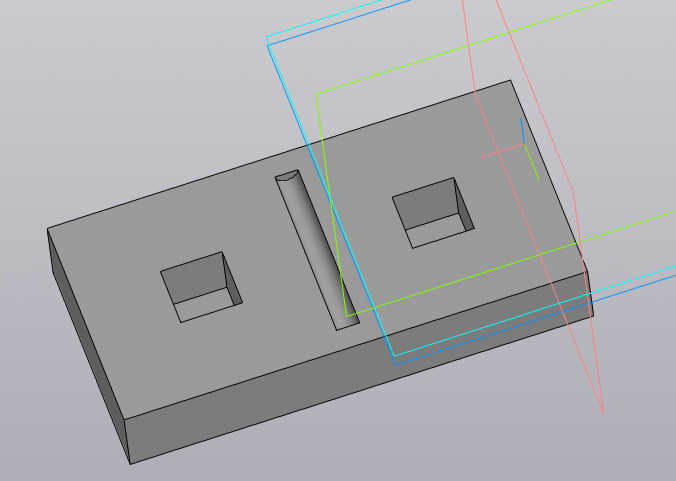


Рисунок 5.3 – Модель кости с квадратными выемками.

На рисунке 5.4 представлена модель с квадратной каемкой.

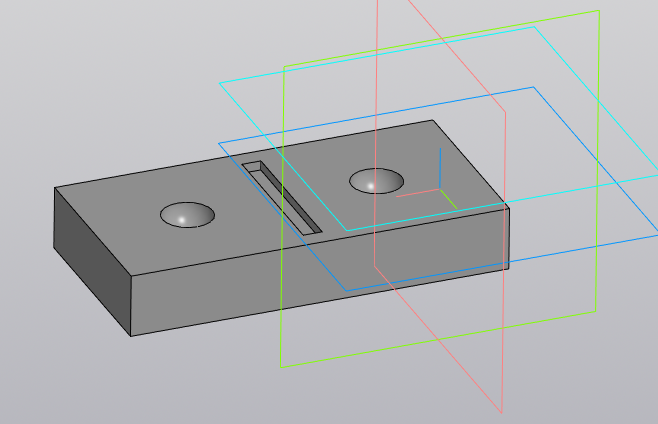


Рисунок 5.4 – Модель кости с квадратной каемкой.

На рисунке 5.5 представлена модель с максимальными значениями параметров.



Рисунок 5.5 – Модель кости с максимальными значениями.

На рисунке 5.5 представлено диалоговое окно при вводе некорректного значения параметра.

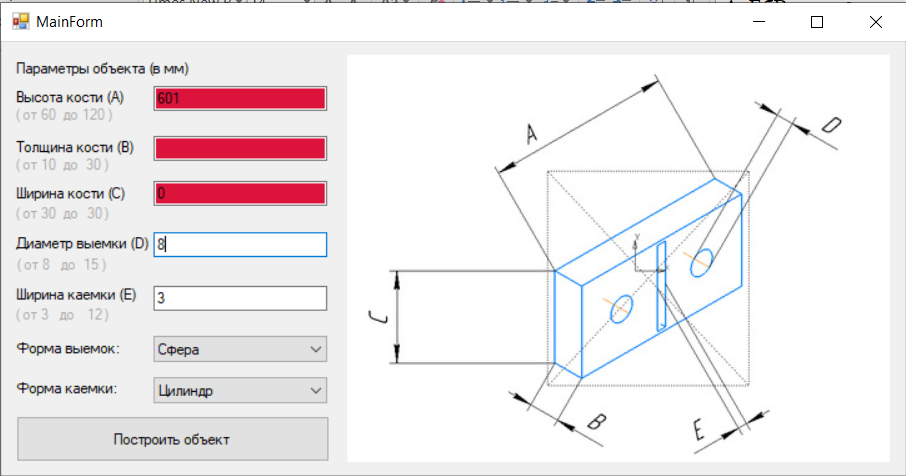


Рисунок 5.5 – Диалоговое окно при некорректном вводе параметра

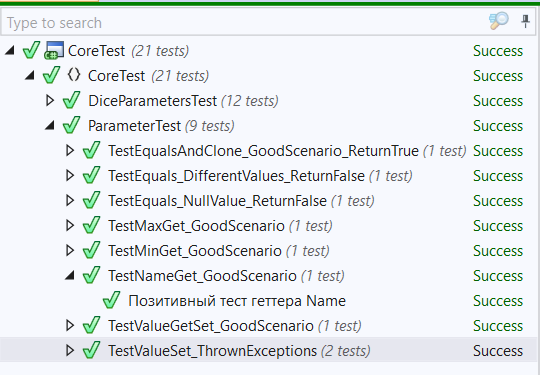
## 5.2 Модульное тестирование

Юнит-тестирование (модульное тестирование) — тестирование отдельного элемента изолированно от остальной системы [7]. На основе тестовых сценариев проводилось тестирование работы методов и свойств классов при помощи обозревателя тестов Visual Studio. Проверялись открытые поля и методы, для этого были созданы тестовые классы:

В качестве проверки корректности работы методов и свойств классов при помощи тестового фреймворка NUnit версия 3.12.0 выполнено модульное тестирование, проверялись открытые поля, свойства и методы.

На рисунке 5.6 представлено модульное тестирование.

Описание тестовых случаев указано в приложении А.



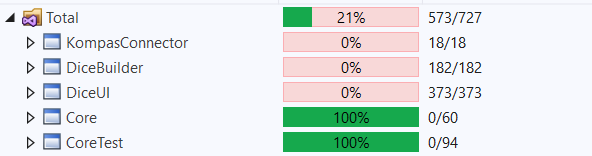


Рисунок 5.6 – Прохождение модульного тестирования.

1. DiceParametersTest. Класс, тестирующий свойства и методы класса DiceParameters.
2. ParameterTest. Класс, тестирующий свойства и методы класса Parameter.

## 5.3 Нагрузочное тестирование

Нагрузочное тестирование — тестирования производительности, сбор показателей и определение производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе (устройству) [7].

1) Тестирование проводилось на Windows 10 x64.

2) Процессор AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx, 2100 МГц, ядер: 4, логических процессоров: 8.

3) 8,00 ГБ (доступно: 7,69 ГБ) ОЗУ;

После построения 81 детали при параметрах по умолчанию (рисунок 5.1), программа завершилась со сбоем, т.к. закончилась память.

На рисунке 5.7 представлено затраченное время на построение моделей кости.

Рисунок 5.7 – График зависимости количества построенных моделей кости от времени построения.

График потребляемой оперативной памяти относительно числа созданных деталей изображен на рисунке 5.8.

Рисунок 5.8 – График зависимости количества построенных моделей кости от потребляемой памяти

Из этого можно сделать вывод, что, время при увеличении количества деталей, увеличивается линейно. Потребление ОЗУ ведет себя так, потому что, начиная с 35-40 детали, через каждые 10 моделей, память значительно провисает, это происходит из-за того, что она пытается освободиться и резко вступает в новую работу. Также, возможно, спустя каждые 10 итераций включается какой-либо фоновый процесс, который нельзя увидеть.

# Заключение

В ходе выполнения лабораторных работ были изучены основные этапы проектирования программного продукта, изучена предметная область предмета проектирования, также было изучено API системы автоматизированного проектирования КОМПАС-3D.

В результате полученные знания были применены для реализации плагина для автоматизации построения модели «Кость домино» в программе КОМПАС-3D, проведены функциональные, модульные и нагрузочные тесты.

# Список используемых источников

1. Компас-3D [электронный ресурс]. – режим доступа: <https://kompas.ru/kompas-3d/about/> (дата обращения 19.10.2021).
2. Базовые интерфейсы API системы КОМПАС [электронный ресурс]. – режим доступа: <https://it.wikireading.ru/23741> (дата обращения 20.10.2021).
3. MechaniCS [электронный ресурс]. – режим доступа: <https://cad.ru/support/bz/archive/62/mechanics/> (дата обращения 20.10.2021).
4. Расширение возможностей системы КОМПАС [электронный ресурс]. – режим доступа: [http://www.k2x2.info/kompas\_3d\_v10\_na\_100/p9.php](http://www.k2x2.info/kompas_3d_v10_na_100/p9.php%20%20)  (дата обращения 21.10.2021).
5. UML [электронный ресурс]. — Режим доступа: https://habr.com/ru/post/458680/ (дата обращения 27.10.2021).
6. Мартин Фаулер. UML. Основы. / Мартин Фаулер; пер. с англ. А. Петухова – 3-е издание. – Спб: Символ-Плюс, 2004 – 192с.
7. Виды тестирования [электронный ресурс]. – режим доступа: <https://qa-academy.by/qaacademy/news/klassifikaciya-vidov-testirovaniya/> (дата обращения: 17.12.2021).

**Приложение А**

(справочное)

Таблица А.1 — Тестовые сценарии

|  |  |  |
| --- | --- | --- |
| **Тестовый метод** | **Входные параметры** | **Описание тестового случая** |
| TestGetParameter\_CorrectValue  (double expected, ParametersType parameter) | ParametersType.DiceWidth, 30 | Получение корректного значения параметра |
| ParametersType.DiceHeight, 60 |
| ParametersType.DiceThickness, 10 |
| ParametersType.DredgingDiameter, 8 |
| ParametersType.EdgeWidth, 3 |
| TestParametersListGet\_GoodScenario() |  | Позитивный тест геттера ParametersList |
| TestIndexerGet\_ReturnValue() |  | При получении значения с помощью индексатора возвращается соответствующий параметр |
| TestIndexerSet\_GoodScenario() |  | При установке значения с помощью индексатора, параметр записывается в коллекцию |

Продолжение таблицы А.1

|  |  |  |
| --- | --- | --- |
| **Тестовый метод** | **Входные параметры** | **Описание тестового случая** |
| TestHasErrorsGet\_OneError() |  | При установке значения с помощью индексатора, параметр записывается в коллекцию |
| TestGetCubeDredging\_CorrectValue() |  | Получение корректного значения параметра CubeDredging |
| TestGetCubeEdge\_CorrectValue() |  | Получение корректного значения параметра CubeEdge |
| TestSetCubeEdge\_CorrectValue() |  | Установление корректного значения параметра CubeEdge |
| TestNameGet\_GoodScenario() |  | Позитивный тест геттера Name |
| TestMinGet\_GoodScenario() |  | Позитивный тест геттера Min |
| TestMaxGet\_GoodScenario() |  | Позитивный тест геттера Max |

Продолжение таблицы А.1

|  |  |  |
| --- | --- | --- |
| **Тестовый метод** | **Входные параметры** | **Описание тестового случая** |
| TestValueGetSet\_GoodScenario() |  | Позитивный тест геттера и сеттера Value |
| TestValueSet\_ThrownExceptions(int value) | 10 | При установке значения меньше минимума выбрасывает исключение |
| 70 | При установке значения больше максимума выбрасывает исключение |
| TestEqualsAndClone\_GoodScenario\_ReturnTrue() |  | При сравнении одинаковых объектов возвращается истина |
| TestEquals\_DifferentValues\_ReturnFalse() |  | При сравнении различных объектов возвращается ложь |
| TestEquals\_NullValue\_ReturnFalse() |  | При сравнении с нулевым объектом возвращается ложь |