Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Двигатели внутреннего сгорания»

ОСНОВНЫЕ ПРИЕМЫ АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ ДЕТАЛЕЙ МАШИНОСТРОЕНИЯ В СИСТЕМАХ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

Учебно-методическое пособие

для студентов специальности 1-37 01 01

«Двигатели внутреннего сгорания»

Учебное электронное издание

Минск БНТУ 2011

УДК 519.6

А в т о р
А.Ю. Пилатов

Р е ц е н з е н т ы :
М.И. Жилевич, зам. декана АТФ, кандидат технических наук, доцент;
А.А. Москалев, старший преподаватель кафедры микропроцессорных систем и сетей ИИТ БГУИР

В пособии предложены лабораторные работы, выполнение которых дает необходимые навыки по основным приемам автоматизации проектирования деталей машиностроения. Рассмотрены задачи автоматизированного создания однотипных графических примитивов в системе автоматизированного проектирования AutoCAD 2007−2010 с использованием AutoLISP. Приведены примеры решения задач для самостоятельной работы. Для изучения учебно-методического пособия требуется знание курса математики, основ информатики и технического черчения в объеме программы технического вуза. Пособие предназначено для студентов специальности 1-37 01 01 «Двигатели внутреннего сгорания», занимающихся 2D/3D−проектированием и расчетами деталей и узлов машиностроения. Приводятся примеры решения конструкторских задач с помощью современного Windows-приложения – SolidWorks.

Дано подробное описание выполнения заданий в лабораторных работах. Издание может быть использовано студентами других технических специальностей.

Белорусский национальный технический университет.
пр-т Независимости, 65. 220013, г. Минск, Республика Беларусь
Тел.(017)292-77-52 факс (017)292-91-37
Регистрационный № ЭИ БНТУ/АТФ16-3.2011

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Системы автоматизированного проектирования, в частности AutoCAD, SolidWorks, предоставляют конструкторам и программистам обширные возможности проектировать 2D/3D−геометрию компьютерных моделей деталей и узлов машиностроения. Этому способствуют адаптация этих систем под логику инженера-механика, функциональность, а также инструменты разработки пользовательских приложений и специальных настроек.

Одной из самых распространенных благодаря своей доступности начинающему разработчику является система AutoCAD. Она дополнена многочисленными возможностями настройки и адаптации, инструментами разработки пользовательских приложений, а также обеспечивает доступ к своей объектной модели с помощью встроенного языка AutoLISP с добавлением средств ActiveX Automation.

Наряду с системой AutoCAD система SolidWorks также обладает широкими возможностями, в частности средствами автоматизации создания 3D−моделей деталей и сборок, которые сложно представить в рамках этого небольшого пособия. Поэтому в данной книге основное внимание сосредоточено на описании приемов, с помощью которых можно выполнить трехмерное моделирование деталей различной степени сложности. В программном обеспечении SolidWorks имеется учебное пособие, позволяющее самостоятельно подробно, пошагово осваивать основные приемы моделирования деталей и сборок.

ЛАБОРАТОРНАЯ РАБОТА № 1
Язык AutoLISP. Общие сведения. Основные функции. Средство разработки приложений

Цель работы: освоить основные приемы программирования в среде AutoLISP САПР AutoCAD.

Основные положения

1. Обзор средств программирования и объектная модель САПР AutoCAD

Система AutoCAD предоставляет программистам и опытным пользователям обширные возможности создавать свои приложения и строить на базе графического процессора новые системы автоматизированного проектирования.

Вместе с системой AutoCAD поставляются среда разработки на языке LISP (Visual LISP) и среда разработки на языке Basic (VBA).

Язык AutoLISP является вариантом стандартного языка LISP, который появился в системе AutoCAD версии 2.1 в 1982 году и стал основным языком создания дополнительных возможностей пользователя. Этот язык достаточно прост в освоении и изучении для не программистов и предоставляет доступ ко всем основным элементам и объектам системы.

Спустя десять лет фирма Autodesk сначала расширила язык за счет средств языка Visual LISP, а затем добавила средства ActiveX Automation, предоставляющие доступ к объектной модели AutoCAD.

Пользователь может работать как с традиционными (AutoLISP, Visual LISP, VBA), так и с нетрадиционными (DCL, ObjectARX) средствами для создания своих приложений, а также использовать другие распространенные в настоящее время языки программирования (Fortran, Delphi, Visual Basic, .NET), в основе чего лежит применение COM-серверов.

COM (Component Object Model − модель компонентных объектов) – концепция фирмы Microsoft, внедрение которой позволяет обращаться к объектам системы AutoCAD из других систем и их приложений и, наоборот, к объектам других систем из системы AutoCAD. COM – это спецификация метода создания компонентов, из которых можно строить приложения. Благодаря COM система AutoCAD дает возможность использовать технологию ActiveX, которая согласно идеологии объектно-ориентированного программирования оперирует не байтами, числами, а объектами прикладной среды, и работает с ними на языке обычных манипуляций. Таким образом, объектная модель работающего приложения представляет собой совокупность объектов, свойств, методов и событий. Для

каждого из этих элементов модели имеется своя реализация в виде данных специальных типов (структур, классов) и операций, обеспечивающих взаимодействие с пользователем.

Объекты (Objects) в системе AutoCAD рассматриваются как иерархия, содержащая не только графические примитивы (*точка*, *отрезок*, *дуга*), но и сложные элементы (*Blocks* (*блоки*), *Layers* (*слои*), *DimStyles* (*размерные стили*)). Корневым элементом такой иерархии является объект *Application* (*AcadApplication*), а остальные находятся на более низких уровнях. На рисунке 1.1 представлены основные иерархические взаимосвязи объектов ActiveX в системе AutoCAD.

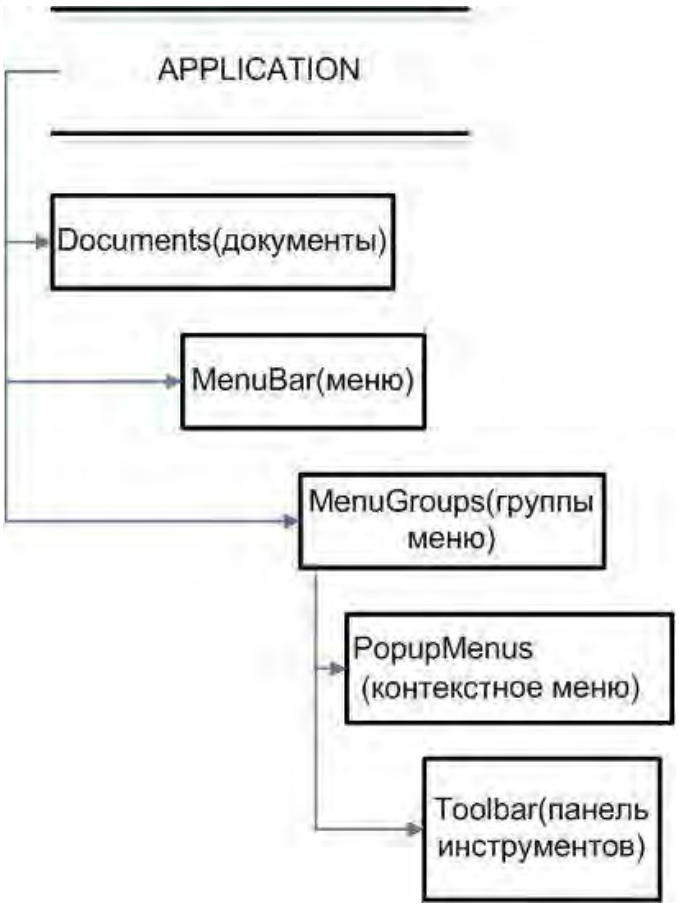

Рисунок 1.1 − Иерархия объектов ActiveX системы AutoCAD

2. Язык программирования Visual LISP и интегрированная среда разработки Visual LISP Editor

Основным средством разработки приложений в системе AutoCAD является язык программирования Visual LISP. Это вариант языка, в который добавлены функции доступа к объектам и таблицам AutoCAD.

Начиная с тринадцатой версии системы AutoCAD, язык расширил свои возможности за счет технологии ActiveX. Он получил интегрированную среду разработки (рисунок 1.2), доступ к которой осуществляется выполнением следующих действий:

Меню **Сервис →AutoLISP→Visual LISP Editor.**

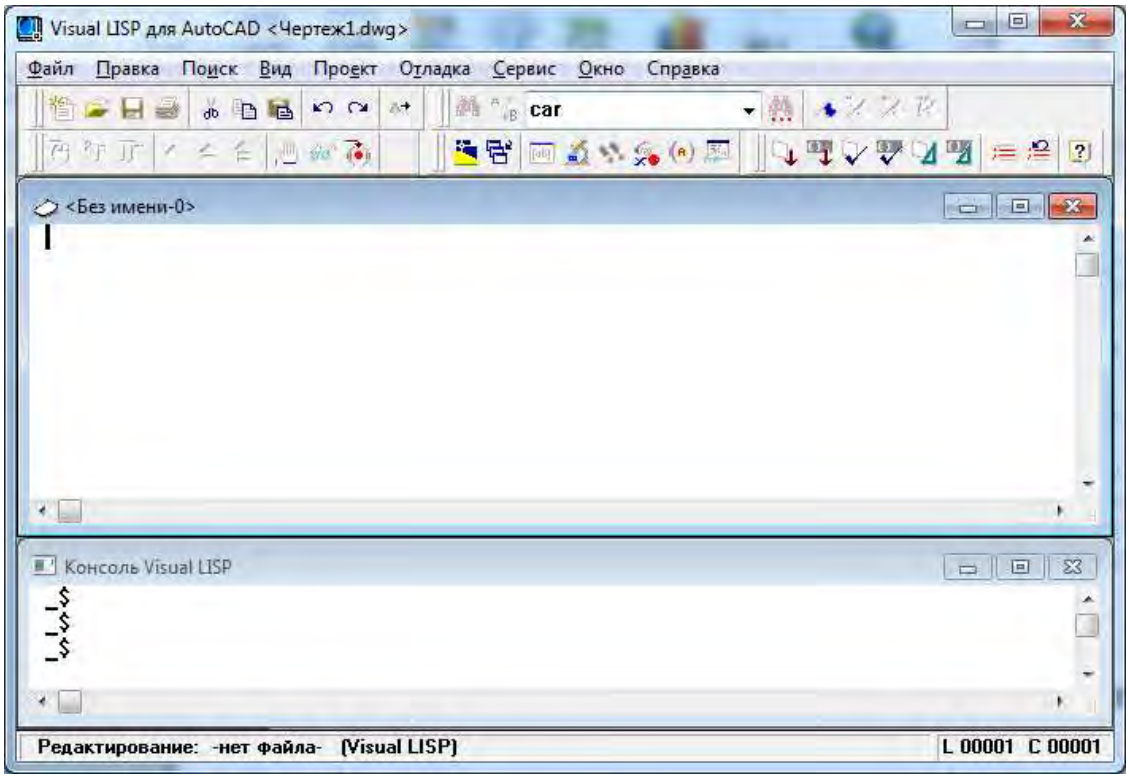

Рисунок 1.2 – Окно редактора Visual LISP системы AutoCAD

С помощью языка Visual LISP можно писать программы или вводить в командной строке выражения, которые затем вычисляет система AutoCAD.

Вычисляемые Visual LISP выражения должны удовлетворять следующей форме:

(<функция> [<аргумент1> [<аргумент2> … [<аргументN>]…]]),

где <функция> − имя функции;

<аргумент1>, <аргумент2>, <аргументN> − аргументы функции, разделяемые одним пробелом.

Квадратные скобки указывают на то, что, начиная с некоторого места, аргументы могут быть опущены. Их количество зависит от синтаксиса функции.

Основное правило Visual LISP – это баланс скобок, т. е. сколько скобок открыто, столько должно быть и закрыто. LISP - выражения могут быть как простыми, так и составными. Последнее указывает на то, что в качестве элементов могут использоваться другие выражения. Разделяющими знаками при этом выступают круглые скобки, пробелы и двойные кавычки.

Например:

(- (+ 14.2 3.09) (*12.5 0,93))

Сначала интерпретатор языка Visual LISP вычислит выражение во внутренних скобках, а затем подставит эти значения и вычислит полученное выражение, установив, что выражение равно 5.655.

Интерпретатор языка LISP вычисляет значение каждого введенного выражения и возвращает его в среду AutoCAD. Возвращаемое значение может быть либо передано для дальнейших вычислений в выражение более высокого уровня, либо сохранено в переменной с помощью специальной функции присвоения, имеющей формат

(setq <переменная1> <выражение1> [<переменная2>
<выражение2> … [<переменнаяN> <выражениеN>] …])

Функция **setq** может использоваться с любым количеством аргументов, которое должно быть обязательно четным и не менее двух. Это основное средство для сохранения значений, возвращаемых другими выражениями:

(setq my_result (- (+ 14.2 3.09) (*12.5 0,93)))

Значение выражения 5.655 будет сохранено в переменной **my_result**. Прочитать значение переменной можно с помощью операции **!** (восклицательный знак). Для этого необходимо в командной строке ввести:

!my_result

Символы языка Visual LISP – это слова, состоящие из группы буквенно-цифровых знаков и являющиеся именами функций или переменных, используемых для хранения данных пользователя. В символах языка не должны присутствовать разделители выражений, к которым относятся круглые скобки, пробелы и двойные кавычки, и служебные знаки – «.» (точка), «,» (запятая), «;» (точка с запятой), «’» (апостроф), «/» (косая черта), «\» (обратная косая черта). Написание символов не должно соответствовать написанию чисел. Зарезервированными являются символы, используемые как имена функций, а также некоторые обозначения констант. Регистр при этом значения не имеет.

Visual LISP работает с объектами следующих типов:

− целое число;
− вещественное число;
− строка;
− список, точечная пара;
− дескриптор файла;
− указатель функции, приложения;
− примитив AutoCAD;
− набор.

3. Основные функции языка Visual LISP

Арифметические функции:

«+» − сложение;

«-» − вычитание;

«*» − умножение;

«/» − деление;

«1+» − инкремент (увеличение на единицу);

«1-» − декремент (уменьшение на единицу);

abs – модуль числа.

Логические функции:

«=» − логическое равенство;

«/=» − логическое неравенство;

«<»− больше;

«>»− меньше;

«and» − логическое и;

«or» − логическое или;

«not» − логическое не;

«if» − условная операция *если*

«zerop» − проверка нуля;

«cond» − условная операция с любым количеством условий (аналог оператора *case* в языках высокого уровня).

Функции вычислений:

«sin» − вычисление синуса числа;

«cos» − вычисление косинуса числа;

«atan» − вычисление арктангенса;

«sqrt» − вычисление квадратного корня;

«expt» − возведение числа в степень;

«log» − вычисление натурального логарифма;

«exp» − экспонента числа;

«fix» − операция округления до целого в меньшую сторону;

«rem» − остаток от деления;

«max» − максимальное из чисел;

«min» − минимальное из чисел.

3.1. Операторы циклов

«while» − выполняет операцию цикла многократно по проверяемому условию:

(while <условие> [<выражение1> [<выражение2> … [<выражениеN>]…]])

«repeat» − выполняет операцию цикла с фиксированным количеством повторений:

(repeat <количество> [<выражение1> …]) – возвращает значение последнего вычисленного выражения.

4. Создание и загрузка программы

Текст программы создается в любом текстовом редакторе, например, Notepad.exe (Блокнот) и сохраняется с расширением *.lsp .

Загрузить файл программы в AutoCAD можно с помощью функции

(load <имя_файла> [<сообщение>]),

где <имя_файла> – это строка с именем загружаемого файла;

[<сообщение>] – строка с текстом сообщения, которое нужно вывести, если загружаемый файл не будет обнаружен по каким-либо причинам. Данный параметр не является обязательным. Если данный параметр опущен, а файл не загружен – при ошибке загрузки выдается стандартное сообщение системы AutoCAD.

Например:

(load “weight_calc.lisp» «Файл с программой не найден»)

Примеры программ

1. Функция if

(if (> a b) 2 8)

возвращает 2, если a > b, и 8 – в других случаях.

(if (= x1 x2) (+ x1 102) (* x1 3.04))

возвращает результат вычисления выражения (+ x1 102), если значения x1 и x2 равны; иначе – результат вычисления выражения (* x1 3.04).

(if (< a b) «a<b»)

возвращает строку «a < b» или **nil** (логическая константа − ложь).

(if nil «Initialization»)

возвращает **nil** (ничего), т. к. данное <условие> всегда ложно.

2. Функция prong

Объединяет несколько выражений в одно, когда по синтаксису LISP может использоваться только одно (например, в функции **if**). Часто данный оператор используется совместно с функцией **if**.

```
(if (> a b)
  (progn
     (setq c-1)
     (setq d ( - c a b))
     (* d b)
); конец progn
  (progn
    (setq c 2)
    (setq d (* a b))
    (* d c)
); конец progn
); конец if
```

В данном примере при истинности условия выполняются выражения (setq c-1), (setq d (- c a b)), (* d b). Если выражение (> a b) – ложь, то рассчитываются выражения (setq c 2), (setq d (* a b)), (* d c).

3. Функция cond

```
(setq mycolor
  ( cond
        ((= cc 1) «красный»)
        ((= cc 2) «желтый»)
        ((= cc 3) «зеленый»)
        ((= cc 4) «синий»)
        ((= cc 5) «голубой»)
        ( T «»)
  ); конец cond
  ); конец setq
```

Функция **setq** присваивает переменной **mycolor** результат вычисления функции **cond**. Это значение вычисляется по следующей схеме. Сначала рассматривается список, заданный в качестве первого аргумента (список – массив постоянных элементов любого типа, например, `(0.0 0.0 1.0) – точка с координатами x=0.0, y=0.0, z=1.0), и проверяется условие, являющееся первым элементом этого списка. Если оказалось, что значение cc равно 1, то дальнейшее рассмотрение аргументов

```
        ((= cc 2) «желтый»)
        ((= cc 3) «зеленый»)
```

((= cc 4) «синий»)
(T «»)

функции **cond** не выполняется, а в качестве возвращаемого значения принимается строка «красный». Если первое условие не выполнилось, проверяется второе, и т. д. При невыполнении первых пяти условий, выполняется шестое условие в любом случае, так как в данном условии стоит логическая константа «T» − true. Указанная ветвь оператора приведет к возвращению в переменную **mycolor** пустой строки «».

4. Циклы

```
(setq n 10) ; Задание числа, факториал которого необходимо вычислить
(setq i 1 factorial 1)
(while (< i n)
  (setq i (1+ i))
  (setq factorial (* factorial i))
); конец while
```

В данном примере производится расчет факториала числа n = 10. В программе используются переменные i (является счетчиком цикла) и факториал (накапливает произведение чисел и формирует факториал).

Функция **while** проверяет условие (< i n) для текущего значения **i**, и если результат вычисления отличен от **nil**, то выполняются внутренние операции. На первом шаге i=1 условие дальнейшего выполнения цикла **while** (< i n) возвращает значение **T** (истина). После этого значение **i** увеличивается на 1 и переменные **factorial** и **i** перемножаются, а результат записывается в переменную **factorial**. Далее передается управление на вход и начинается второй шаг цикла. Выполнение продолжается до тех пор, пока i < 10.

Задание к лабораторной работе

Создайте программу расчета факториала n = 20. Дополнительное условие: из расчетной последовательности исключить числа, делящиеся на Ваш порядковый номер в журнале.

Содержание отчета

1. Титульный лист.
2. Цель работы с учетом выполненного задания.
3. Алгоритм программы.
4. Листинг программы.
5. Скриншоты результатов выполнения программы в системе AutoCAD.

ЛАБОРАТОРНАЯ РАБОТА № 2
Автоматизированное построение графических примитивов в системе AutoCAD

Цель работы: Освоить основные приемы работы с примитивами системы AutoCAD.

Основные положения

1. Примитивы AutoCAD

Рисунок в системе AutoCAD имеет организацию, аналогичную организации базы данных, в которой элементы (графические примитивы и неграфические объекты) имеют списковую структуру. Каждый примитив имеет свой тип. Основные примитивы системы AutoCAD представлены в таблице 2.1

Таблица 2.1 − Основные примитивы системы AutoCAD

Обозначение в системе AutoCAD	Интерпретация обозначения
ARC	дуга
CIRCLE	Круг
DIMENSION	Размер
ELLIPSE	Эллипс
HATCH	Штриховка
IMAGE	Рисунок
LINE	Отрезок
MTEXT	Многострочный текст
POINT	Точка
POLYLINE	Ломаная
SPLINE	Сплайн
TOLERANCE	Допуск

Как правило, имя объекта совпадает с именем команды системы AutoCAD, которая создает графический объект.

С помощью LISP-программы можно получать доступ к спискам с данными примитивов текущего рисунка. Схема работы следующая:

1. Получают имя нужного примитива с помощью функций **entlast** и **entnext** или **entsel**. Данные функции имеют следующий формат, например:

```
( setq eela (entlast))
```

В переменную **eela** возвращается имя последнего неудаленного примитива в следующем виде:

<Entity name: 7ef8bf68>

Функции **entnext** и **entsel** имеют аналогичный формат записи. Функция **entnext** возвращает имя следующего неудаленного объекта, а функция **entsel** возвращает имя выделенного объекта.

2. С помощью функции **entget** получают список с данными примитива, который состоит из точечных пар с различными свойствами и их координатами.

Например, новый примитив с помощью выражения можно записать следующим образом:

(command “_LINE» ` (261.17 23.8 -111.429) `(72.047 0.52 30.622) “”)

Теперь выражение **(setq elst (entget (entlast)))** вернет примерно такой результат:

((-1 . <Entity name: 40065e28>) (0 . “LINE”) (330 . <Entity name: 40065cf8>) (5. “6D”) (100 . “AcDbEntity”) (67 . 0) (410 . “Model”) (8 . “0”) (62 . 5) (100 . “AcDbLine”) (10 261.17 23.8 -111.429) (1172.047 0.52 30.622) (210 0.0 0.0 1.0))

Полученный список состоит из точечных пар и координатных списков, у которых первым элементом является цифровой DXF-код, а оставшаяся часть − данные этого кода − DXF-код − от *drawing interchange format (формат для обмена чертежами* − текстовый (ASCII) или двоичный формат файлов для экспорта чертежей в другие приложения или для импорта чертежей из других приложений).

Вся информация о построенном объекте LINE записывается в переменную **elst**. Для того, чтобы извлечь нужные значения из полученного результата в Visual LISP, предусмотрена комбинация функций **cdr** и **assoc**. Например:

(cdr (assoc -1 est)) возвращает <Entity name: 40065e28> − имя примитива

(cdr (assoc 0 est)) возвращает “LINE” − тип примитива

(cdr (assoc 62 est)) возвращает 5 − цвет (номер цвета) примитива

(cdr (assoc 10 est)) возвращает (261.17 23.8 -111.429) − список с координатами начальной точки примитива.

3. С помощью функций **entmake** и **entmod** создают новые примитивы или редактируют существующие, независимо от включенных в системе режимов, например, **OSNAP** и **SNAP**.

1.1. Создание функций пользователя в VisualLISP

Для создания функции пользователя предназначена специальная функция **defun:**

(defun <символ>
([<символ1арг>…]/[<символ1раб>…])<выражение1>[…<выражениеN>]),

где <символ> − имя вашей функции;
<символ1арг> − символы локальных переменных, которые являются аргументами вашей функции;
<символ1раб> − символы локальных переменных, инициализируемых и используемых только внутри программы;
<выражение1>, … <выражениеN> − LISP выражения, составляющие тело вашей функции.

1.2. Пример функции пользователя, создающей отрезок и окружность

```
(defun your_programm (startPt endPt crnter radius / )

; startPt – координаты 1-й точки
; endpt – координаты 2-й точки
; Построение отрезка по двум точкам startPt и endPt

(entmake (list `(0. “LINE”) (cons 10 startPt) (cons 11 endPt)))

; Построение окружности по центру и радиусу
; center – точка центра
; radius – радиус

(entmake (list `(0 . “CIRCE”) (cons 10 center) (cons 40 radius)))
); defun
```

Вызов созданной функции пользователя осуществляется набором в командной строке следующего текста после стандартной загрузки LISP-файла с помощью функции **load:**

your_programm (list 32 112.5 0) (list 161.2 140.8 0) (list 44 99.9 0) 50.0)

2. Выполнение команд AutoCAD из программы пользователя

Один из способов, применяемых в программах на языке LISP для выполнения построений в текущем рисунке, – вызов команд системы AutoCAD. Основной функцией, которая позволяет это делать, является функция **command**, которая имитирует ввод в командной строке AutoCAD с клавиатуры. Синтаксис данной функции:

(command [<параметр1> [<параметр2> … [<параметрN>]…]])

Данная функция может вызываться без параметров, что равносильно нажатию клавиши <Esc>, что приводит к прерыванию текущей команды.

Для выполнения команд системы AutoCAD из программы можно вместо функции **command** использовать другую функцию – **vl-cmdf**, что бывает предпочтительнее, так как эта функция не прерывает работу программы в случае ошибки.

3. Некоторые специальные функции для работы с графическими примитивами

Для вычисления расстояния между двумя точками служит функция **distance**, имеющая следующий синтаксический формат:

(distance <точка1> <точка2>),

где <точка1> − координаты начальной точки отрезка, представленные в виде списка, например ‘(0.0 0.0 0.0);

<точка2> − координаты конечной точки отрезка, представленные в виде списка.

Если точки задаются в трехмерном пространстве, то функция автоматически рассчитывает угол между осью X и проекцией вектора, идущего из первой точки во вторую, на текущую плоскость построений.

Для вычисления координат точки, отстоящей от первой заданной точки на некоторое расстояние, служит функция **polar**, имеющая следующий синтаксис:

(polar <точка> <угол> <расстояние>),

где <точка> − координаты точки, представленные в виде списка;

<угол> − угол, образованный отрезком, координата конечной точки которого вычисляется данной функцией и осью X;

<расстояние> − длина отрезка, координата конечной точки которого вычисляется.

Примеры программ

В качестве примера рассмотрим текст программы, которая на слое FORMAT рисует линии штампа основной линии.

```
; Задание ширины жирной линии

(setq swidth 0.6)

; Запоминание текущих состояний режимов ORTHO, SNAP и OSNAP

(setq old_ortho (getvar “ORTHOMODE”)
  old_snap (getvar “SNAPMODE”)
  old_osnap (getvar “OSMODE”))

; Отключение режимов ORTHO, SNAP и OSNAP

(setvar “ORTHOMODE” 0)
(setvar “SNAPMODE” 0)
(setvar “OSMODE” 0)

; Создание слоя FORMAT

(command “_.LAYER” “_M” “FORMAT” “”)

; Рисование линий штампа

(command “_.PLINE” “0,55” “_W” swidth swidth “-185,55” “-185,0” “”
(command “_.PLINE” “-200,30” “-120,30” “”)
(command “_.PLINE” “-178,30” “-178,55” “”)
(command “_.PLINE” '(-168 0) '(-168 55) “”)
(command “_.PLINE” '(-145 0) '(-145 55) “”)
(command “_.PLINE” '(-130 0) '(-150 55) “”)
(command “_.PLINE” '(-120 0) '(-120 55) “”)
(command “_.PLINE” '(-50 0) '(-50 40) “”)
```

```
(command “_.PLINE” '(-35 20) '(-35 40) “”)
(command “_.PLINE” '(-18 20) '(-18 40) “”)
(command “_.PLINE” '(-30 15) '(-30 20) “”)
(command “_.PLINE” '(-120 15) '(0 15) “”)
(command “_.PLINE” '(-50 20) '(0 20) “”)
(command “_.PLINE” '(-50 35) '(0 35) “” “_.PLINE" '(-120 40) '(0 40) “”)
(command “_.LINE” “-185,5” “-120,5” “”)
(command “_.LINE” “-185,10” “-120,10” “”)
(command “_.LINE” “-185,15” “-120,15” “”)
(command “_.LINE” “-185,20” “-120,20” “”)
(command “_.LINE” “-185,25” “-120,25” “”)
(command “_.LINE” “-185,40” “-120,40” “”)
(command “_.LINE” “-185,45” “-120,45” “”)
(command “_.LINE” “-185,50” “-120,50” “” “_.LINE” “-45,20”)
(command “-45,35” “” “_.LINE” “-40,20” “-40,35”)
(command “”)

; Восстановление состояние режимов ORTHO, SNAP и OSNAP

(setvar “ORTHOMODE” old_ortho)
(setvar “SNAPMODE” old_snap)
(setvar “OSMODE” old_osnap)
```

В начале построений рамки (рисунок 2.1) задается ширина линий **swidth**, цвет, тип и вес линий, а также производится отключение режимов **SNAP**, **ORTHO**, **OSNAP**.

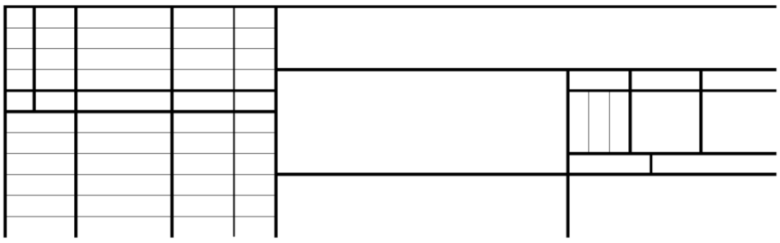

Рисунок 2.1 – Результат выполнения программы построения рамки чертежа

Первой в последнем листинге рисуется жирная полилиния верхней и левой границ штампа, которые имеют координаты (0, 55), (-185,55), (-185,0). Вызывается опция _W (ширина), которой передается значение переменной **swidth** в качестве начального и конечного значения ширины полилинии. После ввода последних двух точек с помощью строки “” команда завершается.

После этого несколько раз вызываются команды PLINE и LINE. В команде PLINE уже не требуется снова задавать ширину полилинии. По умолчанию будет действовать та ширина полилинии, которой мы нарисовали первую линию.

В конце программы с помощью соответствующих системных переменных режимы **SNAP**, **ORTHO**, **OSNAP** приводятся в состояние, в котором они были до начала работы программы.

Задание к лабораторной работе

1. Используя функцию **load** загрузить созданный в текстовом файле скрипт создания рамки чертежа (см. п. «Примеры программ» данной лабораторной работы) и, изучив его последовательность команд, исправить скрипт так, чтобы получилась рамка, представленная на рисунке 2.1.
2. Определить координаты точки касания отрезка окружности. Координаты начала отрезка, радиус и координаты центра окружности получить у преподавателя.

Содержание отчета

1. Титульный лист.
2. Цель работы с учетом выполненного задания.
3. Листинг исправленного скрипта построения рамки.
4. Листинг скрипта расчета координат точки касания отрезка с заданной начальной точкой окружности с заданным центром и радиусом.
5. Скриншоты результатов выполнения программы в системе AutoCAD.

ЛАБОРАТОРНАЯ РАБОТА № 3
Применение технологии COM для автоматизированной интеграции приложений

Цель работы: освоить основные приемы использования технологии COM для интеграции системы AutoCAD с приложениями, работающими с 32- разрядной ОС Windows.

Основные положения

Фирма Microsoft разработала модель COM (Component Object Model – модель компонентных объектов), позволяющую связывать самые разнородные приложения. Построенный по спецификации этой модели программный комплекс предоставляет описание своих компонентов и средств доступа к ним другим программам. Это дает возможность проектировать такие системы, в которых одни приложения (клиенты) обращаются к другим приложениям (серверам) для выполнения некоторых операций. При этом приложение-сервер может работать как в видимом режиме, так и в невидимом (работает в оперативной памяти, не открывая своего окна).

Приложения, поддерживающие технологию COM, при установке заносят в реестр Windows информацию о себе, своих компонентах, объектах. Так, например, имя приложения (**ProgID**), под которым оно может быть вызвано как приложение-сервер, заносится в реестр в раздел **HKEY_CLASSES_ROOT**. Для примера, версия AutoCAD 2006 будет зарегистрирована в реестре Windows как **AutoCAD.Application.16.2**.

Именно под таким именем нужно вызывать объект приложения AutoCAD 2006, если вы будете обращаться к системе AutoCAD из другого приложения. Также в реестре находится информация о том, как можно обратиться к любому зарегистрированному объекту и какие методы (функции обращения) к нему применимы.

1. Обращение к Word 2003 из системы AutoCAD

Приложение Microsoft Word 2003 в реестре имеет регистрацию как «Word.Application.11». Для работы с этим приложением необходимо либо создать в оперативной памяти новый экземпляр приложения, либо соединиться с тем экземпляром, который загружен и уже находится в памяти. При этом приложение, с которым мы работаем с помощью COM-технологии, не обязательно должно находиться в видимом режиме.

Для установления связи с Word может понадобиться одна из трех функций:

− **vlax-create-object** – создание нового объекта;

− **vlax-get-object** – соединение с существующим в памяти объектом;

− **vlax-get-or-create-object** – проверка существования объекта и соединения с существующим, или при его отсутствии, создание нового экземпляра приложения.

Например, создание приложения Word 2003 из системы AutoCAD:

(setq g_ow (vlax-get-or-create-object «Word.Application.11»))

Переменная **g_ow** содержит указатель на COM-объект, который занимает дополнительную память. Такие переменные рекомендуется делать глобальными и по окончании использования обязательно удалять при помощи функции **vlax-release-object**.

Для того, чтобы узнать какие свойства и методы доступны при работе с приложением Word 2003 в системе AutoCAD, рекомендуется воспользоваться функцией отображения этих свойств **vlax-dump-object**. Например:

(vlax-dump-object g_ow T),

где T – параметр, который указывает, что необходим вывод не только свойств, но и методов объекта **g_ow**.

При вызове любой доступной функции (метода) используется функция **vlax-invoke-method,** в которую необходимо передавать два параметра: первым должен быть объект, а вторым – имя метода. Для создания нового документа необходимо к семейству **Documents** (см. рисунок 1.1 в лабораторной работе № 1) применить метод **Add:**

(setq g_docs (vlax-get-property g_ow ‘Documents))
(setq g_doc (vlax-invoke-method g_docs ‘Add))

В приведенном примере новый документ образуется со свойствами по умолчанию и по стандартному шаблону.

2. Обращение к AutoCAD из пользовательского приложения

Применение такого способа основано на том, что сама система AutoCAD оформлена в виде COM-сервера, предоставляющего доступ к своей объектной модели. В основе лежит метод **CreateObject** для запуска приложения AutoCAD как COM-сервера, зарегистрированного в реестре Windows, или метода **GetObject**.

В программах, работающих с такими объектами, обязательно необходимо проверять, освобождается ли выделяемая под них оперативная память. Если этого не происходит, то следует освобождать программно с помощью метода **ReleaseObject**. Иначе может возникнуть эффект «невыгрузки» или «размножения» приложений в памяти.

Примеры программ

1. Создание нового документа Word 2003 в системе AutoCAD

```
(defun writew11_com (/ _re)
(vl-load-com)

; Загрузка  Word 2003 без открытых документов

(setq g_ow (vlax-get-or-create-object «Word.Application.11»))
(vlax-put-property g_ow ‘Visible :vlax-true)

; Создание нового документа

(setq g_docs (vlax-get-property g_ow ‘Documents))
(setq g_doc (vlax-invoke-mrthod g_docs ‘Add))

; Создание диапазона позиций, пределы которого указывают на позиции в строках созданного документа Word 2003

(setq g_r (vlax-invoke-method g_doc ‘Range 0 0 ))

; Вставка текста
(vlax-invoke-method g_r ‘InsertBefore “Заголовок нового документа”)

; Вставка абзаца

(vlax-invoke-method g_r ‘InsertParagraphAfter)
```

```
; Указание рабочего шрифта

(setq g_f (vlax-get-property g_r 'Font ))
(vlax-put-property g_f ' Name "Times New Roman")

; Вставка своего текста в созданный документ

(setq _pos (vlax-get property g_r 'End)
(setq g_r (vlax-invoke-method g_doc 'Range _pos _pos))
(vlax-invoke-method g_r 'InsertAfter "Моя группа")
(vlax-invoke-method g_r 'InsertParagraphAfter)
(vlax-invoke-method g_r 'InsertAfter "Моя фамилия")

; Сохранение документа

(vlax-invoke-method g_doc 'SaveAs "d:\\Users\MyWord.doc")

; Выход из Word

(vlax-invoke-method g_ow "Quit")

; Освобождение объектов и выгрузка Word из памяти

(if (and g_f (not (vlax-object-releaseed-p g_f)))
        (vlax-release-object g_f))
(if (and g_r (not (vlax-object-releaseed-p g_r)))
        (vlax-release-object g_r))
(if (and g_doc (not (vlax-object-releaseed-p g_doc)))
        (vlax-release-object g_doc))
(if (and g_docs (not (vlax-object-releaseed-p g_docs)))
        (vlax-release-object g_docs))
(if (and g_ow (not (vlax-object-releaseed-p g_ow)))
        (vlax-release-object g_ow))

(setq g_f nil g_r nil g_doc nil g_docs nil g-ow nil)
(gc)
); defun
```

2. Рисование в загружаемой системе AutoCAD отрезка, координаты которого определяются из пользовательской программы, созданной в среде разработки Delphi

```
unit Unit1;

interface
uses
    Windows, Messages, SysUntils, Variants, Classes, Graphics, Control, Forms, Dialogs, StdCtrls;
…

procedure TForm1.Button1Click(Sender: TObject);
var

{начальные и конечные точки отрезка}
StartPoint, EndPoint: OleVariant;

{задание пространства модели}

ModelSpace, Acad :OleVariant;

begin

{создание массивов координат в виде 8-байтовых действительных чисел VT_R8, которые предварительно определены в файле Active.pas среды разработки Delphi}

StartPoint:=VarArrrayCreate([0,2], VT_R8);
EndPoint:= VarArrayCreate([0,2], VT_R8);

{присваивание им значений}

StarPoint[0]:=1.0; //x
EndPoint[1]:=1.0; //y
StartPoint[2]:=0.0; //z

EndPoint[0]:=250.0;
EndPoint[1]:=250.0;
EndPoint[2]:=150.0;
```

```
{проверка того, запущен ли AutoCAD}
try
Acad:=GetActiveOleObject(‘AutoCAD.Application.16.2`);
except
        {если не запущен, то его запускаем}
Acad:=CreateOleObject(‘AutoCAD.Application.16.2`);
end;

{делаем AutoCAD видимым}

Acad.Visible:=True;

{создание объекта пространства модели в активном документе}

ModelSpace:=Acad.ActiveDocument.ModelSpace;

{рисуем линию}
ModelSpace.AddLine(VarArrayRef(StartPoint),
                                VarArrayRef(EndPoint)).Update;
end;
end.
```

Рассмотренная программа запускает простейшее диалоговое окно с одной кнопкой, щелчок по которой сначала создает варианты StartPoint и EndPoint в виде массива из трех вещественных чисел. Отрезок строится с помощью метода **Addline**, к которому обращается созданное пользовательское приложение.

Задание к лабораторной работе

1. Создать программу на Delphi, запускающую диалоговое окно, в котором задаются выданные преподавателем координаты отрезка, после закрытия которого открывается приложение AutoCAD с начерченным отрезком по указанными в пользовательском приложении координатам.
2. Создать скрипт на Visual LISP, при загрузке которого осуществляется черчение отрезка с данными координатами, а также открывается документ Microsoft Word, в котором эти координаты должны быть записаны.

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Листинг программы Delphi по заданию 1.
4. Листинг программы на Visual LISP по заданию 2.
5. Скриншоты выполнения программ.

ЛАБОРАТОРНАЯ РАБОТА № 4
Основные приемы моделирования простых деталей в системе SolidWorks 2009

Цель работы: освоить основные приемы трехмерного (3D) проектирования твердотельных простых деталей на примере системы автоматизированного проектирования SolidWorks 2009.

Основные положения

Программа SolidWorks 2009 представляет интегрированную среду трехмерного автоматизированного проектирования как простых, так и сложных деталей и конструкций, использующий графический интерфейс Microsoft Windows. Она предоставляет полный цикл моделирования: проектирование трехмерных деталей, сборка из отдельных деталей, сборочные чертежи и деталировки, а также представление моделей в реалистичном (визуализация) и динамичном (анимация) виде.

Процесс моделирования в SolidWorks начинается с создания эскиза, то есть двухмерного профиля или поперечного сечения. Затем эскиз при помощи определенного конструктивного элемента (бобышки, выреза, отверстия, скругления, фаски, оболочки и т. д.) приобретает трехмерный вид. Эскизы могут быть вытянуты, повернуты, рассечены сложным образом или смещены по контуру. Набор эскизов и конструктивных элементов образует деталь. Затем детали компонуются в сборку с помощью их взаимного расположения и сопряжения. После проверки работоспособности сборки на ее основе создается сборочный чертеж и чертежи входящих в сборку отдельных деталей.

Таким образом, в процессе моделирования создается не деталь, например, модель по трем проекциям (САПР AutoCAD), а алгоритм (последовательность операций) ее создания. Задаются размеры и геометрические взаимосвязи между элементами. Размеры, взаимосвязи и уравнения определяют форму конкретной детали. При изменении размеров изменяются форма и размеры детали, но сохраняется общий замысел проекта.

1. Основные преимущества САПР SolidWorks

Преимуществами САПР SolidWorks являются:

− технология SWIFT (Solid Works Intelligent Feature Technology), которая представляет собой автоматизированную систему анализа ошибок и проблем, которые могут возникнуть после фиксации на данном этапе своего решения проектировщиком при дальнейшем проектировании детали и сборки. «Умная» технология SWIFT предлагает конструктору исправить все выявленные на текущем этапе проектирования ошибки и проблемы;

− адаптированный (head-up) пользовательский интерфейс объединяет одинаковые функции управления проектом в общие группы утилиты

Command Manager, что упрощает проектировщику доступ к функциям работы с проектом и существенно сокращает время разработки сложных объектов;

− широкие возможности коллективной работы с документами и интерактивная поддержка пользователей;

− интеграция SolidWorks с другими САПР.

2. Главное окно SolidWorks

Главное окно (рисунок 4.1) системы автоматизированного проектирования SolidWorks включает в себя следующие элементы: главное меню, панели инструментов, рабочую область, интерактивную помощь, строку состояния.

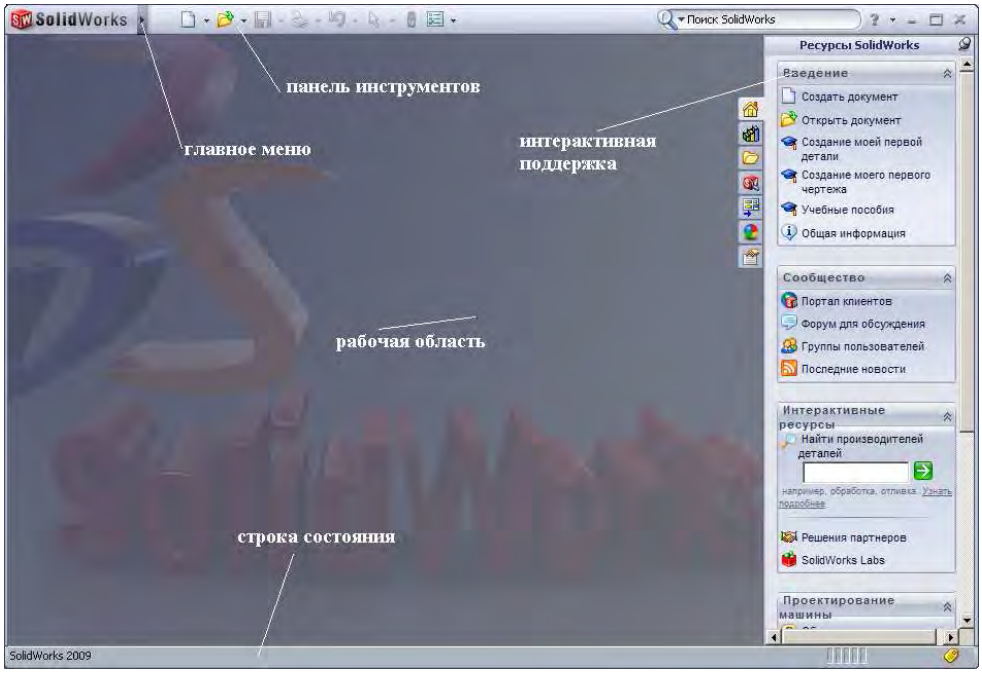

Рисунок 4.1 − Главное окно SolidWorks

Главное меню SolidWorks 2009 находится под строкой заголовка. Оно содержит пункты, доступные в настоящий момент. В зависимости от типа активного документа (деталь, сборка или чертеж) набор пунктов главного меню изменяется. Если не открыт ни один из документов, главное меню содержит четыре пункта: File (Файл), View (Вид), Tools (Инструменты) и Help (Справка). При наличии активного документа в главном меню добавляются следующие пункты: Edit (Правка), Insert (Вставка) и Window (Окно). Если активное окно документа распахнуто на весь экран, левее пункта меню File (Файл) появляется значок, соответствующий типу активного документа:  детали,  сборки или  чертежа. Если размер окна активного документа меньше рабочей области, значок расположен в заголовке соответствующего окна.

Рабочая область занимает все свободное пространство между панелью инструментов и строкой состояний. При отсутствии активных документов она пустая. Обычно рабочая область разделена на две части: диспетчерскую (слева) и графическую (справа). В дальнейшем эти области будут рассмотрены подробнее.

Строка состояний расположена в нижней части главного окна программы SolidWorks 2009. Содержание строки состояний зависит от типа активного документа и отображает такую информацию:

− имя активного документа или краткое описание того пункта меню или конструктивного элемента, на котором в данный момент находится курсор мыши;

− текущие координаты расположения курсора;

− состояние эскиза: Over Defined (Переопределен), Under Defined (Определен не полностью) или Fully Defined (Полностью определен);

− текст «Editing Sketch/Part/Assembly/Drawing» («Редактирование эскиза/детали/сборки/чертежа»).

Панели инструментов. В SolidWorks насчитывается 278 панелей инструментов (рисунок 4.2), не считая панели инструментов дополнительных модулей. Панели инструментов предназначены для ускорения работы в различных режимах. Панель инструментов Standard (Стандартная) в минимальной конфигурации появляется при первом запуске программы. В зависимости от вида выполняемой работы (создания детали, сборки или чертежа) отображаются различные панели инструментов. Панели инструментов могут располагаться как по периметру рабочей области (прикрепленные панели), так и в любом месте на рабочей области (плавающие панели). С помощью технологии **drag&drop** панели инструментов можно перемещать по рабочей области, расставляя их в соответствии со своими требованиями.

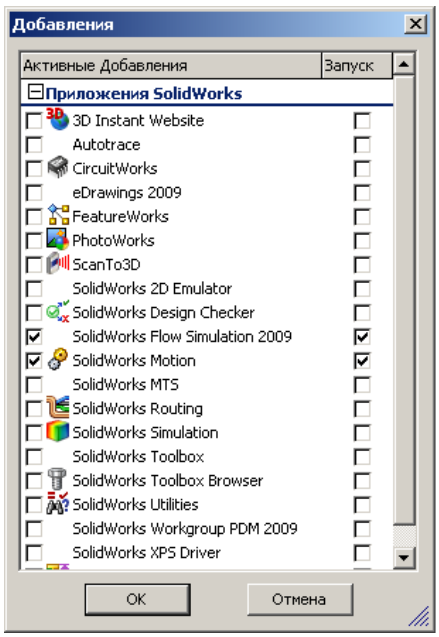

Рисунок 4.2 – Включение установленных приложений

Основные функции работы с системой расположены на стандартной панели инструментов **Стандартная**. При необходимости включения той или иной установленной ранее надстройки (Solid Works Motion, Cosmos FlowWorks) нужно выбрать на кнопке **Настройка** панели **Стандартная** пункт меню **Приложения** и в появившемся окне (см. рисунок 4.2) выбрать ранее установленное требуемое приложение.

3. Основные понятия и определения

В моделях SolidWorks используются следующие термины.

Origin (*Исходная точка*) − отображается в виде двух стрелок серого цвета и представляет (0,0,0) координату модели. Когда эскиз становится активным, исходная точка эскиза отображается красным цветом и представляет (0,0,0) координату эскиза. Размеры и взаимосвязи могут быть добавлены к исходной точке модели, но не эскиза.

Axis (*Ось*) − прямая линия, которая используется для создания геометрии модели, элементов или шаблонов. Ось можно создать различными способами, включая пересечение двух плоскостей.

Plane (*Плоскость*) − плоская вспомогательная геометрия. Можно использовать плоскости для добавления двухмерного эскиза, для разреза модели, а также в качестве нейтральной плоскости для уклона и т.д.

Face (*Грань*) −границы, которые позволяют определить форму модели или поверхности. Грань − это область модели или поверхности, которую можно выбрать. Например, прямоугольная твердотельная деталь имеет шесть граней.

Vertex (*Вершина*) − точка, в которой пересекаются две или несколько линий или кромок. Вершины можно выбрать для создания эскизов, нанесения размеров и множества других операций.

Edge (*Кромка*) − место, в котором две грани или поверхности соприкасаются на определенном расстоянии. Кромки можно выбрать для создания эскизов, нанесения размеров и множества других операций.

Примеры работы с системой SolidWorks 2009

1. Создание активного проекта детали

Выберите в главном меню пункт **Файл→Новый** − появится окно (рисунок 4.3), в котором пользователь осуществляет выбор типа проекта.

Выберите **Деталь** и нажмите снизу кнопку **ОК**.

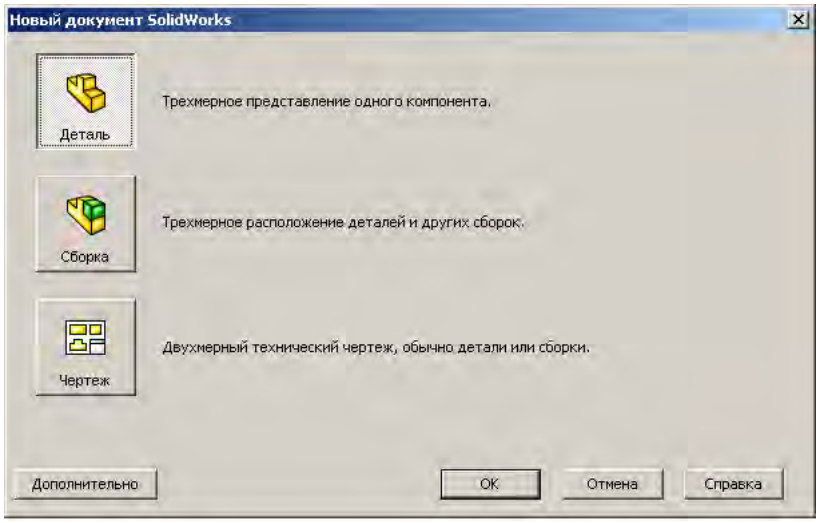

Рисунок 4.3 – Выбор типа проекта

2. Особенности автоматизированного проектирования простых деталей в Solid Works 2009

Данные действия приведут к загрузке в рабочее пространство чертежа дополнительных инструментов (**Command Manager, Центр управления проектом, Вид, Стандартные виды, Кнопки ориентации вида**) (рисунок 4.4), посредством которых будет осуществляться проектирование детали.

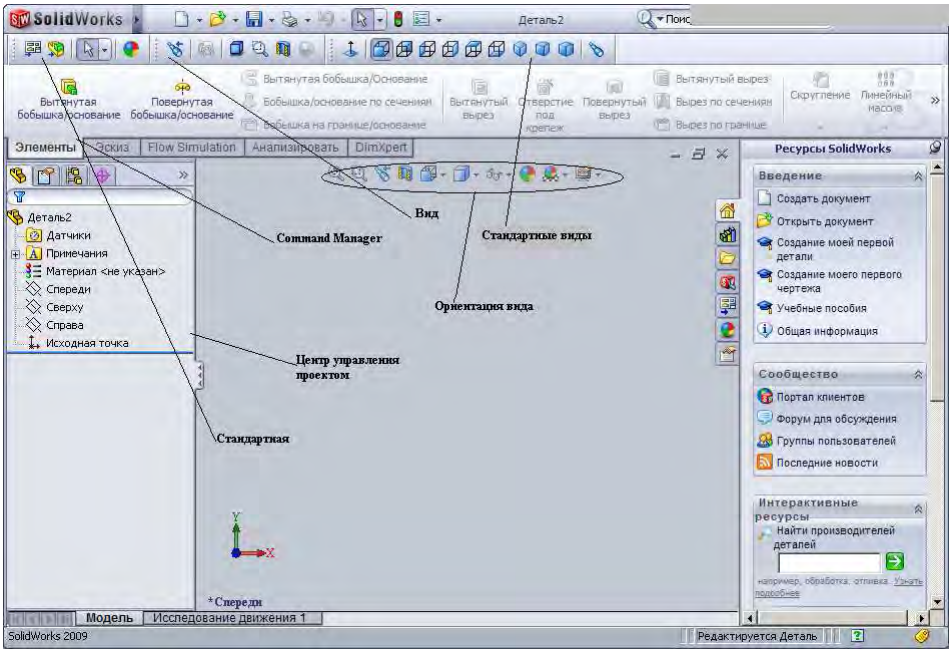

Рисунок 4.4 − Рабочее пространство проектирования простой детали

Основная функциональность (набор функций) инженерной идеологии проектирования SolidWorks 2009 представлена на панели закладок **Command Manager**.

Закладка *Элементы* подсвечивает основные доступные в данный момент элементы, например, *Вытянутая бобышка* или *Повернутая бобышка*, для текущего проектирования **трехмерного** объекта по составленному заранее двухмерному *Эскизу*.

Проектирование новой детали начинается с закладки *Эскиз*, где пользователь моделирует основание (фундамент) последующей трехмерной модели с помощью плоскостных элементов (линий, окружностей, дуг, прямоугольников и т.д.). При этом необходимо в качестве основного выбрать **Центр управления проектом → Дерево конструирования→ Вид** (рисунок 4.5). Следует отметить, что в *Дереве конструирования* отображаются все созданные ранее объекты − части проектируемой детали с учетом их наследственных связей друг с другом, в то время как изменение их свойств с возможным последующим их перестроением осуществляется в *Property Manager*, вызываемом щелчком правой мыши (**Редактировать Определение**), на выбранном объекте в *Дереве конструирования*.

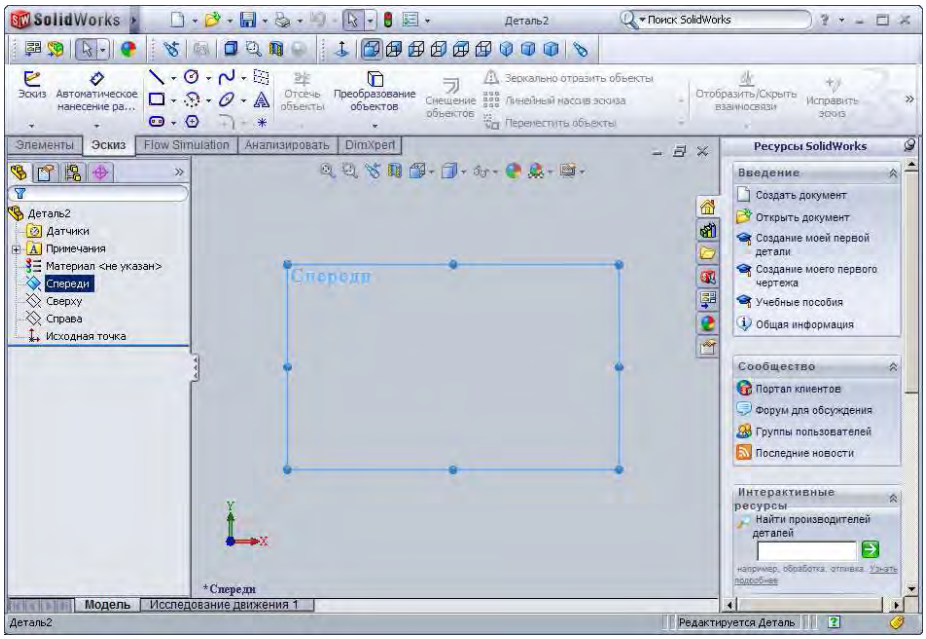

Рисунок 4.5 − Выбор основного вида для проектирования эскиза моделируемой трехмерной детали

Для перестроения любой составной части проектируемого объекта выберите его в *Дереве Конструирования*, вызовите *Property Manager*, после чего нажмите кнопку *Перестроить* (рисунок 4.6).

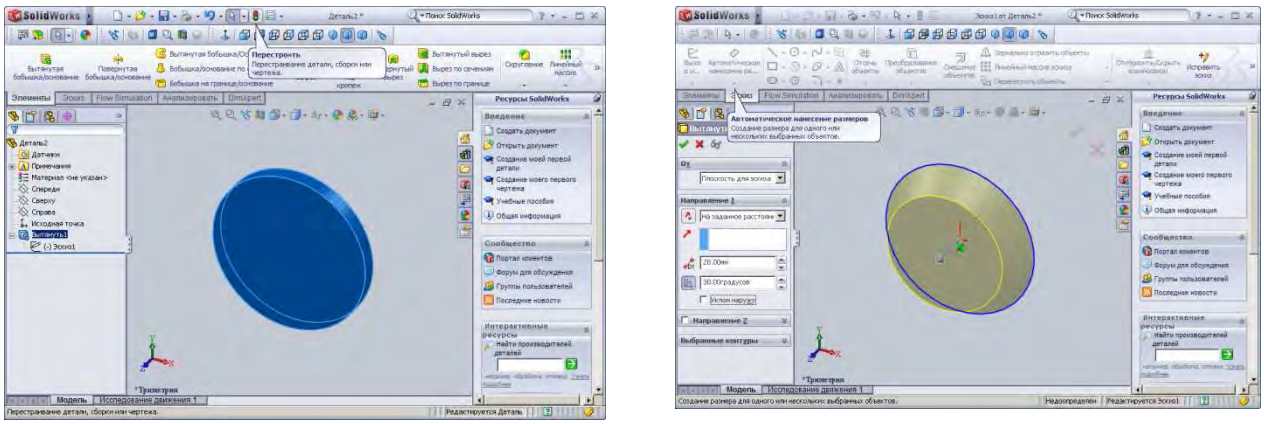

Рисунок 4.6 − Перестроение объекта модели простой детали

Изменение вида при необходимости осуществляется кнопками *Ориентации* либо кнопками панели *Стандартные виды* (рисунок 4.7).

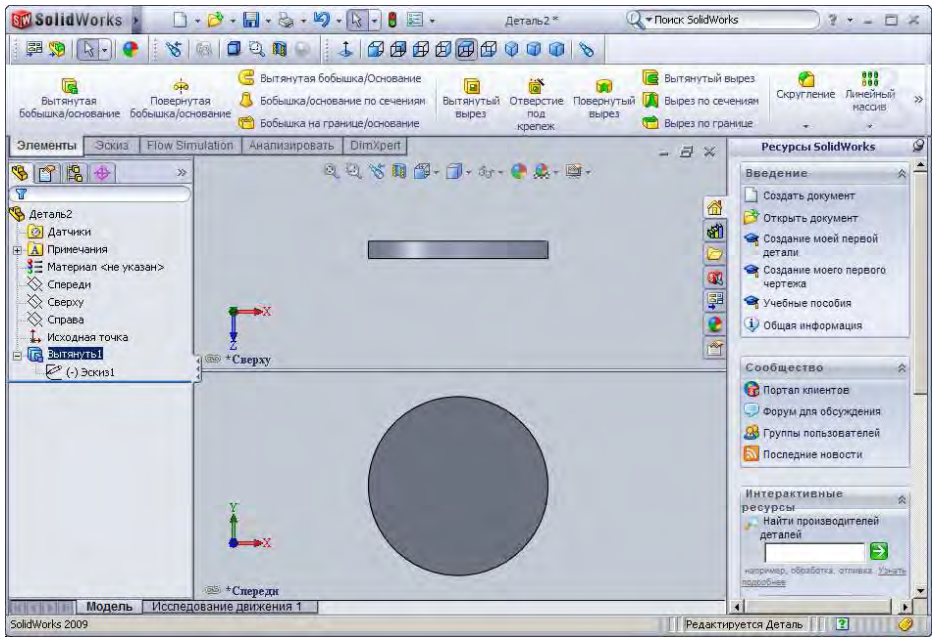

Рисунок 4.7 – Изменение вида кнопками ориентации модели в пространстве.

Задание к лабораторной работе

С помощью подсказок в ресурсах интерактивной справки (пункт *Создание моей первой детали*) создайте модель детали *Нажимная пластина ПИ-1* и затем чертеж модели данной детали (рисунок 4.8). При этом имейте в виду следующее:

− для создания центрального отверстия выбирают элемент *Отверстие под крепеж* в **Command Manager** и указывают его свойства (отверстие под чистовой метчик М14×1.5). Затем указывают **сначала произвольное** расположение на верхней грани, **а затем**, определив координаты центра

верхней грани проекции центральной осевой линии *Нажимной пластины ПИ-1* путем перестроения **трехмерного** эскиза, найдя проекцию (точку) центральной осевой линии *Нажимной пластины* и определив ее пространственные координаты, изменяют координаты фронтальной проекции проектируемого отверстия под крепеж;

− выбор подходящей глубины резьбы осуществляют кнопками последовательного увеличения/уменьшения в зависимости от толщины *Нажимной пластины* (рисунок 4.9);

− создавая фронтальные разрезы (см. рисунок 4.5), при указании «верхней» и «нижней» точек линии разреза необходимо следить за тем, чтобы курсор принимал вид  .

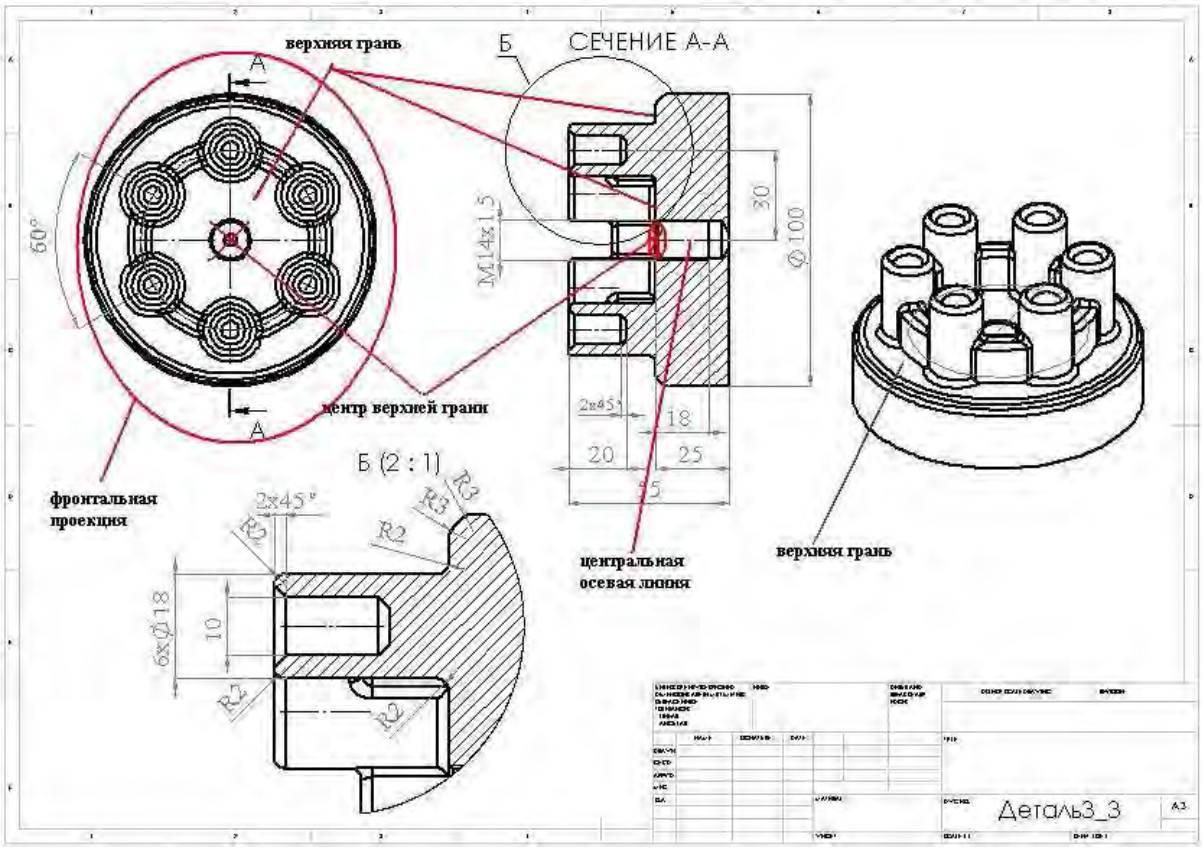

Рисунок 4.8 – Нажимная пластина ПИ-1

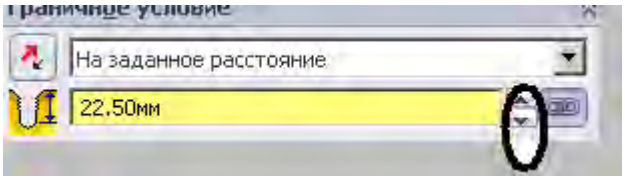

Рисунок 4.9 – Выбор подходящей глубины резьбового отверстия

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Основные положения.
4. Чертеж *нажимной пластины*.

ЛАБОРАТОРНАЯ РАБОТА № 5
Основные приемы моделирования сборок в системе SolidWorks 2009

Цель работы: освоить основные приемы трехмерного (3D) проектирования сборок на примере системы автоматизированного проектирования SolidWorks 2009.

Основные положения

Сборка − это узел, состоящий из двух или более деталей, называемых компонентами, в одном документе SolidWorks. Расположение и ориентация компонентов задается с помощью сопряжений, устанавливающих взаимосвязи между компонентами.

Ограничения − объект (элемент) проекта, обозначающий добавления ограничений в геометрию модели. **Взаимосвязь** и размеры ограничивают степени свободы и геометрию деталей, а сопряжения ограничивают компоненты в сборках путем добавления взаимосвязей.

Взаимосвязи отображаются на эскизе. Существует несколько видов взаимосвязей: *горизонтальность*, *вертикальность*, *зафиксированный*, *совпадение*, *совпадение*, *концентрический*.

Сопряжения − объекты проекта, вид и количество которых определяют геометрические условия существования сопряжения деталей в сборке.

Примеры работы со сборками

1. Выполнение тренировочного упражнения

В меню **Справка → Учебные пособия** выберите пункт **Упражнение 1 → Детали**, и последовательно осуществите его выполнение. Затем выберите пункт **Упражнение 2 → Сборки** и **Упражнение 3 → Чертежи**. Также осуществите последовательное их выполнение.

2. Проектирование корпус нажимной пластины

Разработайте конструкцию корпуса *1* (рисунок 5.1) с учетом последующей вставки в него спроектированной ранее **Нажимной пластины** *2*. Сохраните деталь под именем *Tutor1*. При этом документ НЕ ЗАКРЫВАЙТЕ**.**

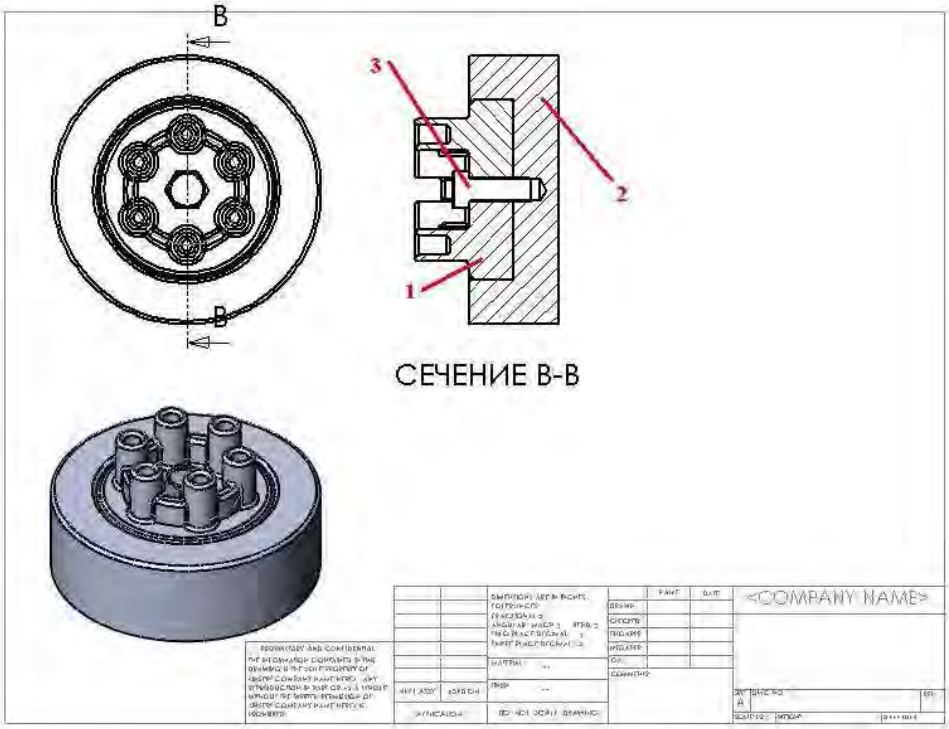

Рисунок 5.1 − Проектируемая сборка

3. Вставка корпуса нажимной пластины

В главном меню выберите пункт *Открыть*, после чего откроется и сделается активным проект **Нажимная пластина**. Сохраните деталь под именем *Tutor 2*, оставляя документ открытым.

4. Создание сборки

На закладке *Сборка* выберите *Вставить компонент*. Слева на закладке *Property Manager* в списке открытие документа отобразятся имеющиеся открытые детали *Tutor1* и *Tutor2* для сборки. Разместите их поочередно в документе *Сборка*.

Определите самостоятельно *Условия сопряжения* размещенных деталей.

Для того чтобы отобразить список имеющихся крепежных деталей в подключаемых библиотеках, при активном документе *Сборка* зайдите в главное меню **Инструменты → Добавления**. Выберите *SolidWorks Toolbox Browser* и *SolidWorks Toolbox*. В правой части чертежа выберите закладку *библиотеки проектирования* разверните список *Toolbox*.

Задание к лабораторной работе

Самостоятельно подберите необходимый крепежный инструмент и определите его параметры (диаметр стержня, длину, длину резьбы), исходя из вашего проекта.

Для того чтобы поместить крепежный инструмент в место отверстия на создаваемой сборке, необходимо, удерживая левую кнопку мыши, поднести выбранный крепежный элемент в зону пустого пространства отверстия. При этом произойдет его автоматическое перемещение, если у вас не будет ошибок в сопряжениях.

В случае необходимости исправлений в **Дереве конструирования** укажите необходимую деталь, нажмите на правую кнопку мыши и выберите пункт *Редактировать деталь*. При этом остальные детали на *Сборке* «скроются».

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Основные положения.
4. Чертеж **с размерами** разработанного корпуса *2* **Нажимной пластины**.
5. Сборочный чертеж всего проекта.

ЛИТЕРАТУРА

1. Полещук, Н.Н. AutoCAD: разработка приложений, настройка и адаптация / Н.Н. Полещук. − СПб.: БХВ-Петербург, 2006. – 992 с.

2. Полещук, Н.Н. AutoLISP и VisualLISP в среде AutoCAD / Н.Н. Полещук. − СПб.: БХВ-Петербург, 2006. − 960с.