Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Кафедра компьютерных систем в управлении и проектировании (КСУП)

**Проект системы**

По дисциплине «Основы разработки САПР»

1. Выдано: студенту группы 589-2 Кравченко Кириллу Андреевичу
2. Тема: разработка плагина “Зубило” для САПР Компас 3D
3. Срок сдачи готовой работы: 31 декабря 2022 г.
4. Исходные данные для работы:

Требования к программному обеспечению:

* Microsoft Windows 10 (64-разрядная версия);
* язык программирования C# с использованием платформы .NET Framework 4.7.2;
* среда разработки Visual Studio 2019;
* плагин для программы Компас 2021;
* библиотека для тестирования NUnit 3.13.2;
* система контроля версии Git.

Требование к аппаратному обеспечению:

* 64-разрядная версия операционной системы;
* Многоядерный процессор(4 ядра и больше) с тактовой частотой 3 ГГц и выше;
* 8 ГБ оперативной памяти и более;
* Видеокарта NVIDIA с поддержкой OpenGL 4.5, c 2 ГБ видеопамяти и более;
* экран расширением 1920 х 1080 пикселов и более.

1. **Введение**

В настоящее время проектирование в своем понимании представляет собой автоматизированный процесс и в некотором роде программно-аппаратный. Проектировщику, который занимается разработкой сложного механизма, или устройства, требующего больших расчетов, математических вычислений при построении модели и высокой точности, подходят системы автоматизации проектных решений — САПР [1].

САПР позволяют уменьшить финансовые затраты на разработку макета (модели) проекта (объекта), а также сократить время, которое тратит проектировщик на создание модели объекта и составление проектной документации.

В каждой крупной САПР есть свой средства для разработки, которые предоставляются с целью дать возможность разработчикам расширить функционал данной системы под свои конкретные нужды. Данным средством является API — программируемый интерфейс приложения [2]. Это набор готовых средств: классов, процедур, функций, структур и т.д. API позволяет определить функциональность, которую предоставляет приложение, при этом абстрагируясь от того, как она реализована.

Расширение функционала в основном подразумевает разработку плагина или библиотеки на основе предоставленного API. В данном курсовом проекте стоит задача разработки плагина для построения 3D модели болта с гайкой в автоматизированном режиме. Плагин — независимо компилируемый программный модуль, динамически подключаемый к основной программе, предназначенный для расширения или использования ее возможностей [3].

В качестве системы, которая предоставляет API и для которой стоит задача разработать плагин, была взята САПР «КОМПАС-3D» версии 15.

1. **Постановка и анализ задачи**
   1. **Описание САПР**
      1. **Описание программы**

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 [4]. Ключевой особенностью продукта является использование собственного математического ядра С3D 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. **Описание API**

В КОМПАС на данный момент существуют API двух версий: API 5 и API 7. [5]. Обе версии реализуют различные функции системы и взаимно дополняют друг друга. Отсюда очевидно, что 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 является KompasObject. Получить указатель на этот интерфейс (если быть точным, на интерфейс приложения API 5) 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 (интерфейсы динамического массива, работы с математическими функциями, библиотек моделей или фрагментов и различных структур параметров определенного типа).

Ниже в таблице 1.1 представлены свойства и методы интерфейса KompasObject, которые были использованы при разработке плагина.

Таблица 1.1 – Методы и свойства интерфейса KompasObject, используемые при разработке плагина

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Document3D() | ksDocument | Метод для получения указателя на интерфейс трехмерного графического документа (детали или сборки) |
| GetParamStruct(short structType) | StructType2D | Метод для получения указателя на интерфейс графического документа (чертежа или фрагмента) |
| Visible | bool | Свойство видимости приложения |
| Quit() |  | Метод для закрытия активного окна приложения КОМПАС |

В таблице 1.2 представлены методы интерфейса ksEntity, которые были использованы при разработке плагина.

Таблица 1.2 – Методы интерфейса ksEntity, используемые при разработке плагина

|  |  |  |
| --- | --- | --- |
| Название | Тип | 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) |

В таблице 1.3 представлены свойства и методы интерфейса ksDocument2D, которые были использованы при разработке плагина.

Таблица 1.3 – Методы интерфейса ksDocument2D, используемые при разработке плагина

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ksLineSeg(double x1, double y1, double x2, double y2, int style) | int | Получить указатель на отрезок на двумерной плоскости либо 0 в случае ошибки |
| ksRegularPolygon(  ksRegularPolygonParam param, int style) | int | Получить указатель на многоугольник на двумерной плоскости либо 0 в случае ошибки |
| ksRectangle(  ksRectangleParam param, int style) | int | Получить указатель на прямоугольник на двумерной плоскости либо 0 в случае ошибки |
| ksCircle(  double xc, double yc, double rad, int style) | int | Получить указатель на окружность на двумерной плоскости либо 0 в случае ошибки |

В таблице 1.4 представлены свойства и методы интерфейса ksDocument3D, которые были использованы при разработке плагина.

Таблица 1.4 – Методы интерфейса ksDocument3D, используемые при разработке плагина

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Create (bool invisible, bool \_typeDoc) | bool | Создать документ-модель (деталь или сборку) |
| GetPart(int type) | ksPart | Получить указатель на интерфейс компонента в соответствии с заданным типом |

В таблице 1.5 представлены методы интерфейса ksPart, которые были использованы при разработке плагина.

Таблица 1.5 – Свойства и методы интерфейса ksPart, используемые при разработке плагина

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

В таблице 1.6 представлены типы объектов документа-модели, которые были использованы при разработке 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\_baseExtrusion | Базовая операция выдавливания | ksBaseExtrusionDefinition |
| o3d\_cutExtrusion | Вырезать выдавливанием | ksCutExtrusionDefinition |
| o3d\_baseLoft | Создание элемента по сечениям | ksBaseLoftDefinition |
| o3d\_baseEvolution | Создание кинематического элемента | ksBaseEvolutionDefinition |
| o3d\_cutEvolution | Вырезать кинематический элемент | ksCutEvolutionDefinition |

1. **Описание объекта проектирования**

Изображение моделируемого объекта в 2D:

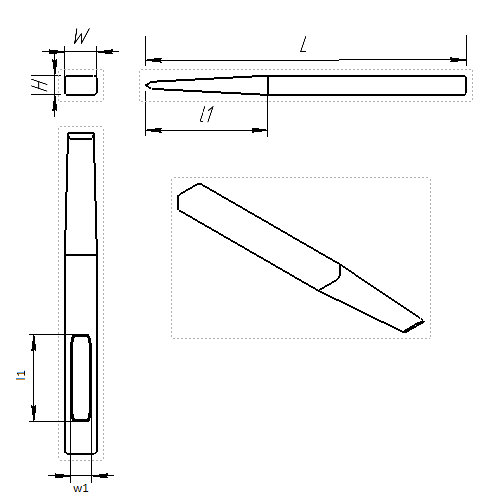


Рисунок 1 – 2D модель ящика

Изображение моделируемого объекта в 3D:

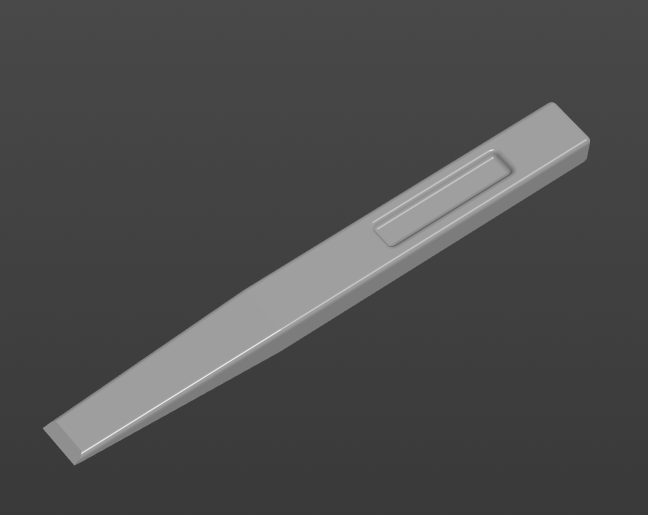


Рисунок 2 – 3D модель ящика

Измеряемые параметры для плагина:

* W – Ширина зубила (10 – 30 мм);
* L – Длина зубила (100 – 300 мм), зависит от ширины зубила, границы вычисляются по формуле L **=** 10W;
* H – Высота зубила (6 – 24 мм), зависит от ширины зубила, границы вычисляются по формуле 0.6W **⩾** H **⩾** 0.8W.
* w1 – Ширина внутреннего выреза зубила (5 – 15 мм), максимальная величина ширины внутреннего выреза зубила w1max рассчитывается по формуле 0.5W, или 0.5 \* l2;
* l1 – Длина лезвия зубила (40 - 150), зависит от длины зубила, границы размеров вычисляются по формуле 0.4L **⩾** l1 **⩾** 0.5L;
* l2 – Длина внутреннего выреза зубила (10 – 75 мм), максимальная величина внутреннего выреза зубила l1max рассчитывается по формуле: 0.25L,

расстояние от края ручки вычисляется по формуле: 0.15L;

Не измеряемые параметры:

- угол фаски лезвия вычисляется по формуле

;

- угол боковой фаски лезвия равен ;

- угол фаски наконечника равен 34º, длина = 1мм;

- радиус боковых скруглений равен 1;

Назначение программы:

Программа предназначена для автоматизации моделирования детали «Зубило»

Плагин позволяет пользователю ввести вышеперечисленные значения через графический интерфейс. В программе предусмотрена проверка корректности введенных данных и сообщение пользователю о неправильно заполненных полях с помощью цветового выделения и всплывающих подсказок.

При запуске моделирования с некорректными значениями программа выводит сообщение об ошибке и отменяет построение модели.

При правильно введенных значениях результатом работы программы будет созданная по ним модель ящика для деталей. Размеры одной ячейки ящика рассчитываются из введенных пользователем параметров автоматически.

1. **Проект системы**

Для графического описания абстрактной модели проекта, а также пользовательского взаимодействия (сценария действий) использован стандарт UML[10].

UML – это 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-моделей возможна генерация кода.

При использовании UML была построена диаграмма классов.

* 1. **Диаграмма классов**

Диаграмма классов для данного проекта представлена на рисунке 3.1.1.

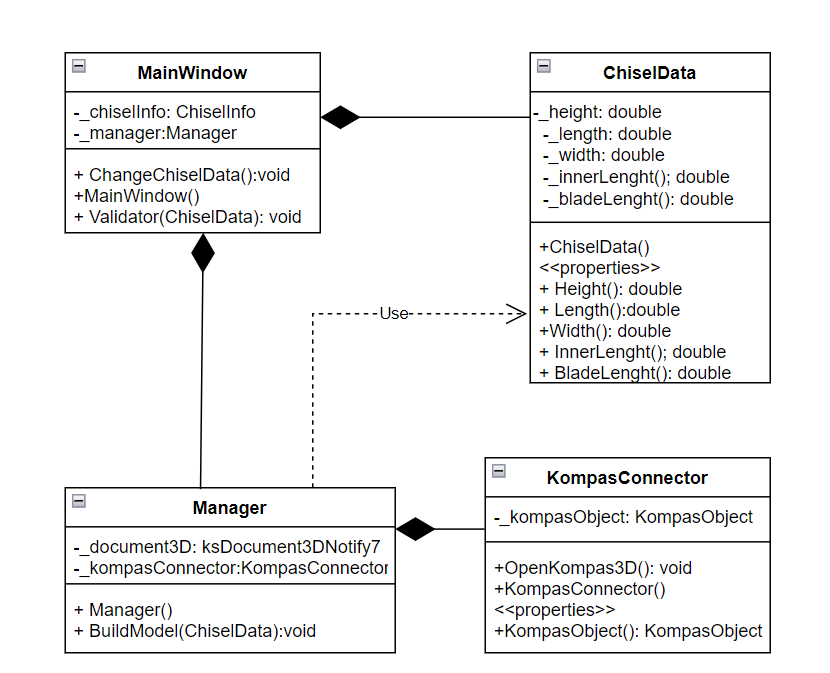


Рисунок 3.1.1 – UML-диаграмма классов

1) MainWindow – класс диалогового окна, который обеспечивает взаимодействие между пользователем и программой.

2) ChiselData − класс, хранящий в себе все параметры 3D-модели;

3) Manager − класс, осуществляющий инициализацию, создание модели и валидацию введенных данных;

4) KompasConnector – класс для работы с API КОМПАС 3D;

Класс MainWindow композирует классы ChiselData и Manager.

Класс Manager композирует класс CompasConnector и использует класс ChiselData.

В таблице 3.1.1 представлено описание полей и методов класса MainWindow.

Таблица 3.1.1 – Описание полей и методов класса MainWindow.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| \_chiselData: ChiselData | Данные для построения зубила |
| \_manager: Manager | Взаимодействие между плагином и КОМПАС 3D |
| MainWindow() | Конструктор |
| ChangeChiselData() | Изменение данных для построения зубила |
| Validator(ChiselData): void | Валидация введенных данных |

В таблице 3.1.2 представлено описание свойств и методов класса RookInfo.

Таблица 3.1.2 – Описание свойств и методов класса ChiselData.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| Height: float | Высота зубила |
| Leight: float | Длина зубила |
| Width: float | Ширина зубила |
| InnerLenght: float | Длина внутреннего выреза зубила |
| BladeLenght: float | Длина лезвия зубила |
| ChiselData() | Конструктор |

В таблице 3.1.3 представлено описание полей и методов класса Manager.

Таблица 3.1.3 – Описание полей и методов класса Manager.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| \_document3D:ksDocument3DNotify7 | Документ, содержащий 3D-модель или сборку |
| \_kompasConnector:KompasConnector | Подключение к компас 3D |
| Manager() | Конструктор |
| BuildModel(ChiselData):void | Создание модели зубила |

В таблице 3.1.4 представлено описание полей и методов класса KompasConnector.

Таблица 3.1.4 – Описание свойств и методов класса KompasConnector.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| openKompas3D(): void | Открытие Компаса |
| KompasConnector() | Конструктор |
| \_kompasObject: KompasObject | Интерфейс API-системы КОМПАС |

* 1. **Макеты пользовательского интерфейса**

Плагин представляет собой пользовательскую форму с ячейками ввода параметров. Сохранение заданных параметров осуществляется кнопкой «Сохранить».

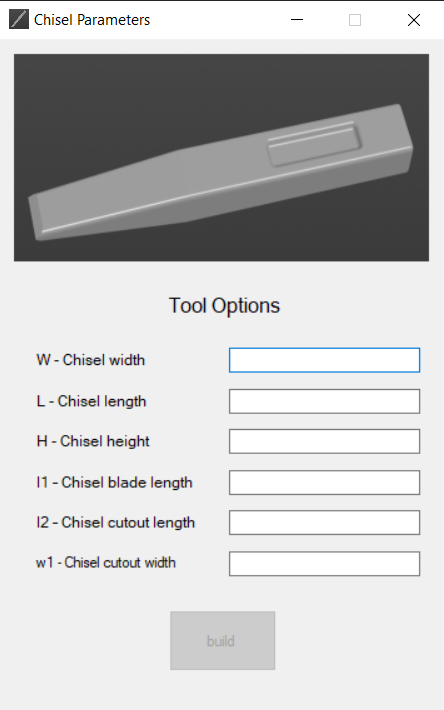
****

Рисунок 3.2.1 – Пользовательский интерфейс при запуске программы

* 1. **Описание программы для пользователя**

Чтобы построить 3Д модель зубила, используя данный плагин, необходимо запустить плагин. В запущенном окне ввести параметры, при вводе некорректных значений программа выдаст сообщение об ошибке, в котором будут указаны некорректные значения с пояснениями и возможными диапазонами корректных значений, а так же поле, с некорректным значением будет выделено красной заливкой. После корректного ввода всех значений кнопка «Build» становится активной, и необходимо нажать на кнопку «Build», чтобы запустить плагин построения модели.

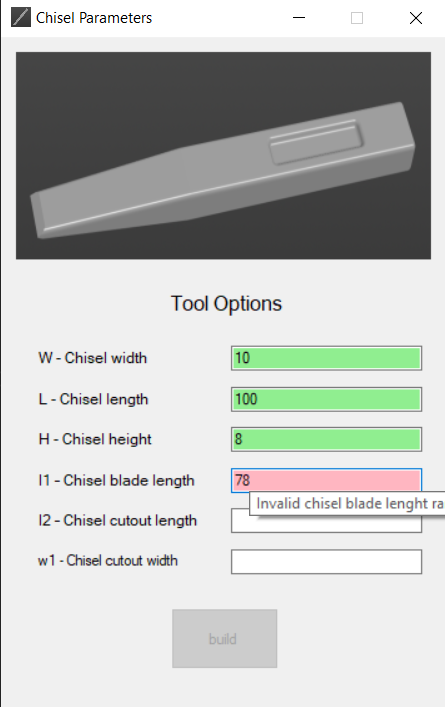


Рисунок 3.3.1 Пользовательский интерфейс при некорректно введенном значении

При вводе некорректных значений поле выделяется красной заливкой и при наведении на поле высвечивается ошибка с пояснениями и возможным решением.



Рисунок 3.3.2 Текст ошибки при наведение на поле с некорректно введенным значением