**Содержание**

[ВВЕДЕНИЕ 4](#_Toc59107397)

[1 РАЗРАБОТКА ПРИЛОЖЕНИЯ 5](#_Toc59107398)

[1.1 Назначение системы автоматизации моделирования выдавливания бобышки/ отверстия по траектории 5](#_Toc59107399)

[1.2 Структура системы автоматизации моделирования 5](#_Toc59107400)

[1.3 Описание приложения 6](#_Toc59107401)

[1.3.1 Таблица для ввода координат кривой 6](#_Toc59107402)

[1.3.2 Построение параллелепипеда 6](#_Toc59107403)

[1.3.3 Построение кривой заданным точкам и плоскости 7](#_Toc59107404)

[1.3.4 Создание основания по траектории, проходящей через кривую 8](#_Toc59107405)

[1.3.5 Создание выреза по траектории, проходящей через кривую 9](#_Toc59107406)

[1.3.6 Размеры 9](#_Toc59107407)

[1.4 Руководство пользователю 11](#_Toc59107408)

[1.4.1 Область применения 11](#_Toc59107409)

[1.4.2 Краткое описание возможностей 11](#_Toc59107410)

[1.4.3 Уровень подготовки пользователя 11](#_Toc59107411)

[1.4.4 Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю 11](#_Toc59107412)

[1.4.5 Минимальные системные требования 11](#_Toc59107413)

[1.4.6 Порядок проверки работоспособности 11](#_Toc59107414)

[2 ЗАКЛЮЧЕНИЕ 12](#_Toc59107415)

[3 ЛИТЕРАТУРА 13](#_Toc59107416)

[4 ПРИЛОЖЕНИЕ А 14](#_Toc59107417)

ВВЕДЕНИЕ

В современном мире информационных технологий задачи построения трёхмерных объектов требуют применения автоматизированного программного обеспечения для расчётов и построения моделей для дальнейшего использования в сложных конструкциях.

Преимуществом SOLIDWORKS является простой доступ к инструментам api[1], которые можно использовать для автоматизирования задач построения различных деталей, начиная по примитивных заканчивая комплексными анимированными сборками.

Особенностью автоматизации построения является возможность точного конструирования элементов, для дальнейшего использования в сложных деталях или для добавления в конструкцию.

Приложение, которое было сделано при выполнении данной курсовой работы, позволяет осуществить построение примитивной детали по траектории, которая задаётся вручную пользователем.

Данное приложение может быть использовано при построении деталей кузова, каркасов, хорошо подходит для конструирования литых изделий[2], а также может быть использовано для демонстрации возможности конструирования при решении учебных задач[3].

РАЗРАБОТКА ПРИЛОЖЕНИЯ

Назначение системы автоматизации моделирования выдавливания бобышки/ отверстия по траектории

Система автоматизации моделирования выдавливания бобышки/отверстия по траектории на основе API Solid Works, предназначена для построения кривых по координатам, введённых пользователем в окно координат и создание вырезов либо основания вдоль данной кривой. Для корректной работы программы в процессе создания выреза либо основания будет создана вспомогательная плоскость и эскиз, на основе которого будет проводиться построение.

Данное приложение представляет из себя приложение для платформы Windows, позволяющее создавать объекты в файле трёхмерной модели в Solidworks. Файл модели должен быть открыт. Тип файла, который должен быть открыт в среде Solidworks – модель.

Объекты, которые можно создать, используя данное приложение являются:

1. Трёхмерный параллелепипед;
2. Кривая;
3. Плоскость;
4. Вырез по кривой;
5. Создание основания по траектории.

Приложение написано на платформе .NET Core с использованием компонентов WindowsForms.

Структура системы автоматизации моделирования

Приложение представляет из себя окно с таблицей для ввода значений, по которым будет происходить построение кривых и кнопки, которые отвечают за построение трёхмерных объектов (рисунок 1).

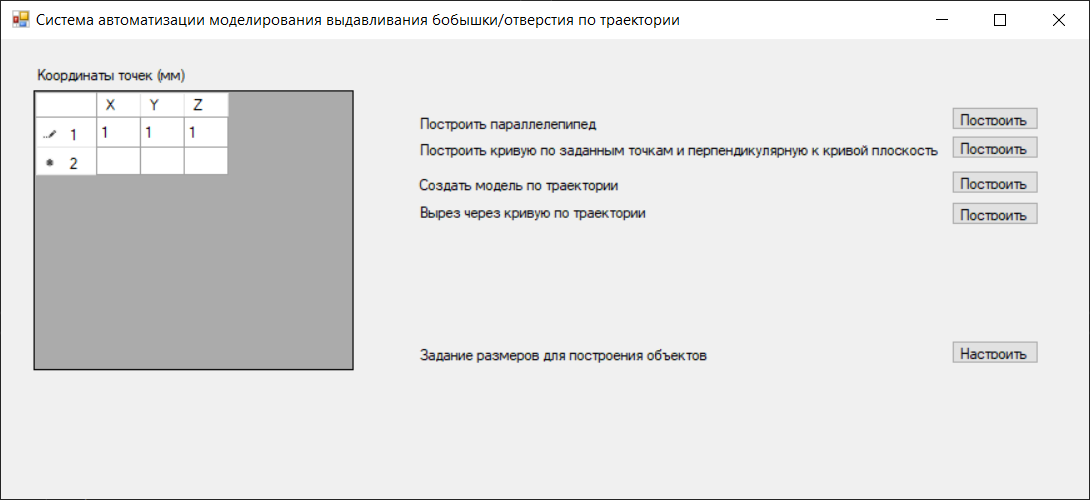


Рисунок 1. Внешний вид приложения

Приложение состоит из следующих модулей:

* 1. Таблицы для ввода координат кривой;
  2. Построение параллелепипеда;
  3. Построение кривой по заданным точкам и плоскости;
  4. Создание основания по траектории, проходящей через кривую;
  5. Создание выреза по траектории, проходящей через кривую;
  6. Задание размеров для построения объектов.

Ввод данных производится пользователем с помощью клавиатуры и мыши, вывод происходит в открытый файл в среде SolidWorks.

Описание приложения

Таблица для ввода координат кривой

Для построения кривой необходимо получить координаты точек в трёхмерном пространстве холста, через которые будет проходить кривая. Для построения кривой необходимо наличие хотя бы двух точек с ненулевым расстоянием между друг другом. В таблице присутствует возможность удалять и добавлять новые строки, которые являются точками, координатам которых соответствуют столбцы таблицы (рисунок 2). Таблицу необходимо заполнять числовыми значениями любой величины, пустые ячейки будут восприняты как число ноль. Абсолютно пустые значения строки не будут учитываться как точки.

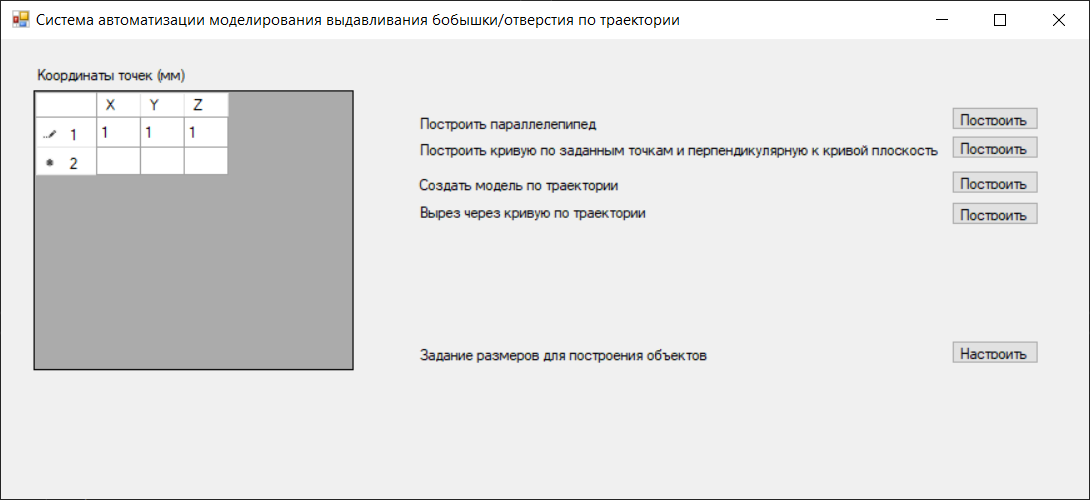


Рисунок 2. Заполненная таблица координат в программе

Построение параллелепипеда

Для демонстрации работы выреза через кривую по траектории, необходимо наличие объекта на открытом холсте в среде SolidWorks. При нажатии на кнопку, соответствующую надписи “Построить параллелепипед”, происходит построение параллелепипеда с размерами, которые были введены в соответствующие поля в окне “ Задание размеров для построения объектов” в графах параллелепипед: высота, ширина, длина, в плоскости, которая была построена после срабатывания модуля “построение кривой по заданной траектории” и иметь открытым документ типа “Деталь” в среде Solidworks, результат работы модуля приведен на рисунке 3.

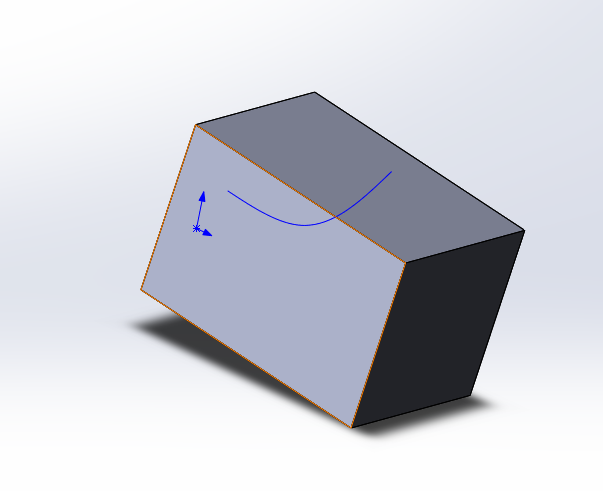


Рисунок 3. Результат работы модуля “Построение параллелепипеда”

При попытке построить параллелепипед через программное средство без предварительного построения плоскости, пользователь получит сообщение об ошибке (рисунок 4).

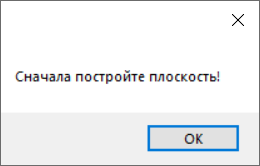


Рисунок 4. Сообщение об отсутствия плоскости для построения параллелепипеда

Построение кривой заданным точкам и плоскости

Для построения выреза и основания необходимо наличие двух компонентов в открытом документе: кривой, через которую будет проходить построение либо вырез и плоскости, от которой будет произведено построение. Также плоскость необходима для построения параллелепипеда.

Компоненты кривой и плоскости строятся через данный модуль. Для работы модуля необходима заполненная таблица координат с хотя бы двумя точками в ней. Плоскость будет создана в центре первой точки и из нее же будет проводиться кривая, при этом плоскость будет расположена перпендикулярно кривой (рисунок 5). Кривая будет создана через заданные точки, первая строка координат в таблице будет являться начальной точкой для кривой, а последняя строка, содержащая числовые данные – будет являться конечной точкой кривой. Для построения необходимо нажать на кнопку, соответствующую надписи “Построить кривую по заданным точкам и перпендикулярную к кривой плоскость” и иметь открытым документ типа “Деталь” в среде Solidworks.

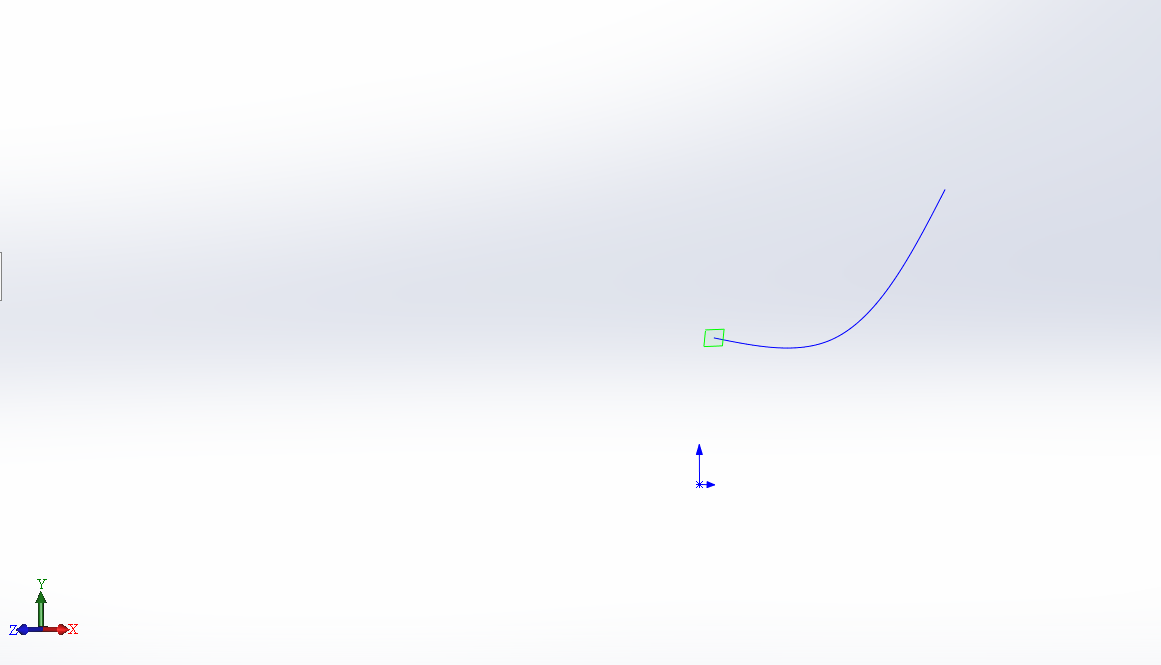


Рисунок 5. Построение кривой и плоскости, перпендикулярной к ней

Создание основания по траектории, проходящей через кривую

Данный модуль позволяет пользователю создать основание от плоскости, вдоль кривой, которые были построены в модуле “Построение кривой заданным точкам и плоскости”. Основание будет иметь радиус, установленный пользователем в окне “ Задание размеров для построения объектов”, и иметь начальную точку, совпадающую с первой точкой в таблице значений координат. Для совершения построения необходимо нажать на кнопку, соответствующую надписи “Создать модель по траектории” и иметь открытым документ типа “Деталь” в среде Solidworks (рисунок 6). Радиус основания зависит от выбранных параметров в окне “ Задание размеров для построения объектов” – в графе: основание, радиус.

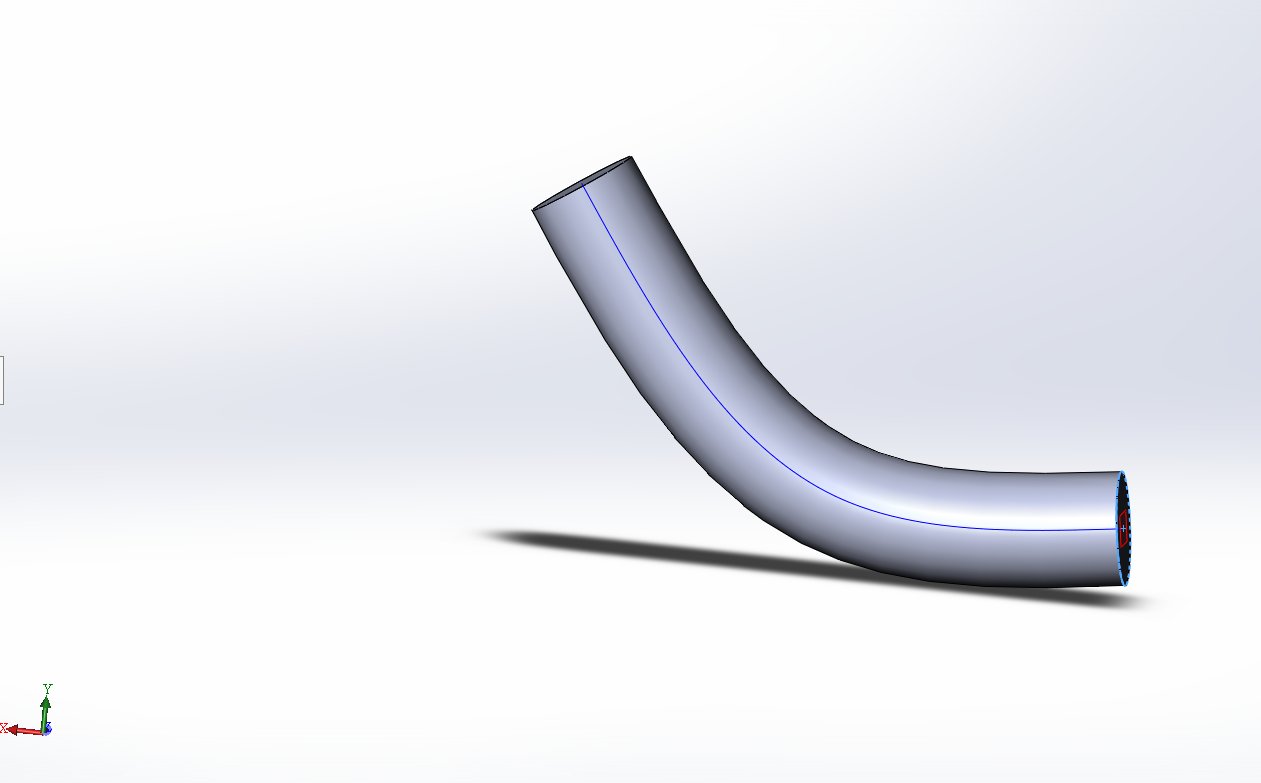


Рисунок 6. Создание основание в плоскости по траектории заданной кривой

Создание выреза по траектории, проходящей через кривую

Данный модуль позволяет пользователю создать вырез от плоскости, вдоль кривой, которые были построены в модуле “Построение кривой заданным точкам и плоскости”. Вырез будет иметь радиус, установленный пользователем в окне “Настройки”, и иметь начальную точку, совпадающую с первой точкой в таблице значений координат. Для совершения построения необходимо нажать на кнопку, соответствующую надписи “Вырез через кривую по траектории” и иметь открытым документ типа “Деталь” в среде Solidworks (рисунок 7). Радиус основания зависит от выбранных параметров в окне “ Задание размеров для построения объектов” – в графе: вырез, радиус.



Рисунок 7. Создание выреза в объекте через траекторию, заданную кривой

Размеры

Данный модуль представляет собой окно, которое появляется при нажатии на кнопку, соответствующую надписи “ Задание размеров для построения объектов”, в данном окне Пользователь имеет возможность изменить следующие размеры: высота, ширина, длина для параллелепипеда, радиус для основания, радиус для выреза, для этого необходимо изменять значения на нужные в соответствующих полях (рисунок 8).

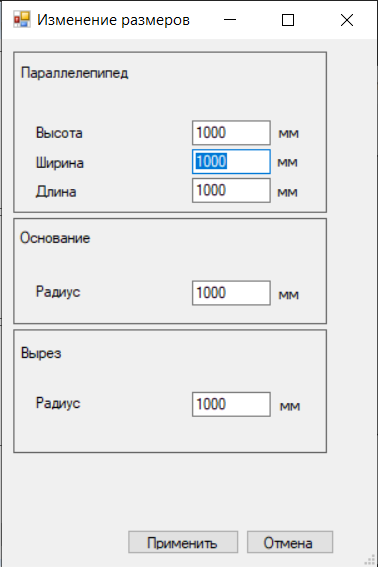


Рисунок 8. Изменение размеров для элементов

В случае, если Пользователь ошибся с вводом данных, появится окно, которое сообщит в каком месте Пользователь ошибся с вводом данных (рисунок 9).

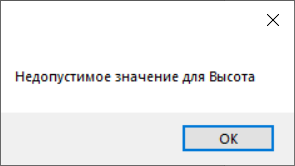


Рисунок 9. Сообщение, появляющееся при ошибке ввода

При нажатии на кнопку “Применить”, текущие размеры сохранятся и будут использованы при построении в дальнейшем (рисунок 10) и окно будет закрыто. При нажатии на кнопку “Отмена” текущие настройки будут сброшены и окно будет закрыто.



Рисунок 9. Кнопки “Применить” и “Отмена”

Руководство пользователю

Область применения

Требования применяются при:

* опытной эксплуатации;

Краткое описание возможностей

Приложение «» предназначен для создания, плоскостей, кривых, оснований и вырезов по координатам, введенным Пользователем.

Уровень подготовки пользователя

Умение пользоваться клавиатурой и мышкой;

Других навыков не требуется.

Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю

Отсутствует.

Минимальные системные требования

ОС: Windows10.

Дисковое пространство: 100 мб;

ОЗУ: 16 гб;

Процессор: Intel I7;

Видеопамять: 1 гб;

Порядок проверки работоспособности

Для проверки приложения с рабочего места пользователя необходимо выполнить следующие действия:

1. Открыть среду SolidWorks;
2. Создать либо открыть документ типа “Деталь”;
3. Запустить приложение.

ЗАКЛЮЧЕНИЕ

Приложение Система автоматизации моделирования выдавливания бобышки/отверстия по траектории на основе API Solid Works, которое было сделано при выполнении данной курсовой работы, позволяет осуществить построение примитивной детали по траектории, которая задаётся вручную пользователем.

Данное приложение может быть использовано при построении деталей кузова, каркасов, хорошо подходит для конструирования литых изделий, а также может быть использовано для демонстрации возможности конструирования при решении учебных задач.

ЛИТЕРАТУРА

1. SOLIDWORKS API Help[Электронный ресурс]: – Режим доступа: <http://help.solidworks.com/2020/english/api/sldworksapiprogguide/Welcome.htm>, свободный.
2. A review of the design methods of complex topology structures for 3D printing[Электронный ресурс] : – Режим доступа: <https://vciba.springeropen.com/articles/10.1186/s42492-018-0004-3>
3. Strategic Development of Lightweight Platforms Made of Steel - [Электронный ресурс]: – Режим доступа: https://link.springer.com/article/10.1007/s41777-018-0039-3

# ПРИЛОЖЕНИЕ А

Листинг программы

using SolidWorks.Interop.sldworks;

using SolidWorks.Interop.swconst;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace bntu\_10702117\_Course\_project\_Zayicev

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

static int plane = 0;

static int curve = 0;

static int stBox = 0;

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

List<GeometryPoint> geometryPoints = new List<GeometryPoint>();

int Count { get; set; }

private void Form1\_Load(object sender, EventArgs e)

{

Settings = new Settings() { CubeX = 1.0f, CubeY = 1.0f, CubeZ = 1.0f, BassR = 1.0f, CutR = 1.0f };

//Заполнение формы

dataGridView1.ColumnCount = 3;

dataGridView1.RowCount = 2;

dataGridView1.RowHeadersWidth = 50;

for (int i = 0; i < dataGridView1.RowCount; i++)

{

dataGridView1.Rows[i].HeaderCell.Value = $"{i + 1}";

}

//dataGridView1.Rows.Add($"X",$"Y",$"Z");

}

private void button1\_Click(object sender, EventArgs e)

{

SldWorks SwApp;

try

{

SwApp = (SldWorks)Marshal.GetActiveObject("SldWorks.Application"); //проверка на то, включен ли солид

}

catch

{

MessageBox.Show("Пожалуйста, включите SolidWorks."); //если не влкючен

return;

}

FillPoints();

IModelDoc2 swModel = SwApp.IActiveDoc2; //стандартное присвоение

swModel.SketchManager.InsertSketch(true); //создать скетч

//Создать 3 точки

for (int i = 0; i < geometryPoints.Count; i++)

{

/\* int x = Convert.ToInt32(geometryPoints.ToArray()[i].X);

int y = Convert.ToInt32(geometryPoints.ToArray()[i].Y);

int z = Convert.ToInt32(geometryPoints.ToArray()[i].Z);

\*/

swModel.CreatePoint(Convert.ToInt32(geometryPoints.ToArray()[i].X), Convert.ToInt32(geometryPoints.ToArray()[i].Y), Convert.ToInt32(geometryPoints.ToArray()[i].Z));

}

swModel.ClearSelection2(true);

swModel.SketchManager.InsertSketch(true);

}

private bool ClearPoints()

{

geometryPoints.Clear();

return true;

}

private bool FillPoints()

{

ClearPoints();

Count = dataGridView1.RowCount-1;

for (int i = 0; i < Count; i++)

{

double[] cellArray = new double[3];

for (int j = 0; j < 3; j++)

{

try

{

cellArray[j] = Convert.ToDouble(dataGridView1.Rows[i].Cells[j].Value);

}

catch (Exception)

{

MessageBox.Show($"Проверьте ячейки в {i} строке, они должы содержать числовые данные");

throw;

}

}

geometryPoints.Add(new GeometryPoint(cellArray));

}

return true;

}

private void BuidPlane()

{

SldWorks SwApp;

try

{

SwApp = (SldWorks)Marshal.GetActiveObject("SldWorks.Application"); //проверка на то, включен ли солид

}

catch

{

MessageBox.Show("Пожалуйста, включите SolidWorks."); //если не влкючен

return;

}

FillPoints();

IModelDoc2 swModel = SwApp.IActiveDoc2; //стандартное присвоение

swModel.SketchManager.InsertSketch(true); //создать скетч

//Создать 3 точки

for (int i = 0; i < 3; i++)

{

/\* int x = Convert.ToInt32(geometryPoints.ToArray()[i].X);

int y = Convert.ToInt32(geometryPoints.ToArray()[i].Y);

int z = Convert.ToInt32(geometryPoints.ToArray()[i].Z);

\*/

swModel.CreatePoint(Convert.ToInt32(geometryPoints.ToArray()[i].X), Convert.ToInt32(geometryPoints.ToArray()[i].Y), Convert.ToInt32(geometryPoints.ToArray()[i].Z));

}

swModel.ClearSelection2(true);

swModel.SketchManager.InsertSketch(true);

}

//построить кривую

private void button2\_Click(object sender, EventArgs e)

{

SldWorks SwApp;

try

{

SwApp = (SldWorks)Marshal.GetActiveObject("SldWorks.Application"); //проверка на то, включен ли солид

}

catch

{

MessageBox.Show("Пожалуйста, включите SolidWorks."); //если не влкючен

return;

}

FillPoints();

IModelDoc2 swModel = SwApp.IActiveDoc2;

FeatureManager swFeatureManager = default(FeatureManager);

swFeatureManager = (FeatureManager)swModel.FeatureManager;

var swModelDocExt = (ModelDocExtension)swModel.Extension;

curve++;

swModel.InsertCurveFileBegin();// начало проведения ломаной

//var status = swModel.InsertCurveFilePoint(0, 0, 0);

for (int i = 0; i < geometryPoints.Count; i++)

{

swModel.InsertCurveFilePoint(Convert.ToInt32(geometryPoints.ToArray()[i].X), Convert.ToInt32(geometryPoints.ToArray()[i].Y), Convert.ToInt32(geometryPoints.ToArray()[i].Z));

}

swModel.InsertCurveFileEnd();// конец проведения ломаной

var sManager = swModel.SelectionManager;

var selObj = sManager.GetSelectedObject5(1);

selObj.name = $"Curve991" + curve.ToString();//test

var boolStatus = swModel.Extension.SelectByID2("", "POINTREF", Convert.ToInt32(geometryPoints.ToArray()[0].X), Convert.ToInt32(geometryPoints.ToArray()[0].Y), Convert.ToInt32(geometryPoints.ToArray()[0].Z), true, 1, null, 0);

var swRefPlane = (RefPlane)swFeatureManager.InsertRefPlane(2, 0, 4, 0, 0, 0);

//var bResult = swModel.Extension.SelectByID2("", "PLANE", geometryPoints.ToArray()[0].X, geometryPoints.ToArray()[0].Y, geometryPoints.ToArray()[0].Z, false, 0, null, 0);

// status = swModelDocExt.SelectByID2("Curve1", "REFERENCECURVES", 0, 0,0, false, 0, null, 0);

swModel.ClearSelection2(true);

}

/\*

\*

\*

\* ?/

private void button3\_Click(object sender, EventArgs e)

{

SldWorks SwApp;

try

{

SwApp = (SldWorks)Marshal.GetActiveObject("SldWorks.Application"); //проверка на то, включен ли солид

}

catch

{

MessageBox.Show("Пожалуйста, включите SolidWorks."); //если не влкючен

return;

}

IModelDoc2 swModel = SwApp.IActiveDoc2; //стандартное присвоение

FeatureManager swFeatureManager = default(FeatureManager);

swFeatureManager = (FeatureManager)swModel.FeatureManager;

var swModelDocExt = (ModelDocExtension)swModel.Extension;

//var boolstatus = swModelDocExt.SelectByID2("", "SKETCHPOINT", 0.028424218552, 0.07057725774359, 0, true, 0, null, 0);

swModel.ClearSelection2(true);

swModel.SketchManager.InsertSketch(true);

var bools = new List<bool>();

foreach (var item in geometryPoints)

{

var boolstatus = swModelDocExt.SelectByID2("", "REFERENCECURVES", Convert.ToInt32(item.X), Convert.ToInt32(item.Y), Convert.ToInt32(item.Z), false, 0, null, 0);

bools.Add(boolstatus);

}

var boolStatus2 = swModel.Extension.SelectByID2("Point1", "SKETCHPOINT", Convert.ToInt32(geometryPoints.ToArray()[0].X), Convert.ToInt32(geometryPoints.ToArray()[0].Y), Convert.ToInt32(geometryPoints.ToArray()[0].Z), false, 0, null, 0);

var swRefPlane = (RefPlane)swFeatureManager.InsertRefPlane(4, 0, 2, 0, 0, 0);

}

\*/

private void button4\_Click(object sender, EventArgs e)

{

var SwApp = (SldWorks)Marshal.GetActiveObject("SldWorks.Application"); //проверка на то, включен ли солид

IModelDoc2 swModel = SwApp.IActiveDoc2;

swModel.InsertCurveFileBegin();

var status = swModel.InsertCurveFilePoint(0, 0, 0);

status = swModel.InsertCurveFilePoint(0, 0, 0.0127);

status = swModel.InsertCurveFilePoint(0, 0, 0.0254);

status = swModel.InsertCurveFilePoint(0, 0, 0.0381);

status = swModel.InsertCurveFilePoint(0, 0.0254, 0.0381);

status = swModel.InsertCurveFilePoint(0, 0.0381, 0.0381);

status = swModel.InsertCurveFileEnd();

ModelDocExtension swModelDocExt = default(ModelDocExtension);

SelectionMgr swSelectionMgr = default(SelectionMgr);

//Get free point curve feature

swModelDocExt = (ModelDocExtension)swModel.Extension;

swSelectionMgr = (SelectionMgr)swModel.SelectionManager;

swModel.ClearSelection2(true);

status = swModelDocExt.SelectByID2("Curve1", "REFERENCECURVES", 0, 0, 0, false, 0, null, 0);

// swModel.ClearSelection2(true);

}

// создание модели по траектории

private void button5\_Click(object sender, EventArgs e)

{

if (geometryPoints.Count == 0)

{

MessageBox.Show("Сначала постройте плоскость!");

return;

}

SldWorks SwApp;

try

{

SwApp = (SldWorks)Marshal.GetActiveObject("SldWorks.Application"); //проверка на то, включен ли солид

}

catch

{

MessageBox.Show("Пожалуйста, включите SolidWorks."); //если не влкючен

return;

}

IModelDoc2 swModel = SwApp.IActiveDoc2; //стандартное присвоение

var bResult = true;

bResult = swModel.Extension.SelectByID2("", "PLANE", geometryPoints.ToArray()[0].X, geometryPoints.ToArray()[0].Y, geometryPoints.ToArray()[0].Z, false, 0, null, 0);

swModel.SketchManager.InsertSketch(true);

swModel.ClearSelection2(true);

var circle = swModel.SketchManager.CreateCircleByRadius(0, 0, 0, Settings.BassR);

var activeSketch = swModel.GetActiveSketch2();

activeSketch.Name = $"Sketch991" + plane.ToString();

swModel.SketchManager.InsertSketch(true);

swModel.ClearSelection2(true);

bResult = swModel.Extension.SelectByID2($"Sketch991" + plane.ToString(), "SKETCH", 0, 0, 0, false, 1, null, 0);

//test//bResult = swModel.Extension.SelectByID2("", "REFERENCECURVES", geometryPoints.ToArray()[0].X, geometryPoints.ToArray()[0].Y, geometryPoints.ToArray()[0].Z,true,2,null,0);

bResult = swModel.Extension.SelectByID2($"Curve991" + curve.ToString(), "REFERENCECURVES", 0, 0, 0, true, 2, null, 0);//test

var swFeature = (Feature)swModel.FeatureManager.InsertProtrusionSwept4(false, false, 0, false, false, 0, 0, false, 0, 0,

0, 0, true, true, true, 0, true, false, 0.1, 0);

plane++;

//var mf = swModel.FeatureManager.InsertProtrusionSwept4(false, false, 0, false, false, 0, 0, false, 0, 0, 0, 0, true, true, true, 0, true, false, 0.01,1);

//var swFeature = (Feature)swModel.FeatureManager.InsertProtrusionSwept4(false, false, (int)swTwistControlType\_e.swTwistControlFollowPath, false, false, (int)swTangencyType\_e.swTangencyNone, (int)swTangencyType\_e.swTangencyNone, false, 0, 0, (int)swThinWallType\_e.swThinWallOneDirection, 0, true, true, true, 0, true, false, 0.1, 0);

}

//вырез через кривую

private void button6\_Click(object sender, EventArgs e)

{

if (geometryPoints.Count == 0)

{

MessageBox.Show("Сначала постройте плоскость!");

return;

}

SldWorks SwApp;

try

{

SwApp = (SldWorks)Marshal.GetActiveObject("SldWorks.Application"); //проверка на то, включен ли солид

}

catch

{

MessageBox.Show("Пожалуйста, включите SolidWorks."); //если не влкючен

return;

}

IModelDoc2 swModel = SwApp.IActiveDoc2; //стандартное присвоение

var bResult = true;

bResult = swModel.Extension.SelectByID2("", "PLANE", geometryPoints.ToArray()[0].X, geometryPoints.ToArray()[0].Y, geometryPoints.ToArray()[0].Z, false, 0, null, 0);

swModel.SketchManager.InsertSketch(true);

swModel.ClearSelection2(true);

var circle = swModel.SketchManager.CreateCircleByRadius(0, 0, 0, Settings.CutR);

var activeSketch = swModel.GetActiveSketch2();

activeSketch.Name = $"Sketch991" + plane.ToString();

swModel.SketchManager.InsertSketch(true);

swModel.ClearSelection2(true);

bResult = swModel.Extension.SelectByID2($"Sketch991" + plane.ToString(), "SKETCH", 0, 0, 0, false, 1, null, 0);

//test

bResult = swModel.Extension.SelectByID2($"Curve991" + curve.ToString(), "REFERENCECURVES", 0, 0, 0, true, 4, null, 0);

//

//test //bResult = swModel.Extension.SelectByID2("", "REFERENCECURVES", geometryPoints.ToArray()[0].X, geometryPoints.ToArray()[0].Y, geometryPoints.ToArray()[0].Z, true, 4, null, 0);

/\* var swFeature = (Feature)swModel.FeatureManager.InsertProtrusionSwept4(false, false, 0, false, false, 0, 0, false, 0, 0,

0, 0, true, true, true, 0, true, false, 0.1, 0);

\*/

var swFeature2 = (Feature)swModel.FeatureManager.InsertCutSwept4(false, true, 0, false, false, 0, 0, false, 0, 0, 0, 0, true, true, 0, true, true, true, false);

//var swFeature = (Feature)swModel.FeatureManager.InsertCutSwept5(false, true, 0, false, false, 0, 0, false, 0, 0, 0, 0, true, true, 0, true, true, true, false, true, 0.1, 0);

plane++;

/\* var SwApp = (SldWorks)Marshal.GetActiveObject("SldWorks.Application"); //проверка на то, включен ли солид

IModelDoc2 swModel = SwApp.IActiveDoc2; //стандартное присвоение

var swFeature = (Feature)swModel.FeatureManager.InsertProtrusionSwept4(false, false, 0, false, false, 0, 0, false, 0, 0,

0, 0, true, true, true, 0, true, false, 0.1, 0);

\*/

}

//Построение параллелепипеда

private void button7\_Click(object sender, EventArgs e)

{

if (geometryPoints.Count == 0)

{

MessageBox.Show("Сначала постройте плоскость!");

return;

}

SldWorks SwApp;

try

{

SwApp = (SldWorks)Marshal.GetActiveObject("SldWorks.Application"); //проверка на то, включен ли солид

}

catch

{

MessageBox.Show("Пожалуйста, включите SolidWorks."); //если не влкючен

return;

}

IModelDoc2 swModel = SwApp.IActiveDoc2; //стандартное присвоение

var bResult = true;

bResult = swModel.Extension.SelectByID2("", "PLANE", geometryPoints.ToArray()[0].X, geometryPoints.ToArray()[0].Y, geometryPoints.ToArray()[0].Z, false, 0, null, 0);

swModel.SketchManager.InsertSketch(true);

swModel.ClearSelection2(true);

var box = swModel.SketchManager.CreateCenterRectangle(0, 0, 0, Settings.CubeY/2, Settings.CubeX/2, Settings.CubeZ);

swModel.FeatureManager.FeatureExtrusion2(true, false, false, 0, 0, Settings.CubeZ, Settings.CubeZ, false, false, false, false, 1.74532925199433E-02, 1.74532925199433E-02, false, false, false, false, true, true, true, 0, 0, false);

swModel.ClearSelection2(true);

//swModel.SelectionManager.EnableContourSelection =false;

//var activeSketch = swModel.GetActiveSketch2();

//activeSketch.Name = $"SketcBoxh991" + box.ToString();

stBox++;

}

private void Form1\_KeyPress(object sender, KeyPressEventArgs e)

{

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

}

private void dataGridView1\_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)

{

dataGridView1.Rows[e.RowIndex].HeaderCell.Value = $"{e.RowIndex+1}";

}

private void dataGridView1\_RowsRemoved(object sender, DataGridViewRowsRemovedEventArgs e)

{

dataGridView1.Rows[e.RowIndex].HeaderCell.Value = $"{e.RowIndex+1}";

}

SizeForm sizeformRef;

public Settings Settings;

private void button1\_Click\_1(object sender, EventArgs e)

{

sizeformRef = new SizeForm();

sizeformRef.settings = Settings;

sizeformRef.form1Ref = this;

sizeformRef.ShowDialog();

}

public void UpdateSettings(Settings newSettings)

{

Settings = newSettings;

}

}

}

/\* swModel.InsertProtrusionSwept4(false, false, (int)swTwistControlType\_e.swTwistControlFollowPathFirstGuideCurve, false, false, (int)swTangencyType\_e.swTangencyNone, (int)swTangencyType\_e.swTangencyNone, true, 0, 0, (int)swThinWallType\_e.swThinWallMidPlane);

//var mf2 = (Feature)swModel.InsertProtrusionSwept4(false, false, (int)swTwistControlType\_e.swTwistControlFollowPath, false, false, (int)swTangencyType\_e.swTangencyNone, (int)swTangencyType\_e.swTangencyNone, false, 0, 0, (int)swThinWallType\_e.swThinWallOneDirection, 0, true, true, true, 0, true, false, 0, 0);

\*/ //var mf2 = (Feature)swModel.FeatureManager.InsertProtrusionSwept4(false, false, (int)swTwistControlType\_e.swTwistControlFollowPathFirstGuideCurve, false, (int)swTangencyType\_e.swTangencyNone, (int)swTangencyType\_e.swTangencyNone, false, 0, 0, (int)swThinWallType\_e.swThinWallOneDirection, 0, true, true, true, 0, true, false, 0, 0);