УДК 004.4

ПРОГРАММНОЕ СРЕДСТВО ЧТЕНИЯ ДАННЫХ ДЛЯ АВТОМАТИЗАЦИИ ПРОВЕРКИ КОНСТРУИРОВАНИЯ УЧЕБНОЙ 3D МОДЕЛИ

Гулиев Э. М. О., студент
Научный руководитель – к.т.н., доцент Полозков Ю. В.

Процесс обучения графическим дисциплинам связан с выполнением заданий по построению двумерных чертежей и трехмерных учебных моделей [1–3]. С точки зрения преподавания одним из наиболее трудоемких и субъективных этапов является проверка выполненных заданий [1]. Поэтому автоматизация контроля правильности выполнения геометро-графических построений является актуальной научно-технической задачей. Для ее решения разрабатывается специальное программное средство на языке программирования C# с использованием программного интерфейса приложения (*API*) *SolidWorks API*. В данном программном средстве предусмотрена реализация двух основных действий: автоматизация конструирования заданной учебной трехмерной модели и автоматизация контроля построений такой модели, которая на основе выданного варианта аксонометрического изображения сделана обучающимся в интерактивном режиме в среде *SolidWorks* (рис. 1). Первая часть полностью реализована. Вторая, более сложная часть, заключается в выполнении двух основных задач: чтение свойств модели для получения данных о ее геометрическом строении и сравнение полученных с заранее заданными значениями [2]. Очевидно, что от реализации алгоритмов чтения данных о модели во многом зависит окончательный результат автоматизированного контроля конструирования 3D модели.

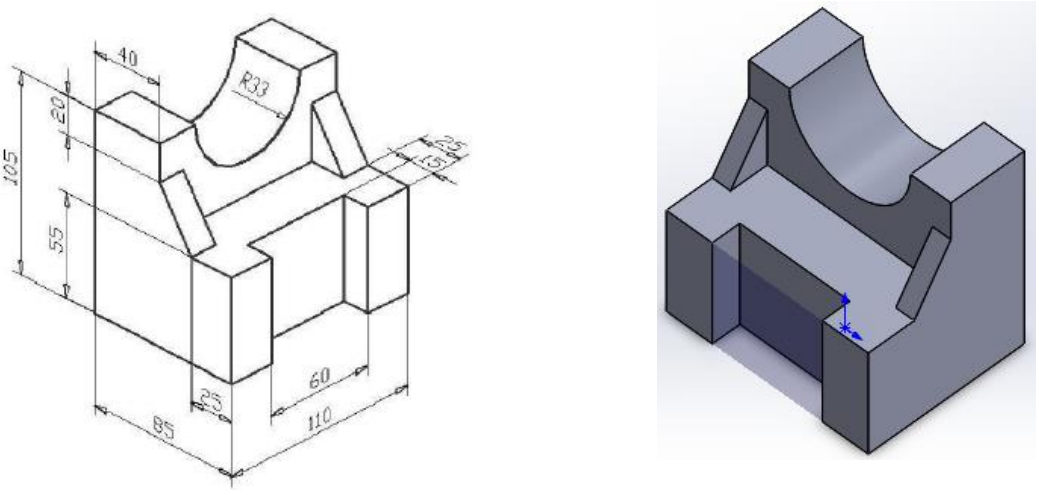

Рис. 1 – Пример задачи по конструированию учебной трехмерной модели

В процессе твердотельного моделирования 3D модель представляет собой совокупность 3D конструктивных элементов, которые строятся на основе создания двухмерных эскизов. Для задания эскиза вначале указывается системная плоскость,

грань модели или плоскость вспомогательной геометрии. Затем в эскизах создаются замкнутые двумерные контуры, которые в последующем служат основой для выполнения трехмерных операций твердотельного моделирования, таких как выдавливание, вращение, вырезание выдавливанием, вырезание вращением, операции выдавливания по траектории, по сечениям и др. Выполненные в ходе конструирования операции отображаются в виде узлов в дереве модели (рис. 2).

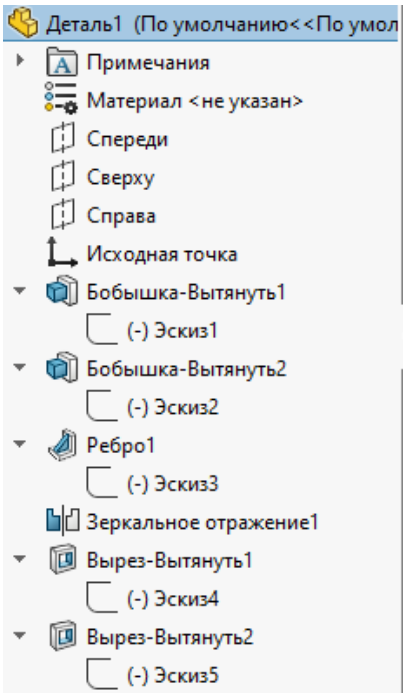

Рис. 2 – Дерево модели с выполненными конструктивными элементами

Автоматизация чтения конструктивных элементов 3D модели реализуется за счет получения доступа к информации узлов дерева трехмерной модели. Это обеспечивается с использованием следующих библиотек API SolidWorks: *SolidWorks.Interop.sldworks.dll* и *SolidWorks.Interop.swconst.dll*. Для извлечения информации из дерева модели используется метод GetFeatureTreeRootItem2() интерфейса *interface SolidWorks.Interop.sldworks.FeatureManager*, наследуемый от *interface SolidWorks.Interop.sldworks.IFeatureManager*. Для использования GetFeatureTreeRootItem2() передается параметр панели swFeatMgrPane_e. swFeatMgrPaneBottom интерфейса *SolidWorks.Interop.swconst.swFeatMgrPane_e*. Возвращаемым результатом метода GetFeatureTreeRootItem2(swFeatMgrPane_e. swFeatMgrPaneBottom) будет TreeControlItem интерфейса *interface SolidWorks. Interop. sldworks. TreeContrilItem*. Объект TreeControlItem будет иметь доступ к информации первого узла дерева. Для получения информации о следующем узле используется метод GetFirstChild() интерфейса TreeControlItem. Для получения свойств узла используются следующие методы: Object, ObjectType, Expanded интерфейса TreeControlItem: Object – получает объект SOLIDWORKS, связанный с этим элементом в дереве конструирования FeatureManager; Ob-

jectType – получает тип объекта SOLIDWORKS для этого элемента в дереве конструирования FeatureManager; Expanded – получает или задает, следует ли разворачивать этот элемент в дереве конструирования FeatureManager.

После получения доступа к узлу, требуется проверить, что узел не пустой. Если узел имеет объект, то проверяется тип и свойства этого объекта, чтобы узнать является ли он экземпляром Feature интерфеса *interface SolidWorks.Interop.sldworks.Feature*. Если объект равен true, то мы преобразовываем объект Object, в объект Feature. Для получения типа объекта вызывается метод GetTypeName(), а для названия объекта свойство Name интерфейса Feature. Если объект типа равен ProfileFeature, то данный объект является эскизом. Для получения свойств эскиза используется метод GetSpecificFeature2(). Полученный объект требуется преобразовать в Sketch интерфейса *interface SolidWorks.Interop.sldworks.Sketch,* наследуемый от интерфейса *interface SolidWorks.Interop.sldworks.ISketch.*

Для получения количества отдельных объектов, а именно двумерных примитивов, в эскизе используются следующие методы:

– *GetLineCount* – возвращает количество отрезков в эскизе;

– *GetArcCount* – возвращает количество дуг в эскизе;

– *GetUserPointsCount* – возвращает количество точек в эскизе;

– *GetParabolaCount* – возвращает количество парабол в эскизе;

– *GetEllipseCount* – возвращает количество эллипсов в эскизе.

Если количество обнаруженных объектов в эскизе не нулевое, то для получения свойств объектов в эскизе используются следующие методы: *GetLines2, GetArcs2(), GetUserPoints(), GetEllipses3(), GetParabolas2().* Например, для извлечения данных об отрезке используется метод *GetLines2*, имеющий следующую сигнатуру:

– *GetLines2(short CrossHatchOption)* – возвращает массив значений $[Color, Type, Line\ Font, Line\ Width, Layer\ ID, Layer\ Override, [Start], [End]]$, где *CrossHatchOption* – 4, только сплошные линии;

– *Color – COLORREF* определяет цвет как целое число. Возвращаемое значение может быть 0 или -1 для цвета по умолчанию;

– *Type* – тип линии;

– *Line Font* – стиль линии;

– *Line Width* – ширина линии;

– *Layer ID* – слой объекта;

– *Layer Override* – целое число с битовыми флагами, установленными для определения того, какие свойства, если таковые имеются, были переопределены по отношению к свойствам слоя по умолчанию;

– [*Start*] – массив из трех нецелочисленных значений (*X*, *Y*, *Z*), описывающее начало отрезка;

– [*End*] – массив из трех нецелочисленных значений (*X*, *Y*, *Z*), описывающее конец отрезка.

На основе этого метода программно реализована следующая процедура, позволяющая извлекать значения координат начальной и конечной точек отрезка:

```
    ///                                                                 <summary>
///Процедура извлечения значений координат начальной и конечной точек отрезка.
///                                                                 </summary>
///          <param name="sketch">Название эскиза</param>
private static void LineListener(string sketch)
{
    var selectedSketch = (Sketch) _featureNode.GetSpecificFeature2();
    var lineCount = selectedSketch.GetLineCount();
    var getLinesProperties = selectedSketch.GetLines2(4);
        _startEndLine = new List<string>();
    string start;
    string end;

    if (getLinesProperties is IEnumerable lineEnumerable && lineCount != 0)
    {
        var line = lineEnumerable.Cast<double>().ToArray();

        for (var i = 0; i < lineCount; i++)
        {
            TreeNode.LastNode.LastNode.Nodes.Add("Отрезок");

            if (i == lineCount) continue;
            start = "Начало: x = " + line[12 * i + 6] * 1000 + ", y = " + line[12
*  i + 7] * 1000 +
                    ", z = " + line[12 * i + 8] * 1000 + ";";
            end = "Конец: x = " + line[12 * i + 9] * 1000 + ", y = " + line[12 *
i + 10] * 1000 + ", z = " +
                  line[12 * i + 11] * 1000 + ";";

            _startEndLine.Add(start + "\n" + end);

            TreeNode.LastNode.LastNode.LastNode.Nodes.Add(start);
            TreeNode.LastNode.LastNode.LastNode.Nodes.Add(end);
        }
    }
     }.
```

Результат работы указанной процедуры представлены на рис. 3.

Более сложно описывается эллипс, данные о котором извлекаются с помощью метода *GetEllipses3()*. Этот метод имеет следующую сигнатуру:

$$\left[\begin{matrix}Color, Line\ Type, Line\ Font, Line\ Width, Layer\ ID, Layer\ Override, StartPt[3],\\ EndPt[3], CenterPt[3], MajorPt[3], MinorPt[3], Direction\end{matrix}\right],$$

где *Line Type* – тип линии; *StartPt*[3] – массив из трех нецелочисленных значений (*X*, *Y*, *Z*), описывающая начальную точку эллипса; *EndPt*[3] – массив из трех нецелочисленных значений (*X*, *Y*, *Z*), описывающая конечную точку эллипса; *CenterPt*[3] – массив из трех нецелочисленных значений (*X*, *Y*, *Z*), описывающая центральную точку эллипса; *MajorPt*[3] – массив из трех нецелочисленных значений (*X*, *Y*, *Z*), описывающую точку эллипса на большой оси; *MinorPt*[3] – массив из трех нецелочисленных значений (*X*, *Y*, *Z*), описывающую точку эллипса на малой

оси; *Direction* – направление вращения (по часовой стрелке = –1, против часовой стрелки = 1).

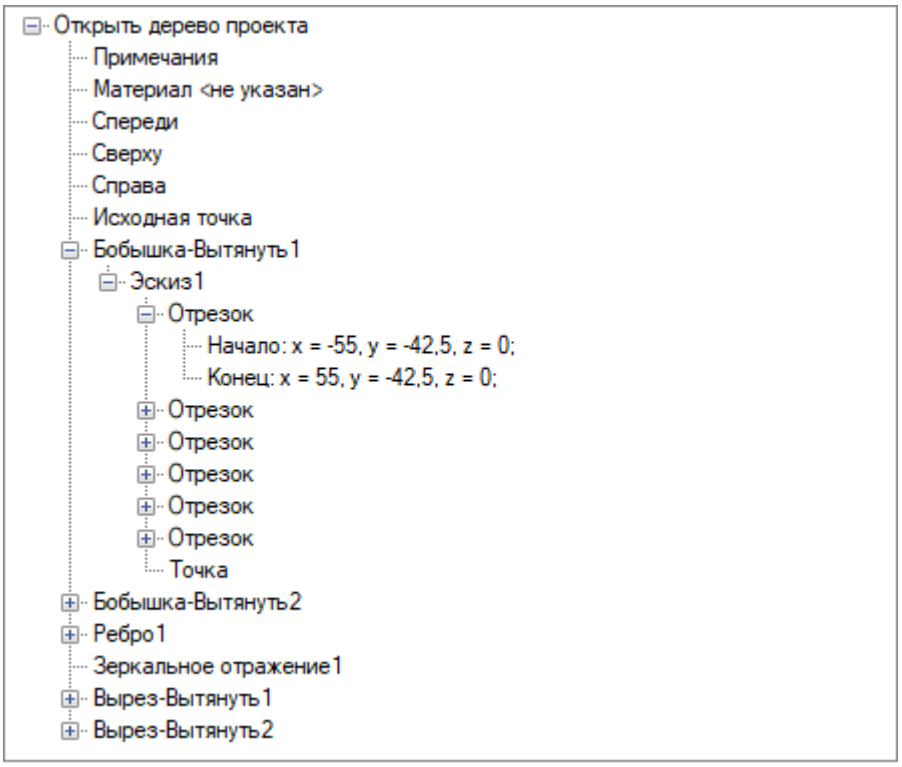

Рис. 3 – Пример результата работы метода GetLines2()

На основе этого метода, была реализована процедура по извлечению координат центра, начала, конца, точки на малой и большой оси эллипса, а также направления вращения:

```
     /// <summary>
///     Процедура позволяющая извлекать значения координат эллипса.
/// </summary>
/// <param name="sketch">Название эскиза</param>
private static void EllipseListener(string sketch)
{
    var selectedSketch = (Sketch) _featureNode.GetSpecificFeature2();
    var ellipseCount = selectedSketch.GetEllipseCount();
    var getEllipseProperties = selectedSketch.GetEllipses3();
    _startEndEllipse = new List<string>();
    string start;
    string end;

if (getEllipseProperties is IEnumerable ellipseEnumerable && ellipseCount != 0)
 {
    var ellipse = ellipseEnumerable.Cast<double>().ToArray();

    for (var i = 0; i < ellipseCount; i++)
    {
        TreeNode.LastNode.LastNode.Nodes.Add("Эллипс");

        if (i == ellipseCount) continue;
```

```
        start = "Начало: x = " + ellipse[16 * i + 6] * 1000 + ", y = " + el-
lipse[16 * i + 7] * 1000 +
                ", z = " + ellipse[16 * i + 8] * 1000 + ";";
        end = "Конец: x = " + ellipse[16 * i + 9] * 1000 + ", y = " + ellipse[16
* i + 10] * 1000 +
              ", z = " +
              ellipse[16 * i + 11] * 1000 + ";";
        var center = "Центр: x = " + ellipse[16 * i + 12] * 1000 + ", y = " +
ellipse[16 * i + 13] * 1000 +
                     ", z = " + ellipse[16 * i + 14] * 1000 + ";";
        var majorPoint = "Точка на большой оси: x = " + ellipse[16 * i + 15] *
1000 + ", y = " +
                         ellipse[16 * i + 16] * 1000 + ", z = " +
                         ellipse[16 * i + 17] * 1000 + ";";
        var minorPoint = "Точка на малой оси: x = " + ellipse[16 * i + 18] *
1000 + ", y = " +
                         ellipse[16 * i + 19] * 1000 + ", z = " +
                         ellipse[16 * i + 20] * 1000 + ";";

        _startEndEllipse.Add(start + "\n" + end + "\n" + center + "\n" + major-
Point + "\n" + minorPoint);
        TreeNode.LastNode.LastNode.LastNode.Nodes.Add(center);
        TreeNode.LastNode.LastNode.LastNode.Nodes.Add(start);
        TreeNode.LastNode.LastNode.LastNode.Nodes.Add(end);
        TreeNode.LastNode.LastNode.LastNode.Nodes.Add(majorPoint);
        TreeNode.LastNode.LastNode.LastNode.Nodes.Add(minorPoint);
    }
 }
     }.
```

На рис. 4 представлен результат работы процедуры по чтению параметров задания эллипса.

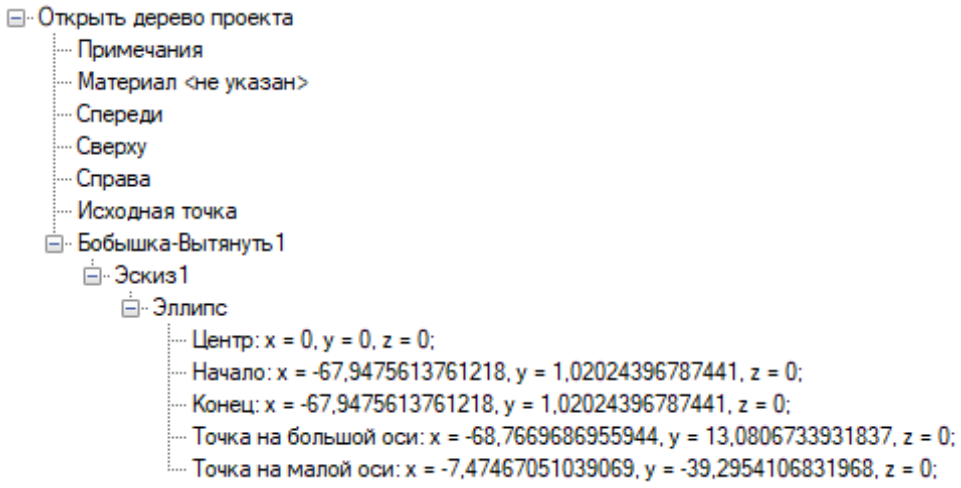

Рис. 4 – Пример результатов работы метода GetEllipses3()

Представленное программное средство предназначено для поддержки обучения конструктивному твердотельному моделированию в среде Solid Works. Его возможности позволяют автоматизировать конструирование учебной 3D модели для демонстрации преподавателем правильного варианта ее построения. Для этого приложения разработаны новые функции, реализованные на основе API Solid Works, для автоматизации чтения данных о конструктивных элементах

трехмерной модели, полученной в результате интерактивного твердотельного моделирования с применением теоретико-множественных операций. Получаемые с помощью этих функций данные позволят реализовать автоматизированный контроль правильности конструирования учебной 3D модели на каждом шаге ее построения. В связи с этим перспективные исследования направлены на алгоритмизацию сценариев контроля задач 3D моделирования, а также разработку алгоритмов по добавлению исходных данных для любой трехмерной модели и хранению их базе данных.

Литература

1. Полозков, Ю. В. Концепция интерактивного программного комплекса обучения и контроля знаний по начертательной геометрии / Ю. В. Полозков, В. И. Луцейкович, Вестник ПГУ. Сер. Е, Педагогические науки. – 2013. – № 15. – С. 48–56.

2. Полозков, Ю. В. Способы интерактивного взаимодействия при автоматизированном обучении начертательной геометрии / Ю. В. Полозков, В. И. Луцейкович // Инновации в образовании: материалы VI международной научно-практической конференции: сборник материалов докладов VI международной науч.-практ. конф., Орел, 14 мая 2014 г. / Орловский гос. аграрный. ун-т. – Орел, 2014. – С. 222–295.

3. Полозков, Ю. В. Сценарий автоматизированного контроля решения задач на построение 3D моделей деталей в Solid Works / Ю. В. Полозков, В. М. Будчанин // Информационные технологии и системы: проблемы, методы, решения (ИТС – 2017): материалы Республиканской научно-технической конференции; (Минск, 23–24 ноября 2017 г.): редкол.: С. В. Харитончик [и др.]. – Минск: Четыре четверти, 2018. – С. 246–249.