Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

**РАЗРАБОТКА ПЛАГИНА «ДЕТСКАЯ ГОРКА» ДЛЯ «КОМПАС-3D V18»**

Проект системы по лабораторному проекту

по дисциплине «ОСНОВЫ РАЗРАБОТКИ САПР»

|  |
| --- |
| Выполнил:  студент гр. 586-2  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.В. Евсюков  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020г. |
| Руководитель:  к.т.н., доцент каф. КСУП:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Калентьев  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020г. |

Содержание

[1 Описание САПР 3](#_Toc38483339)

[1.1 Описание программы 3](#_Toc38483340)

[1.2 Описание API 3](#_Toc38483341)

[1.3 Обзор аналогов 15](#_Toc38483342)

[Anthen приложение к AutoCAD для проектирования 3D-Конструкции. 15](#_Toc38483343)

[2 Описание предмета проектирования 16](#_Toc38483344)

[3 Проект программы 18](#_Toc38483345)

[3.1 Диаграмма вариантов использования (Use Cases) 18](#_Toc38483346)

[3.2 Диаграмма классов 20](#_Toc38483347)

[3.3 Макет пользовательского интерфейса 21](#_Toc38483348)

[Список литературы 22](#_Toc38483349)

# 1 Описание САПР

# 1.1 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 [1]. 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.

# 1.2 Описание API

Взаимодействие внешнего приложения или подключаемого модуля с системой КОМПАС (с функциями моделирования, математическими функциями ядра системы и пр.) осуществляется посредством программных интерфейсов (API).

На данный момент в КОМПАС существуют API двух версий: API 5 и API 7 [2]. Обе версии реализуют различные функции системы и не являются взаимозаменяемыми.

При создании подключаемых модулей в большей мере используется интерфейс API 5.

Для начала взаимодействия с интерфейсом API необходимо выполнить следующие методы:

* Activator.CreateInstance(Type) – создание экземпляра указанного типа, используя конструктор этого типа без параметров. (метод пространства имен System);
* UIElement.ActivateControllerAPI() – активация контроллера API.

Необходимо использовать свойство видимости UI:

* UIElement.Visibility – получает или задает видимость пользовательского интерфейса данного элемента. Это свойство зависимостей (Пространство имен: System.Windows).

Методы взаимодействия:

* **Cr****eate** – создать документ-модель (деталь или сборку);

Синтаксис Autimation:

BOOL Create (BOOL invisible, BOOL typeDoc);

Синтаксис COM:

BOOL Create (BOOL invisible, BOOL \_typeDoc);

Входные параметры:

invisible – признак режима редактирования документа (TRUE – невидимый режим, FALSE – видимый режим);

typeDoc – тип документа (TRUE - деталь, FALSE – сборка).

Возвращаемое значение:

True – в случае успешного завершения.

* **Create** – создать объект в модели;

Синтаксис Automation:

BOOL Create();

Синтаксис COM:

BOOL Create();

Возвращаемое значение:

TRUE – в случае успешного завершения.

Примечание:

Данный метод необходимо вызвать для создания объекта в модели после создания интерфейса с помощью метода NewEntity и изменения параметров создаваемого объекта;

Если до вызова данного метода свойства объекта не изменены, то объект создается с параметрами, принятыми по умолчанию;

Если объект уже создан, то функция вызовет метод Update.

* **ActiveDocument3D** – получить указатель на интерфейс текущего документа трехмерной модели;

Синтаксис Automation:

LPDISPATCH ActiveDocument3D();

Возвращаемое значение:

Указатель на интерфейс документа трехмерной модели ksDocument3D.

Синтаксис COM:

Экспертная функция LPDOCUMENT3D ksGetActive3dDocument();

Возвращаемое значение:

Указатель на интерфейс документа трехмерной модели IDocument3D.

Примечание:

Если документ трехмерной модели не активен, функция возвращает NULL.

* **Document3D** – получить указатель на интерфейс документа трехмерной модели;

Синтаксис Automation:

LPDISPATCH Document3D();

Возвращаемое значение:

Указатель на интерфейс документа трехмерной модели ksDocument3D.

Синтаксис COM:

Экспертная функция LPDOCUMENT3D ksGet3dDocument();

Возвращаемое значение:

Указатель на интерфейс документа трехмерной модели IDocument3D.

* **GetPart** – получить указатель на интерфейс компонента в соответствии с заданным типом;

Синтаксис Automation:

LPDISPATCH GetPart(long type);

Синтаксис COM:

LPPART GetPart(int type);

Входные параметры:

type - тип компонента из перечисления “Типы компонентов”.

Типы компонентов:

pInPlace\_Part – компонент, редактируемый на месте;

pNew\_Part – новый компонент;

pEdit\_Part – редактируемый компоненты;

pTop\_Part – главный компонент, в составе которого находится новый или редактируемый или указанный компонент.

Возвращаемое значение:

Указатель на интерфейс компонента ksPart или IPart.

Примечание:

Тип задается для нового, редактируемого или главного компонента либо тип равен номеру компонента в документе;

Функция используется, чтобы получить доступ к компоненту документа;

Деталь или сборка являются компонентами. Сборка, в свою очередь, состоит из компонентов - деталей и подсборок.

* **GetDefaultEntity** –получить указатель на интерфейс объекта, создаваемого системой по умолчанию;

Синтаксис Automation:

LPDISPATCH GetDefaultEntity(short objType);

Синтаксис COM:

LPENTITY GetDefaultEntity(short objType);

Входные параметры:

objType – тип объекта.

Возвращаемое значение:

Указатель на интерфейс ksEntity или IEntity.

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

* **N****e****wEntity** – создать новый интерфейс объекта и получить указатель на него;

Синтаксис Automation:

LPDISPATCH NewEntity(short objType);

Синтаксис COM:

LPENTITY NewEntity(short objType);

Входные параметры:

objType – тип объекта.

Возвращаемое значение:

Указатель на интерфейс ksEntity или IEntity.

Примечание:

Реальный объект создается в модели после вызова метода Create.

* **B****egin****Edit** – войти в режим редактирования эскиза;

Синтаксис Automation:

BeginEdit();

Возвращаемое значение:

SketchDoc – указатель на интерфейс IFragmentDocument.

Синтаксис COM:

HRESULT BeginEdit([out, retval] IFragmentDocument \*\* SketchDoc);

Возвращаемое значение:

SketchDoc – указатель на интерфейс IFragmentDocument.

Примечание:

После редактирования необходимо вызвать функцию ISketch::EndEdit.

* **EndEdit** – выйти из режима редактирования эскиза;

Синтаксис Automation:

BOOL EndEdit();

Синтаксис COM:

HRESULT EndEdit([out, retval] VARIANT\_BOOL \* res);

Возвращаемое значение:

TRUE – в случае успешного завершения;

FALSE – в случае неудачи.

* **ksLineSeg** – создать отрезок;
* Синтаксис Automation:

long ksLineSeg(double x1, double y1, double x2,   
 double y2, long style);

Входные параметры:

x1, y1 – координаты первой точки отрезка;

x2, y2 – координаты второй точки отрезка;

style – стиль линии.

Возвращаемое значение:

Указатель на отрезок – в случае удачного завершения;

0 – в случае неудачи.

* **ExtrusionParam** – получить указатель на интерфейс параметров элемента выдавливания;

Синтаксис Automation:

LPDISPATCH ExtrusionParam();

Синтаксис COM:

LPEXTRUSIONPARAM ExtrusionParam();

Возвращаемое значение:

Указатель на интерфейс объекта ksExtrusionParam или IExtrusionParam.

* **SetSketch** – изменить указатель на интерфейс эскиза элемента;

Синтаксис Automation:

BOOL SetSketch(LPDISPATCH sketch);

Синтаксис COM:

BOOL SetSketch(LPENTITY sketch);

Входные параметры:

sketch – указатель на интерфейс эскиза ksEntity или IEntity.

Возвращаемое значение:

TRUE – в случае успешного завершения.

* **ksCircle** – создать окружность;

Синтаксис Automation:

long ksCircle(double xc, double yc, double rad,   
long style);

Входные параметры:

xc, yc – координаты центра окружности;

rad – радиус окружности;

style – стиль линии.

Возвращаемое значение:

указатель на окружность – в случае удачного завершения;

0 – в случае неудачи.

* **array** – получить указатель на интерфейс массива скругляемых объектов;

Синтаксис Automation:

LPDISPATCH Array();

Синтаксис COM:

LPENTITYCOLLECTION Array();

Возвращаемое значение:

Указатель на интерфейс массива объектов ksEntityCollection или IEntityCollection.

Примечание:

В возвращаемом массиве включен контроль, не позволяющий добавить в массив нулевые указатели на элементы.

* **A****ddMateConstraint** – добавить сопряжение в сборку;

Синтаксис Automation:

BOOL AddMateConstraint(long constraintType,   
LPDISPATCH object1, LPDISPATCH object2, short direction,   
short fixed, double value);

Синтаксис COM:

BOOL AddMateConstraint(long constraintType,   
LPENTITY obj1, LPENTITY obj2, short direction, short fixed,   
double val);

Входные параметры:

|  |  |
| --- | --- |
| constraintType | - тип сопряже­ния из перечи­сления MateConstraintType; |
| object1 | - указатель на интерфейс первого объ­екта, на кото­рый наклады­вается сопряжение (ksEntity или IEntity); |
| object2 | - указатель на интерфейс второго объек­та, на который накладывает­ся сопряжение (ksEntity или iEntity); |
| val | - параметр для ограничений (расстояние или угол меж­ду объектами). |
| direction | - ориентация  (1 - объекты однонаправ­ленные,   0 - направле­ние не учиты­вается,  -1 - объекты разнонаправ­ленные); |
| fixed | - признак фик­сации деталей перед выпол­нением  (0 - детали не фиксируются,   1 - фиксирует­ся первая де­таль,  2 - фиксирует­ся вторая де­таль); |

Возвращаемое значение:

TRUE - в случае успешного завершения.

Примечание:

Направление задаётся знаком параметра val.

* **M****at****eConstraintType** – типы сопряжений;

|  |  |  |
| --- | --- | --- |
| mc\_Coincidence | 0 | совпадение объектов |
| mc\_Parallel | 1 | параллельность |
| mc\_Perpendicular | 2 | перпендикулярность |
| mc\_Tangency | 3 | касательность |
| mc\_Concentric | 4 | концентричность |
| mc\_Distance | 5 | постоянное расстояние между объек­тами |
| mc\_Angle | 6 | постоянный угол между объектами |
| mc\_InPlace | 7 | создание компонента "на месте" (эквивалентно совпадению системы координат плоскости, на которой со­здается компонент, и системы коорди­нат плоскости первого эскиза этого компонента) |

* **ksCreateInsertionFragment** – создать вставку фрагмента;

Синтаксис Automation:

int LIB\_FUNC ksCreateInsertionFragment(char\*fileName);

Входные параметры:

fileName – полное имя файла фрагмента.

Возвращаемое значение:

1 – в случае удачи.

* **SaveAs** – сохранить документ – модель (деталь или сборку) с новым именем;

Синтаксис Automation:

BOOL SaveAs(BSTR fileName);

Синтаксис COM:

BOOL SaveAs(LPOLESTR fileName);

Входные параметры:

filename – имя файла документа.

Возвращаемое значение:

TRUE – в случае успешного завершения.

* **SaveAsEx** – сохранить документ с новым именем файла;

Синтаксис Automation:

BOOL SaveAsEx(BSTR fileName, long saveMode);

Синтаксис COM:

BOOL SaveAsEx(LPOLESTR fileName, long saveMode);

Входные параметры:

|  |  |
| --- | --- |
| fileName | - новое имя файла доку­мента; |
| saveMode | * версия для сохранения: 1 – в предыду­щую версию; 0 – в текущую версию; 1 – в версию 5.11. |

Возвращаемое значение:

TRUE – в случае успешного завершения;

FALSE – в случае неудачи.

Примечание:

Если fileName = NULL, то используется имя файла из документа. Если же и в документе отсутствует имя файла, то выводится ошибка.

Используемые свойства:

* **thin** – признак формирования тонкой стенки элемента;

Тип данных: BOOL.

Значения свойства:

|  |  |
| --- | --- |
| TRUE | – тонкая стенка формируется; |
| FALSE | – тонкая стенка не формируется. |

Синтаксис Automation:

|  |  |
| --- | --- |
| thin = iThinParam.thin | Получить свойство |
| iThinParam.thin = thin | Установить свойство |
| thin = iThinParam.GetThin() | Получить свойство |
| iThinParam.SetThin(thin) | Установить свойство |

* **radius** – радиус скругления;

Тип данных: double.

Синтаксис Automation:

|  |  |
| --- | --- |
| radius = iFilletDefinition.radius | Получить свойство |
| iFilletDefinition.radius = radius | Установить свойство |
| radius= iFilletDefinition.GetRadius() | Получить свойство |
| iFilletDefinition.SetRadius(radius) | Установить свойство |

* **directionType** – направление выдавливания;

Тип данных: short.

Синтаксис Automation:

|  |  |
| --- | --- |
| directionType= iBaseExtrusion.directionType | Получить свойство |
| iBaseExtrusion.directionType= directionType | Установить свойство |
| directionType= iBaseExtrusion.GetDirectionType() | Получить свойство |
| iBaseExtrusion.SetDirectionType  (directionType) | Установить свойство |

Значения свойства:

dtNormal – прямое направление (для тонкой стенки наружу);

dtReverse – обратное направление (для тонкой стенки внутри);

dtBoth – в обе стороны;

dtMiddlePlane – от средней плоскости.

Примечание:

1.Нормаль, проведенная к грани, всегда направлена наружу ("из тела детали");

2.Прямое направление совпадает с нормалью, проведенной к плоскости эскиза;

3.Для вырезаемого элемента выдавливания направление противоположно нормали.

* **Depth** – глубина выдавливания;

Тип данных: double.

Синтаксис Automation:

|  |  |
| --- | --- |
| Depth = iObject.Depth(normal); | Получить свойство |
| iObject.Depth(normal) = Depth; | Установить свойство |
| Depth = iObject.GetDepth(normal); | Получить свойство |
| iObject.SetDepth(normal, Depth); | Установить свойство |

Синтаксис COM:

|  |  |
| --- | --- |
| iObject->get\_Depth(normal,&Depth); | Получить свойство |
| iObject->put\_Depth(normal,Depth); | Установить свойство |

Входные параметры:

TRUE - глуби­на выдавлива­ния в прямом направлении;

FALSE - глу­бина выдавли­вания в обрат­ном направлении.

Примечание:

Позволяет считывать и устанавливать глубину выдавливания.

* **Plane** – опорная плоскость;

Тип данных: указатель на интерфейс IModelObject.

Синтаксис Automation:

|  |  |
| --- | --- |
| Plane = iObject.Plane; | Получить свойство |
| iObject.Plane = Plane; | Установить свойство |
| Plane = iObject.GetPlane(); | Получить свойство |
| iObject.SetPlane(Plane); | Установить свойство |

Синтаксис COM:

iObject->get\_Plane(&Plane); Получить свойство

iObject->put\_Plane(Plane); Установить свойство

* **A****ngle** –угол поворота эскиза относительно проекции системы координат модели на плоскость эскиза (в градусах).

Тип данных: double.

Синтаксис Automation:

|  |  |
| --- | --- |
| Angle = iObject.Angle; | Получить свойство |
| iObject.Angle = Angle; | Установить свойство |
| Angle = iObject.GetAngle(); | Получить свойство |
| iObject.SetAngle(Angle); | Установить свойство |

Синтаксис COM:

iObject->get\_Angle(&Angle); Получить свойство

iObject->put\_Angle(Angle); Установить свойство

# 1.3 Обзор аналогов

# Anthen приложение к AutoCAD для проектирования 3D-Конструкции.

Основой для создания всевозможных 3D-конструкций служит осевая модель, осям которой могут назначаться отдельные профили или целые группы профилей [3]. Работу существенно упрощает команда анализа осевой модели, в ходе выполнения которой определяются плоскости заполнений и наружная сторона. Благодаря анализу ATHEN знает все углы, размеры полей и выверку в модели. Таким образом, можно, применяя свои собственные группы, достичь высокой степени автоматизации, что позволяет создать, например стоечно-ригельный фасад со всеми раскроями.

Модуль «Лестница» позволяет конструировать лестницы и получать спецификации элементов. Результат может выводиться как в 2D, так и в 3D и охватывает план лестницы, линию хода, тетивы и ступени.

На рисунке 1.3 представлен интерфейс программы.

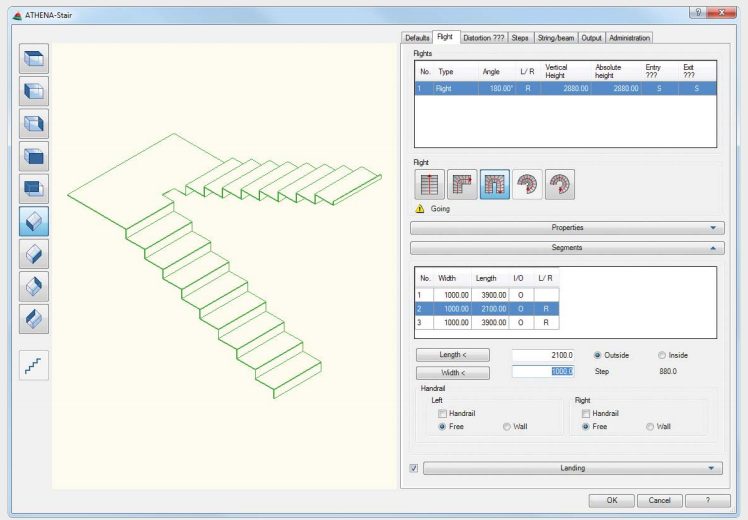
****

Рисунок 1.3 – интерфейс программы

# 2 Описание предмета проектирования

Предметом проектирования является Детская горка.

Детская горка — сооружение с гладким наклонным спуском и лесенкой, позволяющей забираться на верхнюю площадку и скатываться вниз [4]. Предназначено для развлечения и спортивного развития детей, устанавливается на [детских площадках](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%82%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D0%BB%D0%BE%D1%89%D0%B0%D0%B4%D0%BA%D0%B0), в парках и других местах детского отдыха.

На рисунке 2.1 представлена 3D модель детской горки.

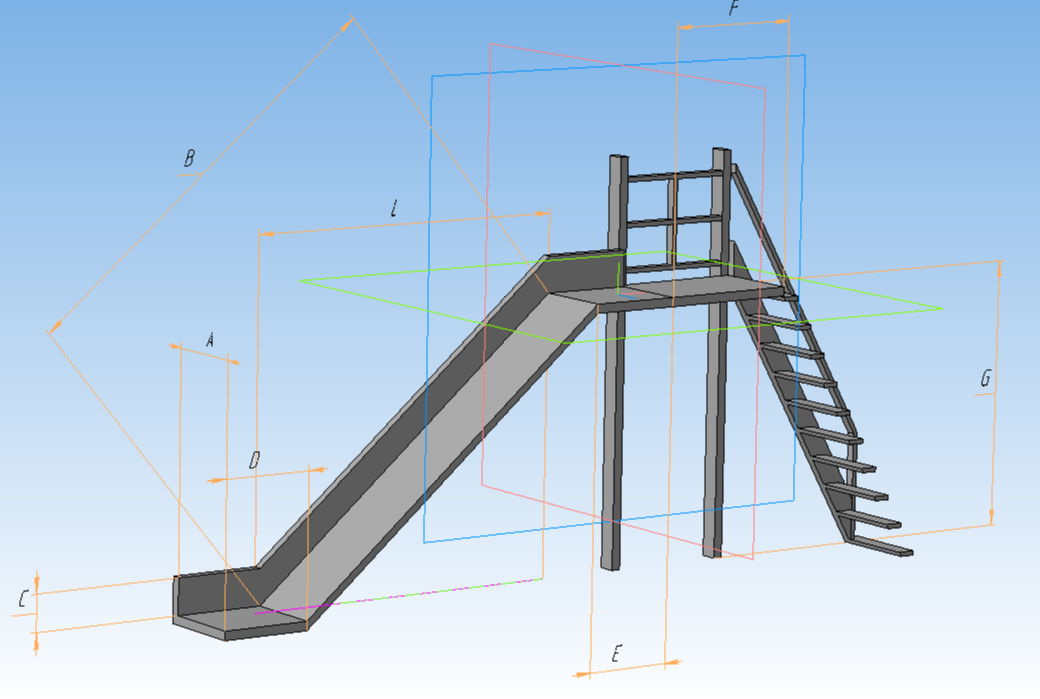


Рисунок 2.1 – 3D модель детской горки

Плагин должен обладать графическим интерфейсом для ввода следующих параметров:

* длина платформы F (от 40 до 120 см);
* длина начала горки E (от 20 до 60 см);
* длина конца горки D (от 20 до 60 см);
* расстояние горки от начала (E) до конца (D) L (от 80 до 240 см);
* ширина A (от 20 до 80 см);
* высота бордюра C (от 8 до 32 см);
* высота платформы G (от 40 до 160 см).

# 3 Проект программы

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

UML – это язык графического описания для объектного моделирования в области разработки программного обеспечения[5]. 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 были построены: диаграмма использования и диаграмма классов.

# 3.1 Диаграмма вариантов использования (Use Cases)

Диаграмма вариантов использования ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *use case diagram)* в [UML](https://ru.wikipedia.org/wiki/UML) — диаграмма, отражающая отношения между [актерами](https://ru.wikipedia.org/wiki/%D0%90%D0%BA%D1%82%D0%BE%D1%80_(UML)) и [прецедентами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5%D0%BD%D1%82_(UML)) и являющаяся составной частью *модели прецедентов*, позволяющей описать систему на концептуальном уровне[5].

Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних [требований](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E) к системе.

Основное назначение диаграммы — описание функциональности и поведения, позволяющее [заказчику](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%B0%D0%B7%D1%87%D0%B8%D0%BA), [конечному пользователю](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C) и [разработчику](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82) совместно обсуждать проектируемую или существующую [систему](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)).

На рисунке 3.1 представлена диаграмма вариантов использования.

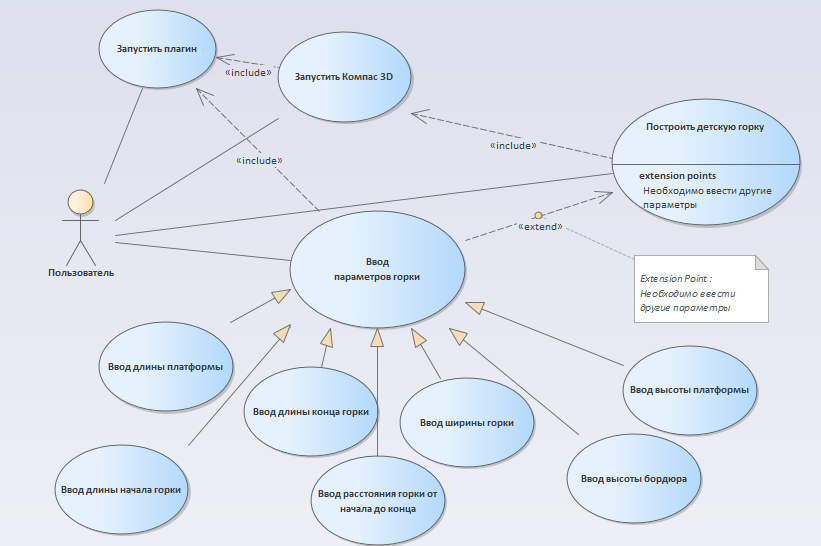


Рисунок 3.1 – Диаграмма вариантов использования

# 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 [5].

На рисунке 3.2. показана диаграмма классов.

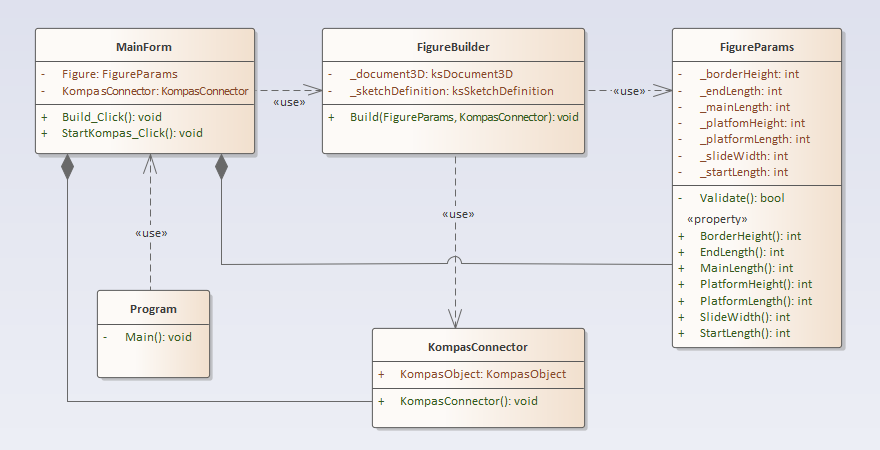


Рисунок 3.2 – Диаграмма классов

Для реализации подсистемы были спроектированы следующие классы:

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

# 3.3 Макет пользовательского интерфейса

После запуска приложения перед пользователем появляется главное окно рисунок 3.3.

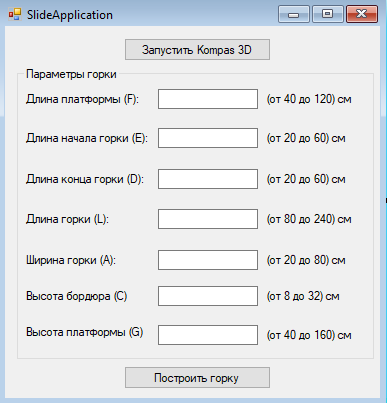


Рисунок 3.3. – Макет пользовательского интерфейса при запуске программы

Кнопка “Запустить Kompas 3D” отвечает за запуск Kompas 3D.

Область “Параметры горки” отвечает за параметры горки. Все поля предварительно заполнены минимально возможными значениями.

Кнопка “Построить горку” отвечает за построение горки. При нажатии кнопки “Построить горку” до нажатия кнопки “Запустить Kompas 3D” будет открываться диалоговое окно с сообщением: для построения необходимо нажать на кнопку запустить Kompas3D. При неправильном вводе параметров горки будет открываться дополнительное диалоговое с ошибками в определенной строке, пример на рисунке 3.4.

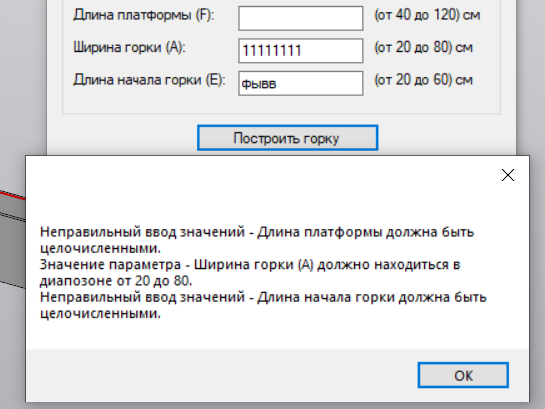


Рисунок 3.4. – Сообщение с ошибками

# Список литературы

КОМПАС-3D: О программе. Официальный сайт САПР КОМПАС [Электронный ресурс]. – Режим доступа: <http://kompas.ru/kompas-3d/about/> (дата обращения 01.03.2020)

Кидрук Максим. КОМПАС-3D V18 / М. Кидрук. – СПб.: Питер, 2009 – 560 с. (дата обращения 01.03.2020)

Плагин Anthena [Электронный ресурс]. — Режим доступа: <https://cad-plan.com/> (дата обращения 01.03.2020)

Детская горка – Википедия. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%82%D1%81%D0%BA%D0%B0%D1%8F_%D0%B3%D0%BE%D1%80%D0%BA%D0%B0> (дата обращения 01.03.2020)

М. Фаулер. UML. Основы, 3-е издание. Книга по UML для начинающих – 2018 – 192 с.(дата обращения: 01.03.2020)