Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждения

высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по дисциплине

«Основы разработки САПР» (ОРСАПР)

Студент гр. 586-2

\_\_\_\_\_\_\_\_\_\_\_\_\_И.В. Евсюков

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020г.

Руководитель

К.т.н., доцент каф. КСУП

\_\_\_\_\_\_\_\_\_\_А.А. Калентьев

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020г.

2020

**РЕФЕРАТ**

Пояснительная записка 21 с., 15 рис., 1 таблицы, 10 источника.

КОМПАС-3D, ПЛАГИН, МОДЕЛЬ ДЕТСКОЙ ГОРКИ, АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ, САПР.

Целью данной работы является разработка плагина, автоматизирующего построение модели детской горки, для системы автоматизированного проектирования КОМПАС-3D 18.1, с помощью интегрированной среды разработки Visual Studio Сommunity 2019.

В процессе работы изучено внешнее устройство детской горки и основные функции системы автоматизированного проектирования «Компас-3D» v 18.

В результате работы был создан плагин, автоматизирующий построение детской горки.

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

**Содержание**

1 Введение 4

2 Постановка и анализ задачи 5

2.1 Описание предмета проектирования 5

2.2 Выбор инструментов и средств реализации 7

2.3 Назначение плагина 7

2.4 Описание аналогов разрабатываемого продукта 8

2.4.1 Anthen приложение к AutoCAD для проектирования 3D.8

3 Описание реализации 9

3.1 Диаграмма прецедентов плагина 9

3.2 Диаграмма классов 11

3.3 Диаграмма пакетов 13

4 Описание программы для пользователя 14

5 Тестирование программы 15

5.1 Функциональное тестирование 15

5.2 Модульное тестирование 16

5.3 Нагрузочное тестирование 18

Заключение 20

Список использованных источников 21

Приложение А 22

**1 Введение**

Автоматизация моделирования имеет огромное значение для развития науки, техники и производства в современном обществе. В настоящее время автоматизация – основной способ повышения производительности и эффективности труда инженерно-технических работников, занимающихся моделированием сложных устройств. Использование автоматизации в проектировании позволяет создавать все более сложные технические объекты и гибко реагировать на появление новых решений и технологий в той или иной области техники.

Практическая реализация методов и идей автоматизированного моделирования происходит в рамках систем автоматизированного проектирования (САПР). Однако мало создать высокопроизводительные современные САПР. Надо уметь их эффективно использовать. Для этого нужны квалифицированные инженеры-пользователи САПР. В рамках современного «компьютеризированного» общества инженер любой специальности, занимающийся разработкой технических устройств, должен уметь использовать средства автоматизированного проектирования. Это позволяет повысить эффективность моделирования, улучшить его качество, снизить материальные затраты и уменьшить число разработчиков.

Таким образом, целью данной работы является разработка плагина, автоматизирующего построение модели детской горки, для системы автоматизированного проектирования «Компас-3D» v 18 [1], с помощью интегрированной среды разработки Visual Studio Сommunity 2019 [3].

Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

**2 Постановка и анализ задачи**

В рамках лабораторных работ в соответствии с технически заданием требовалось разработать плагин, который на основе входных параметров, интегрируя с системой КОМПАС 3D [1], строит деталь «Детская горка». Более того, требовалось, чтоб плагин позволял изменять входные параметры в соответствии с требованиями пользователей, а именно: высота бордюра, длина конца горки, длина горки, высота платформы, длина платформы, ширина горки, длина начала горки, толщина платформы.

**2.1 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 [2]. 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.

Обозначенные параметры с ограничениями:

F – Длина платформы (от 40 до 120 см);

E – Длинна начала горки (от 40 до 120 см);

D – Длина конца горки (от 40 до 120 см);

L – Расстояние длины горки (от 40 до 120 см);

A – Ширина горки (от 40 до 120 см);

С – Высота бордюра (от 40 до 120 см);

G – Высота платформы (от 40 до 120 см);

T – Толщина платформы (от 3 до 10 см).

Рассмотрим чертеж детской горки с размерными выносками в изометрии (рисунок 2.1) и на плоскости XZ (рисунок 2.2):

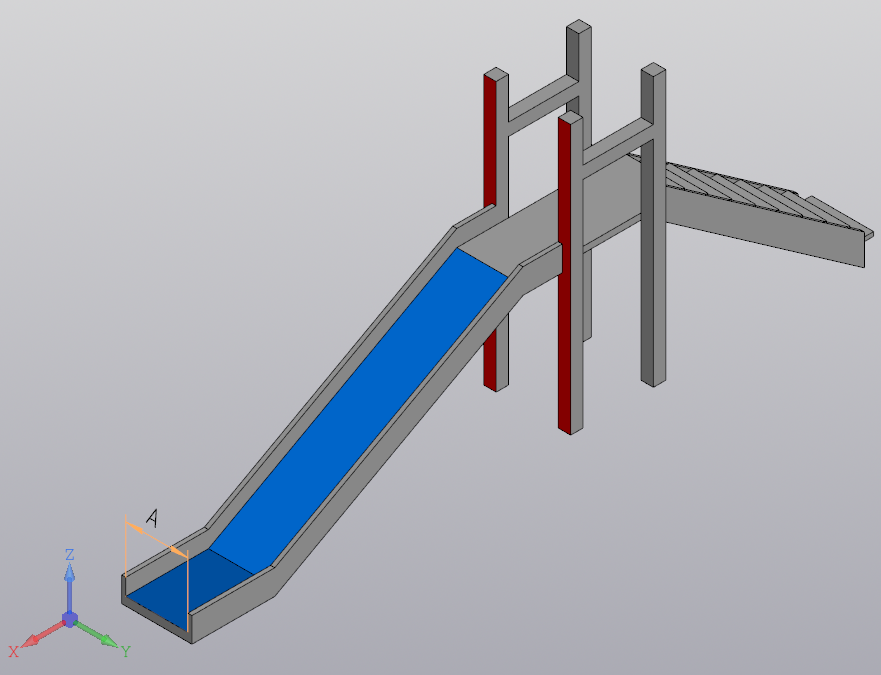


Рисунок 2.1 – Горка в изометрии

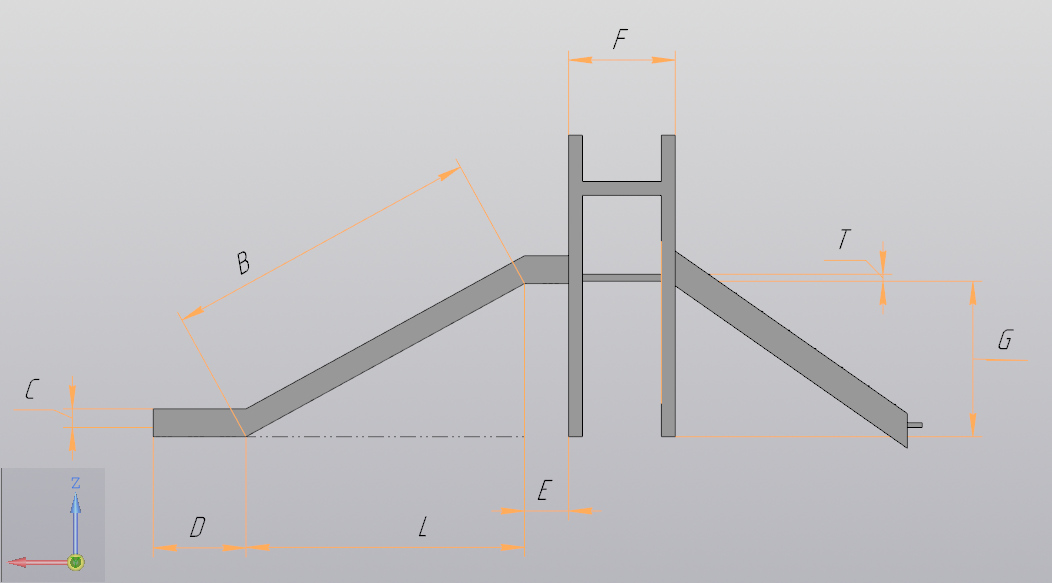


Рисунок 2.2 – Горка вид в плоскости XZ

**2.2 Выбор инструментов и средств реализации**

В связи с требованием технического задания программа выполнена на языке C# в среде Microsoft Visual Studio 2019 [3], с использованием .NET Framework 4.7.0, для системы КОМПАС 3D V18. Инструментом тестирования и создания модульных тестов был выбран стандартный обозреватель тестов среды Microsoft Visual Studio 2019 [3] с тестовым фреймворком NUnit версии 3.12.0.

Для реализации пользовательского интерфейса использовался WinForm.

Взаимодействие плагина с системой КОМПАС 3D [1] осуществляется посредством интерфейсов, называемых API. В КОМПАС 3D на данный момент существует API двух версий API 5 [4] и API 7 [4]. Явных преимуществ между версиями нет, поскольку обе версии реализуют различные функции системы и взаимно дополняют друг друга. Для выполнения лабораторных была выбрана версия API 5, так как для полноценной реализации плагина «Детская горка» достаточно методов и свойств интерфейсов API 5.

**2.3 Назначение плагина**

Назначение разрабатываемого плагина обусловлено быстрым моделированием различных параметров горки. Благодаря данному расширению, проектировщики могут наглядно рассмотреть спроектированную модель, при необходимости перестроить под необходимые им параметры.

# 2.4 Описание аналогов разрабатываемого продукта

# 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 [7]. 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 2.3 представлен интерфейс программы.

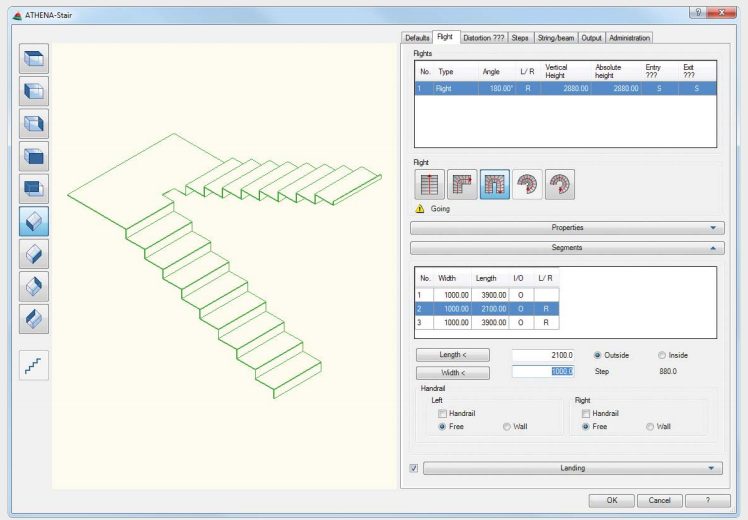
****

Рисунок 2.3 – интерфейс программ

**3 Описание реализации**

На этапе разработки проекта программы для формального описания архитектурной особенности, пользовательского сценария системы был выбран унифицированный язык моделирования (UML) [8]. На основе UML построены: диаграммы вариантов использования диаграммы классов, диаграммы пакетов.

В процессе реализации диаграммы классов, вариантов использования и пакетов были дополнены.

Дополнения были обусловлены нарастанием функционала в реализации мелких детальных особенностей и изменений требований заказчика.

**3.1 Диаграмма прецедентов плагина**

Прецеденты [8] – это технология определения функциональных требований к системе. Работа прецедентов заключается в описании типичных взаимодействий между пользователем системы и самой системой.

Прецеденты представляют собой ценный инструмент для понимания функциональных требований к системе.

Рассмотрим изначальную версию диаграммы прецедентов на рисунке 3.1.

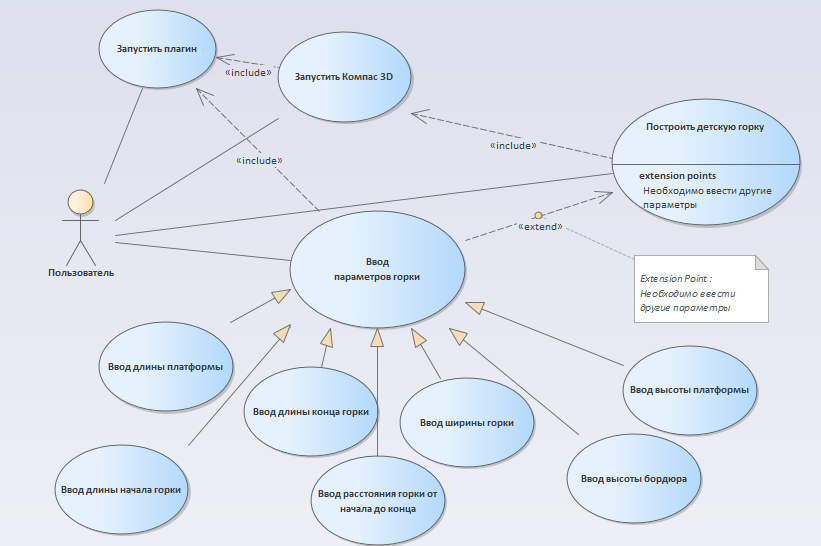


Рисунок 3.1 – Изначальная диаграмма прецедентов плагина

Действующим лицом выступает пользователь.

Для пользователя доступны следующие возможности:

* Запустить плагин;
* Запустить компас;
* Построить детскую горку;
* Ввод параметров горки.

В результате изменений, внесенных заказчиком, диаграмма прецедентов подверглась изменениям.

Добавлены возможности для изменения толщины платформы.

Рассмотрим измененную диаграмму прецедентов плагина на рисунке 3.2.

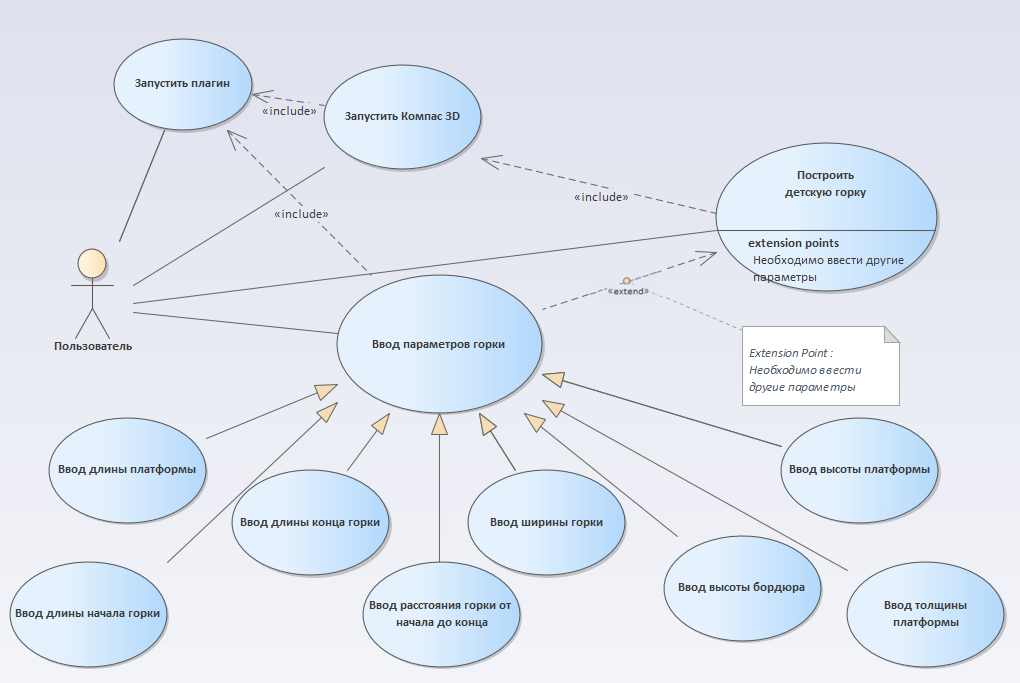


Рисунок 3.2 – Измененная диаграмма прецедентов плагина

**3.2 Диаграмма классов**

Диаграмма классов [8] описывает типы объектов системы и различного рада статические отношения, которые существуют между ними. На диаграммах классов отображаются также свойства классов, операции классов и ограничения, которые накладываются на связи между объектами.

Рассмотрим изначальную версию диаграммы классов на рисунке 3.4.

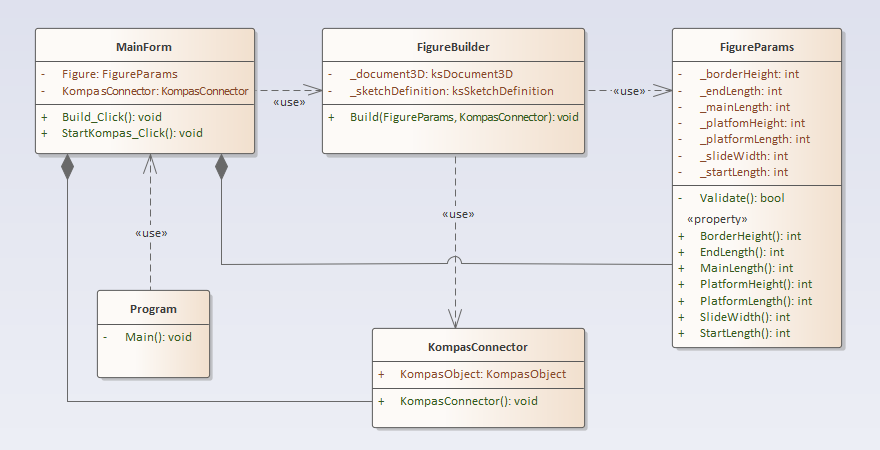


Рисунок 3.4 – Изначальная диаграмма классов плагина

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.

В результате изменений, внесенных заказчиком, диаграмма классов подверглась изменениям.

Была изменена связь между классами FigureBuilder и FigureParams c использования на композицию так-как необходимо хранить экземпляр класса FigureParams в классе FigureBuilder.

Была добавлена новое свойство PlatformThicknessT в классе FigureParams для изменения толщины платформы.

Рассмотрим измененную диаграмму классов плагина на рисунке 3.3

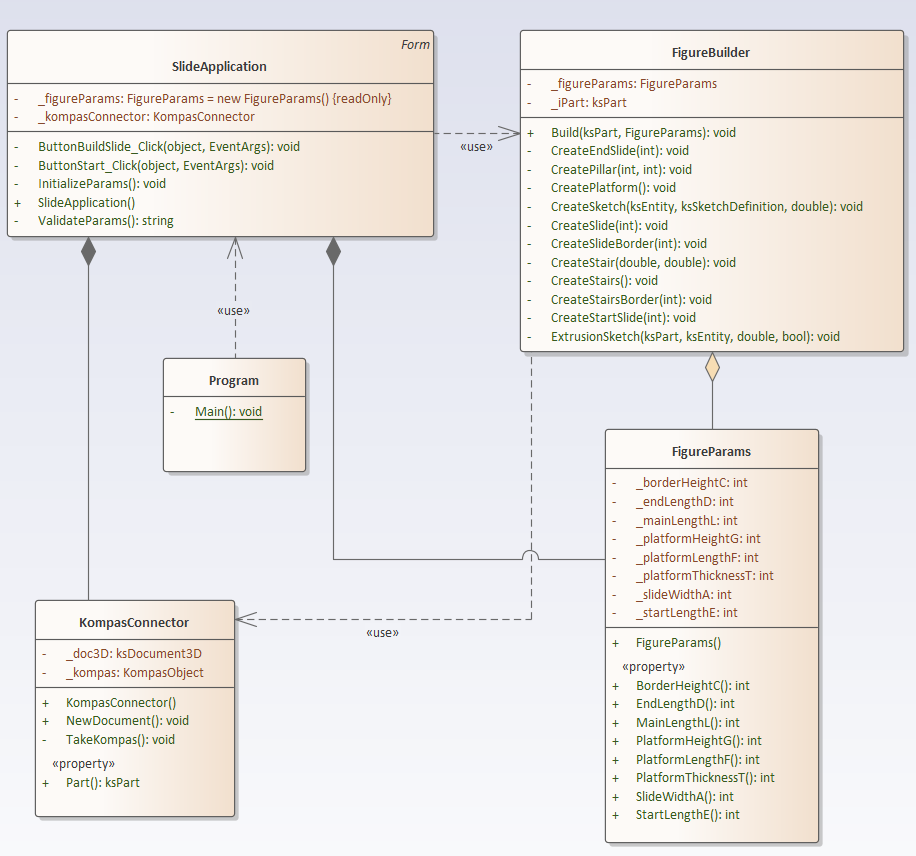


Рисунок 3.3– Измененная диаграмма классов плагина

**3.3 Диаграмма пакетов**

Пакет (package) [8] – это инструмент группирования, который позволяет взять любую конструкцию UML объединить ее элементы в единицы высокого уровня. В основном пакеты служат для объединения классов в группы, но могут применяться для любой другой конструкции языка UML.

Рассмотрим диаграмму пакетов плагина на рисунке 3.5.

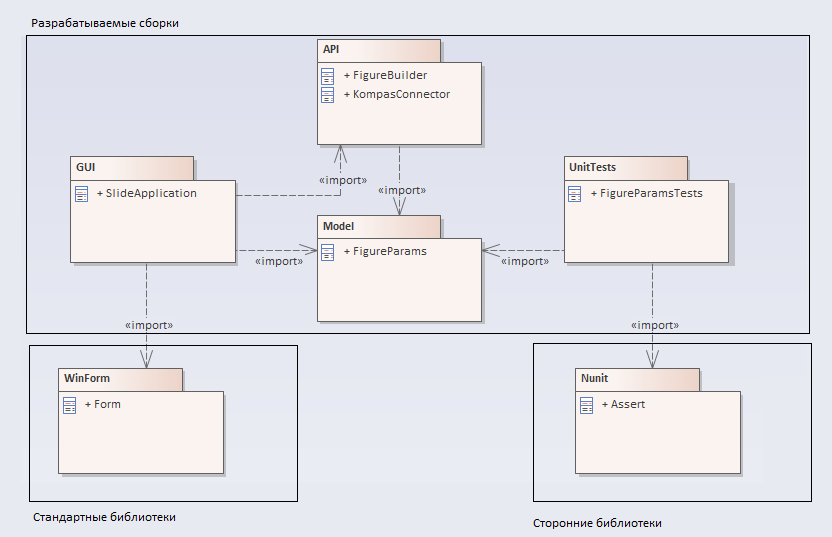


Рисунок 3.5– Диаграмма пакетов плагина

Пакет API хранит классы, отвечающие за работу с API Кompas3D.

Пользовательский интерфейс содержится в пакете GUI.

Пакет Model хранит класс с параметрами горки.

Пакет UnitTests хранит класс с юнит-тестами приложения.

**4 Описание программы для пользователя**

После запуска приложения перед пользователем появляется главное окно рисунок 4.1

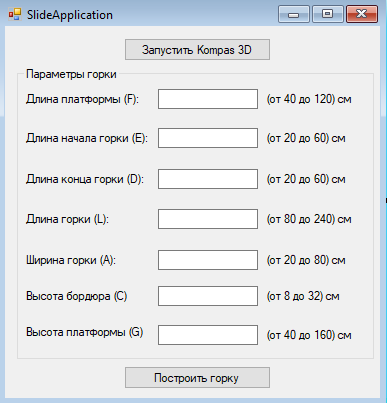


Рисунок 4.1 – Макет пользовательского интерфейса при запуске программы

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 4.2

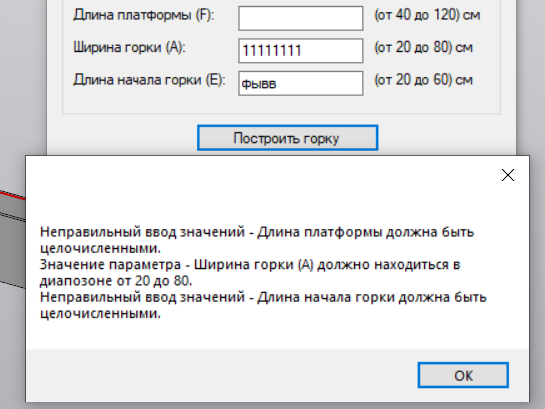


Рисунок 4.2 – Сообщение с ошибками

**5 Тестирование программы**

Тестирование позволяет выявлять ошибки в программе в процессе разработки и при выпуске промежуточных и финальных версий приложения.

**5.1 Функциональное тестирование**

При функциональном тестировании [9] проверялась корректность работы плагина “Детская горка”, а именно, соответствие полученного результата в виде трехмерной модели, с входными параметрами.

Проведем тестирование максимальных и минимальных параметров модели.

Модель с минимальными параметрами представлена на рисунке 5.1.

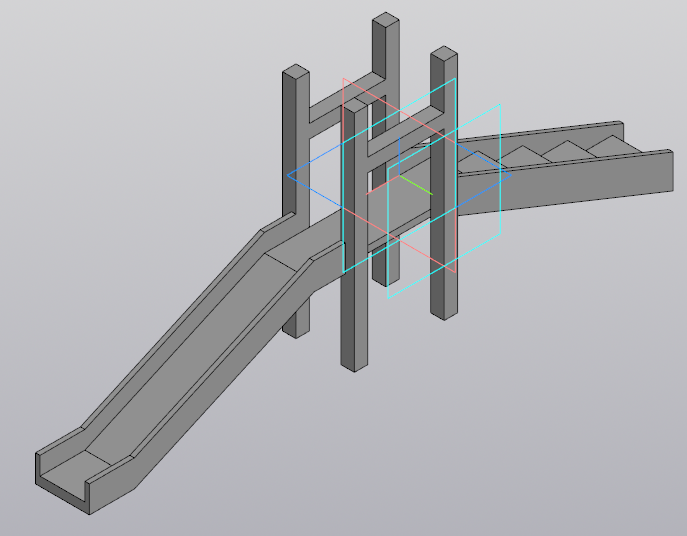


Рисунок 5.1 – Модель горки с минимальными возможными параметрами

Модель с максимальными параметрами представлена на рисунке 5.2.

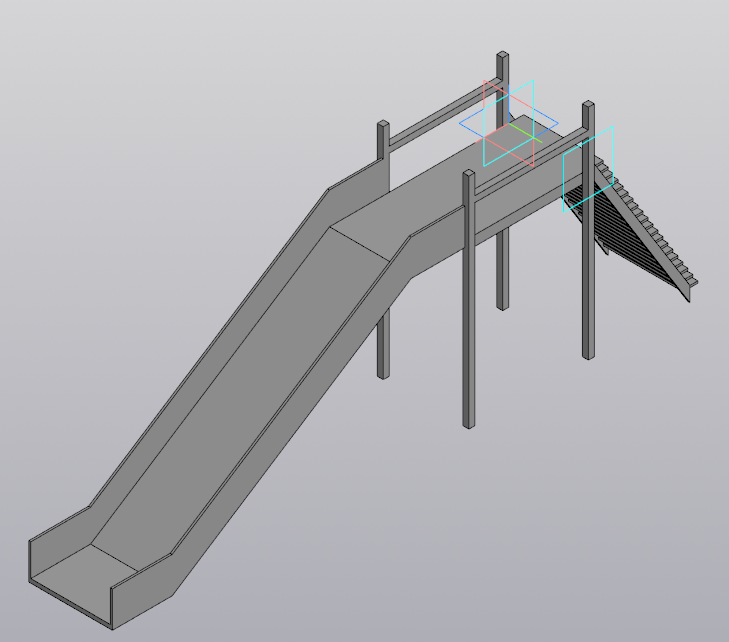


Рисунок 5.2 – Модель горки с максимальными возможными параметрами

**5.2 Модульное тестирование**

В целях проверки корректности работы методов и свойств классов при помощи обозревателя тестов Visual Studio было проведено модульное тестирование [10], проверялись открытые поля и методы, для этого были созданы тестовые классы:

* FigureParamsTests – класс тестирующий свойства и методы класса Model. Описание класса FigureParamsTests в Приложении А (Таблица А.1);

Диалоговое окно состояний запущенных тестов для класса FigureParamsTests изображены на рисунке 5.3

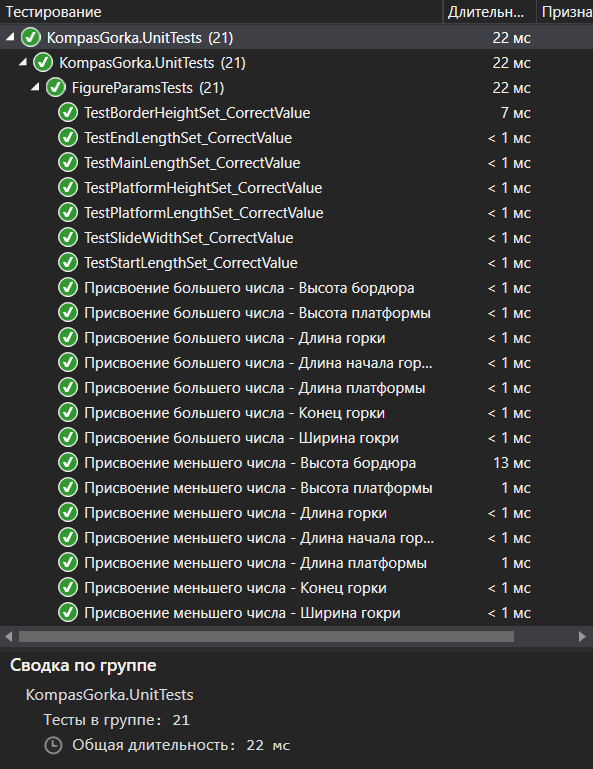


Рисунок 5.3 - Диалоговое окно состояний запущенных тестов для класса FigureParamsTests.

**5.3 Нагрузочное тестирование.**

Нагрузочное тестирование – это тестирование 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) [11].

Нагрузочное тестирование проводилось со следующими параметрами кровати:

Длина платформы - 40 см;

Длинна начала горки - 60 см;

Длина конца горки - 40 см;

Расстояние длины горки - 60 см;

Ширина горки - 40 см;

Высота бордюра - 40 см;

Высота платформы - 40 см;

Толщина платформы - 3 см.

Тестирование производилось на ПК со следующей конфигурацией:

Процессор: Intel Core i7-4710HQ

Графическая система: Intel HD Graphics 4600

ОЗУ: 2 GB DDR 1333 MHz

После построения 104 детали программа завершилась со сбоем из-за нехватки памяти.

На рисунке 5.6 представлен график потребляемой оперативной памяти относительно количества построенных деталей.

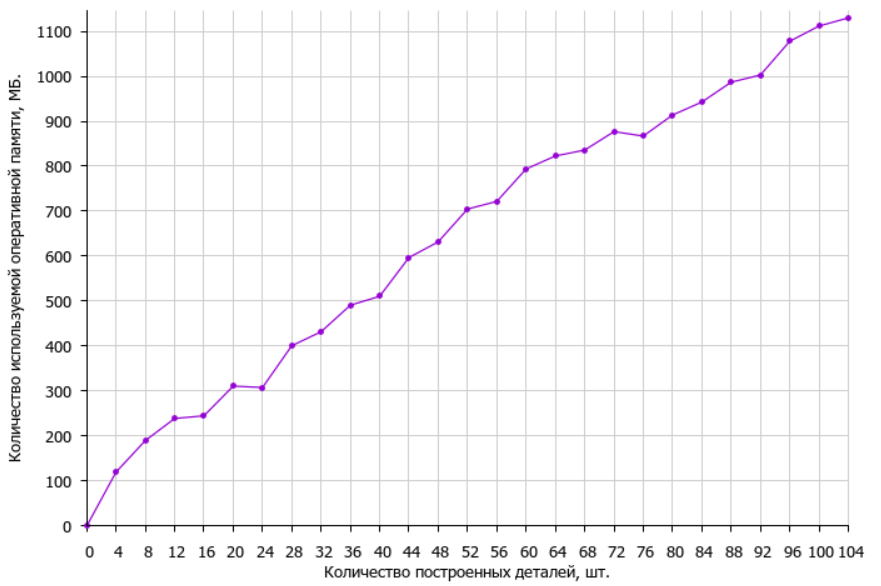


Рисунок 5.6 – График потребляемой оперативной памяти относительно числа созданных деталей.

График зависимости загрузки программой центрального процессора от количества построенных деталей изображен на рисунке 5.7.

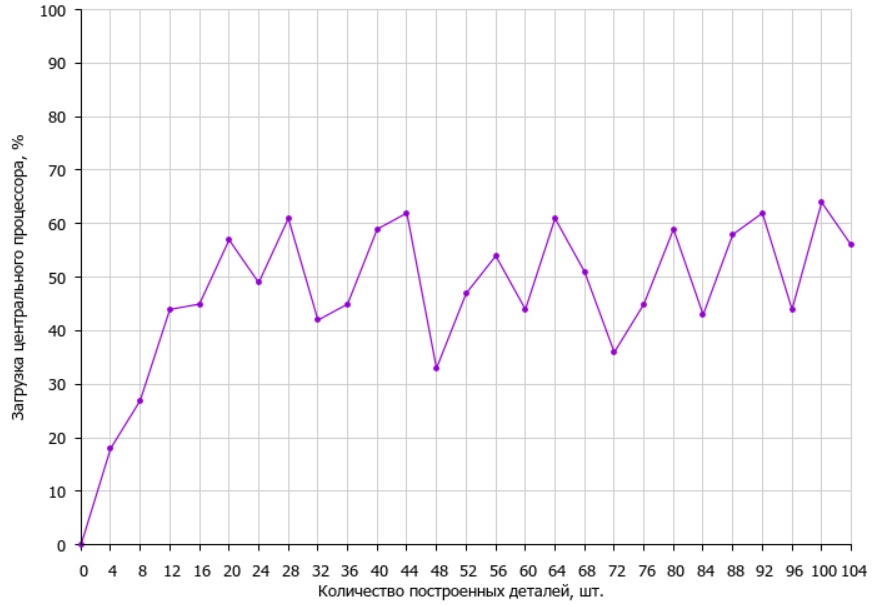


Рисунок 5.7 – График зависимости загрузки программой центрального процессора от количества построенных деталей.

При израсходовании свободных ресурсов нагрузка на процессор остается постоянной, а затраты оперативной памяти линейно увеличиваются.

**Заключение**

В ходе выполнения лабораторных работ были изучены предметная область проектирования, предмет проектирования, аналоги предмета проектирования, API и на основании полученных данных были спроектированы архитектура и макет системы, создан плагин «Детская горка», проведены модульные, функциональные и нагрузочные тесты.

**Список использованных источников**

1. КОМПАС – 3D [Электронный ресурс]. – URL: <https://kompas.ru/> (дата обращения: 13.04.2020);

# 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)

1. Microsoft Visual Studio 2019 [Электронный ресурс]. – URL: <https://visualstudio.microsoft.com/ru/vs/> (дата обращения: 13.04.2020);

# Базовые интерфейсы API системы КОМПАС [Электронный ресурс]. – URL: <https://it.wikireading.ru/23741> (дата обращения: 13.04.2020);

1. Приложение “Оборудование: Металлоконструкции” [Электронный ресурс]. – URL: <https://kompas.ru/kompas-3D/application/machinery/steel-constructions-3d/> (дата обращения: 13.04.2020);
2. Приложение “Оборудование: Кабели и жгуты” [Электронный ресурс]. – URL: <https://kompas.ru/kompas-3D/application/instrumentation/equipment-cables-and-harness/> (дата обращения: 13.04.2020);
3. Плагин Anthena [Электронный ресурс]. — Режим доступа: <https://cad-plan.com/> (дата обращения 01.03.2020)
4. Мартин Фаулер UML Основы. Краткое руководство по стандартному языку объектного моделирования [Электронный ресурс]. – URL: litportal.ru›trial/pdf/24500318.pdf (дата обращения: 13.04.2020);
5. Функциональное тестирование [Электронный ресурс]. – URL: <https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/> (дата обращения: 13.04.2020);
6. Модульное тестирование [Электронный ресурс]. – URL: <http://espressocode.top/unit-testing-software-testing/> (дата обращения: 13.04.2020);
7. Нагрузочное тестирование [Электронный ресурс]. – URL: <https://daglab.ru/nagruzochnoe-testirovanie-programmnogo-obespechenija/> (дата обращения: 13.04.2020);

**Приложение А**

(Справочное)

В таблицах приложения для обозначения модификаторов доступа полей приняты следующие условные знаки:

* “ – ” обозначение private (закрытого) поля;
* “ + ” обозначение public (открытого) поля;

Описание полей и методов используемых для проверки тестовых случаев класса FigureParamsTests представлено в таблице А.1

Таблица А.1 – Класс FigureParamsTests

|  |  |
| --- | --- |
| Наименование | Описание |
| − \_figureParams: FigureParams | Поле хранит параметры фигуры |
| − InitContact() | Метод для инициализации начальных параметров фигуры. |
| + TestBorderHeightSet\_CorrectValue() | Метод для проверки корректных данных высоты бордюра. |
| +TestEndLengthSet\_CorrectValue() | Метод для проверки корректных данных длины конца горки. |
| + TestMainLengthSet\_CorrectValue() | Метод для проверки корректных данных длины горки. |
| + TestPlatformHeightSet\_CorrectValue() | Метод для проверки корректных данных высоты платформы. |
| + TestPlatformLengthSet\_CorrectValue() | Метод для проверки корректных данных длины платформы. |
| +TestSlideWidthSet\_CorrectValue() | Метод для проверки корректных данных ширины горки. |
| +TestStartLengthSet\_CorrectValue() | Метод для проверки корректных данных длина начала горки. |

Продолжение таблицы А.1.

|  |  |
| --- | --- |
| TestBorderHeightSet\_ArgumentException() | Метод для проверки не корректных данных высоты бордюра. |
| + TestEndLengthSet\_ArgumentException() | Метод для проверки не корректных данных длины конца горки. |
| + TestMainLengthSet\_ArgumentException() | Метод для проверки не корректных данных длины горки. |
| + TestPlatformHeightSet\_ArgumentException() | Метод для проверки не корректных данных высоты платформы. |
| +TestPlatformLengthSet\_ArgumentException() | Метод для проверки не корректных данных длины платформы. |
| +TestStatLengthSet\_ArgumentException() | Метод для проверки не корректных данных длина начала горки. |
| TestSlideWidthSet\_ArgumentException | Метод для проверки не корректных данных ширины горки. |