УДК 621.865.8

Методика разработки моделей микроэлектромеханических систем (МЭМС). API–функции виртуальной среды проектирования,тестирования и отладки МЭМС[1]

О. В. Даринцев, А. Б. Мигранов

Институт механики УНЦ РАН, Уфа

Аннотация. В статье рассматриваются вопросы моделирования микросистем, проблемы, связанные с разработкой виртуальных моделей и их использованием как интерфейсной надстройки в реальных микротехнологических модулях. Приводятся методики создания моделей микророботов и микросистем для виртуальной среды проектирования, тестирования и отладки МЭМС с API (Application Programm Interface) функциями ее открытой, модульной, наращиваемой архитектуры.

Ключевые слова: микроэлектромеханические системы, виртуальная среда, трехмерная визуализация вычислительных экспериментов, API–функции

1 Введение

В настоящее время микроэлектромеханические системы (МЭМС, на англ. – MicroElectroMechanical Systems – MEMS) с трехмерной структурой, объединяющие в своем составе электронные и механические компоненты (редукторы, датчики, двигатели, приводы и т.д.), с габаритными размерами в несколько миллиметров и менее, собираются вручную под микроскопом. Это, в первую очередь, сказывается на их надежности и стоимости. Для производства микросистем, аналогично «большим» системам,

[1] Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (№ 02–01–97916) и в рамках программы фундаментальных исследований РАН (гос.контракт № 10002–25/ОЭММПУ–4/080–093–535)

необходим переход к автоматизированным техпроцессам. Так, ожидается, что использование мобильных микроманипуляционных систем (микроботов) станет основой технологических операций при изготовлении и сборке MEMS–систем. В Институте механики УНЦ РАН предложено использовать гибридную структуру микротехнологической среды, реализованной на основе реальной [1] и полунатурной (виртуальной) [2] составляющих. Мобильные микроботы, сборочные узлы, технологическая оснастка, многоуровневая система технического зрения и др. оборудование — это реальная составляющая; библиотеки систем управления и планирования, графическая подсистема 3D–визуализации, средства наблюдения и контроля технологических процессов и др. обеспечение — полунатурная (виртуальная) составляющая. Разработка новых конструкций микроботов, новых алгоритмов планирования и управления для микротехнологических комплексов требует значительных аппаратно-временных затрат, особенно на этапах отладки и тестирования. Авторами для автоматизации рутинных операций производства микросистем предлагается более широко использовать полунатурные системы и модели виртуальной реальности при разработке микромеханических устройств и их дальнейшей эксплуатации.

Это позволит отказаться от изготовления натурных образцов, ограничить эксплуатацию сложного и дорогостоящего оборудования в лаборатории при проведении экспериментов. К примеру, средняя стоимость реальных прототипов микротехнологических сред составляет десятки тысяч у.е. Используемые в микроботототехнике электронные микроскопы и системы технического зрения могут дать только плоское изображение, а свойственная им недостаточная глубина резкости не позволяет одновременно получить отчетливое изображение детали и микроманипулятора. Поэтому еще одно применение виртуальной среды — это использование как более удобного инструмента восприятия, наблюдения и контроля человеком–оператором или технологом. Для рассмотрения выполняющихся техпроцессов с любого направления, с любым планом, «разрезов» интересующих устройств используются встроенные средства API DirectX 8.1, на основе которой построена графическая подсистема виртуальной среды. Другим вариантом практического применения виртуальной среды является возможность ее использования при обучении персонала навыкам работы с оборудованием, проведение в учебных заведениях виртуальных лабораторных работ для изучения микросистемотехники. Для конструктора наибольший интерес представляет использование виртуальной среды при разработке новых микроустройств и микромеханизмов. К примеру, с использованием прототипа виртуальной среды разработаны оригинальные MEMS–системы и отработаны принципы действия этих устройств; новизна разработок подтверждена патентами РФ.

Особенности виртуальной среды, связанные с необходимостью ее использования для управления, контроля в реальном микрокомплексе и тестирования на этапе отладки, накладывают такие требования к архитектуре программного средства, как модульность и открытость. Такой вариант реализации позволяет без каких-либо изменений основного кода внедрять в среду новые модели в виде объектов, микросистем, блоков систем управления и планирования, получая тем самым новые качества и характеристики. Разработка моделей микромеханических систем и микроманипуляционных устройств сопровождается рядом особенностей, которые связаны с их масштабом, конструктивным исполнением и принципами функционирования (сборки). Необходимо также учитывать возмущающие факторы при манипулировании и проблемы, связанные с позиционированием микророботов. Далее в статье рассматриваются методики создания моделей для практического применения в конкретных (виртуальных) средах с учетом перечисленных особенностей разработки микросистем и микроустройств.

2 Методика разработки моделей микророботов. Интерфейс MD–API виртуальной среды

В виртуальной среде обеспечивается возможность работать, как с моделями реальных микророботов, так и с моделями, прототипов которых не существует. Это позволяет уже на этапе проектирования оценивать эффективность конструкции, особенности управления и планирования. С точки зрения программирования модель микроробота — это динамическая библиотека, которая разрабатывается независимо от главных модулей виртуальной среды. Существуют различные области применения динамических библиотек:

1. отдельные библиотеки, содержащие дополнительные функции (библиотеки обработки изображений);
2. части программы (окна программы, формы и т.п.);
3. хранилища ресурсов (числовые массивы, двоичные данные и т.д.);
4. библиотеки поддержки (DirectX, OpenGL);
5. дополнения к программе, расширяющие ее возможности (plug-ins);
6. разделяемые ресурсы (используются сразу несколькими процессами).

Для связи с главными модулями в них используются интерфейсные функции. Далее под библиотекой будет пониматься программный модуль, являющийся дополнением к виртуальной среде и расширяющий ее возможности.

Таблица 1. Интерфейсные функции MD–API

Функция/Процедура	**Возвращаемое значение/выполняемое действие**
Базовые функции MD–API	
`GetPluginType`	Тип библиотеки расширения
`mmbGetPluginVersion`	Версия библиотеки
`mmbGetPluginAutor`	Авторы разработки; e-mail, www и т.д.
`mmbGetDescription`	Текстовое описание модели; основные геометрические размеры; параметры манипуляционного устройства: относительные координаты закрепления, число степеней свободы, тип кинематической пары, нулевые значения присоединенных параметров и т.д.
`mmbGetDnkStk`	Динамические и массогабаритные характеристики
`mmbGet_Avertices`	Массив пространственных координат вершин треугольников (граней)
`mmbGet_face_indicies`	Массив индексов на координаты вершин и направления нормалей
`mmbGet_normals`	Массив направления нормалей к граням
`mmbGet_Vertices`	Общее число вершин в 3D–модели
`mmbGet_Triangles`	Общее число граней в 3D–модели
`mmbGet_nNormals`	Общее число нормалей в 3D–модели
`mmbSlvIK`	Решение обратной задачи кинематики для манипулятора

Библиотека с моделью микроробота содержит описание геометрии микроробота (3D–модели платформы, основания, приводов, манипуляционной схемы и т.д.), прямой и обратной кинематики манипулятора, массогабаритных характеристик и динамических параметров (ускорение, линейная скорость, угловая скорость платформы; скоростные характеристики манипуляционного устройства).

Интерфейсные функции MD–API (Microrobot Development Application Program Interface), реализованные для связи с библиотеками моделей микророботов, перечислены в Табл. 1.

Создание библиотеки начинается с построения пространственной модели в пакете 3D–моделирования или САПР (AutoCAD, 3D Studio MAX, Lightwave и т.д.). Пространственная модель отражает конструктивные особенности реального (проектируемого) микроробота, соотношение масштабов и размеры. Моделирование в программе 3D Studio Max позволяет получить высокую степень реалистичности объектов за счет использования большого числа граней, мультитекстурирования, сглаживания поверхностей и т.д. Однако для отработки особенностей управления и планирования достаточно эскизной пространственной модели. Тем более, что визуализация с высокой степенью реалистичности впоследствии потребует соответствующих аппаратных ресурсов. К примеру, из двух представленных на Рис. 1 пространственных моделей манипуляционного устройства

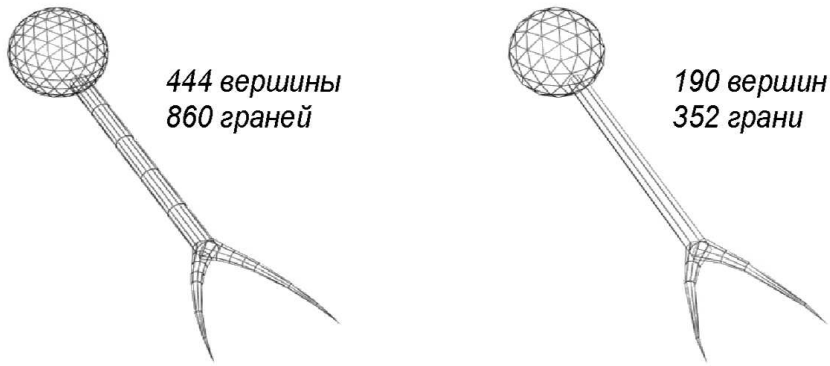

Рис. 1. 3D–модели манипуляционного устройства

более предпочтительным является эскиз справа. Для его построения использовалось меньшее число граней и вершин.

Интерфейсная часть библиотеки на языке PASCAL выглядит следующим образом:

```
function GetPluginType: PChar;export;stdcall;
function mmbGetPluginVersion: PChar;export;stdcall;
function mmbGetPluginAutor: PChar;export;stdcall;
function mmbGetDescription: TMicrorobotDescription;export; stdcall;
function mmbGetDnkStk: TDynamicVar;export;stdcall;
procedure mmbGet_Avertices(const Element: TMicrorobotElement; var
Result: array of single);export;stdcall;
procedure mmbGet_face_indicies(const Element: TMicrorobotElement;
var Result: array of integer);export;stdcall;
procedure mmbGet_normals(const Element: TMicrorobotElement; var
Result: array of single);export;stdcall;
function mmbGet_Triangles(const Element: TMicrorobotElement): integer;
export;stdcall;
function mmbGet_Vertices(const Element: TMicrorobotElement): integer;
export;stdcall;
function mmbGet_nNormals(const Element: TMicrorobotElement): integer;
export;stdcall;
function mmbSlvIK(const X: single; const Y: single; const Z: single;
const Angle: single): TMicrorobotManipulator;export;stdcall.
```

Далее приведем комментарии к функциям MD–API.

Функция `GetPluginType()` возвращает тип библиотеки расширения. Данная функция реализуется в библиотеках всех типов, и в соответствии с возвращаемым значением проводится идентификация библиотеки и реали-

зованных в ней функций. Для модулей с моделями микророботов функция должна всегда возвращать значение 0×101, если библиотека реализована PASCAL и 0×102, если на C (C++).

Функции `mmbGetPluginVersion()` и `mmbGetPluginAutor()` возвращают, соответственно, версию библиотеки и информацию об авторах–разработчиках, в которой содержатся ФИО авторов, координаты для связи (e-mail, www) и т.д.

В функции `mmbGetDescription()` типа *TMicrorobotDescription* реализуются геометрические характеристики модели и особенности построения манипуляционной схемы. В текстовом поле *Description* поясняются принципы работы, конструктивные и другие особенности. Габаритные размеры, занимаемые в рабочей зоне, возвращаются полями *Length, Width*. Эти данные используются планировщиком траекторий при нахождении оптимальных путей, предотвращения столкновений с препятствиями и столкновений микророботов между собой. В поле *MNP* типа *TMicrorobotManipulator* описывается кинематическая схема микроманипулятора. Число степеней свободы и, соответственно, число звеньев передаются через поле *DOF* структуры *TMicrorobotManipulator*. Последнее звено всегда является губками механизма захвата или инструментом. Поле *Joint* определяет класс кинематической пары в сочленении i-го и $i-1$ звена и может принимать одно из следующих значений: *RotXYZ, RotX, RotY, RotZ, TranX, TranY, TranZ.* Значение *RotXYZ* соответствует кинематической паре третьего класса, значения *RotX, RotY, RotZ, TranX, TranY, TranZ* определяют вращательные и поступательные сочленения относительно осей X, Y и Z (Рис. 2). В трехмерном векторе *Variable* содержатся значения присоединенных параметров i-го звена, при которых манипулятор находится в начальном положении. Однородная матрица *MatrixTransform* — это координаты и ориентация точки крепления i-го звена относительно $i-1$, для первого звена матрица определяет крепление относительно основания микроробота.

Функция `mmbGetDnkStk()` является опциональной, возвращает динамические и массогабаритные характеристики в виде структуры *TDynamicVar,* содержащей поля с максимальными значения линейной, угловой скорости и ускорения, массы роботов и отдельных звеньев.

Функции `mmbGet_Avertices(...)`, `mmbGet_face_indicies(...)`, `mmbGet_normals(...)`, `mmbGet_Triangles(...)`, `mmbGet_Vertices(...)`, `mmbGet_nNormals(...)` возвращают массивы данных, используемых графической подсистемой в процессе 3D–визуализации. Полная пространственная модель состоит из подмоделей, которые выделяются по функциональному назначению (позиционирование, манипулирование, захват и т.д.) или по конструктивным признакам (звено, схват, основание и т.д.). В качестве параметра функциям передается номер подмодели N. При $N = 1...m$ под-

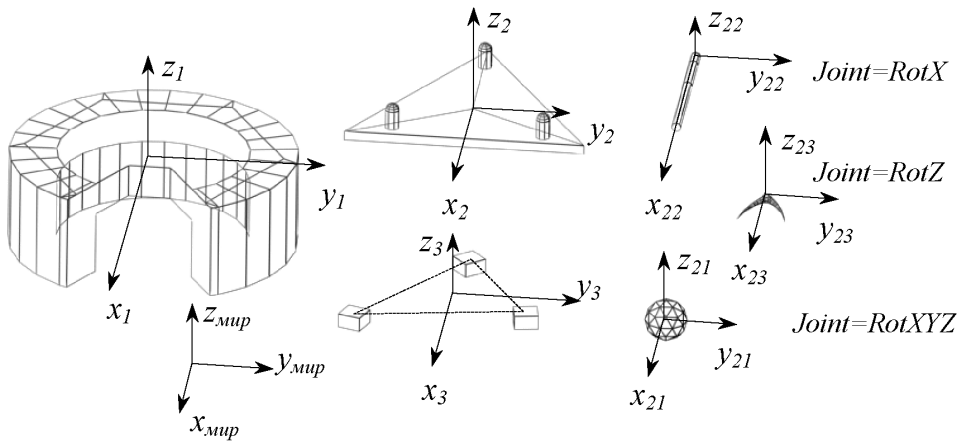

Рис. 2. Пространственные подмодели: основание, коммуникационная плата, звенья микроманипулятора и т.д.

модель является неподвижной частью конструкции (платформа, основание, приводы и др.) и при $N > m$ подвижной частью конструкции (шаровая основа, звенья манипулятора, схват и т.п.). Положение и ориентация в пространстве подвижных частей определяются дополнительными однородными матрицами. Общее число элементов (число подмоделей) является произвольным, однако число неподвижных частей ограничено величиной $m = 20$. Функциями `mmbGet_Avertices(...)` и `mmbGet_normals(...)` возвращаются, соответственно, массив пространственных координат вершин треугольников (граней) и массив направления нормалей. Для предотвращения избыточности данных используется функция `mmbGet_face_indicies (...)`, которая возвращает массив индексов на координаты вершин и направления нормалей к каждой грани. Данные для функций `mmbGet_Avertices(...)`, `mmbGet_face_indicies (...)` и `mmbGet_normals(...)` подготавливаются в трехмерном редакторе или извлекаются из файлов открытого формата, например, *.DXF. Функции `mmbGet_` Triangles(...), `mmbGet_Vertices (...)`, `mmbGet_nNormals(...)` возвращают, соответственно, общее число граней, вершин и нормалей в пространственной подмодели.

В функции `mmbSlvIK(...)` реализуется решение обратной задачи кинематики манипулятора. В качестве параметров передаются пространственные координаты целевой точки X, Y, Z и угол *Angle*, задающий желаемую ориентацию. Решение возвращается в виде структуры *TMicrorobotManipulator*.

Рассмотренная методика создания моделей микророботов была апробирована при тестировании, отладке алгоритмов в виртуальной микротех-

нологической среде и управлении реальными системами. Методики проходили тестирование при разработке новых конструкций микроустройств и механизмов. К примеру, разработанная по данной методике модель мобильного вакуумного микроробота после доработки конструктивной части и уточнения алгоритмов работы, получила патент на изобретение.

3 Методика разработки моделей микросистем. Интерфейс MSD–API виртуальной среды

Выделяется три этапа создания модели микросистемы. На первом этапе разрабатываются пространственные подмодели сборочных узлов, деталей и других компонентов, составляющих микросистему «в сборке». В сложных системах общее число компонентов может достигать 50, в более простых до 10. На этом этапе в качестве инструментов разработки используются трехмерные редакторы и программы 3D–моделирования. С их помощью возможно создание практически любых по сложности и конфигурации пространственных моделей и подмоделей микросистем. Используя средства программы 3D–моделирования (инструменты трансформация, ротация, линейка и др.) из разработанных подмоделей последовательно составляется единое целое (получение предварительного виртуального продукта сборки). Из множества подмоделей сборочных узлов выбирается один — базовый элемент, относительно которого определяются положение, ориентация, вектора атрибутов и другие параметры остальных компонентов микросистемы.

Для моделирования факторов окружающей среды на втором этапе оцениваются физические величины, которые в условиях микромира создают возмущающие воздействия различной природы. Это контактная разность потенциалов, поверхностная плотность заряда, коэффициент поверхностного натяжения, межатомное расстояние между поверхностями и др. параметры, относящиеся к сборочным узлам и характеризующие такие явления, как электростатические силы, силы поверхностного натяжения и силы Ван-дер-Ваальса. Наибольшее влияние на величины этих сил оказывают вещества, из которых изготавливаются сборочные узлы. Чаще всего это кремний, металлы или биоматериалы.

Третий этап — это разработка динамической библиотеки на языке C или PASCAL с использованием интерфейса MSD–API (Microsystems Development Application Programm Interface). Список интерфейсных функций для реализации библиотеки приведен в Табл. 2.

Интерфейсная часть библиотеки выглядит следующим образом:

```
function comGetPluginType: shortstring;export;stdcall;
function comGetPluginVersion: shortstring;export;stdcall;
```

Таблица 2. Интерфейсные функции MSD–API

Функция/Процедура	**Возвращаемое значение/выполняемое действие**
Базовые функции MSD-API	
`GetPluginType`	Тип библиотеки расширения
`comGetPluginVersion`	Версия библиотеки
`comGetPluginAutor`	Авторы разработки; e-mail, www и т.д.
`comGetCount`	Количество сборочных узлов в микросистеме
`comGetDescription`	Атрибутивные функции сборочного узла, взаимосвязи в конфигурации микросистемы и другие параметры сборки
`comGetPhysics`	Параметры моделирования окружающей среды
`comGet_Avertices`	Массив пространственных координат вершин треугольников (граней)
`comGet_face_indicies`	Массив индексов на координаты вершин и направления нормалей
`comGet_normals`	Массив направления нормалей к граням
`comGet_Vertices`	Общее число вершин в 3D–модели
`comGet_Triangles`	Общее число граней в 3D–модели
`comGet_nNormals`	Общее число нормалей в 3D–модели

```
function comGetPluginAutor: shortstring;export;stdcall;
function comGetCount: integer;export;stdcall;
function comGetDescription(PartNumber:integer): TPartDescription;
export;stdcall;
function comGetPhysik: TPhysics; export;stdcall;
procedure comGet_Avertices(const PartNumber:integer; var Result: array
of single);export;stdcall;
procedure comGet_face_indicies(const PartNumber:integer; var Result:
array of integer);export;stdcall;
procedure comGet_normals(const PartNumber:integer; var Result: array
of single);export;stdcall;
function comGet_Triangles(PartNumber:integer): Integer;export;stdcall;
function comGet_Vertices(PartNumber: integer): Integer;export;stdcall;
function comGet_nNormals(PartNumber: integer): Integer;export;stdcall;
```

Далее приведем комментарии к функциям MSD–API.

Функции `GetPluginType()`, `comGetPluginVersion()` и `comGetPluginAutor()` аналогичны одноименным функциям интерфейса MD–API. Для библиотек с моделями микросистем `GetPluginType()` должна всегда возвращать значение 0×201, если библиотека реализована PASCAL и 0×202, если на C (C++).

Функция `comGetCount()` возвращает количество сборочных узлов в микросистеме.

Функция `comGetDescription(...)` возвращает конфигурацию, геометрию, взаимосвязи между частями и механизмами сборочного узла *PartNumber* в виде структуры *TPartDescription*. Поля Fk, FMk, FVk, объявленные в структуре, являются массивами шестимерных булевых векторов типа *TAttributiveFunction*, определяющие атрибутивные функции сборочного узла. Степени свободы сборочного узла *PartNumber* относительно сборочного узла i в направлениях трехмерной системы координат описываются атрибутной функцией $Fk[j] = (d_{+x}, d_{-x}, d_{+y}, d_{-y}, d_{+z}, d_{-z})$, где j — индекс атрибутивной функции в массиве Fk, соответствующий сборочному узлу i. При $Fk[j][k] = d_p = 1$, $k = 1 \ldots 6$ сборочный узел *PartNumber* может свободно перемещаться в направлении относительно сборочного узла i, иначе $Fk[j][k] = d_p = 0$. Информация о свободе манипуляции одного сборочного узла относительно другого (критерий геометрической достижимости) передаются переменной FMk. Степени манипуляции сборочного узла *PartNumber* относительно сборочного узла i в направлениях трехмерной системы координат описываются атрибутной функцией $FMk[j] = (d_{+x}, d_{-x}, d_{+y}, d_{-y}, d_{+z}, d_{-z})$, где j — индекс атрибутивной функции в массиве FMk, соответствующий сборочному узлу i. При $FMk[j][k] = d_p = 1$, $k = 1 \ldots 6$ сборочный узел *PartNumber* имеет свободу манипуляции в направлении относительно сборочного узла i, иначе $FMk[j][k] = d_p = 0$. Критерий видимости определяется переменной FVk, которая описывает степень наблюдаемости сборочного узла *PartNumber* относительно узла i. Атрибутивная функция имеет вид $FVk[j] = (d_{+x}, d_{-x}, d_{+y}, d_{-y}, d_{+z}, d_{-z})$, где j — индекс атрибутивной функции в массиве FVk, соответствующий сборочному узлу i. При $FVk[j][k] = d_p = 1$, $k = 1 \ldots 6$ сборочный узел *PartNumber* наблюдается из точки, удаленной в направлении относительно сборочного узла i, иначе $d_p = 0$. Переменная RS это матрица размером $N \times N$, где N — количество сборочных узлов, переменная определяет все взаимосвязи в конфигурации микросистемы. Элементами матрицы являются относительные коэффициенты взаимосвязи сборочных узлов. В примере, показанном на Рис. 3, для первого сборочного узла (*PartNumber*$= 1$) переменные Fk, FMk и FVk имеют вид: $Fk[1] = (0,0,0,0,0,0)$; $Fk[2] = (0,0,0,0,0,1)$; $Fk[3] = (1,1,1,1,0,1)$; $Fk[4] = (1,1,1,1,1,1)$; $FMk[1] = (0,0,0,0,0,0)$; $FMk[2] = (0,0,0,0,1,0)$; $FMk[3] = (1,1,1,1,1,0)$; $FMk[4] = (1,1,1,1,1,1)$; $FVk[1] = (1,1,1,1,1,1)$; $FVk[2] = (1,1,1,1,1,0)$; $FVk[1] = (1,1,1,1,1,0)$; $FVk[1] = (1,1,1,1,0,1)$. Похожим образом переменные Fk, FMk и FVk определяются для сборочных узлов *PartNumber*$= 2, 3, 4$. Матрица RS для данной конфигурации имеет вид: $((1;0,4;0;0),(0,4;1;0,5;0),(0;0,5;1;0,4),(0;0;0,4;1))$. *XPurpose*, *YPurpose*, *Zpurpose* определяют положение точки начала локальной системы координат, связанной со сборочным узлом *PartNumber* в относительной системе координат базового узла. В примере, изображенном на

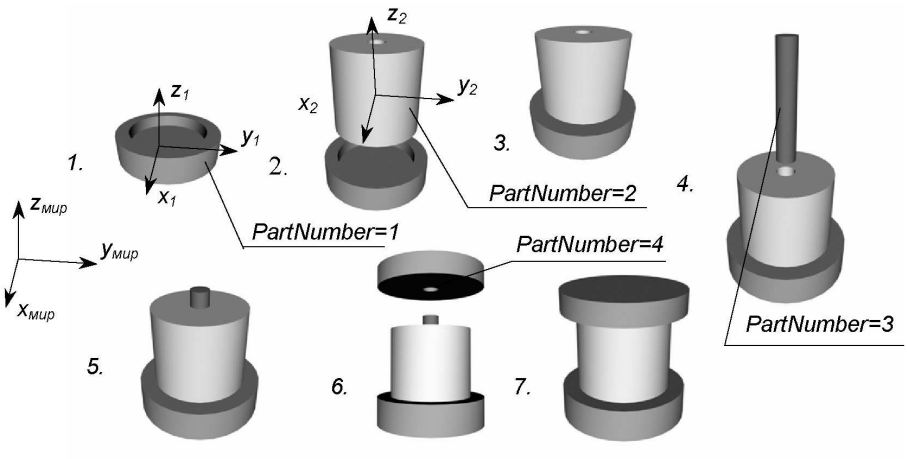

Рис. 3. Пример виртуальной сборки

Рис. 3, показаны системы координат $X_1Y_1Z_1$ и $X_2Y_2Z_2$ для сборочных узлов *PartNumber*= 1 (базовый) и *PartNumber*= 2. Точки начала систем координат выбираются на этапе разработки модели в программе 3D–моделирования. *CornerInitail*, *CornerFinal* определяют ориентацию микроробота в радианах при захвате (в локальной системе координат сборочного узла) и установке (относительно мировой системы координат). Переменная *CaptureDm* определяет расстояние между пальцами схвата манипулятора, при котором происходит захват сборочного узла. Дополнительные параметры сборки передаются переменной *AssemblyOptions* типа *TAssemblyOptions*, которая может принимать комбинацию из следующих значений: *AsNut*, *AsAdv*, *As180Deg*. Значение *AsNut* определяет, что для сборочного узла необходима операция ввинчивания, *AsAdv* движение основания микроробота должно обеспечивать дополнительную степень свободы при операциях со сборочными узлами, сопрягаемых с внутренними полостями других компонентов (например, втулки), *As180Deg* определяет необходимость операции крена на 180° вдоль оси OX манипулятора. Параметры для 3D–визуализации в графической подсистеме передаются переменной *Material*, это диффузионные свойства поверхности, рассеянная и зеркальная составляющая. Для их инициализации используется библиотечная функция `InitMateral(...)` API Direct3D. Переменная *ComponentName* — название сборочного узла.

Функция `comGetPhysics()` возвращает параметры моделирования внешней окружающей среды. Данные передаются структурой *TPhysics*, в которой определены поля *vc, sigma, eps, gamma, zm, d*. Поле *vc* — приблизительная контактная разность потенциалов для расчета электростатических сил, вызванных контактом сборочного узла с текущей конфигура-

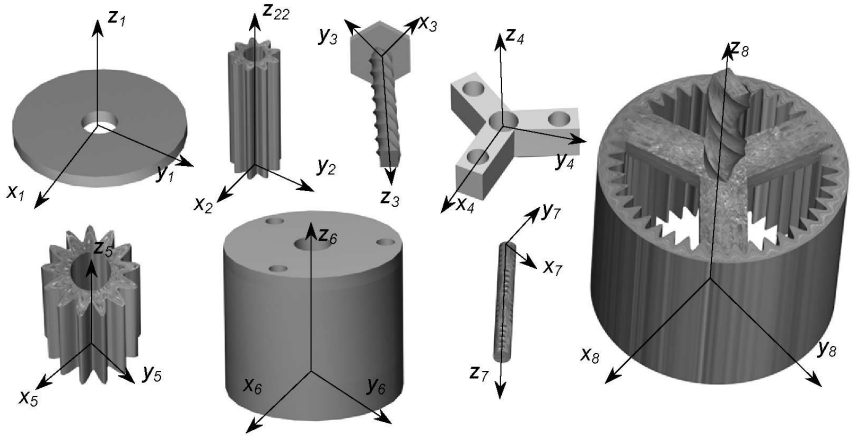

Рис. 4. Модели сборочных узлов

цией сборки [В]; *sigma* — поверхностная плотность заряда [Кл/м^2]; *eps* — диэлектрическая проницаемость среды [Ф/м]; *gamma* — коэффициент поверхностного натяжения [Н/м]; *zm* — среднее межатомное расстояние между поверхностью микрообъекта и схвата [м]; *d* — условная величина диаметра сборочного узла [м].

Функции `comGet_Avertices(...)`, `comGet_face_indicies(...)`, `comGet_normals(...)`, `comGet_Triangles(...)`, `comGet_Vertices(...)`, `comGet_nNormals(...)` возвращают структуры данных, используемых графической подсистемой в процессе 3D–визуализации. В качестве параметра передается номер сборочного узла *PartNumber*. В остальном функции полностью аналогичны одноименным интерфейса MD–API.

В качестве примера на Рис. 4 представлены модели некоторых деталей планетарного редуктора с направлением локальных систем координат.

Рассмотренная методика создания моделей микросистем используется при работе с новыми типами сборочных узлов и механизмов. Оператору технологических процессов недостаточно реальных изображений получаемых с микроскопа для восприятия и контроля рабочей сцены. Виртуальные модели микросистем используются для получения дополнительных проекций, разрезов, сечений микрокомпонентов и устройств, а также при тестировании и отладке интеллектуальных алгоритмов планирования траекторий (при описании рабочего пространства), системы планирования операций микросборки (с применением нейронной сети) и определения оптимальной последовательности сборки.

4 Заключение

Рассмотренные методики разработки моделей микромеханизмов и систем позволили ограничить эксплуатацию дорогостоящего оборудования в лаборатории, в 2–3 раза сократить время на проведение экспериментов, а также провести диагностику новых перспективных конструкций микророботов и их узлов без изготовления натурных образцов, оценить влияние различных параметров на функционирование, как отдельных компонентов микротехнологического модуля, так и всего комплекса в целом. Отладка и тестирование интеллектуальных алгоритмов планирования и управления микротехнологическим модулем проводилась также на базе виртуальных моделей.

В настоящее время ведутся работы по разработке методик создания подключаемых к виртуальной среде модулей с системами управления и планирования. В перспективах предполагается внедрение в микротехнологическую среду микророботов, а также их моделей, зарубежного производства, например, KHEPERA.

Список литературы

[1] Kusimov S., Ilyasov B., Darintsev O. and other. Flexible Multiagent Microrobotic Station for Microtechnologies and Precision Assembly (Desktop Microfactories) // Proceedings ASI2000&IIMB2000, Bordeaux, France, 2000 P. 359–365.

[2] Ilyasov B. G., Darincev O. V., Migranov A. B., Munasypov R. A., H. Woern. Using virtual microfactory as intelligence interface for the control of real microassembly station // Proceedings of the 5th International Workshop on Computer Science and Information Technologies (CSIT-2003), 16–18 Sept. 2003, Ufa. P. 300–305.

[3] Ilyasov B. G., Darincev O. V., Munassypov R. A., Migranov A. B. Decision Making on Control of Real Microsystems on the Virtual Model's Basis // Proc. of the IARP Intern. Workshop «MicroRobots, MicroMashines and MicroSystems» April 24–25, 2003, Moscow. P. 157–163

[4] Даринцев О. В., Мигранов А. Б. Виртуальная роботизированная микросборочная фабрика, алгоритмы интеллектуального планирования и управления // Научно–теоретический журнал «Искусственный интеллект» № 4. 2002 – ТШІ «Наука І Освіта», стр. 397–404.

[5] Охоцимский Д. Е., Ильясов Б. Г., Даринцев О. В. Мунасыпов Р. А., Мигранов А. Б. Проблемы взаимодействия виртуальных и реальных микросистем // Мобильные роботы и мехатронные системы: Мат. научн. шк.–конф. (Москва, 2–3 декабря 2002 года). М.: Изд-во МГУ, 2002. С. 5–17.