МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»
ВОЛЖСКИЙ ФИЛИАЛ

Кафедра гуманитарных и естественнонаучных дисциплин

Методические указания к лабораторным работам
по дисциплине

3-D ПРОГРАММИРОВАНИЕ

Направление подготовки

09.03.01 Информатика и вычислительная техника

Направленность (профиль, специализация) образовательной программы

«Автоматизированные системы обработки информации и управления»

Квалификация

бакалавр

Чебоксары
2019

Составитель: Михайлова О.В.

3-D программирование. Методические указания к лабораторным работам для студентов 09.03.01 «Информатика и вычислительная техника». – Чебоксары: Волжский филиал Московского автомобильно-дорожного государственного технического университета (МАДИ), 2019. – 110 с.

Печатается по решению Учебно-методического совета Волжского филиала МАДИ

Оглавление

Введение

В современных условиях требования рынка труда к выпускникам вузов значительно выросли, что потребовало создания последовательной, рассчитанной на весь период обучения, научно-обоснованной системы подготовки кадров, важное место, в которой отводится лабораторной форме обучения.

1.Цели и задачи учебной дисциплины

Целью освоения дисциплины является формирование у обучающихся компетенций в соответствии с требованиями ФГОС ВО и образовательной программы.

Задачами освоения дисциплины являются:

- приобретение обучающимися знаний, умений, навыков и (или) опыта профессиональной деятельности, характеризующих этапы формирования компетенций в соответствии с учебным планом и календарным графиком учебного процесса;
- оценка достижения обучающимися планируемых результатов обучения как этапа формирования соответствующих компетенций.

2. Компетенции студента, формируемые в результате освоения учебной дисциплины

В результате освоения дисциплины «3-D программирование» у обучающихся формируются следующие компетенции и должны быть достигнуты следующие результаты обучения как этап формирования соответствующих компетенций:

Код компетенции	Результаты освоения образовательной программы	Перечень планируемых результатов обучения по дисциплине
ОПК-1	способностью инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем	**знать:** базовые понятия; **уметь:** применять информационные технологии для создания трехмерных моделей.
ПК-2	способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования	**владеть:** навыками 3D программирования и моделирования трехмерных объектов.

3. Тематический план лабораторных работ

№ п./п.	№ раздела дисциплины	Темы лабораторных занятий	Трудоемкость, ч.	Формы контроля
1	2	Трехмерная каркасная модель детали средствами базовой графики	2	Защита л/работы
2	2	Создание простой трехмерной модели детали в «Компас-3D» и ассоциативных видов	2	Защита л/работы
3	2	Чтение сборочного чертежа. Создание трехмерных моделей и ассоциативных чертежей в «Компас-3D»	2	Защита л/работы
4	2	Создание сборки из трехмерных моделей, спецификации и ассоциативного сборочного чертежа в «Компас-3D»	2	Защита л/работы
5	3	Создание анимации на языке C#	2	Защита л/работы
6	3	Создание трехмерной модели на управляемой трехмерной сцене на языке C#	2	Защита л/работы
7	4	Создание трехмерной модели детали средствами библиотеки OpenGL	2	Защита л/работы
8	5	Создание трехмерной модели на языке VRML	2	Защита л/работы
9	6	Создание трехмерной модели с применением DirectX	2	Защита л/работы

Лабораторная работа № 1. ***Трехмерная каркасная модель детали средствами базовой графики***

Работа предназначена для знакомства студента с методами трехмерного геометрического моделирования: построения каркасной модели, построения поверхностной модели и знакомства с основами твердотельного моделирования. Данная лабораторная работа нужна студенту для приобретения навыков работы в среде программирования C++ Builder

Цели работы:

Познакомиться с конкретным примером выполнения программы (листинги 1 и 2) и переписать программу в среде программирования C++ Builder.

Выполнить задание по указанию преподавателя, оформить файл отчета, сдать исходные и исполняемый файл.

Задание

По заданию своего номера варианта создать трехмерную каркасную модель детали средствами графики C++ Builder. Написать текст программы. Предусмотреть параметризацию размеров детали. Привести рисунок копии экрана для своего варианта.

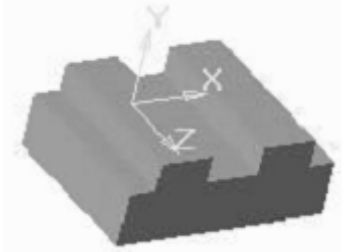
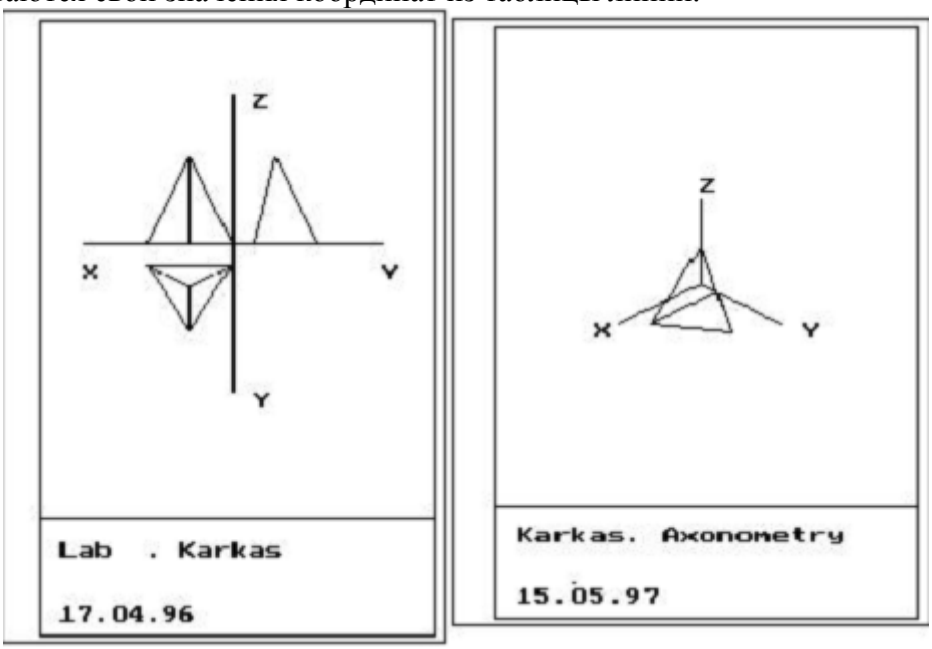


Рисунок задания

Пояснение

Прежде чем начать выполнять задание обратимся к принципам проецирования на примере трехгранной пирамидки и разберем решение данной задачи для нее. Кроме того, чтобы студент не только подставил свои числа в готовую программу, а немного подумал о Windows программировании, приведем текст решения на языке C++ с использованием DOS функций (line, moveto, lineto).

Для ускорения работы над заданием желательно нарисовать эскизы ортогональной и аксонометрической проекций детали, а затем составить таблицы для координат вершин (таблица 1) и линий (таблица 2). Далее в тексте программы по указанному образцу вписываются свои значения координат из таблицы линий.



Ортогональные и аксонометрическая проекции каркасной модели.

Таблица 1 Значения координат вершин пирамиды

Номер вершины	X	Y	Z
1	40	10	0
2	0	10	0
3	20	40	0
4	20	20	40

Таблица 2 Значения координат начала и конца линии ребер каркасной модели

Номер линии (между вершинами)	Значения координат начала линии			Значения координат конца линии		
	Xn	Yn	Zn	Xk	Yk	Zk
0 (1-2)	40	10	0	0	10	0
1 (2-3)	0	10	0	20	40	0
2 (3-1)	20	40	0	40	10	0
3 (1-4)	40	10	0	20	20	40
4 (2-4)	0	10	0	20	20	40
5 (3-4)	20	40	0	20	20	40

Листинг 1 Текст программы вычерчивания ортогональных проекции.

```
// Выполнил студент 321 группы Вариант 01 Петров С.Ю., 12.03.2004
// файл g32101S1.cpp
/*Задание: Написать программу на языке Си  для вычерчивания
ортогональных проекций пирамиды.*/
#include <stdio.h>
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
main(void){int i,Driver,Mode,errorcode;  int x0, y0;
int  xn[20]={40, 0,20,40, 0,20},
 yn[20]={10,10,40,10,10,40},
 zn[20]={ 0, 0, 0, 0, 0, 0},
 xk[20]={ 0,20,40,20,20,20},
 yk[20]={10,40,10,20,20,20},
 zk[20]={ 0, 0, 0,40,40,40};
/* Установка параметров */
x0=110.;y0=110.;
/* Инициализация графики  */
Driver=DETECT; initgraph(&Driver,&Mode," ");
errorcode=graphresult( );  if(errorcode !=grOk)
{printf(" Ошибка графики: %s  Код =%d", grapherrormsg(errorcode),
errorcode);  exit(1);}
/* рамка  */ setbkcolor(0); rectangle(2,1,210,297);
rectangle(20,5,205,294); line(20,239,205,239);
outtextxy(30,250,"Tast S1. Karkas ");  outtextxy(30,260,"Petrov");
outtextxy(30,270,"gr.321 number 01");outtextxy(30,280,"17.04.96");
/* Оси координат и их обозначение */
line(x0-70.,y0,x0+70,y0 ); line(x0,y0-70.,x0,y0+70);
outtextxy(x0-70.,y0+10.,"X"); outtextxy(x0+70.,y0+10.,"Y");
outtextxy(x0+10.,y0-70.,"Z"); outtextxy(x0+10.,y0+70.,"Y");
/* Основные циклы вычислений проекции XOY*/
for(i=0;i<11;i++){  line(x0-xn[i],y0+yn[i],x0-xk[i],y0+yk[i] );};
/* Основные циклы вычислений проекции XOZ*/
```

```
for(i=1;i<11;i++){  line(x0-xn[i],y0-zn[i],x0-xk[i],y0-zk[i] );};
/* Основные циклы вычислений проекции YOZ*/
for(i=1;i<11;i++){  line(x0+yn[i],y0-zn[i],x0+yk[i],y0-zk[i] );};
/* закрытие графического режима*/
while(!kbhit( )); closegraph( );return(0);}
```

Листинг 2 Текст программы вычерчивания прямоугольной изометрической проекции.

```
// Выполнил студент 321 группы Вариант 01 Петров С.Ю., 12.03.2004
// файл g32101S1.cpp
/*Задание: По ортогональным проекциям каркасной модели написать
программу на языке Си  для вычерчивания аксонометрической проекции
пирамиды.*/
#include <stdio.h>
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
main(void){int i,Driver,Mode,errorcode; int sx0, sy0;
float  kx,ky,kz,ax,ay,az,sxn,syn,sxk,syk; double  sin30,cos30;
/* 1. Ввод значений исходных данных */
int n=6;
int xn[20]={40, 0,20,40, 0,20},
 yn[20]={10,10,40,10,10,40},
 zn[20]={ 0, 0, 0, 0, 0, 0},
 xk[20]={ 0,20,40,20,20,20},
 yk[20]={10,40,10,20,20,20},
 zk[20]={ 0, 0, 0,40,40,40};
sx0=110;sy0=130;sin30=sin(30./180.*3.14);cos30=cos(30./180.*3.14);
kx=0.82;ky=0.82;kz=0.82;
/*    2. Инициализация графики  */
Driver=DETECT; initgraph(&Driver,&Mode," ");
errorcode=graphresult(); if(errorcode !=grOk)
{printf("Ошибка графики:%s код=%d",
 grapherrormsg(errorcode), errorcode); exit(1);}
/*    вычерчивание рамки формата А4*/
rectangle(2,1,210,297); rectangle(20,5,205,294);
line(20,239,205,239);
outtextxy(30,250,"Tast s1. Karkas "); outtextxy(30,260,"Petrov");
outtextxy(30,270,"gr. 321 num 01"); outtextxy(30,280,"15.05.97");
/* 3. Организация основных циклов вычислений линии ребер каркасной
модели */
for(i=0;i<=n ;i++){
ax=xn[i];ay=yn[i];az=zn[i];
sxn=sx0-ax*kx*cos30+ ay*ky*cos30;syn=sy0- (az*kz-ax*kx*sin30-
ay*ky*sin30);
ax=xk[i];ay=yk[i];az=zk[i];
sxk=sx0-ax*kx*cos30+ ay*ky*cos30;syk=sy0- (az*kz-ax*kx*sin30-
ay*ky*sin30);
line(sxn,syn,sxk,syk); };/*next i*/
/* оси координат и их обозначение */
ax=50;ay=0;az=0;
sxk=sx0-ax*kx*cos30+ ay*ky*cos30;syk=sy0- (az*kz-ax*kx*sin30-
ay*ky*sin30);
line(sx0,sy0,sxk,syk);outtextxy(sxk-10 ,syk,"X");
ax=0;ay=50;az=0;
```

```
sxk=sx0-ax*kx*cos30+ ay*ky*cos30;syk=sy0- (az*kz-ax*kx*sin30-
ay*ky*sin30);
line(sx0,sy0,sxk,syk);outtextxy(sxk+10,syk,"Y");
ax=0;ay=0;az=50;
sxk=sx0-ax*kx*cos30+ ay*ky*cos30;syk=sy0- (az*kz-ax*kx*sin30-
ay*ky*sin30);
line(sx0,sy0,sxk,syk);outtextxy(sxk,syk-10,"Z");
/*   4. Закрытие графического режима*/
 while(!kbhit()); closegraph();
/* окончание программы */
 return(0);}
```

Решение

При программировании под Windows функция MoveTo изменяет текущую позицию пера на заданную с координатами X и Y, а функция LineTo рисует отрезок прямой линии. Таким образом, можно заменить функцию line(X1,Y1,X2,Y2) в листингах 1 и 2 на две функции: Canvas->MoveTo(X1,Y1) и Canvas->LineTo(X2,Y2). Другим вариантом решения является создание своей функции. Для удобства работы сразу сделаем ее производной от класса TForm1, тогда функция вычерчивания линии примет вид:

```
void  TForm1::line (int X1,int Y1,int X2,int Y2)
{Canvas->MoveTo(X1,Y1); Canvas->LineTo (X2,Y2);}
```

Текст программы

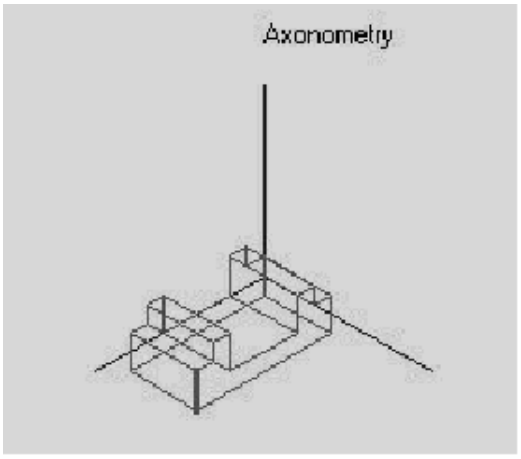
Приведем значения координат для подстановки их в текст программы. Для указанной в задании детали массивы значений координат для начала и конца линий приведены в таблице 3 (№ xn yn zn xk yk zk).

Таблица 3 Значения координат начала и конца линии ребер каркасной модели

N Xn Yn Zn Xk Yk Zk	13 20 50 20 20 10 20	25 10 50 20 10 50 00
1 90 10 00 10 10 00	14 10 50 20 10 10 20	26 90 10 00 90 10 20
2 10 10 00 10 50 00	15 90 50 00 90 50 20	27 90 10 20 80 10 20
3 10 50 00 90 50 00	16 90 50 20 80 50 20	28 80 10 20 80 10 30
4 90 50 00 90 10 00	17 80 50 20 80 50 30	29 80 10 30 70 10 30
5 90 50 20 90 10 20	18 80 50 30 70 50 30	30 70 10 30 70 10 10
6 80 50 20 80 10 20	19 70 50 30 70 50 10	31 70 10 10 30 10 10
7 80 50 30 80 10 30	20 70 50 10 30 50 10	32 30 10 10 30 10 30
8 70 50 30 70 10 30	21 30 50 10 30 50 30	33 30 10 30 20 10 30
9 70 50 10 70 10 10	22 30 50 30 20 50 30	34 20 10 30 20 10 20
10 30 50 10 30 10 10	23 20 50 30 20 50 20	35 20 10 20 10 10 20
11 30 50 30 30 10 30	24 20 50 20 10 50 20	36 10 10 20 10 10 00
12 20 50 30 20 10 30		

Копия экрана

После подстановки этих массивов координат в программу получаем решение. Для придания своей программе более красивого внешнего вида (см. рисунок 5.4) можно использовать свои знания среды программирования C++ Builder. Суть программы при этом не меняется.



Экранная копия работы программы

Оформление результатов работы

Результаты работы оформляются в виде отчета по выдаваемому в электронном виде образцу.

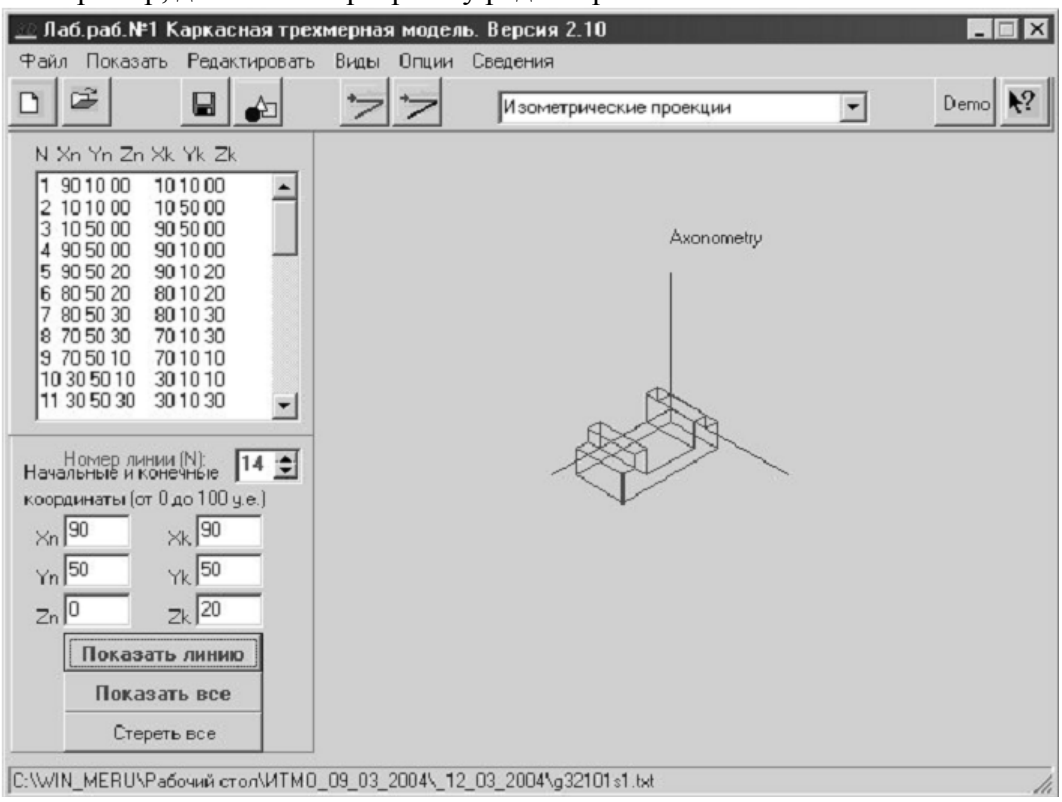
В отчет входят следующие разделы: задание, решение, текст программы, копия экрана, литература и интернет- ссылки. К отчету прилагается исходные и исполняемые файлы программы. Нумерация файлов и папки осуществляется по личному коду выдаваемому преподавателем. Все материалы сдаются в электронном виде. Допускается распечатка отчетных материалов.

Например, имя папки с файлами: g151103S1, где g- студенческая группа, 1511- номер группы, 03- номер студента по списку, S- самостоятельная работа, 1- порядковый номер работы. В папке находятся следующие файлы: g151103L1.doc- файл отчета по работе (образец выдается преподавателем), исходные файлы (Unit).

- Ug151103S1.cpp, Ug151103L2.h, Ug151103S1.dfm, файл проекта (Project)- g151103S1.bpr, g151103S1.cpp, g151103S1.res, исполняемый файл g151103S1.exe.

Дополнительное задание

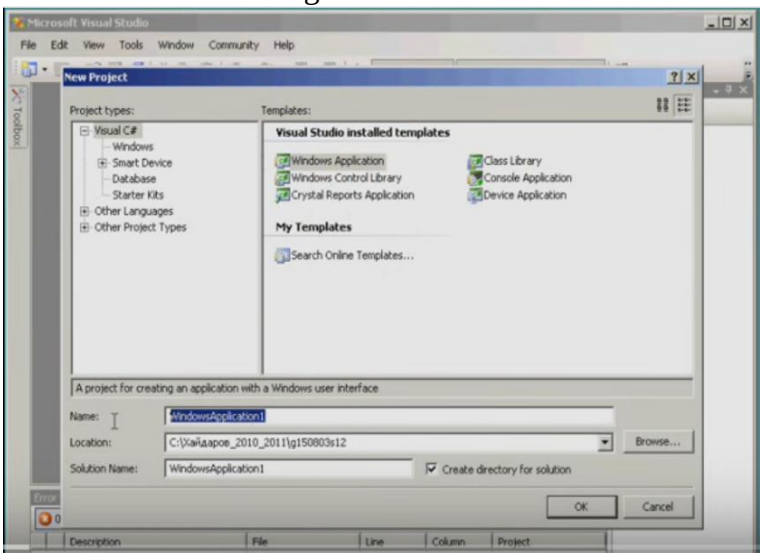
Кроме общетипового задания по согласованию с преподавателем выдается самостоятельное задание, в котором студент может показать свои знания и умения работы. Например, дополнить программу редактором линий.



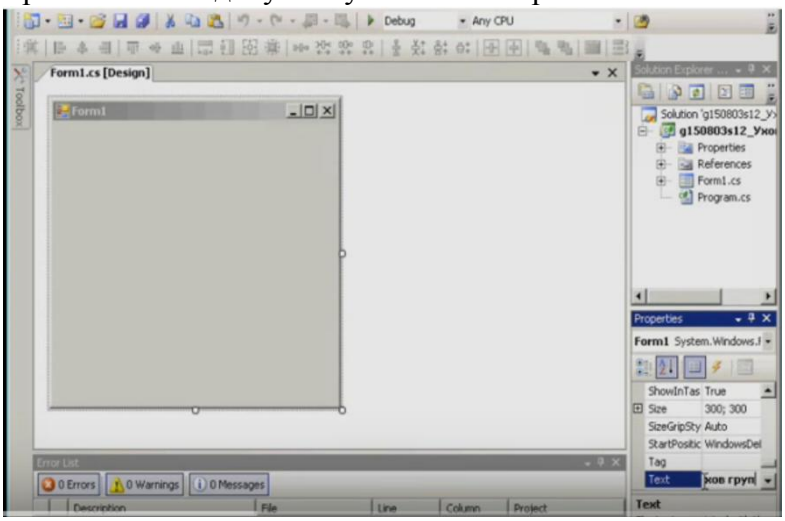
Экранная копия работы программы

Ниже приведен пример выполняя работы.

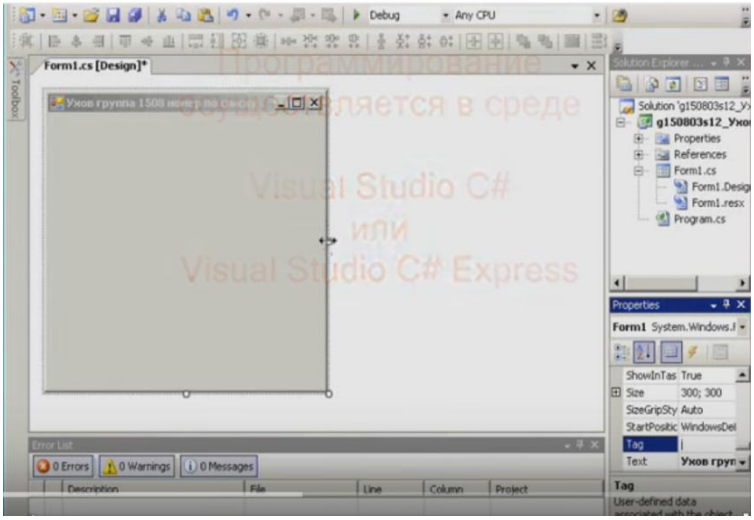
Откройте программу Microsoft Visual Studio C# (либо Visual Studio C# Express). Выберите «File»-«New»-«Progect»



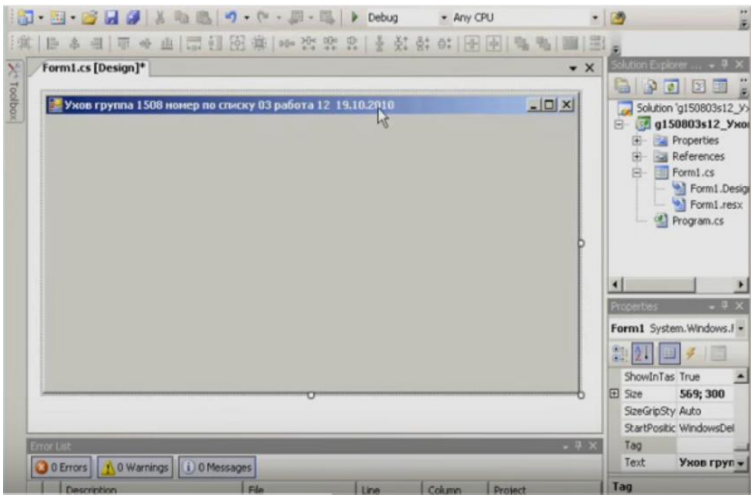
Присвойте имя документу в нижней строке «Name»



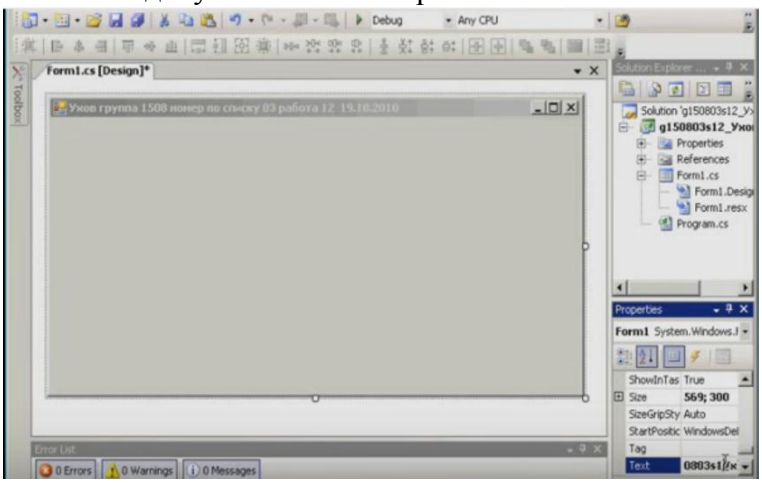
Откройте документ Form1. В правом нижнем окне «Properties» в строке «Text» напечатайте свою фамилию, группу, лабораторная работа № 1, дату выполнения.



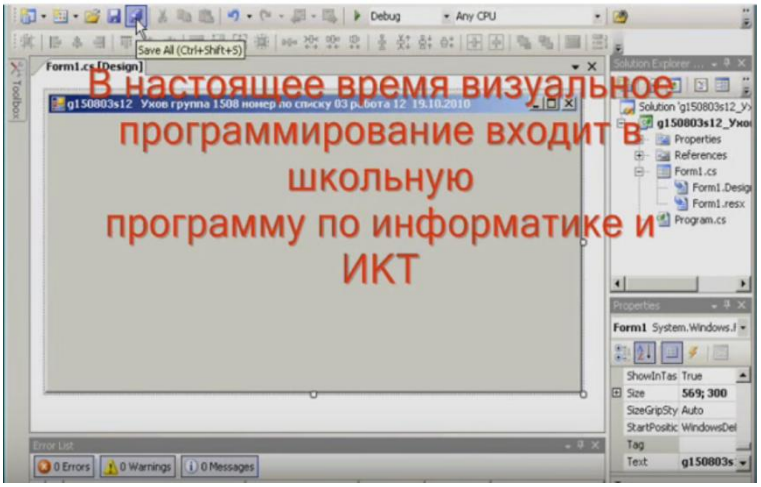
Перейдите в строку «Tag». После этого документу «Form1» присвоится «Фамилия, группа, лабораторная работа № 1, дата выполнения».



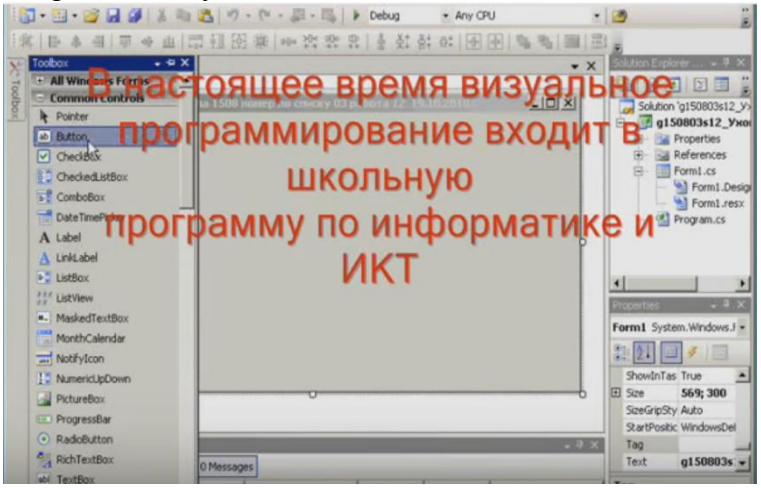
Растяните документ во весь экран.



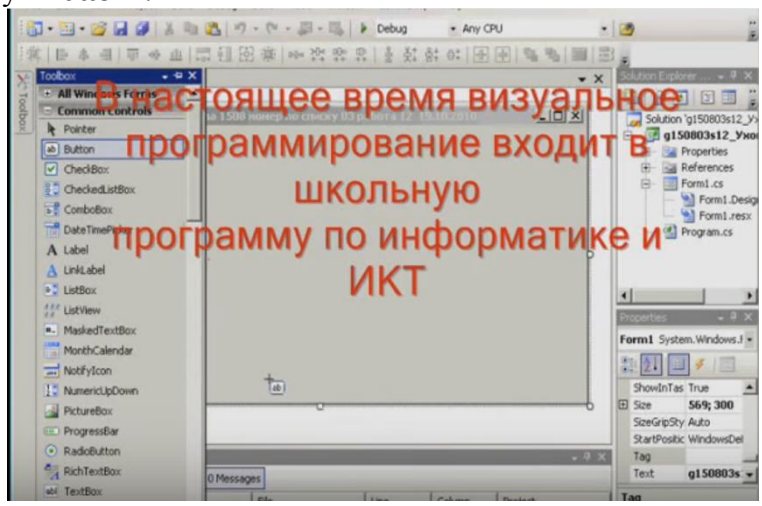
Переместитесь в строку «Text» и в начале текста наберите имя присвоенного вами ранее документа, которое вы вводили в строке «Name»



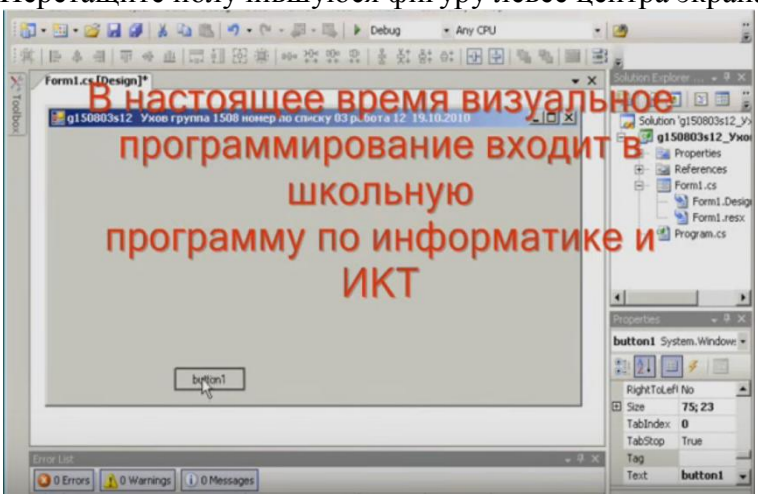
Сохраните документ.



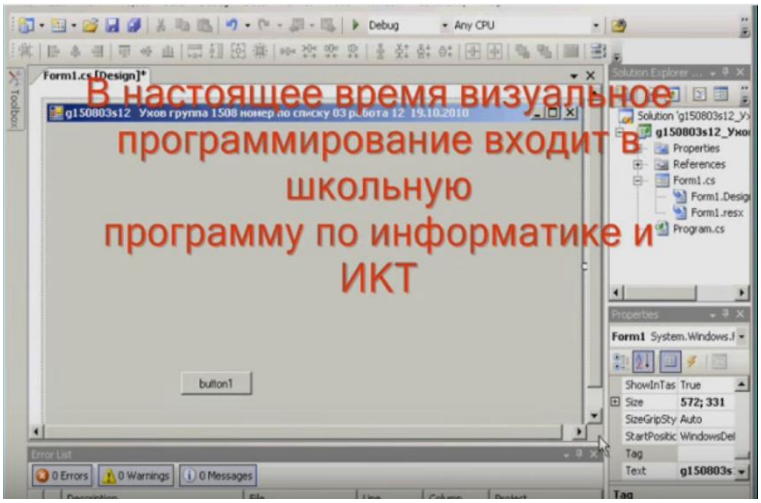
Нажмите кнопку «Toolbox», находящуюся на левой боковой панели. Выберите вкладку «Button».



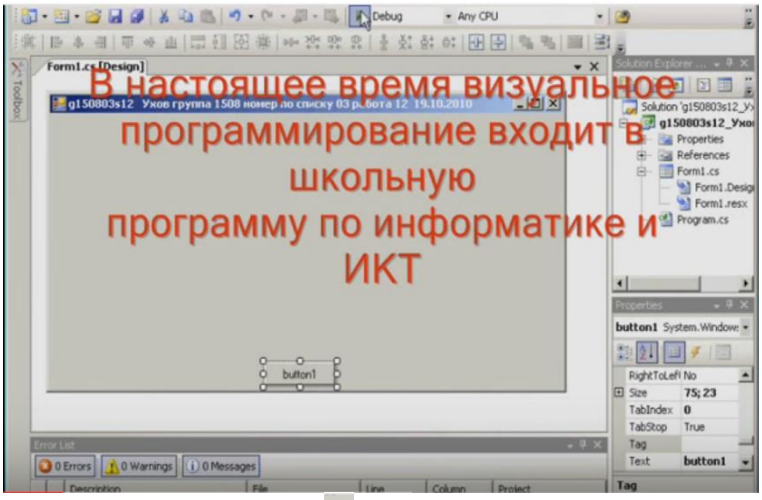
Перетащите получившуюся фигуру левее центра экрана (можно по центру).



Возьмите за правый нижний угол рабочего окна, потяните вниз. У вас получится следующее.

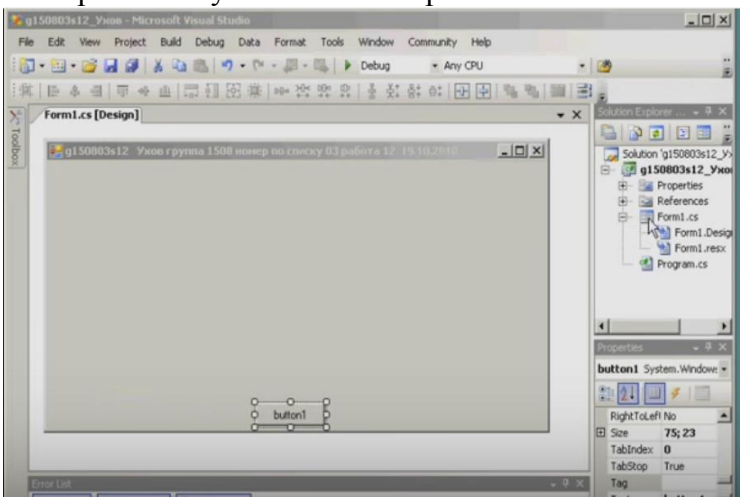


Не забывайте периодически сохранять свою работу.

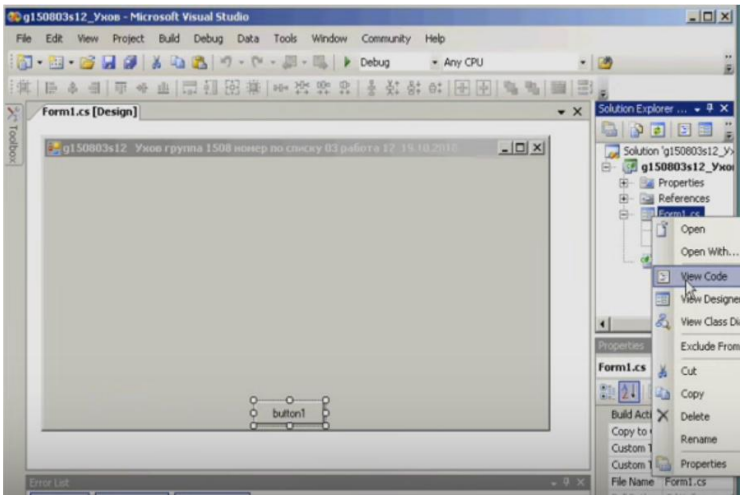


Далее нажмите кнопку  находящуюся в верхней рабочей панели.

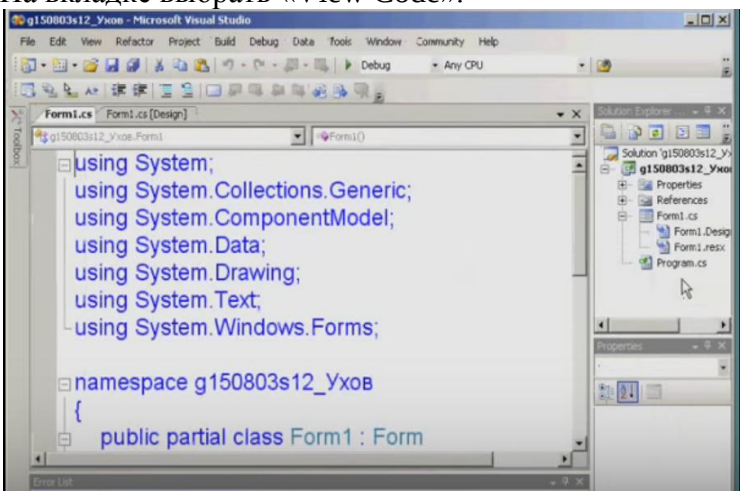
Ваше рабочее окно продублируется. Перетащите новое окошко поверх старого. Нажмите 10 раз кнопку «button1». Закройте новое окно.



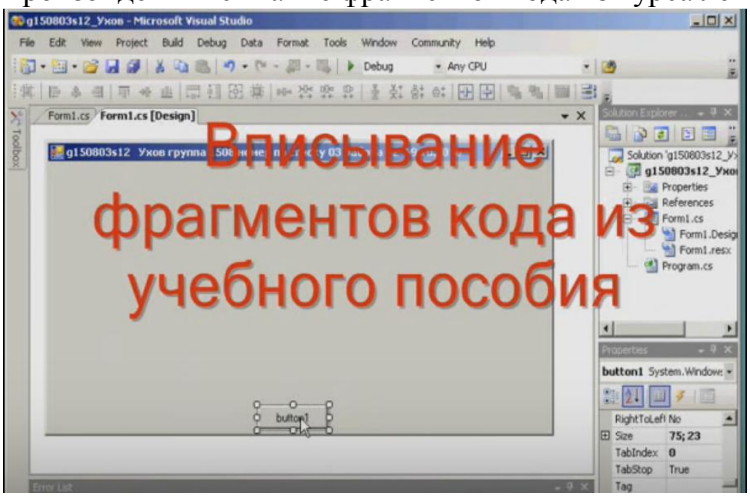
Переместите курсор мыши в правое верхнее окно на документ «Form1.cs»



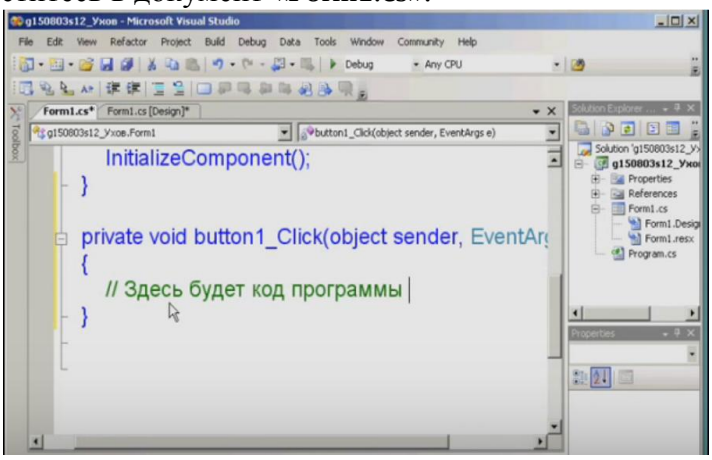
На вкладке выбрать «View Code».



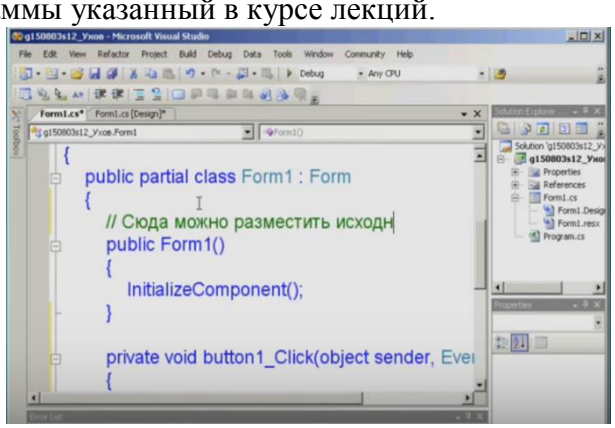
Произойдет вписывание фрагментов кода из курса лекций.



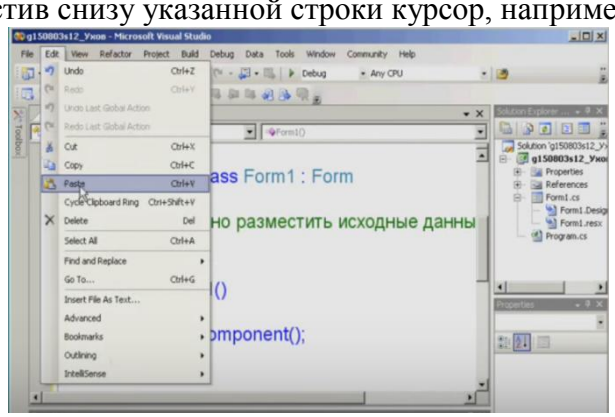
Перейдите в документ «Form1.cs (Design)». Нажмите кнопку «button1» и вы переместитесь в документ «Form1.cs».



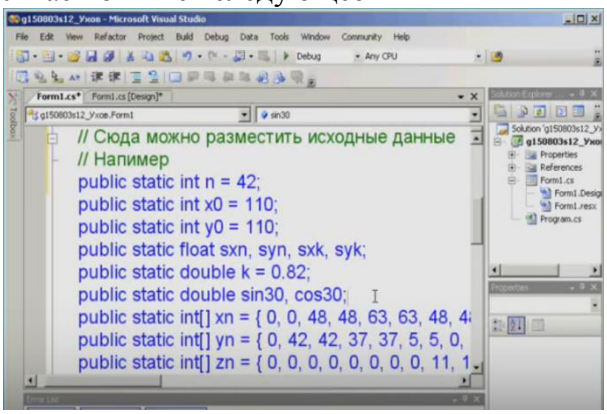
Переместитесь в конец документа. На рисунке показано, где будет располагаться код программы указанный в курсе лекций.



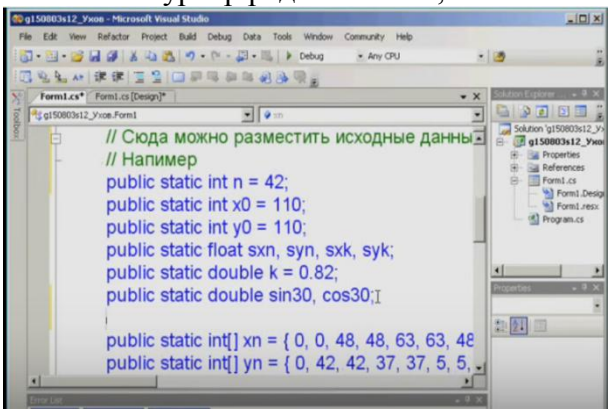
На рисунке показано где можно разместить исходные данные по вариантам. Разместив снизу указанной строки курсор, например открыть «Edit» - «Paste».



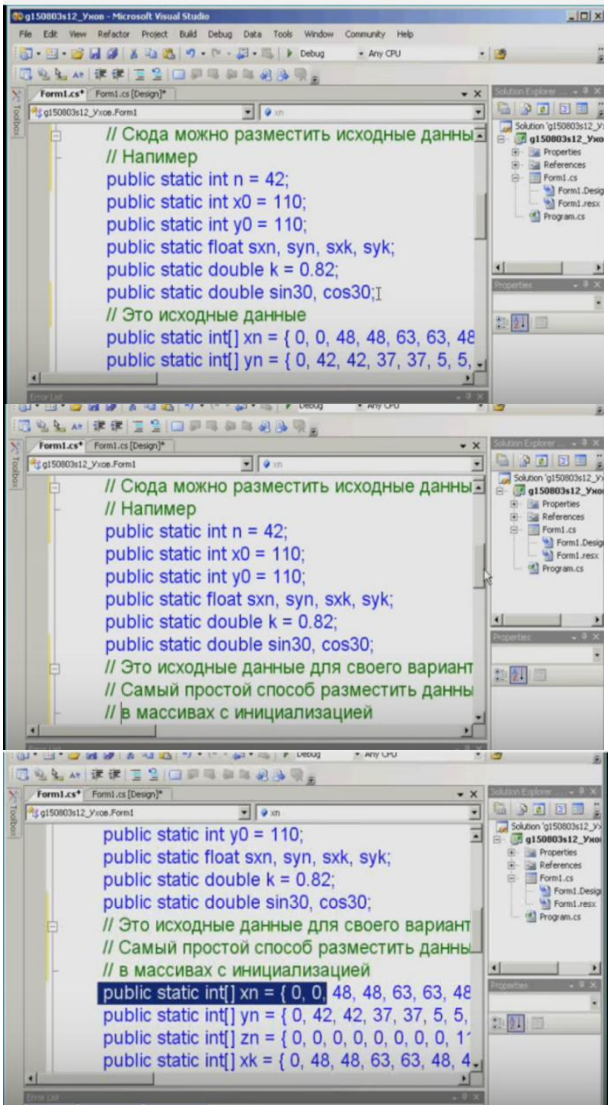
У вас появится следующее



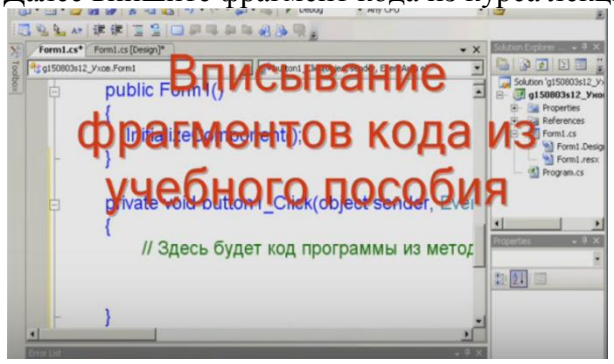
Поставьте курсор рядом с cos30, нажмите кнопку «Enter» для создания новой строки.

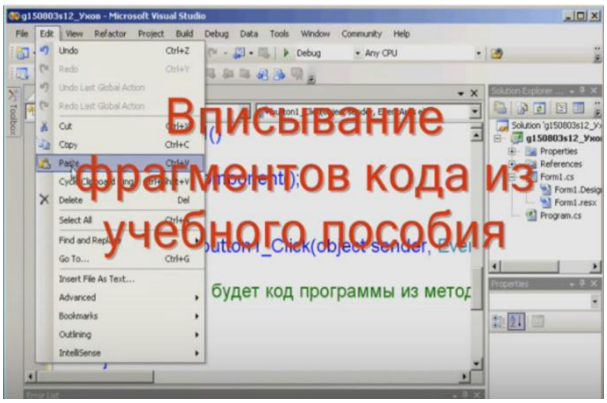


В этой строке вписываются исходные данные для своего варианта.

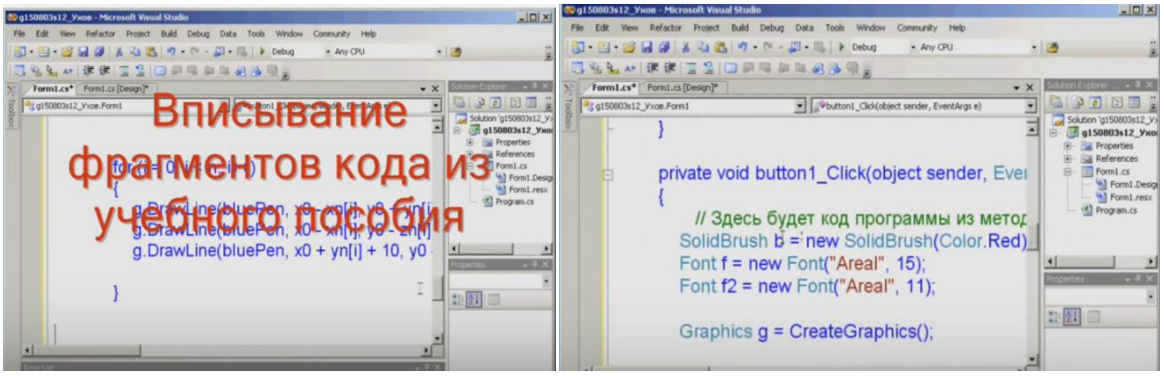


Далее впишите фрагмент кода из курса лекций

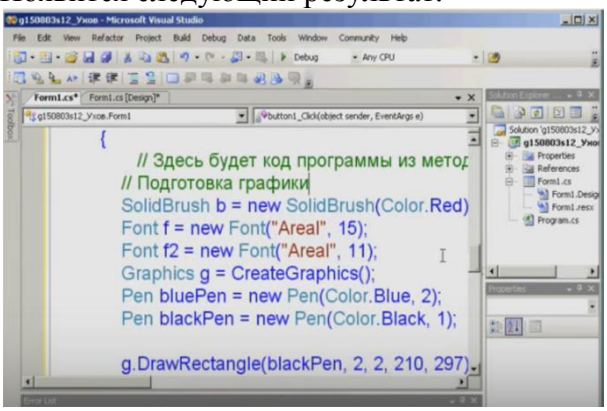




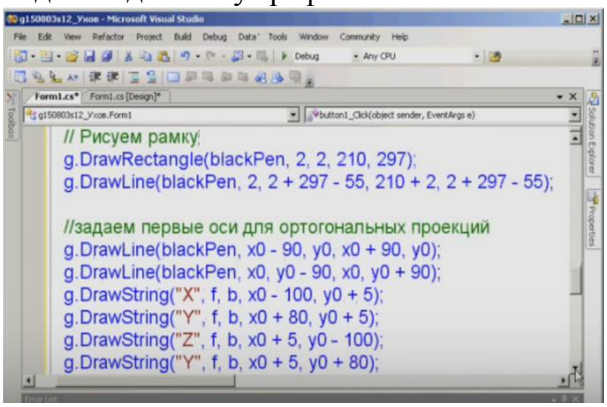
Переместитесь в верхнюю панель нажмите «Edit» - «Paste».



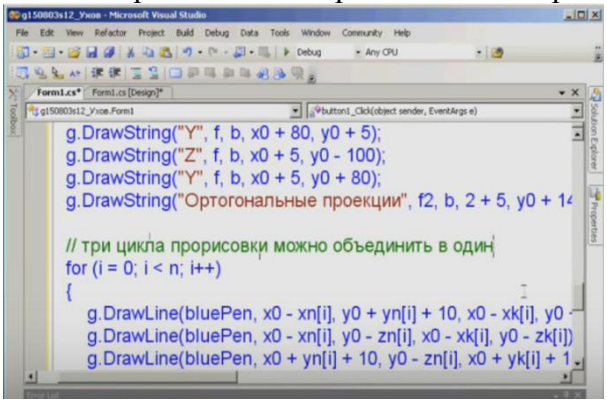
Появится следующий результат.



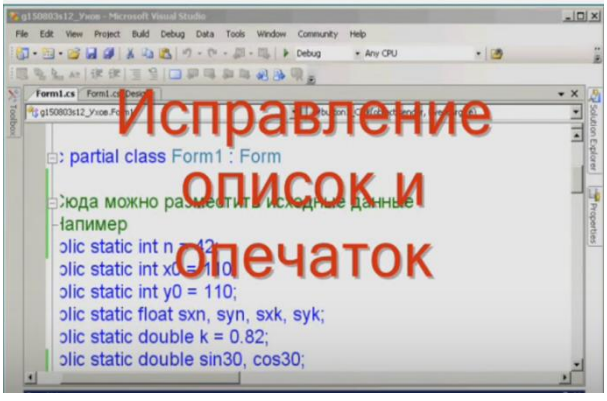
Ведем подготовку графики



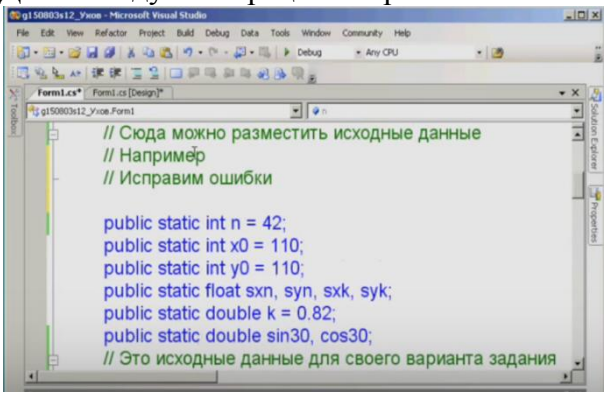
Задаем первые оси для ортогональных проекций.



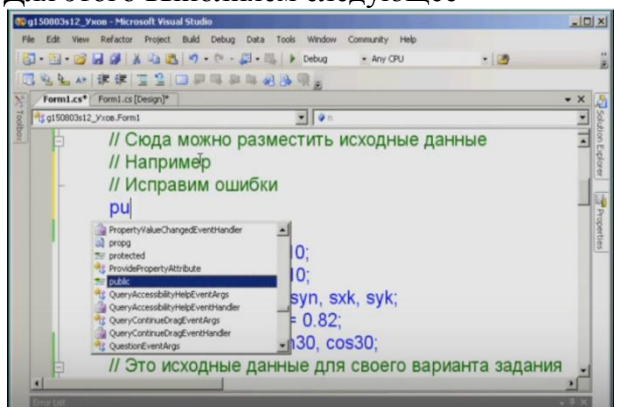
Три цикла прорисовки можно объединить в один.



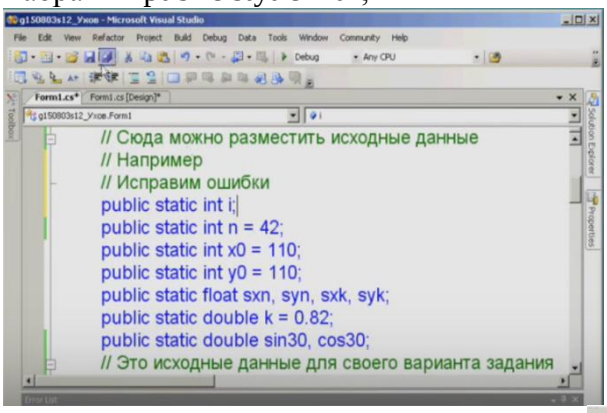
Далее следует операция исправления описок и опечаток.



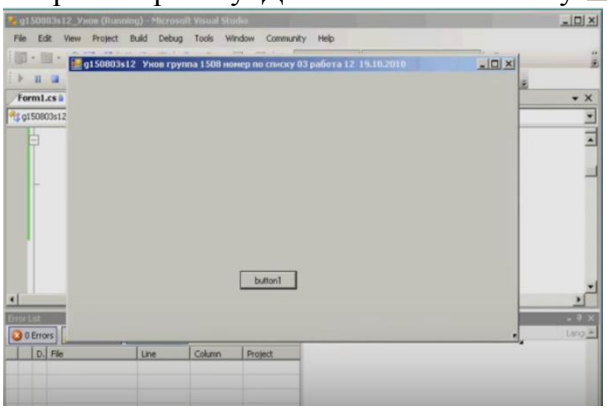
Для этого выполняем следующее



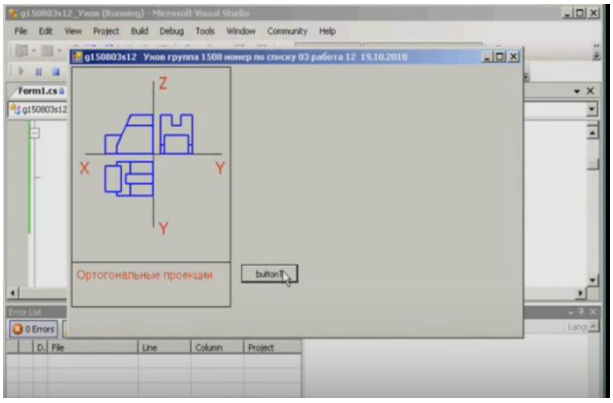
Набрать «public stytic int i;»



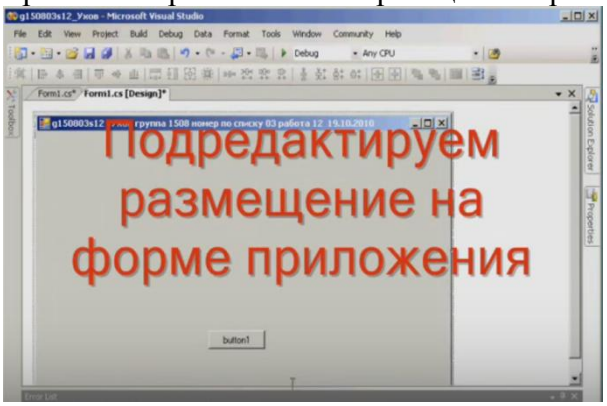
Сохраните работу. Далее нажмите кнопку  находящуюся в верхней рабочей панели.



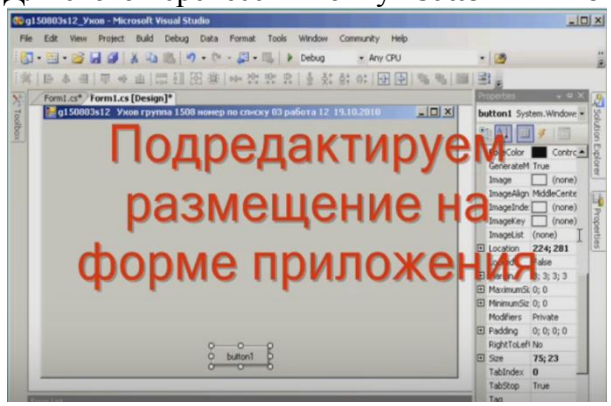
Откроется второе рабочее окно. Нажмите в нем кнопку «button1»



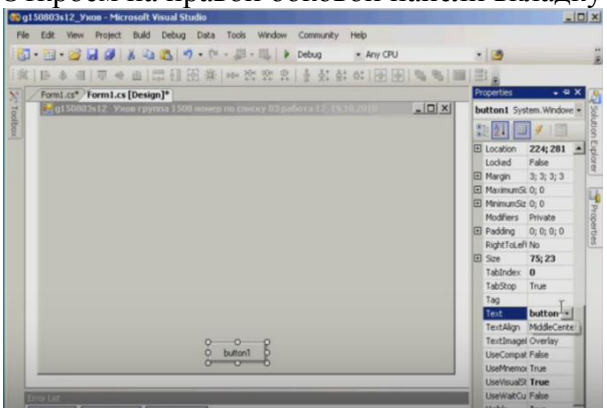
Проявится ортогональная проекция. Закройте это окошко.



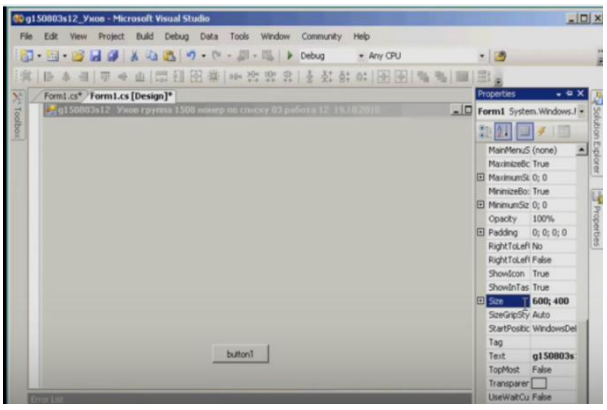
Для этого перенесем кнопку «button1» вниз окна формы приложения.



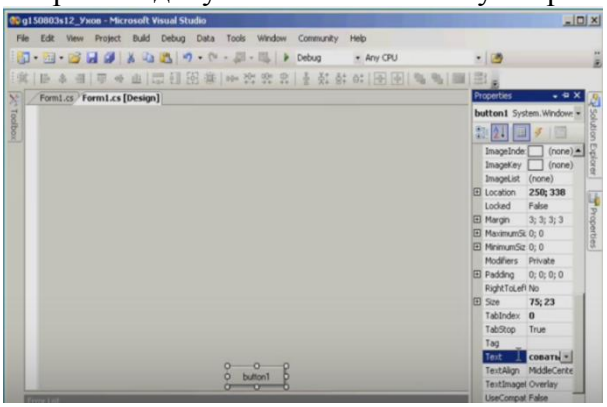
Откроем на правой боковой панели вкладку «Properties».



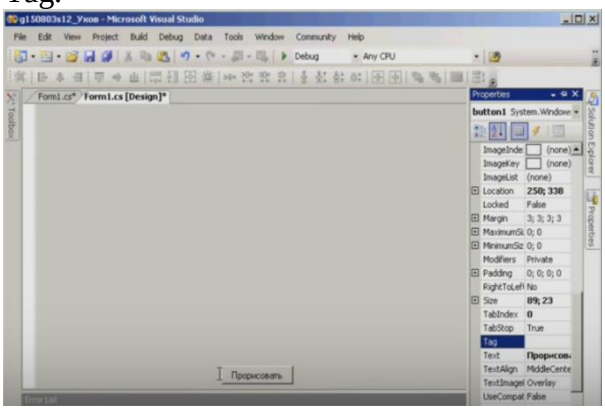
Перейдем во вкладку «Size», изменим цифры на 600; 400.



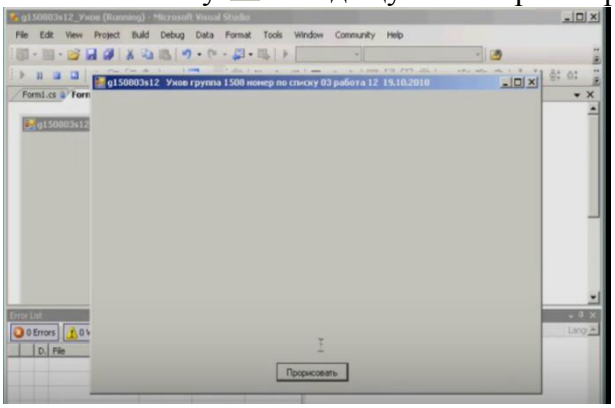
Сохраните документ. Нажав кнопку сохранить и  .



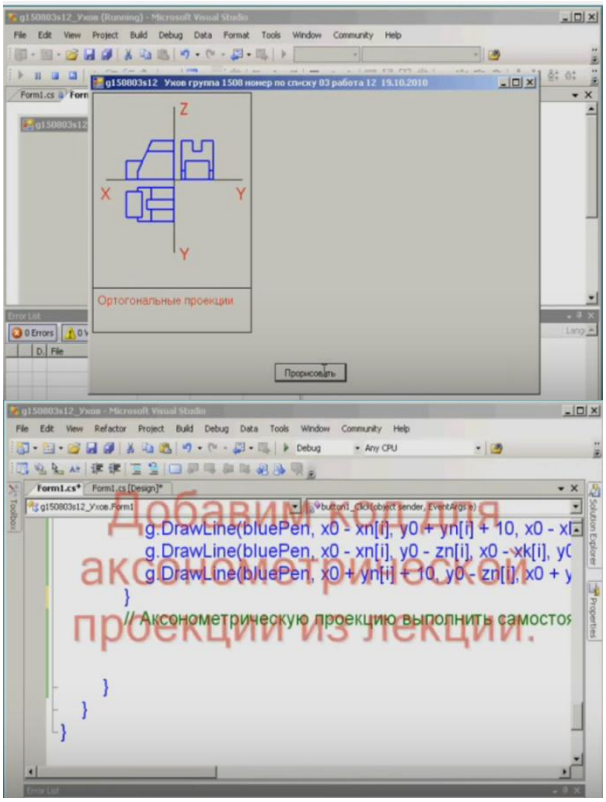
В строке «Text» написать русскими буквами «прорисовать». Курсор переместить в строку Tag.

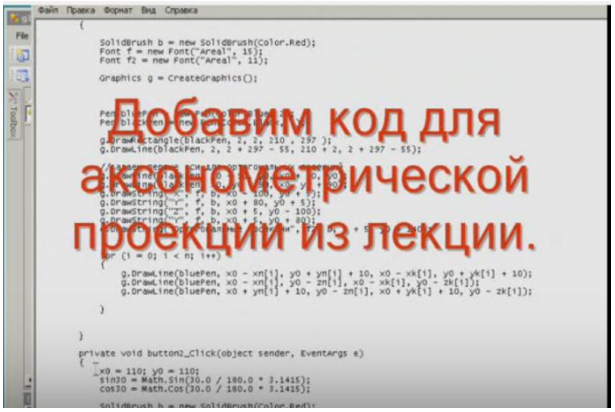


Изменится наименование кнопки «button1» на «Прорисовать». Сохраните работу. Далее нажмите кнопку  находящуюся в верхней рабочей панели.

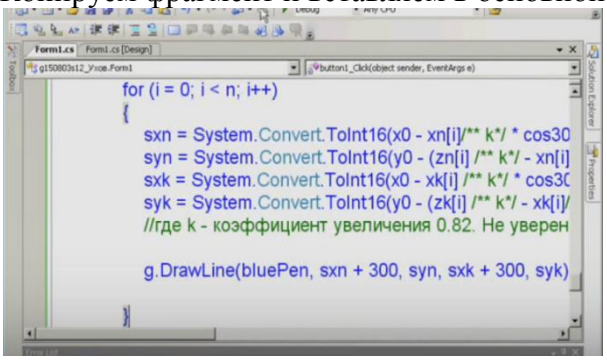


Нажмите кнопку «Прорисовать» появится ортогональная проекция.

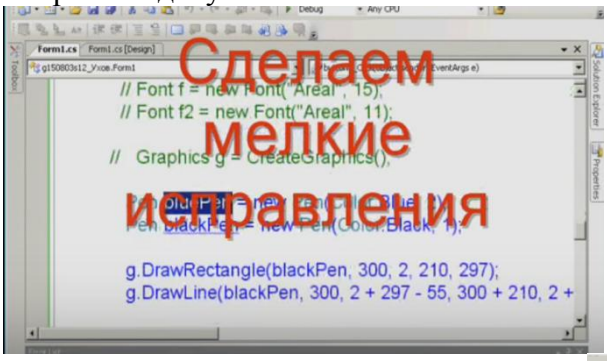




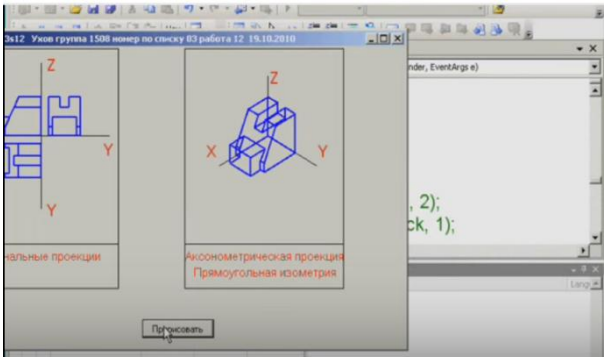
Копируем фрагмент и вставляем в основной документ.

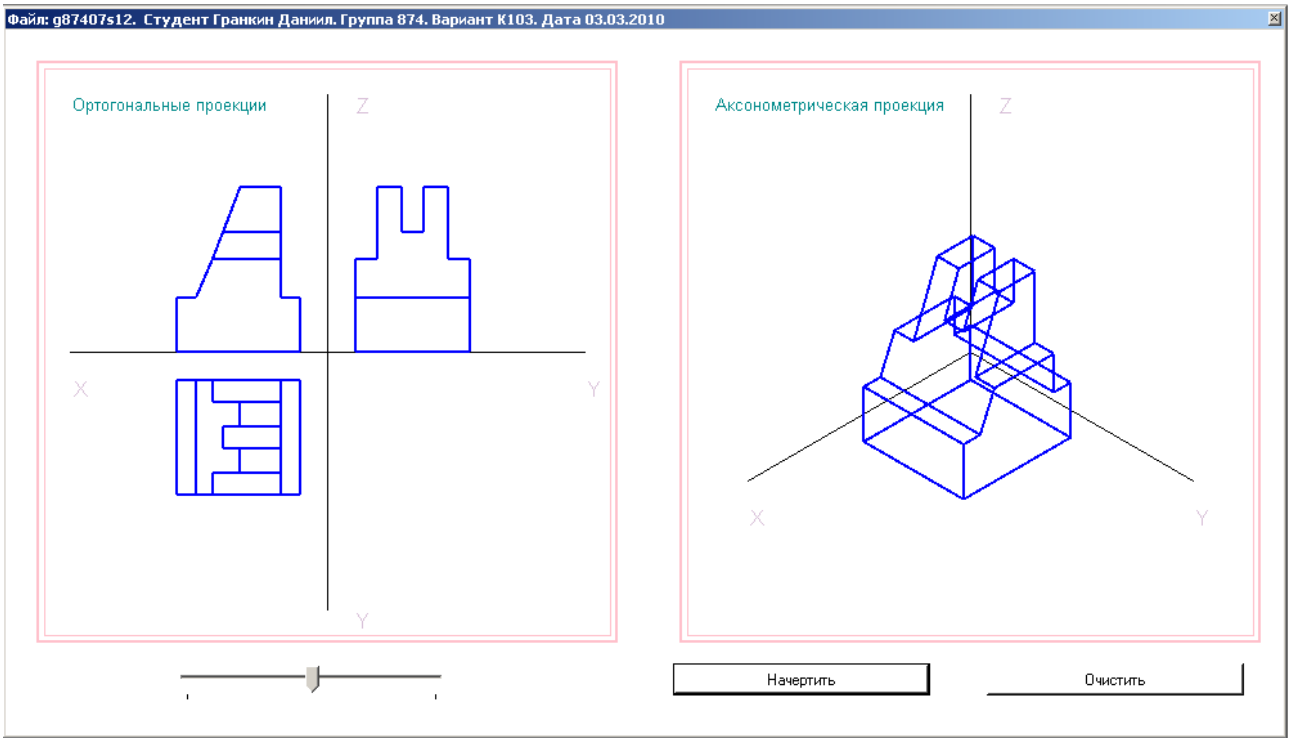


Сохраняем документ.



Сохраните работу. Далее нажмите кнопку  находящуюся в верхней рабочей панели.


Z
Y
X
Y
Аксонометрическая проекция
Прямоугольная изометрия


Файл: g87407s12. Студент Гранкин Даниил. Группа 874. Вариант К103. Дата 03.03.2010
Ортогональные проекции
Z
X
Y
Y
Аксонометрическая проекция
Z
X
Y
Начертить
Очистить

Лабораторная работа № 2 Создание простой трехмерной модели детали в «Компас-3D» и ассоциативных видов

В настоящее время широко развивается виртуальное моделирование трехмерных объектов, которое, по сравнению с чертежом, является более наглядным способом представления оригинала. и более мощным и удобным инструментом решения геометрических задач. Кроме этого по полученной модели можно создать ассоциативный чертеж.

Система КОМПАС-3D позволяет нам в автоматическом режиме получить любые стандартные и дополнительные виды модели.

Откроем рабочее окно ***<Чертеж>***  **.**На панели переключений выберем кнопку <***Ассоциативные виды***>  , после чего откроется панель **Создание ассоциативных видов** (рис. 2.1).

Ассоциативный вид – это вид неразрывно связанный с трехмерной моделью, по образу которой формируется данный чертеж. Любое изменение формы и размеров модели неизбежно повлечет к соответствующим изменениям в ассоциативных видах. На рабочей панели введем кнопку ***<Стандартные виды>***  **,** при этом на экране появится диалоговое окно, с помощью которого можно открыть папку, где находится необходимый файл, соответствующий модели.

После чего на поле чертежа отобразится фантом в виде прямоугольников, условно обозначающих три основных вида. В строке **параметров объектов** можно установить ориентацию детали, и тем самым определить главный вид, масштаб, включить или выключить невидимые линии, линии переходов, а также назначить цвет изображения.

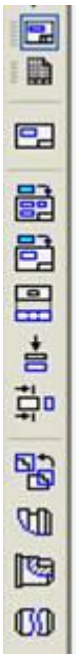


Рис. 2.1

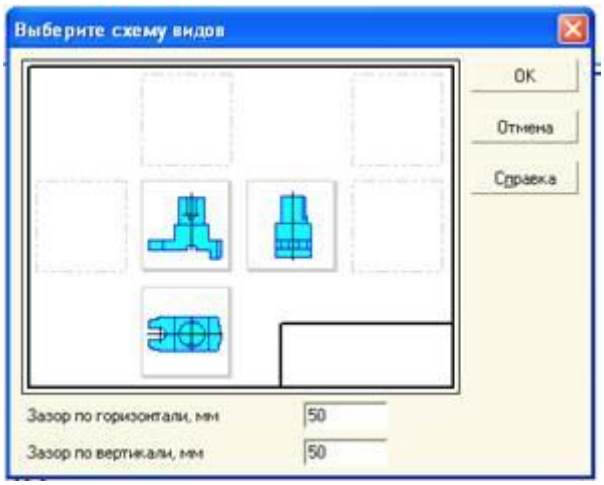


Рис. 2.2

Для того чтобы наиболее рационально расставить виды на поле чертежа, введем кнопку <***Схема видов***>  **.**В результате откроется диалоговое окно (рис. 2.2), в котором можно установить набор стандартных видов, необходимых для полного представления о форме данной детали.

По умолчанию в диалоговом окне установлены три вида: главный вид; вид снизу; вид слева. Остальные основные виды представлены условными прямоугольниками. Если понадобится показать еще какой-нибудь вид, то необходимо указать его мышью. Аналогично можно удалить любой вид, кроме главного. Отменить построение главного вида невозможно.

В нижней части диалогового окна необходимо указать **Зазор по горизонтали**и **Зазор по вертикали,**то есть ввести числовое значение расстояния между видами в горизонтальном и вертикальном направлении.

Выбрав основные виды и установив их настройку, нужно указать положение точки привязки изображения – начала системы координат главного вида. После того, как на поле чертежа появятся выбранные виды, в основной надписи в автоматическом режиме будут установлены все необходимые сведения об изделии. Они передадутся из файла модели.

На рис. 2.3 и 2.4 показана модель и ассоциативные виды, полученные в автоматическом режиме.

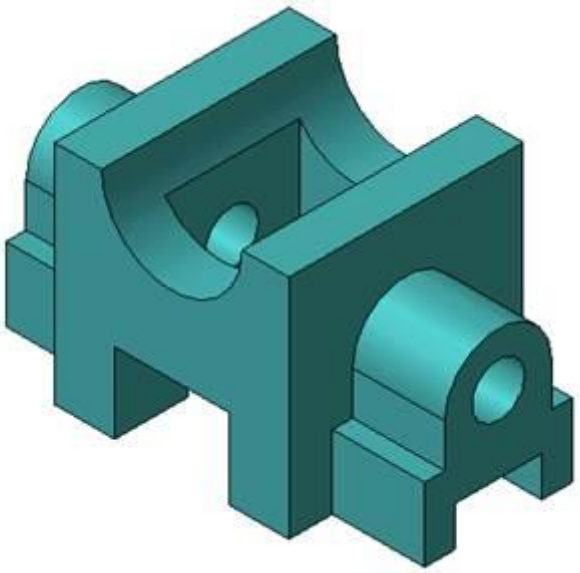


Рис. 2.3

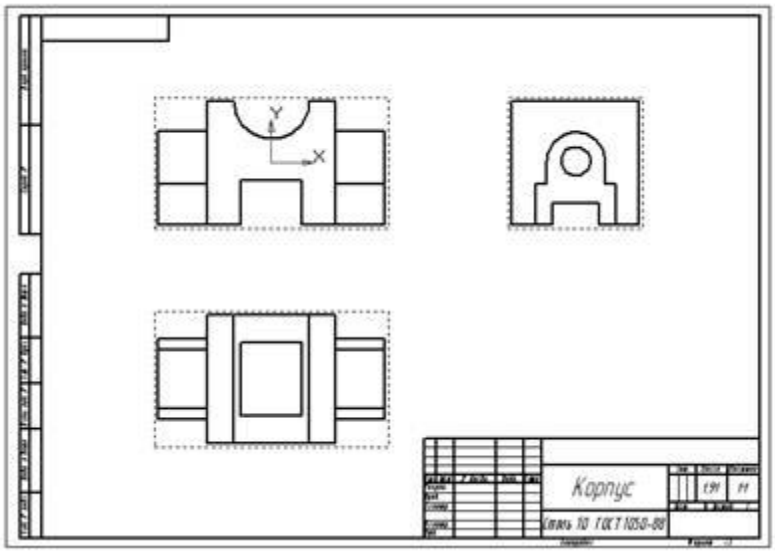


Рис. 2.4

При построении ассоциативных видов необходимо помнить, что понятие «вид» в КОМПАС – 3D и машиностроительном черчении несколько различно. В черчении видом называется изображение видимой части изделия, обращенной к наблюдателю, и между отдельными видами должна быть установлена проекционная связь. Напротив, в КОМПАС – 3D под видом понимается любое логически завершенное изображение, и отдельные виды могут быть не связанными между собой.

Вид при формировании чертежа на компьютере – это средство, управляющее структурой изображения. Любой вид обладает рядом параметров: номер; масштаб; угол поворота в градусах; имя (необязательный параметр): точка привязки

В левой стороны в строке **текущего состояния** находится кнопка ***<Состояние видов>,*** справа находится кнопка ***<Список видов>*** и поле **Текущий вид,** где указывается номер текущего вида. (рис. 2.5).

Рис. 2.5

Для получения информации о видах документа введем кнопку ***<Состояние видов>***, при этом откроется диалоговое окно (рис. 2.5).

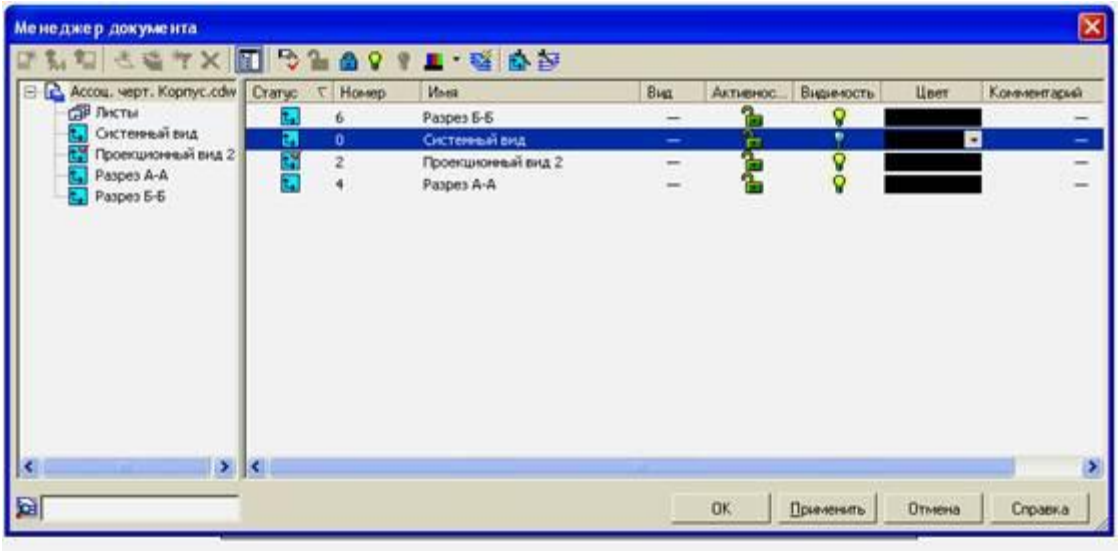


Рис. 2.5

В этом окне приводятся все сведения о видах, которые будут представлены на чертеже. Кроме этого, система автоматически формирует специальный **Системный вид** с нулевым номером. В этом виде выполняется внутренняя рамка и основная надпись. Любой из параметров вида может меняться пользователем в процессе работы. Исключение составляет **Системный вид.** Его параметры неизменны.

Аналогично, начало абсолютной системы координат чертежа всегда находится в левом нижнем углу.

При расстановке изображений система определяет положение начала координат каждого вида на основе данных о системе координат трехмерной модели. Если вид на чертеже создается вручную, то пользователь сам устанавливает его начало координат. Поэтому, **точка привязки** вида – это его начало координат по отношению к системе координат листа.

При создании чертежа можно манипулировать отдельными видами (удалять, перемещать, поворачивать). Если необходимо удалить вид следует ввести кнопку ***<Delete>.*** Если необходимо повернуть или переместить вид, то можно воспользоваться одноименными командами в группе команд **Редактор**.

Важным этапом оформления чертежа является изображение **разрезов.** Построение разреза следует выполнять в следующей последовательности.

1. Вид, на котором будем изображать линию сечения, необходимо перевести в состояние **Текущий.** В нашем случае это – **Проекционный вид 2** (вид сверху).

2. В диалоговом окне **«Установка глобальных привязок»**включить привязку **Выравнивание.**

3. На панели **Обозначения** необходимо выбрать кнопку ***<Линия разреза***> 

4. С помощью привязки **Выравнивание** следует указать точки две точки линии разреза (рис. 2.6).

5. При создании линии сечения нужно проверить направление взгляда, которое указывается специальными стрелками. Если оно выбрано неправильно, то его можно

поменять на противоположное с помощью специальной кнопки  в строке **Параметров объектов** и ввести кнопку ***<Создать объект>***.

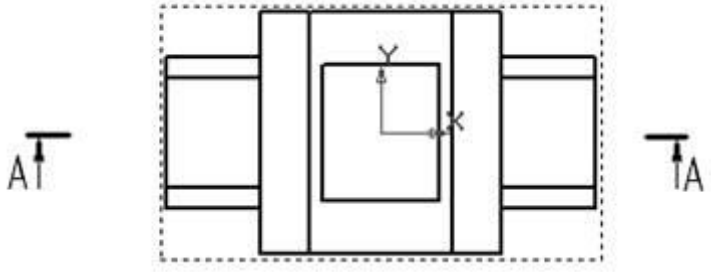


Рис. 2.6

6. На панели **Создание ассоциативных видов** введем кнопку ***<Разрез/Сечение>***  , после чего курсором необходимо указать линию сечения. Если все предыдущие операции были выполнены правильно, то линия сечения обозначится красным цветом. На экране появится фантом в виде габаритного прямоугольника.

7. Далее в строке **Параметров объектов** следует ввести закладку **<Штриховка>** и задать все параметры штриховки.

8. Мышью следует указать направление расположения разреза. Он установится в проекционной связи с видом С**верху** на месте вида **Спереди.** Новый вид будет текущим и автоматически получит имя **Разрез А – А.**

Полученный в автоматическом режиме чертеж необходимо оформить. Оформление предусматривает: построение осей, простановку размеров, введение технологических обозначений, введение технических требований, заполнение основной надписи.

На рис. 2.7 показан завершенный вариант ассоциативного чертежа корпуса.

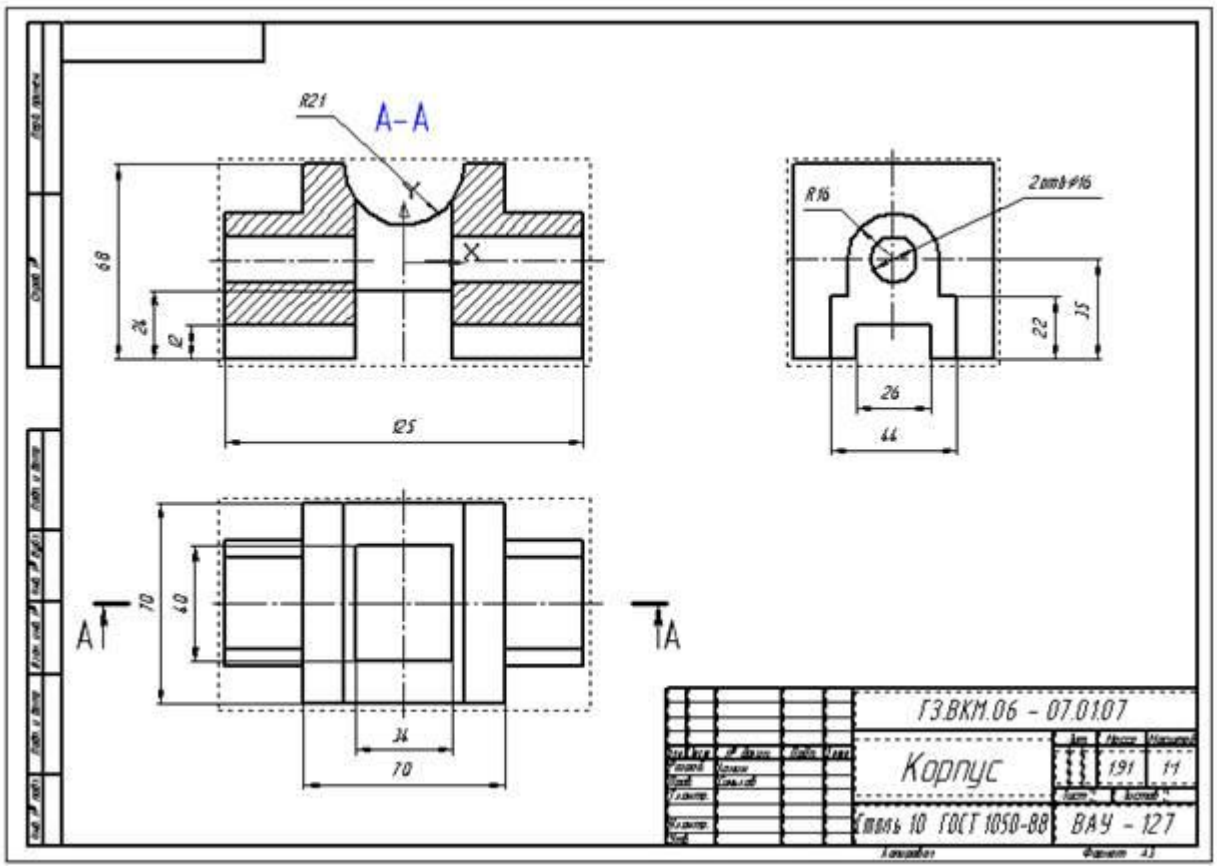
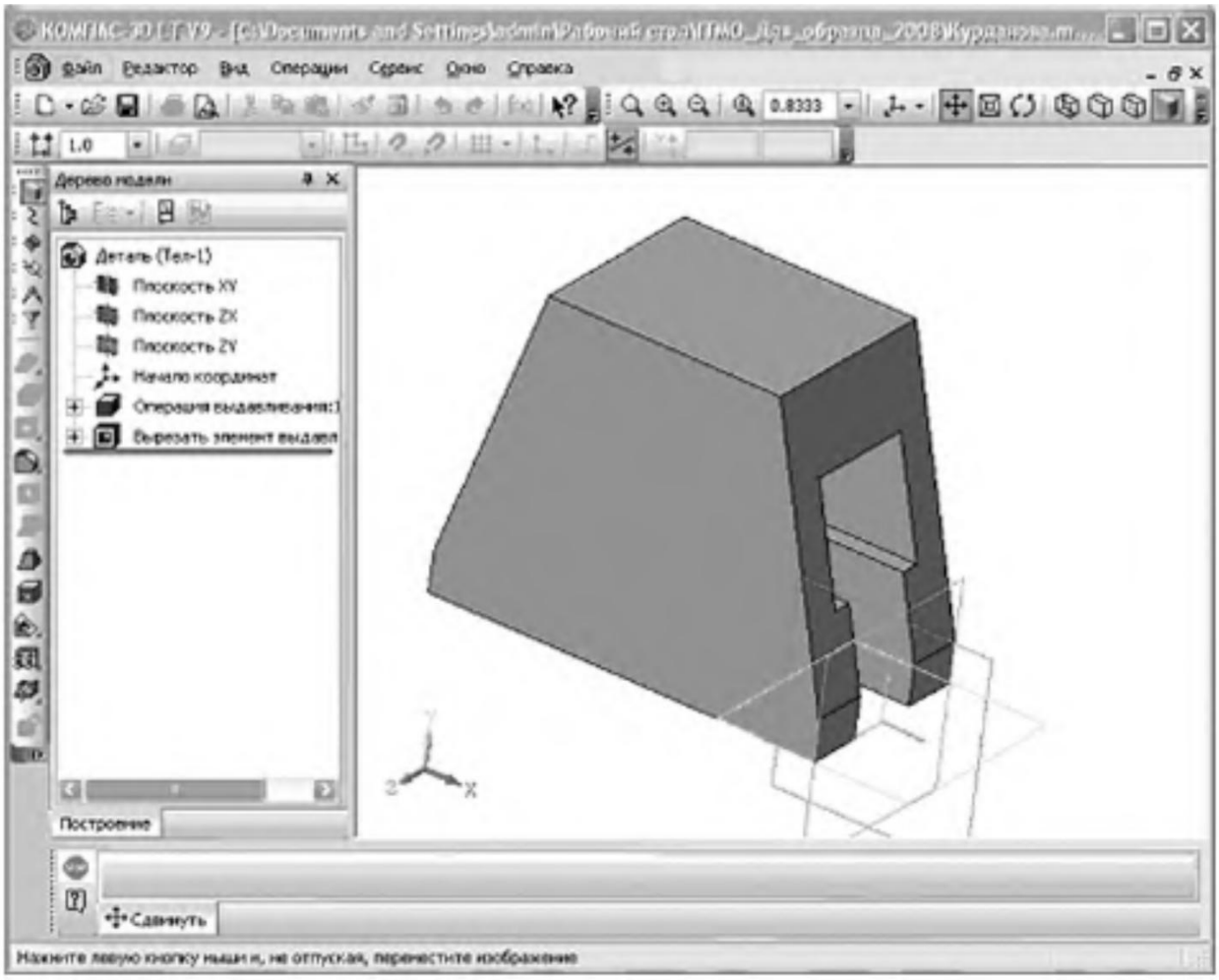
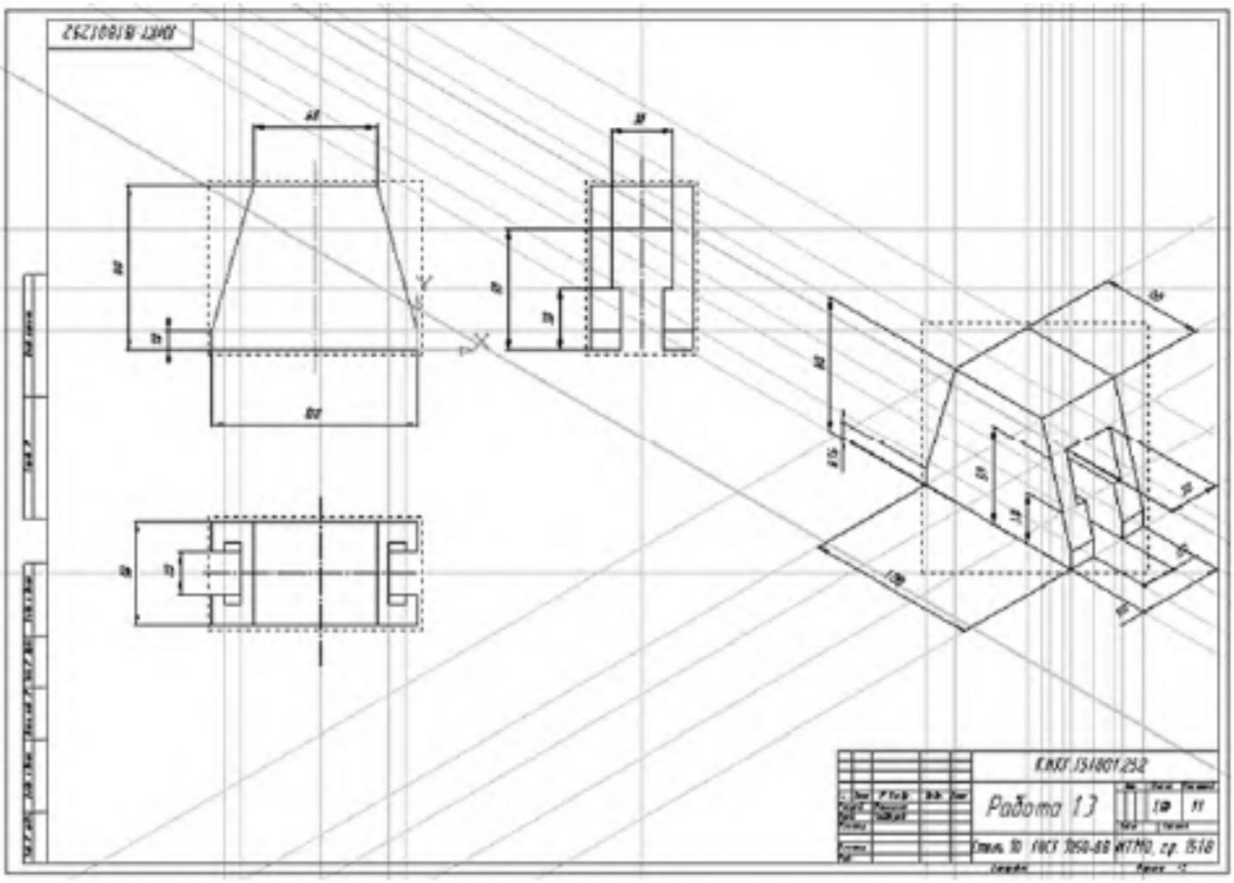
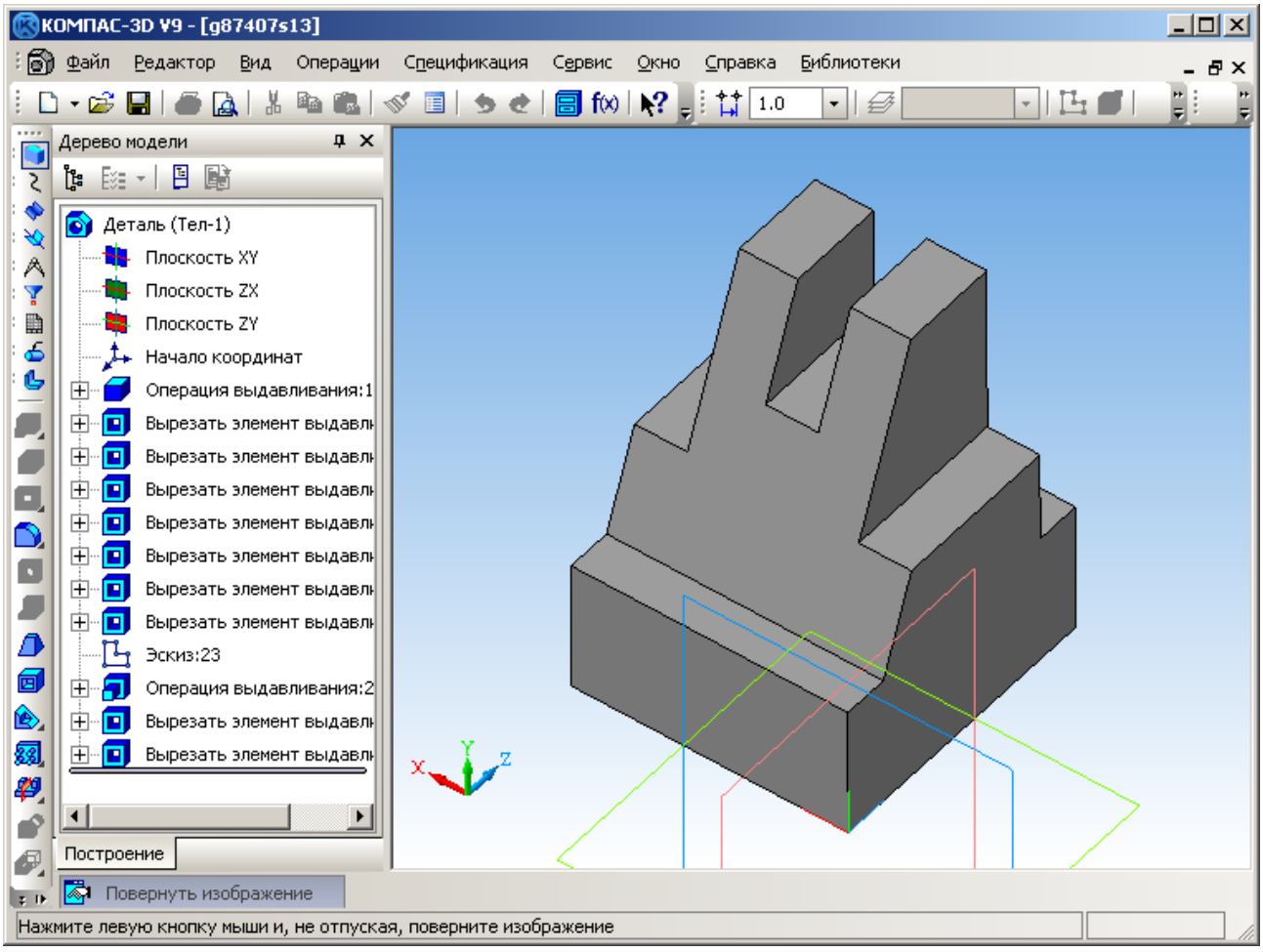


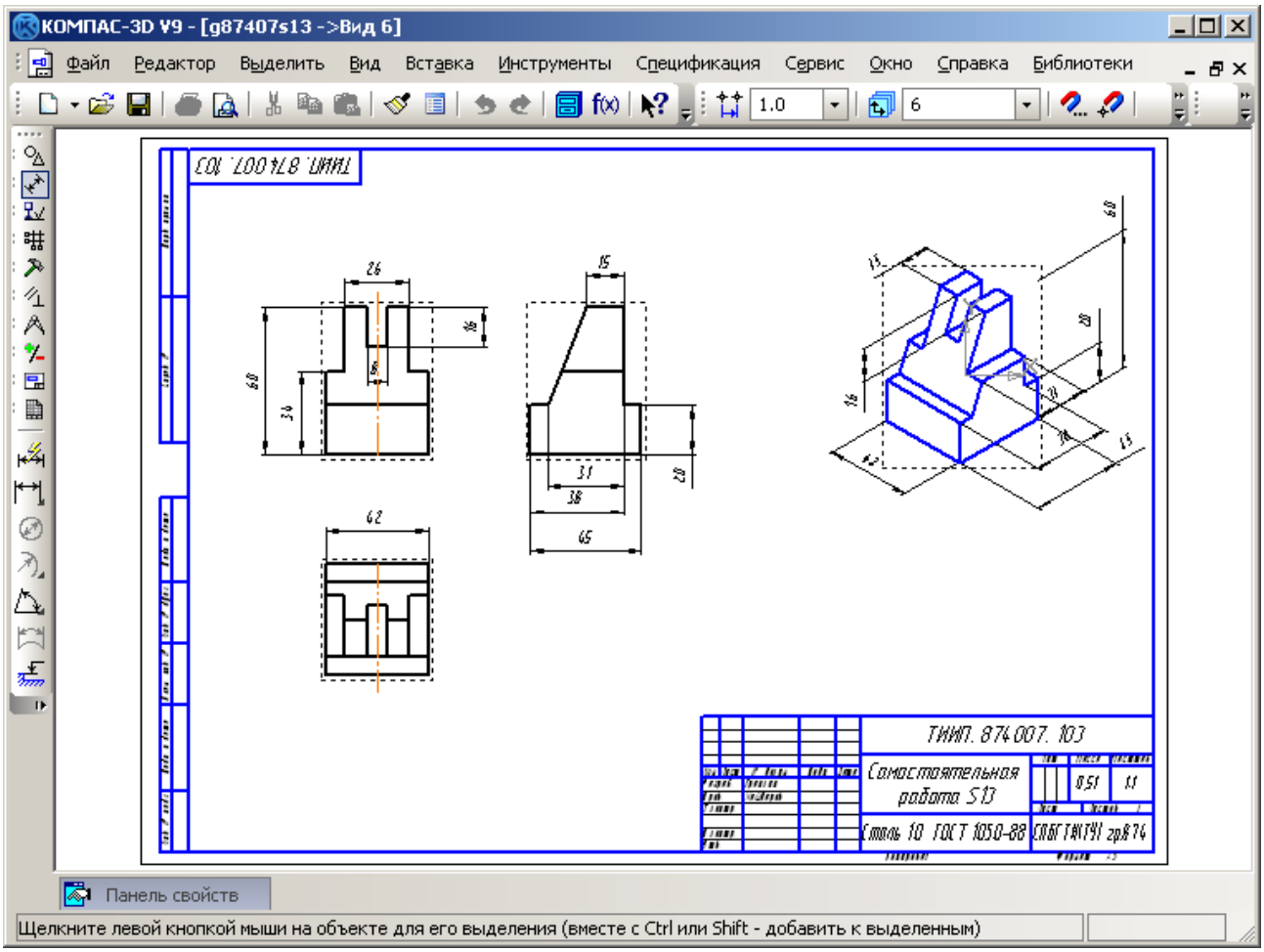
Рис. 2.7

Ниже приведен пример выполнения лабораторной работы






КОМПАС-3D V9 - [g87407s13]
Файл Редактор Вид Операции Спецификация Сервис Окно Справка Библиотеки
Дерево модели
Деталь (Тел-1)
Плоскость XY
Плоскость ZX
Плоскость ZY
Начало координат
Операция выдавливания:1
Вырезать элемент выдавл
Вырезать элемент выдавл
Вырезать элемент выдавл
Вырезать элемент выдавл
Вырезать элемент выдавл
Вырезать элемент выдавл
Вырезать элемент выдавл
Эскиз:23
Операция выдавливания:2
Вырезать элемент выдавл
Вырезать элемент выдавл
Построение
Повернуть изображение
Нажмите левую кнопку мыши и, не отпуская, поверните изображение


КОМПАС-3D V9 - [g87407s13 ->Вид 6]
Файл Редактор Выделить Вид Вставка Инструменты Спецификация Сервис Окно Справка Библиотеки
ТИИП. 874007. 103
Самостоятельная работа S13
Сталь 10 ГОСТ 1050-88
Панель свойств
Щелкните левой кнопкой мыши на объекте для его выделения (вместе с Ctrl или Shift - добавить к выделенным)

Лабораторная работа № 3 Чтение сборочного чертежа. Создание трехмерных моделей и ассоциативных чертежей в «Компас-3D»

Задание

Выполнить комплект конструкторской документации на сборочную единицу. В комплект конструкторской документации входят:

1. 3D–модели всех оригинальных деталей, входящих в сборочную единицу;
2. Ассоциативные рабочие чертежи всех оригинальных деталей, входящих в сборочную единицу;
3. 3D–модель сборочной единицы;
4. Ассоциативный сборочный чертеж;
5. Спецификация.

Описание сборочной единицы. Сборочная единица - изделие, составные части которого подлежат соединению между собой путем различных сборочных операций: свинчиванием, сваркой, пайкой, склеиванием и т.п. В данном случае сборочная единица имеет несложную конструкцию и состоит из четырех оригинальных деталей и четырех шпилечных соединений, обеспечивающих неподвижное разъемное соединение. Схема изделия приведена на рис. 3.1.

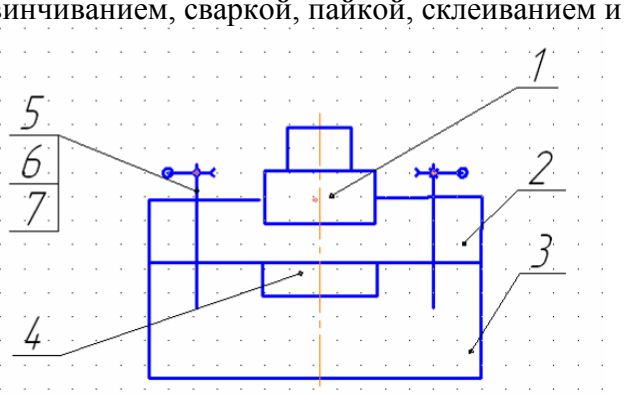


Рис. 3.1

К **Основанию** (поз.3) при помощи шпилечных соединений крепится **Корпус** (поз.2). В состав каждого шпилечного соединения входят шпилька М12, гайка и шайба (поз.5,6,7). Плотное соединение деталей обеспечивается их конструктивной формой. Сверху к **Корпусу** (поз.2) при помощи резьбы присоединяется **Штуцер** (поз.1). К **Основанию** (поз.3) при помощи пайки или склеиванием крепится **Табличка** (поз.4) с габаритными размерами 15 х 20 х 0,3 мм. Тип неразъемного неподвижного соединения выбирается в зависимости от условий эксплуатации изделия. На рис.2 даны незаконченные чертежи **Штуцера** и **Корпуса**. Детали поз.3 и 4 необходимо сконструировать в процессе моделирования, при этом модель **Таблички** создается непосредственно в 3D-сборке.

Материал деталей поз. 1, 2, 3, 4 - Сталь 45 ГОСТ 1050-88.

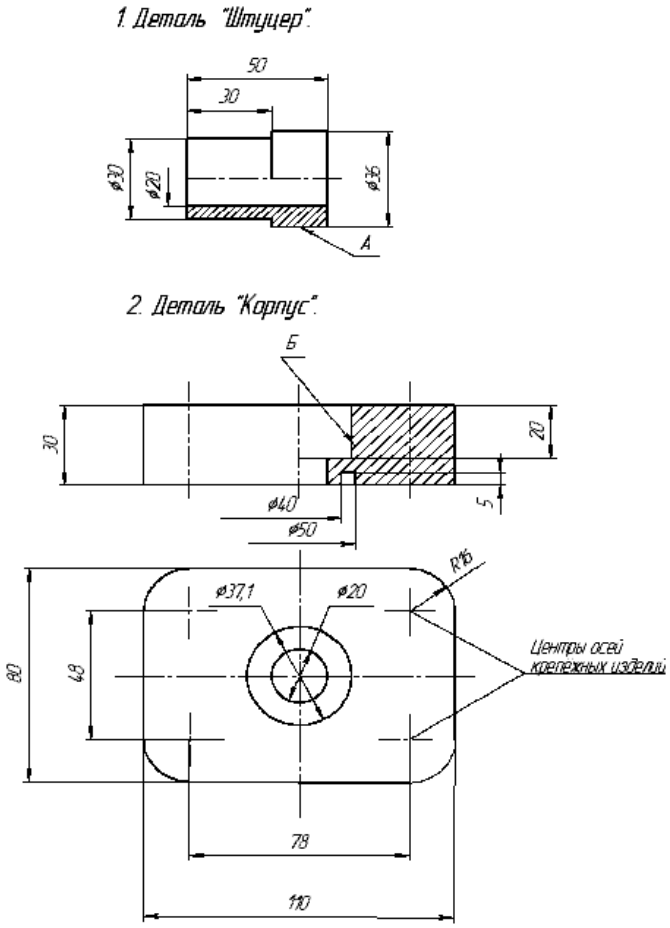


Рис. 3.2

Условия эксплуатации: вибрация повышенная, температура повышенная.

Порядок выполнения работы

Давайте шаг за шагом рассмотрим процесс создания конструкторской документации в соответствии с заданием.

1**.** Начнем с того, что дадим сборочной единице имя – **Приспособление**.

2. Присвоим создаваемым документам обозначение, например, **МИРЭА.001000.ИКГ2** для **Сборки**; **МИРЭА.001001.ИКГ2** для **Штуцера** и т.д.

3.Составим предварительный перечень деталей, входящих в состав **Приспособления** (табл.1).

Таблица 1

№ дет.	Наименование	Кол-во	Обозначение	Материал	Примечание
1	Штуцер	1	МИРЭА.001001.ИКГ2	Сталь 45 ГОСТ 1050-88	Необходимо дополнить чертеж конструктивно-технологическими элементами
2	Корпус	1	МИРЭА.001002.ИКГ2	Сталь 45 ГОСТ 1050-88	Необходимо дополнить чертеж конструктивно-технологическими элементами
3	Основание	1	МИРЭА.001003.ИКГ2	Сталь 45 ГОСТ 1050-88	Сконструировать
4	Табличка	1	МИРЭА.001004.ИКГ2	Сталь 45 ГОСТ 1050-88	Сконструировать
5	Шпилька	4	На стандартные крепежные изделия чертежи не выполняются		Провести расчет шпилечного соединения
6	Гайка	4			
7	Шайба	4			

4.Определим способы соединения составных частей сборочной единицы:

Штуцер присоединяется к **Корпусу** при помощи резьбы. На обеих деталях параметры резьбы должны быть одинаковыми и соответствовать стандарту. Для резьбового соединения в данном случае в соответствии с ГОСТ 8724-81 «Диаметры и шаги первого ряда метрической цилиндрической резьбы общего назначения» целесообразно выбрать метрическую резьбу М36ĥ1. Выбор мелкого шага 1 мм продиктован повышенной вибрацией при эксплуатации изделия.

По условию задания **Корпус,** присоединяется к **Основанию** при помощи шпилек М12.

Табличка (Бирка) крепится к **Основанию** при помощи пайки или склеивания. Выбираем пайку припоем ПОС 40 ГОСТ 1499-70, так как изделие эксплуатируется в условиях повышенной температуры.

5.Результаты проведенных расчетов:

Для соединения заданных деталей применяются следующие стандартные крепежные изделия:

Шпилька М12 х 1,25 х 50 ГОСТ 22032 – 76.

Гайка М12 х 1,25 ГОСТ 5915 – 70.

Шайба 12 ГОСТ 11371 - 78..

В **Основании** необходимо спроектировать четыре глухих резьбовых отверстия под шпилечное соединение со следующими параметрами:

d_0 (диаметр отверстия) – 10, 75 мм;

D (диаметр зенковки) – 13, 95 мм;

H (глубина отверстия) - 19 мм;

H (длина резьбы полного профиля) – 15 мм.

Параметрами положения являются межцентровые расстояния: 78 мм х 48 мм

В детали **Корпус** необходимо спроектировать четыре сквозных отверстия с диаметром $d_{отв}$ =14 мм. Параметрами положения являются межцентровые расстояния: 78 мм х 48 мм.

6. Построим трехмерные модели деталей поз.1, 2 и 3. Начинать надо с моделирования детали поз.2, так как две другие детали будут к ней присоединяться. После моделирования детали выполняется её ассоциативный рабочий чертеж.

7. Процесс моделирования детали предполагает выполнение нескольких этапов:

Предварительный анализ детали;

Создание и сохранение документа для модели детали;

Создание эскиза детали;

Создание базовой детали;

Создание дополнительных элементов детали;

Создание ассоциативного чертежа детали.

МОДЕЛИРОВАНИЕ ДЕТАЛИ ПОЗ.2 (КОРПУС)

Предварительный анализ детали

Для построения модели **Корпуса** (рис.2) проведем предварительный анализ детали:

Представим конструкцию детали и мысленно разобьем ее на основные формообразующие элементы (тела – примитивы). **Корпус** представляет собой призму со скругленными углами. Параметрами формы для призмы являются длина 110 мм, ширина 80 мм и высота 32 мм (рис.3.3).

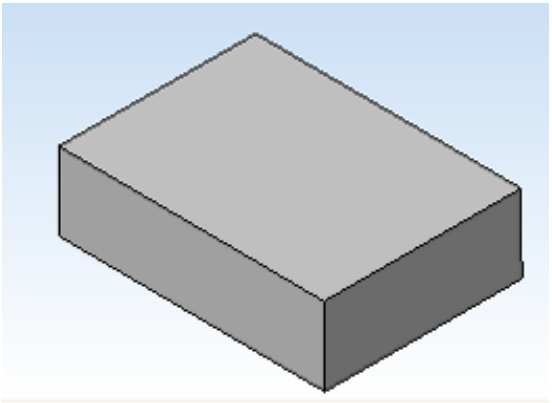
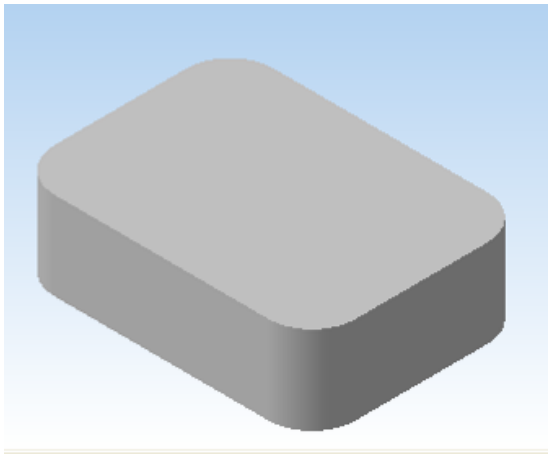


Рис. 3.3 Формообразующий элемент **Корпуса**

Скругления ограничены цилиндрическими поверхностями с параметрами: радиус 16мм и высота 32 мм (рис.3.4).

Рис. 3.4. **Корпус**– призма со скругленными углами

В центре **Корпуса** имеется цилиндрическое отверстие (рис. 3.5) с параметрами: диаметр 37.1 мм и глубина (высота) 20 мм. Отверстие предназначено для нарезания резьбы под **Штуцер.** Согласно ГОСТ 8724-81 выполняем метрическую резьбу с наружным диаметром 36 мм и мелким шагом 1мм. Шаг резьбы выбирается в зависимости от условий эксплуатации: при нормальной вибрации - крупный (как наиболее дешевый), а при повышенной - мелкий. Считается, что температурный режим не влияет на выбор шага резьбы.

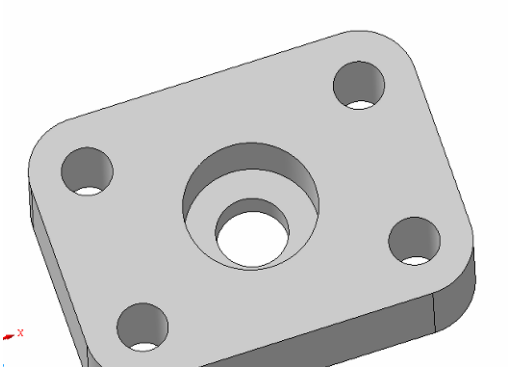


Для получения полного профиля по всей длине резьбы спроектируем специальную проточку (канавку) для выхода резьбонарезного инструмента. Ее параметры зависят от шага резьбы и регламентированы ГОСТ 10549-80. Следует помнить, что стандартизованные элементы резьбы (фаски и проточки) входят в длину резьбы.

Для обеспечения плотного соединения **Корпуса** с **Основанием** служит канавка, образованная цилиндрическими поверхностями со следующими параметрами: диаметр 50мм, высота 5мм: диаметр 40мм, высота 5мм (рис. 3.6).

Деталь имеет продольную и поперечную симметрию.

В углах детали должны быть выполнены четыре сквозных отверстия под шпильки М12х1,25х50 ГОСТ 22032-76. Диаметр этих отверстий зависит от параметров шпилечного



соединения и равен 14 мм. Параметрами положения отверстий являются межцентровые расстояния 78мм и 48мм. Все размеры в данном случае являются сопряженными.

Рис. 3.5. Корпус с отверстием М36х1 и сквозными отверстиями

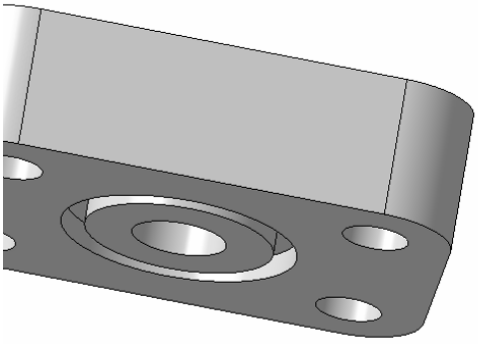


Рис. 3.6. Цилиндрическая канавка под шпильки

Построение 3D модели Корпуса

Перед началом работы необходимо получить лицензию на работу с КОМПАС-3D, записанную в памяти ключа. Для этого выполните команду **Сервис–Получить лицензию на КОМПАС-3D**.

Построение эскиза

Выполните команду **Файл** | **Создать** или нажмите кнопку **Создать** на **Панели стандартная** в диалоговом окне выберите тип документа **Деталь** и нажмите кнопку **ОК.** Перед Вами раскроется окно новой детали с рабочим полем, деревом построения детали и дополнительные панели.

Сразу после создания документа рекомендуется дать ему конкретное имя и сохранить файл на диске:

Выполните команду **Файл** | **Сохранить** или нажмите кнопку **Сохранить** на **Панели стандартная.**

В диалоговом окне **Укажите имя файла для записи** выберите папку, где вы хотите сохранить свой документ.

В поле **Имя файла** диалогового окна сохранения документов введите имя детали **Корпус**.

Нажмите кнопку **Сохранить**. В окне **Информация о документе** просто нажмите кнопку **ОК**. Поля этого окна заполнять необязательно.

Построение модели любой детали необходимо начать с основания – первого формообразующего элемента. Для **Корпуса** это призма.

Создайте эскиз на плоскости **ZX**. Для чего, укажите щелчком мыши в **Дереве построения** плоскость **ZX** (рис. 3.7). При этом пиктограмма плоскости будет выделена зеленым цветом, а в окне детали появится условное обозначение плоскости – квадрат с узелками управления.

Дерево построения детали - это представленная в графическом виде последовательность элементов, составляющих деталь. В **Дереве построения детали** отображаются: обозначение начала координат, плоскости, оси, эскизы, операции и Указатель окончания построения модели.

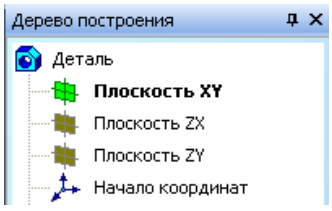


Рис. 3.7

Выполните команду **Операции** | **Эскиз** или нажмите кнопку **Эскиз** в **Панели текущего состояния**.

Выполните команду **Инструменты** | **Геометрия** | **Прямоугольники** | **Прямоугольник** или на Панели геометрия нажмите кнопку **Прямоугольник.**

Вычертите произвольный прямоугольник так, чтобы точка начала координат эскиза оказалась внутри контура (рис. 3.8).

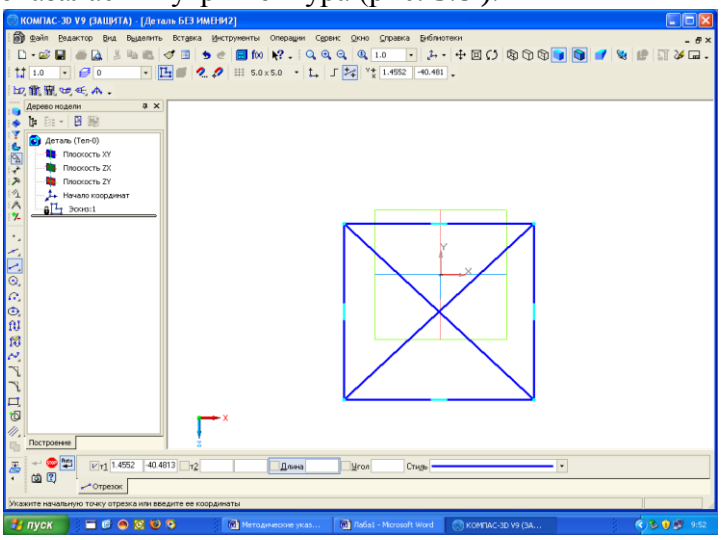


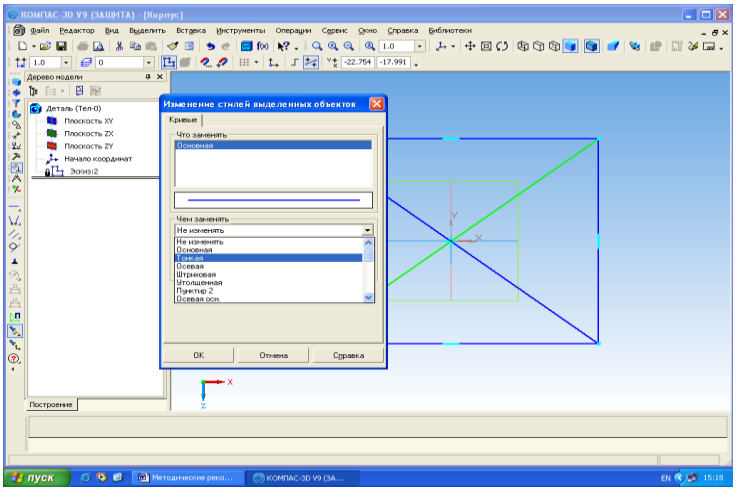
Рис.3.8. Эскиз произвольного прямоугольника

Нажмите кнопку **Прервать команду**  на Панели специального управления или кнопку **Esc**.

Для того чтобы центр прямоугольника всегда находился в начале координат, постройте диагонали прямоугольника.

Выполните команду **Инструменты** | **Геометрия** | **Отрезки** | **Отрезок** или на **Панели геометрия** нажмите кнопку **Отрезок** и с помощью привязки **Ближайшая точка** укажите две вершины прямоугольника. Будет построена одна диагональ. Повторите операцию для двух других вершин. Нажмите кнопку **Прервать команду** на **Панели специального управления**.

Выполните команду **Инструменты** | **Геометрия** | **Точки** | **Точка** или на **Панели геометрия** нажмите кнопку **Точка** и с помощью привязки **Пересечение** зафиксируйте точку пересечения диагоналей прямоугольника. Нажмите кнопку **Прервать команду** на Панели специального управления.



Измените стиль диагоналей с **Основная** (синяя линия) на **Тонкая** (черная линия) при помощи контекстного меню, предварительно выделив диагонали щелчком левой мыши при нажатой кнопке **Shift** (рис. 3.9).

Рис. 3.9. Изменение стиля выделенных объектов

На **Панели параметризация** нажимаем кнопку **Объединить точки** и указываем точку пересечения диагоналей и точку начала координат (рис. 3.10).

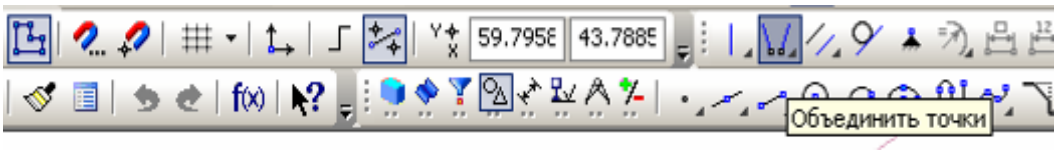


Рис. 3.10.

Результат команды **Объединить точки**

Выполним команду **Инструменты** | **Размеры** | **Линейные** | **Линейный размер** и проставим длину и ширину прямоугольника 110 и 80 (рис. 3.11).

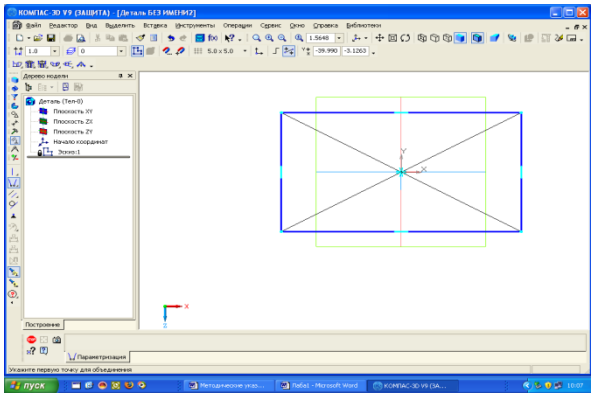


Рис. 3.11. Параметрический эскиз прямоугольника

Построим четыре окружности для отверстий под шпильки диаметром 14 мм. Для этого на **Панели геометрия** активируем кнопку **Окружность по центру и радиусу**.

Щелкните мышью примерно в том месте грани, где должно располагаться отверстие, на **Панели свойств** задайте радиус 7 мм и

нажмите кнопку **Создать объект**. При помощи команды **Линейный размер** на странице **Размеры** задайте положение центра отверстия (рис. 3.12).

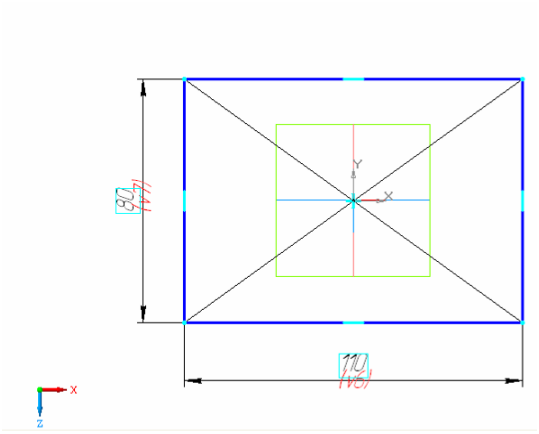


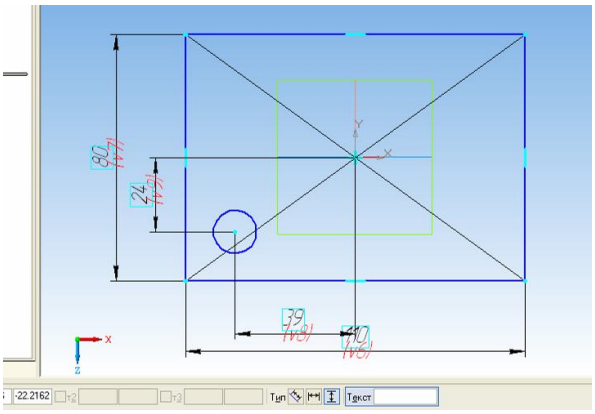
Рис. 3.12. Параметрический эскиз отверстия под шпильку

Выделяем окружность щелчком левой мыши и на вкладке «**Редактирование**» командой **Симметрия**, получим еще три окружности.

Рис. 3.13. Эскиз отверстий в параметрическом режиме

Чтобы выровнять центры отверстий, воспользуйтесь командами **Выровнять точки** по вертикали и **Выровнять точки по** горизонтали на странице **Параметризация**.

Командой **Геометрия** │**Окружности** │**Окружность** построим окружность диаметром 20 мм с центром в начале координат (рис. 3.14).



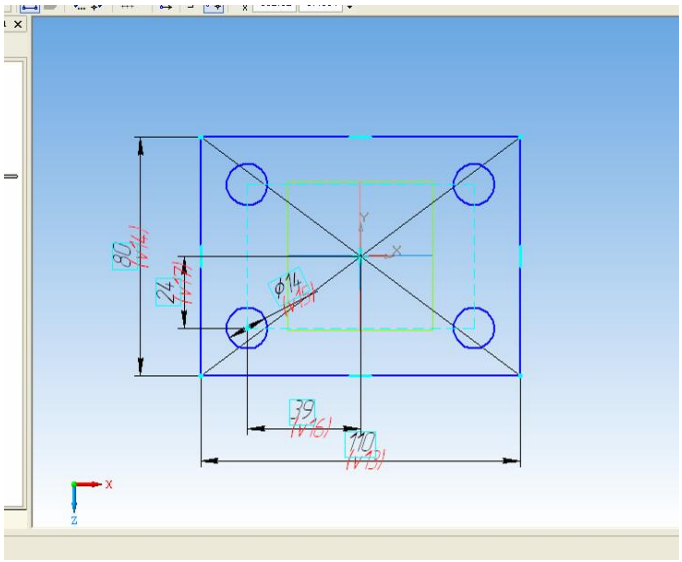
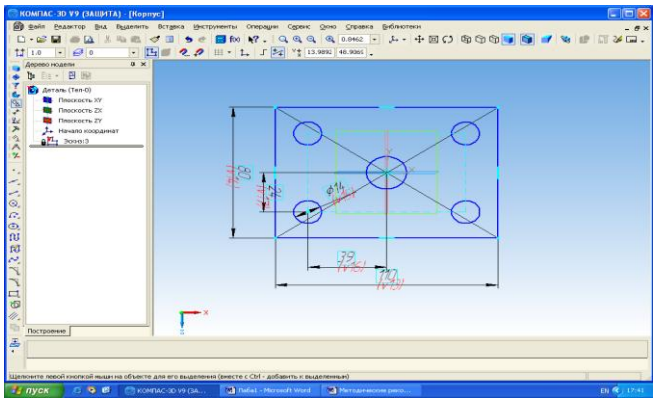


Рис.3.14. Окончательный эскиз



Закройте эскиз. Для этого нажмите еще раз кнопку **Эскиз.**

Построение 3D модели

Примените команду **Операции** │ **Выдавливание –** в раскрывающемся окне **Способ выдавливания** выберите **На расстояние,** в поле **Расстояние 1 –** высоту призмы 32 мм. Проследите, чтобы в поле **Тип построения тонкой стенки** была активна опция **Нет**.

Нажмите кнопку **Создать объект** и заготовка **Корпуса** появится в окне документа (рис. 3.15).

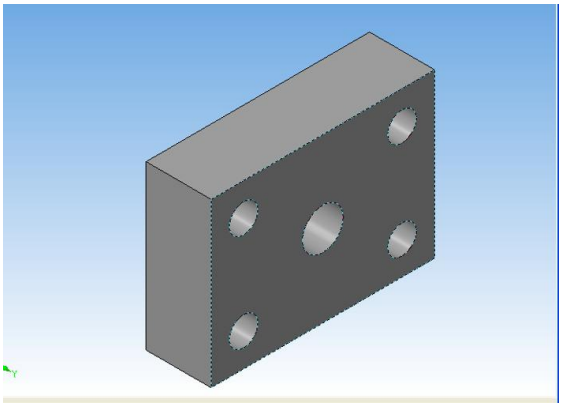


Рис. 3.15. Заготовка **Корпуса**

Выполните скругление ребер призмы со следующими параметрами: радиус 16 мм, высота равна высоте призмы, т.е. 32 мм. Примените команду **Скругление** на странице **Редактирование детали**. В раскрывающемся окне укажите: Тип скругления – **Постоянный радиус** (по умолчанию активен), значение радиуса скругления - 16 мм.

Укажите ребра призмы щелчком мыши. На модели появятся фантомы скруглений (рис. 3.16).

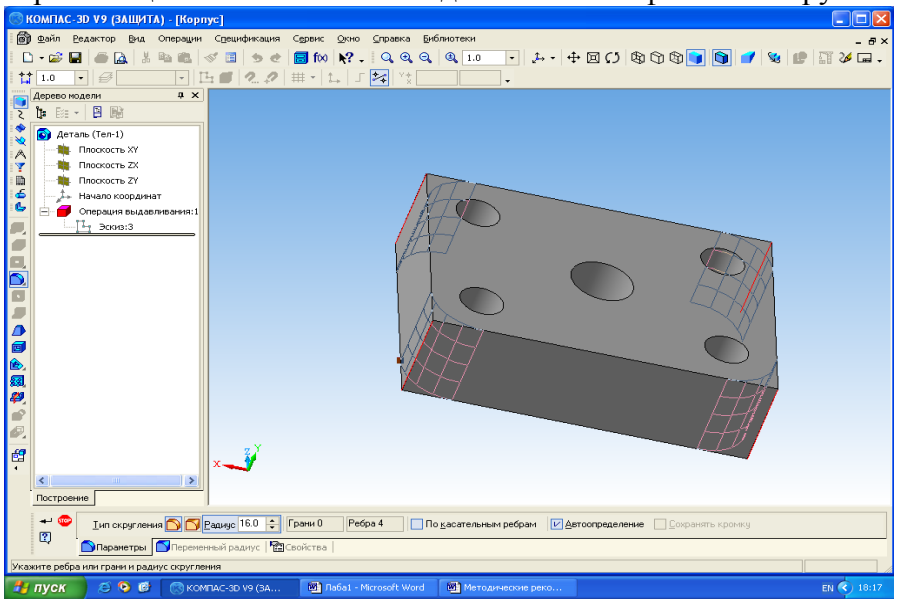


Рис. 3.16. Фантомы скруглений

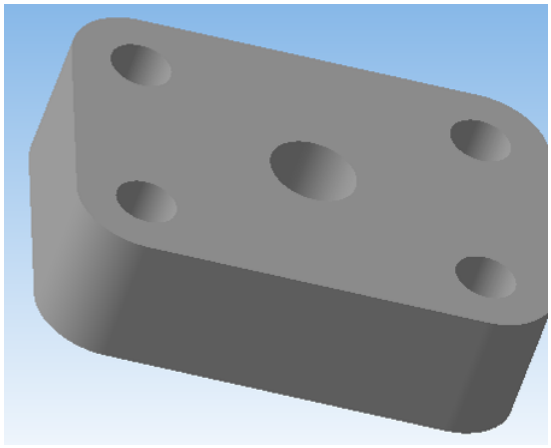


Рис. 3.17. Моделирование **Корпуса**

Нажмите кнопку **Создать объект** и скругления будут выполнены (рис. 3.17).

Требуется выполнить отверстие с резьбой под **Штуцер**(деталь поз.1). Для этого выделите плоскость верхнего основания, щелкнув по ней курсором. Она окрашивается в зеленый цвет (рис.18).

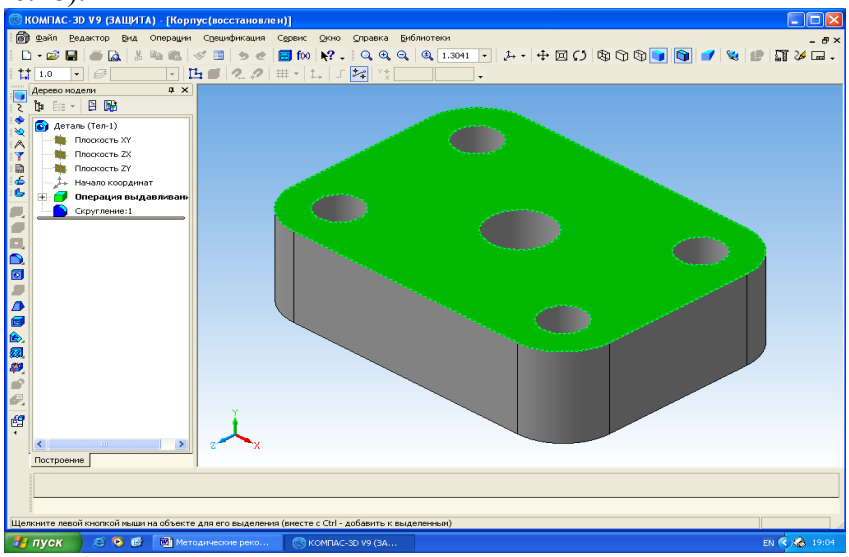


Рис. 3.18. Выделение плоскости верхнего основания

Нажимаем кнопку **Эскиз**. Плоскость занимает нормальное положение. Активируем панель **Геометрия** и чертим окружность с диаметром 37,1 мм. Выходим из команды **Эскиз.**

На инструментальной панели активируем кнопку **Вырезать выдавливанием.** Расстояние указываем 20 мм.→**Создать объект**.

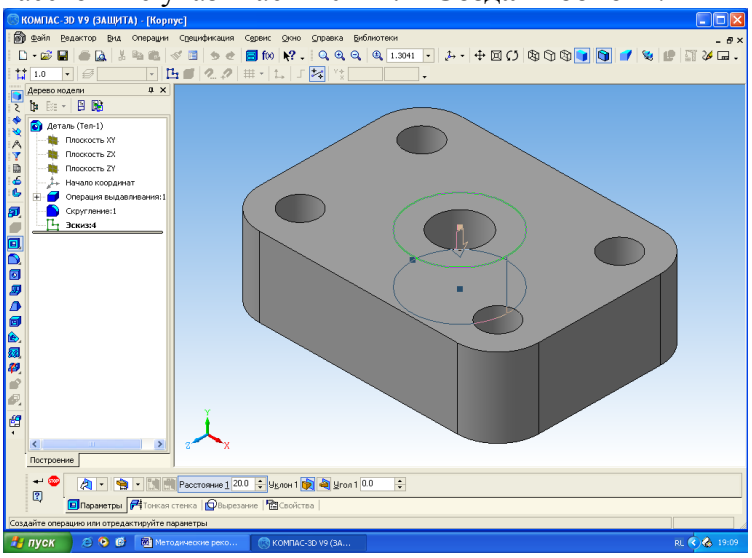


Рис. 3.19. Моделирование отверстия под резьбу

Следующим этапом является моделирование резьбы М36х1 в центральном отверстии. Начнём с технологических элементов резьбы: фаски и проточки.

Моделирование фаски и проточки.

Фаска в отверстии с резьбой необходима для выхода резьбонарезного инструмента и для облегчения завинчивания. Форму и размеры фасок для внутренней метрической резьбы устанавливает ГОСТ 10549-80. Определяющим размером служит шаг резьбы. Для резьбы М36х1 мелкий шаг равен 1 мм, следовательно, размер фаски 1х45^0.Для того, чтобы на требуемых кромках модели выполнить построение фасок, надо последовательно выполнить следующие шаги.

1. Вызовите команду **Фаска** на инструментальной панели **Редактирование детали.**
2. На панели свойств выберите способ построения фаски – **Построение по стороне и углу.** Введите значения полей: **Длина** -1 и**У гол**- 45^0.
3. Щелчком правой кнопки выделите кромку (ребро) для создания фаски. В окне покажется фантом фаски (рис. 3.20).

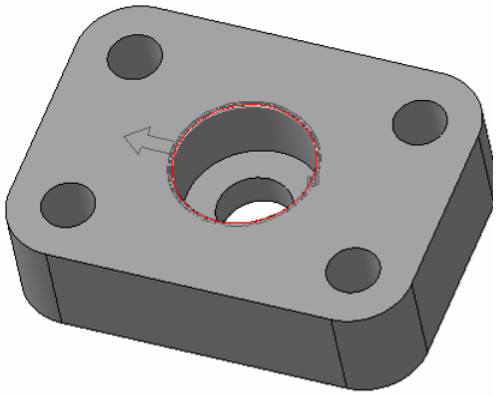


Рис. 3.20. Фантом фаски и направление её построения

4. После задания параметров фаски и настройки её свойств нажмите кнопку **Создать объект** и система отстроит фаску на кромке выделенной грани (рис. 3.21).

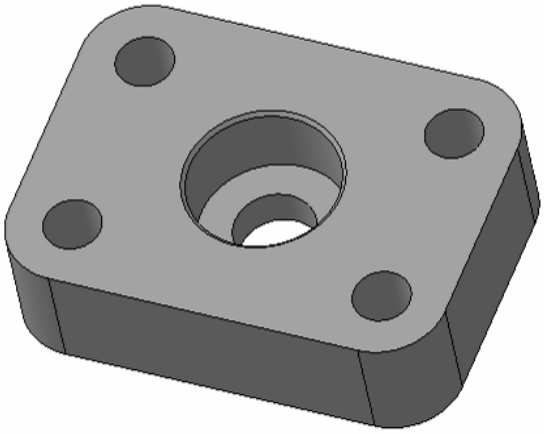


Рис. 3.21. Моделирование фаски в резьбовом отверстии

Для получения полного профиля по всей длине резьбы спроектируем специальную **проточку (канавку**) для выхода резьбонарезного инструмента. Выполним следующие шаги:

1. Нажмите кнопку **Менеджер библиотек** на **Стандартной панели**. Раскройте раздел **Машиностроение** и в правой части диалога дважды щелкните пункт **Библиотека канавок для КОМПАС -3D.**Также двойным щелчком активизируйте вкладку **Канавка по ГОСТ 10549-80**.

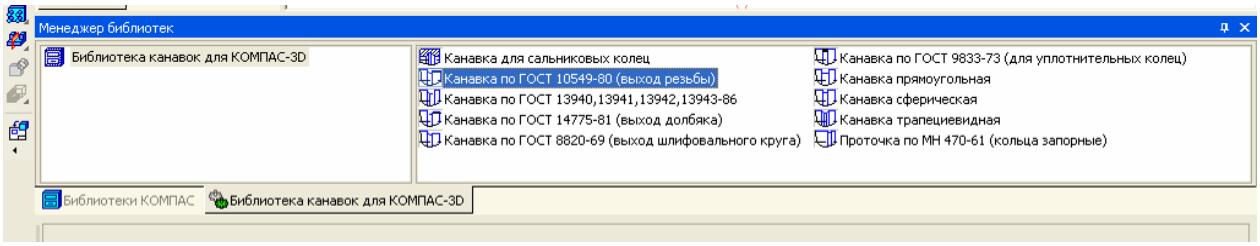


Рис. 3.22. Вызов команды **Канавка по ГОСТ 10549-80**

2. В окне документа укажите на цилиндрическую поверхность, на которой выполняется канавка (проточка). Эта поверхность выделится ярким аквамариновым цветом. В диалоговом окне **Сообщение библиотеки** поставьте флажок в разделе **Внутренняя** и нажмите **ОК**. Появится диалоговое окно, в котором будут указаны форма и размеры канавки в соответствии с ГОСТ 10549-80. Установите **Шаг резьбы– мелкий, тип проточки – нормальный** (рис. 3.23).

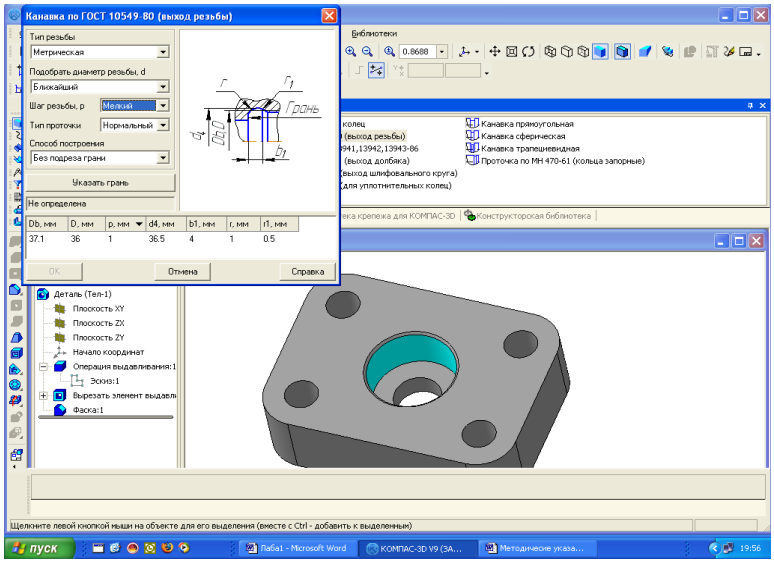


Рис. 3.23. Задание параметров проточки

3. Нажмите на клавишу **Указать грань.** В окне документа укажите базовую грань, рядом с которой будет построена канавка. На рис. 3.24 она выделена зелёным цветом.

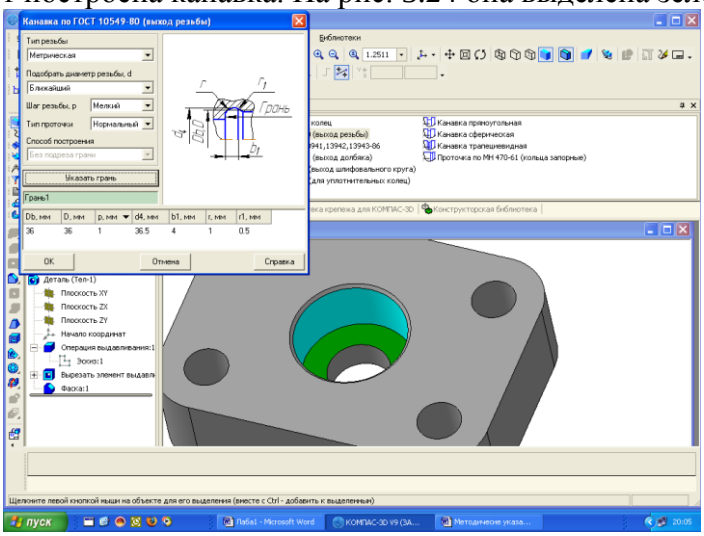


Рис. 3.24. Моделирование канавки (проточки) для выхода резьбонарезного инструмента

4. Нажмите **ОК**. Система автоматически построит канавку (проточку) для выхода резьбонарезного инструмента со стандартными параметрами (рис. 3.25).

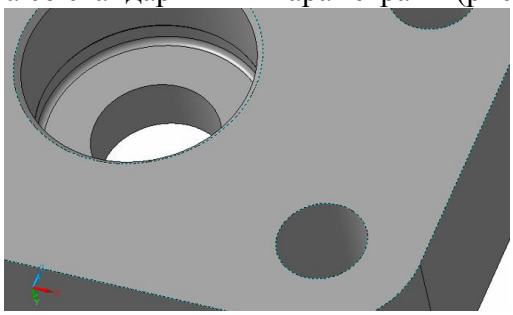


Рис. 3.25. Канавка для выхода резьбонарезного инструмента

<u>В отверстии создадим изображение резьбы М36х1</u>.

1. Нажмите кнопку **Условное изображение резьбы** на странице **Условные обозначения.** Укажите на кромку, где должна быть построена резьба (рис. 3.26).

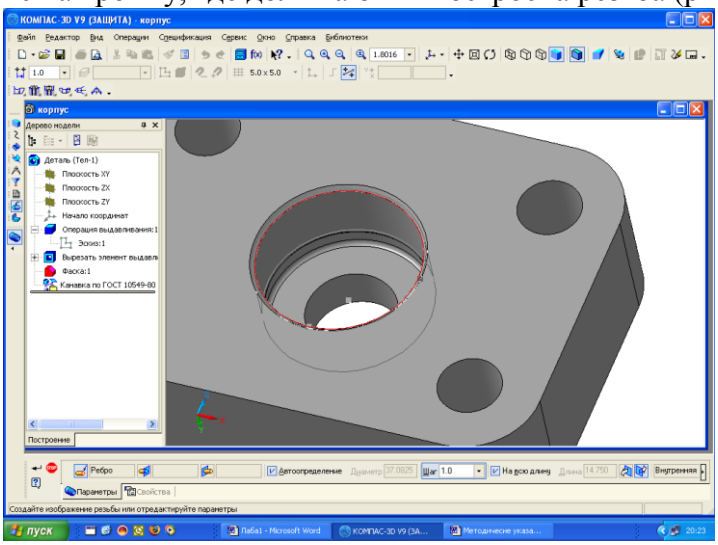


Рис. 3.26. Указание кромки

2. Во вкладке **Параметры** в группе переключателей **Направление** активизируйте **Прямое направление**. Активизируйте переключатель **Начальная граница** и щелкните на торце отверстия, от которого нужно построить резьбу (если этого не сделать, то резьба будет построена от линии, разделяющей цилиндрическую поверхность отверстия и коническую поверхность фаски) (рис. 3.27).

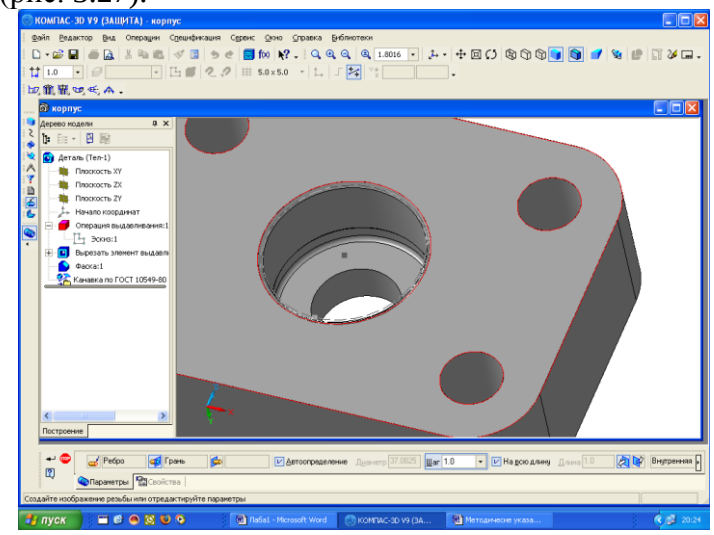


Рис. 3.27. Указание **Начальной границы резьбы**

В поле **Шаг** раскройте список и выберите значение 1.0. Отмените построение **На всю длину** и укажите **Объект для конечной границы резьбы** (рис. 3.28).

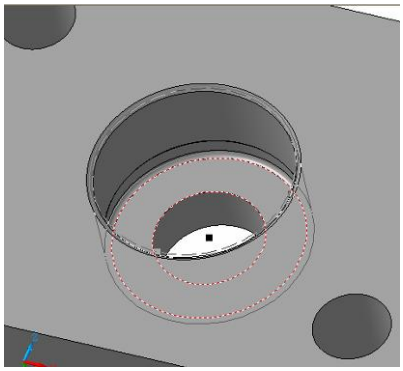


Рис. 3.28. Указание **Объекта для конечной границы резьбы**

Нажмите кнопку **Создать объект** и система отрисует резьбу на внутренней поверхности отверстия (рис 3.29). Резьба условно обозначается жёлтыми линиями.

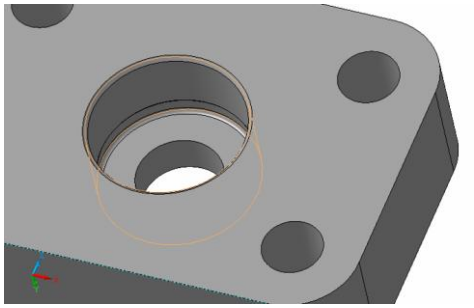


Рис. 3.29. Моделирование резьбы

Следующим этапом является моделирование конструктивного элемента, обеспечивающего плотное прилегание двух смежных деталей – **Корпуса** и **Основания**. Этот элемент представляет собой канавку, ограниченную цилиндрическими поверхностями со следующими параметрами: диаметр 50мм, высота 5мм: диаметр 40мм, высота 5мм. Выполним следующие шаги:

Командой **Вид** | **Повернуть** поверните деталь так чтобы нижняя плоскость была хорошо видна. Для вызова команды нажмите кнопку на панели **Вид** или выберите ее название из меню **Вид**. После вызова команды изменяется внешний вид курсора (он превращается в две дугообразные стрелки). Или нажмите левую кнопку мыши в окне модели и, не отпуская ее, перемещайте курсор. Модель будет поворачиваться вокруг центральной точки габаритного параллелограмма. Для выхода из команды поворота модели нажмите кнопку **Прервать команду** на **Панели специального управления** или клавишу <**Esc**> на клавиатуре. Подведите курсор к нужной плоскости и щелкните на ней. Цвет плоскости изменится на зеленый (рис. 3.30).

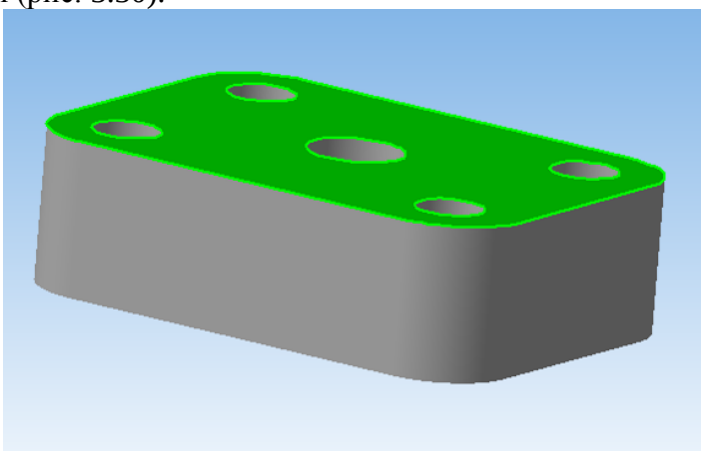


Рис. 3.30. Указание плоскости для моделирования цилиндрической канавки

Активируйте кнопку **Эскиз.** Плоскость займет нормальное положение.

На панели геометрия выберем команду **Окружность** по центру и радиусу и начертим две окружности диаметром 50 и 40 мм.

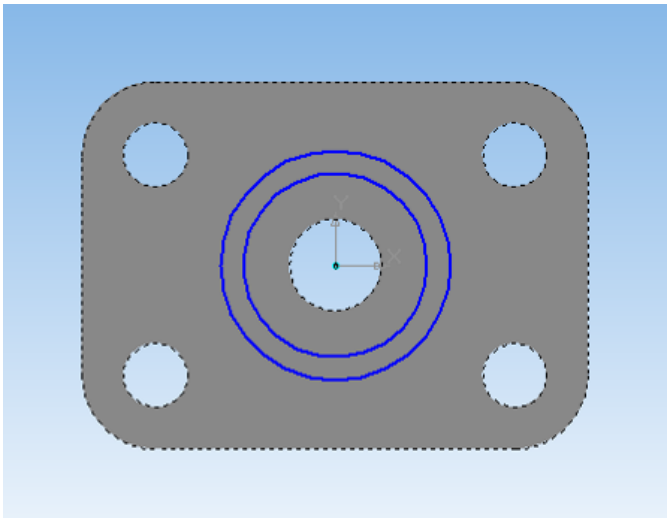


Рис. 3.31. Эскиз канавки

3. Выходим из команды **Эскиз** и операциями **Вырезать выдавливанием** на расстояние 5 мм и **Создать объект** моделируем канавку.

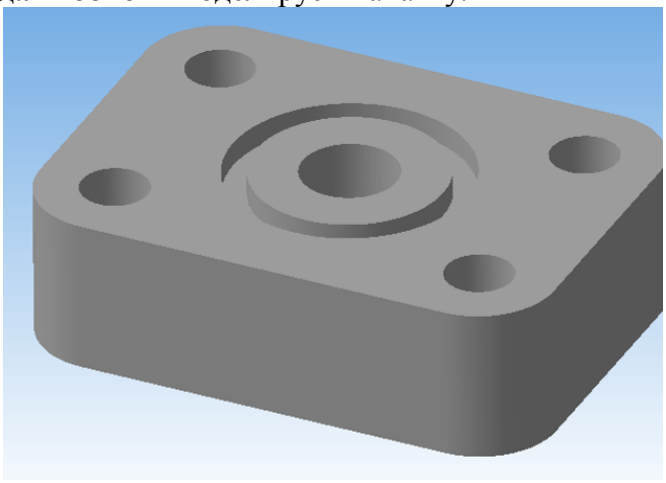


Рис. 3.32. Моделирование канавки

4. В любом месте документа щелкните правой кнопкой мыши и из появившегося контекстного меню выберите команду **Свойства детали.** На панели свойств в поле **Обозначение** введите**МИРЭА.001002.ИКГ2 →Enter**, в поле **Наименование –Корпус** и выберите из списка материалов **Сталь 45 ГОСТ 1058-88** (рис. 3.33).

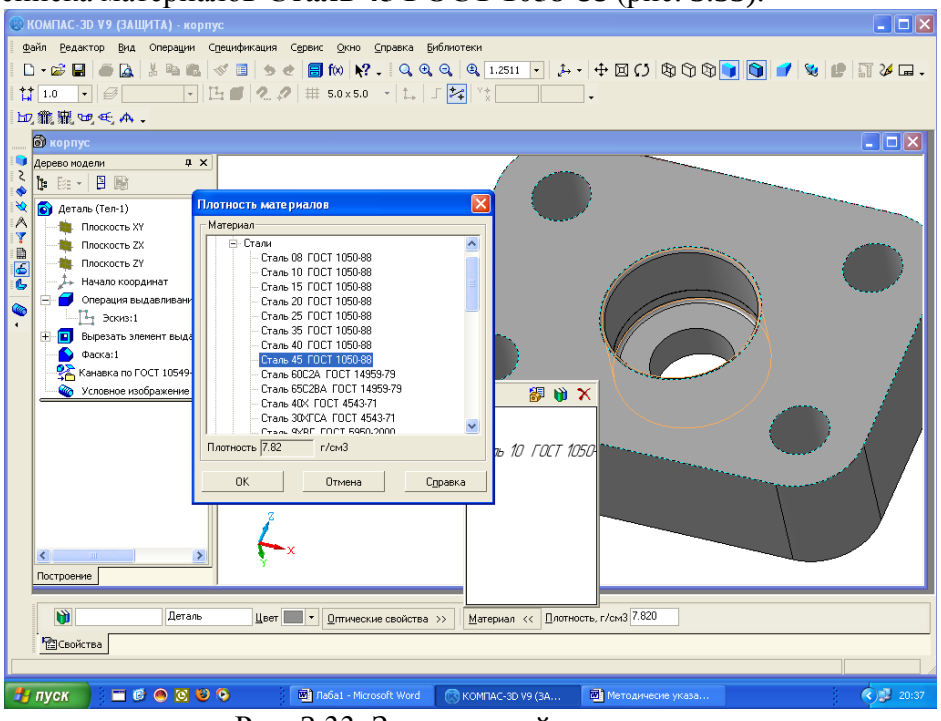


Рис. 3.33. Задание свойств модели

5. На **Панели свойств** задайте цвет детали (рис. 3.34).

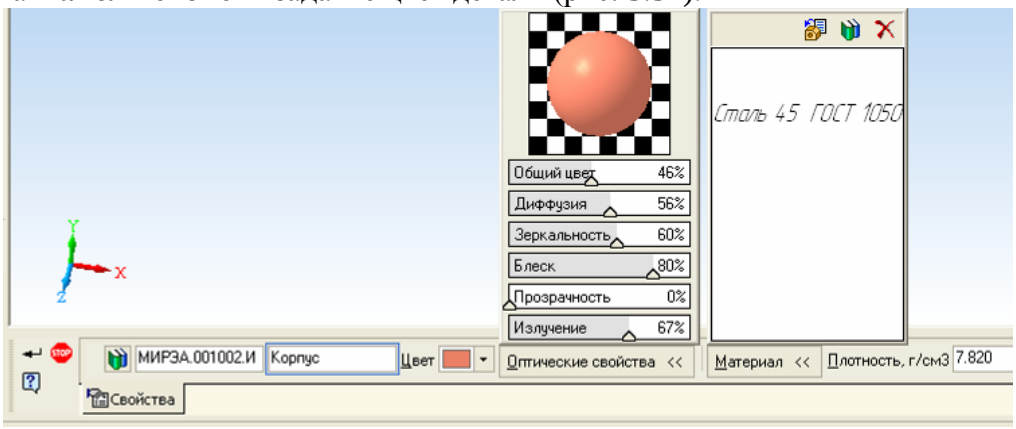


Рис. 3.34. Панель свойств детали **Корпус**

Модель **Корпуса** построена (рис. 3.35).

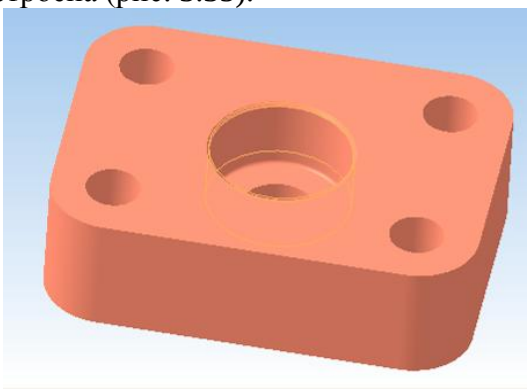


Рис. 3.35. Трехмерная параметрическая модель **Корпуса**

Построение ассоциативного чертежа Корпуса

Построив 3D-модель, можно построить ее ассоциативный рабочий чертеж, при этом сами изображения будут ассоциативно связаны с исходной трехмерной моделью. Это означает, что при изменении формы или размеров 3D-модели будут меняться изображения на ее рабочем чертеже. Для этого между геометрическими элементами в чертеже, размерами и обозначениями, необходимо сформировать ассоциативные связи. Это достигается за счет включения режима параметризации. Выполните команду **Сервис** | **Параметры**. На экране будет открыто диалоговое окно **Параметры**. На закладке **Текущий чертеж**, в левой части окна, сделайте текущей "ветвь" **Параметризация** в нижней части **Дерева параметров**. В правой части окна включите два флажка **Все** в группах **Ассоциировать при вводе** и **Параметризовать**. Нажмите кнопку **ОК.**

Рабочий чертеж детали – это конструкторский документ, содержащий изображения детали и другие данные, необходимые для ее изготовления и контроля. Следует помнить, что количество изображений на чертеже должно быть минимальным, но достаточным, чтобы обеспечить полное представление о форме детали и нанесения всех, необходимых для ее изготовления размеров.

Чтобы полностью раскрыть форму детали **Корпус** достаточно двух изображений: главного вида и вида сверху. Для выявления формы внутренних отверстий выполним сложный ступенчатый разрез. Такой разрез полностью раскроет форму отверстий и кольцевой канавки

Выполните команду **Файл** | **Создать** и создайте новый файл типа **Чертёж** формата А4. Сохраните его с тем же названием **Корпус**.

Вызовите команду **Стандартные виды** на странице **Ассоциативные виды**.

Если у вас уже были открыты документы с трехмерными моделями, то на экране появится диалоговое окно с активными файлами (рис. 3.36). Выберите модель **Корпус** для построения ассоциативного чертежа и щелкните на кнопке **ОК**.

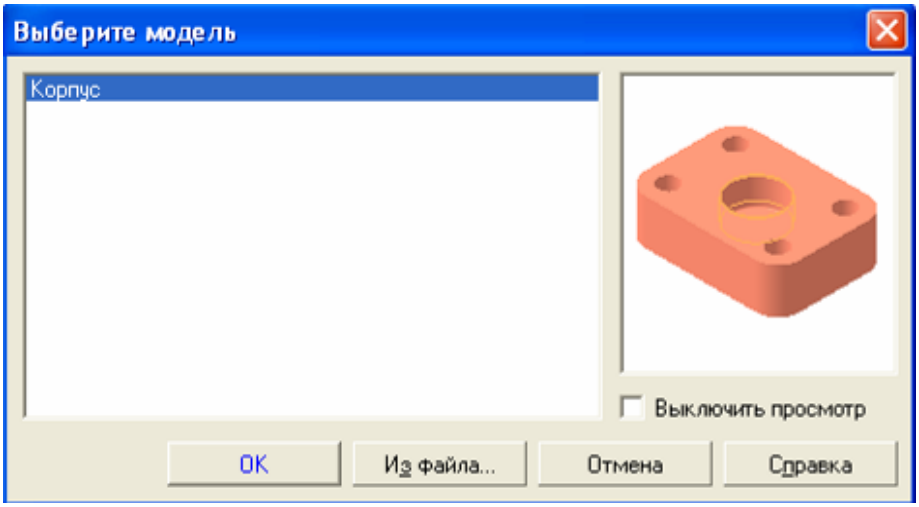


Рис. 3.36. Диалоговое окно **Выберите модель**

Если в предложенном списке нет модели для построения ассоциативного чертежа, то следует нажать кнопку **Из файла**. Появится обычный диалог, в котором следует выбрать нужный файл **Корпус** с исходной 3D-моделью и нажать на кнопку **Открыть.**

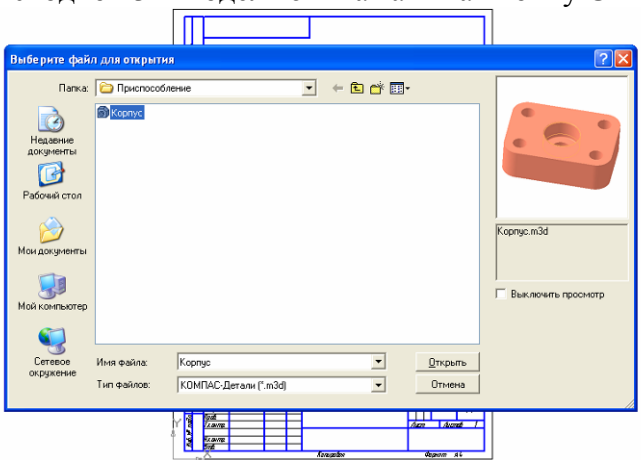


Рис. 3.37. Диалоговое окно **Выберите файл для открытия**

В окне документа появится фантом изображения трех видов (вида спереди, вида слева и вида сверху) в виде габаритных прямоугольников.

В нижней части окна расположена **Панель свойств**, с помощью которой можно управлять процессом создания видов. На запрос системы: **Укажите точку привязки вида**– пока не отвечайте и не щелкайте мышью в окне документа. Целесообразно сначала задать все необходимые параметры на **Панели свойств.**

На **Панели свойств** установите необходимые параметры создаваемого чертежа. **Ориентацию** модели **Корпус** в окне **Ориентация главного вида** оставляем без изменения – **Спереди**.

Нажмите кнопку **Схема** на **Панели свойств**. Появится диалог (рис. 3.38), в котором можно установить любой набор стандартных видов, в том числе и изометрию. В окне **Схема** видов отмените построение вида слева, щелкнув по нему мышью.

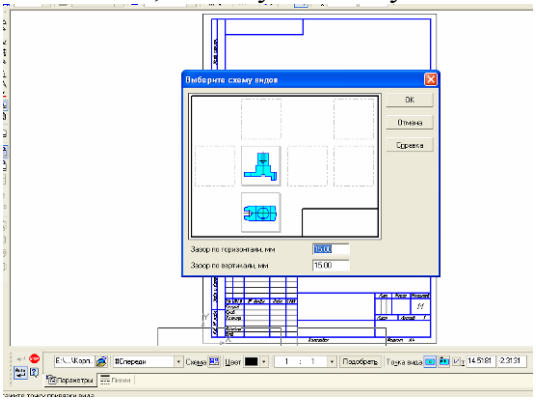


Рис. 3.38. Окно **Схема видов**

Во вкладке **Линии** установите отрисовку линий переходов тонкими линиями и выключите отображение линий невидимого контура, поскольку мы воспользуемся разрезами, чтобы показать данный контур видимым.

Задайте масштаб (например, 1:1).

Поместите фантомы ассоциативных видов в любом месте чертежа (при окончательном оформлении чертежа вы проведёте его компоновку) и щелкните мышью (рис. 3.38.1).

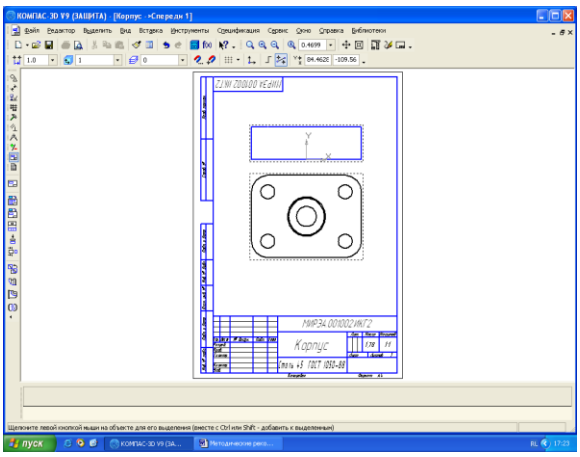


Рис. 3.38.1. Ассоциативные виды **Корпуса**

Два ассоциативных вида **Корпуса** готовы. Приведём их в соответствие с ЕСКД.

При дальнейшей работе над чертежом, состоящим из нескольких видов, необходимо следить за тем, чтобы изменения происходили в текущем (активном) виде. Чтобы вид сделать текущим, можно дважды щелкнуть левой кнопкой мыши на габаритной рамке вида или на любом геометрическом объекте, созданном в данном виде, и состояние всего вида станет текущим или выбрать его имя из списка видов на панели текущего состояния.

Отверстия в детали и глубину кольцевой канавки можно показать при помощи сложного ступенчатого разреза:

Проконтролируйте, чтобы вид сверху был активным. Для этого дважды щелкните левой кнопкой мыши на габаритной рамке вида и состояние всего вида станет текущим.

С помощью кнопки **Увеличить масштаб рамкой** на панели **Вид** увеличьте масштаб вида.

Нажмите кнопку **Установка глобальных привязок** на панели **Текущее состояние** и включите привязку **Выравнивание** (рис. 3.39).

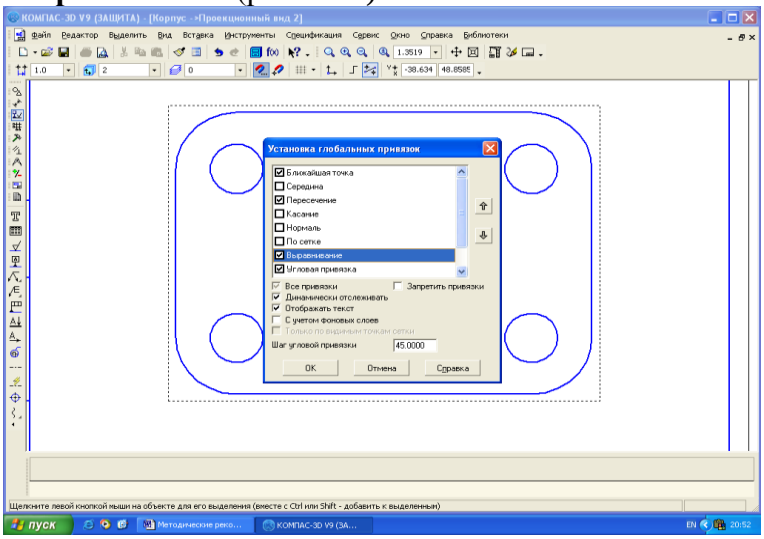


Рис. 3.39 Установка привязки **Выравнивание**

Выполните команду **Инструменты** | **Обозначения** | **Линия разреза** или включите кнопку **Линия разреза** на инструментальной панели **Обозначения** (рис. 3.40).

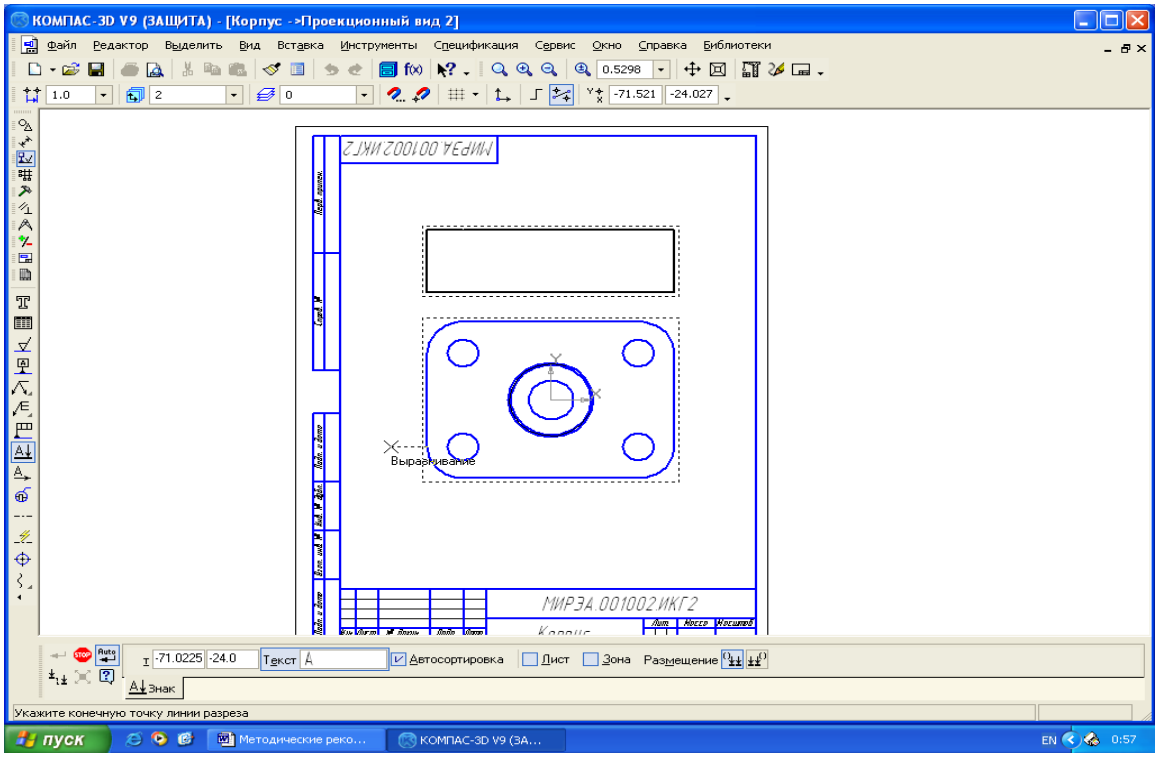


Рис. 3.40. Компактная панель с раскрытой панелью инструментов

Обозначения и панель свойств для построения линии разреза

С помощью привязки **Выравнивание** укажите точки, через которые должна пройти линия разреза. Точки должны опираться на точки центров окружностей. Сначала укажите начальную (расположенную ближе к изображению) точку первого штриха разомкнутой линии.

Затем нажмите кнопку **Сложный разрез** на **Панели свойств**.

Укажите точки перегиба и конечную точку штриха разомкнутой линии.

Еще раз нажмите кнопку **Сложный разрез** на **Панели свойств**, чтобы изменить положение стрелок.

Щелкните левой кнопкой на пустом месте чертежа. Линия разреза будет построена (рис. 3.40.1).

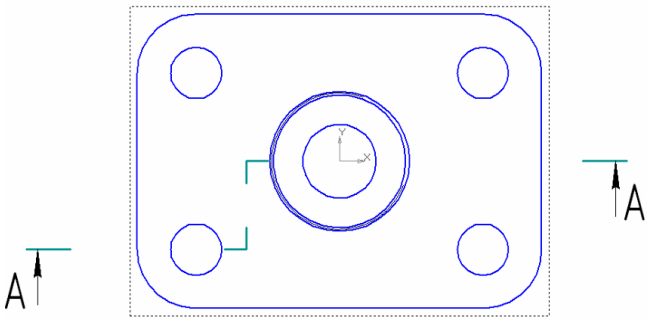


Рис. 3.40.1. Результаты построения линии разреза

Далее возможны два варианта:

1. Сразу после создания линии разреза автоматически запускается команда создания нового **Вида,** обозначение которого будет ассоциативно связано с созданной линией разреза (вы можете отказаться от создания нового вида, нажав кнопку **Прервать команду**).

Появится панель свойств **Разрез \ Сечение** с тремя вкладками**: Параметры, Линии, Штриховка.** На панели свойств можно настроить элементы разреза. Оставьте все параметры без изменений.

На экране появится фантом габаритного прямоугольника разреза **А-А,** который располагается в проекционной связи со своим опорным видом, что ограничивает его перемещение.

Переместите фантом разреза на место **Вида спереди**. Щелчком мыши зафиксируйте его.

2. Если вы выполнили команду **Ассоциативные виды** | **Разрез \ Сечение,** сделайте следующие шаги**:**

Переместите указатель мыши на любой элемент линии разреза. **Линия разреза** высветится красным цветом.

Щелкните мышью по любому элементу линии разреза. Появится фантом разреза.

Переместите фантом разреза на место **Вида спереди** и щелчком мыши зафиксируйте его.

Удалим **Вид спереди**, так как на его месте выполнен сложный ступенчатый разрез:

Выполните **Вид** | **Дерево построения.** Щелчком правой кнопки на названии вида **Спереди:** вызовите контекстное меню и выполните **Удалить вид.**

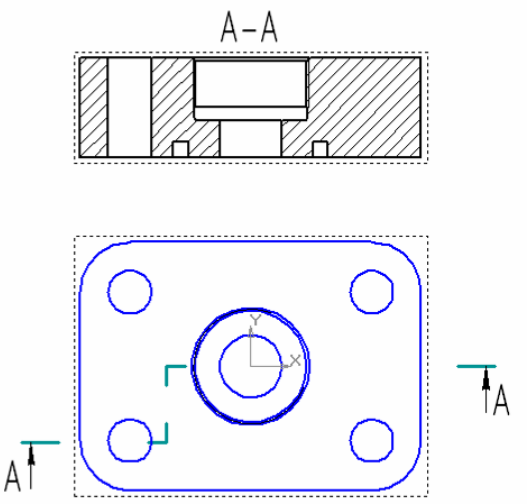


Рис. 3.41. Построение сложного ступенчатого разреза

Дополните изображения Корпуса осевыми и центровыми линиями (**Инструменты** | **Обозначения** | **Автоосевая** или **Инструменты** | **Обозначения** | **Осевая по двум точкам** или **Инструменты** | **Обозначения** | **Обозначение центра**). При нанесении осевых линий пользуйтесь такими локальными привязками, как **Середина** и т.п. Для вызова меню локальных привязок щелкните правой кнопкой мыши. В появившемся контекстном меню выполните, например, **Привязка** | **Середина.**

Не забывайте делать текущим вид, в котором вы наносите линии.

На изображениях проставьте размеры {**Инструменты** | **Размеры** | **Авторазмер**}, используя рис. 3.42 в качестве образца. При этом также не забывайте делать текущим то изображение, на котором вы ставите размеры.

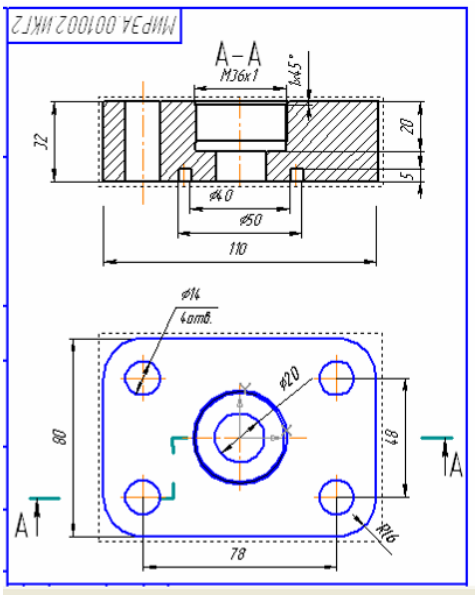


Рис. 3.42. Нанесение осевых линий и размеров на ассоциативный чертёж детали **Корпус**

Завершите заполнение основной надписи (рис. 3.43). Для заполнения основной надписи активизируйте её двойным щелчком мыши. Признаком активности является появление в ней границ ячеек с учётом заданных отступов текста. Заполните следующие ячейки основной надписи:

- В поле **Разраб.** Введите свою фамилию.
- В ячейки **Пров.** и **Утв.** введите фамилии преподавателей, ведущих занятия.
- В поле **Дата** дважды щелкните до появления диалогового окна **Ввод даты** (рис. 3.43) и опять щелкните дважды по новой дате.

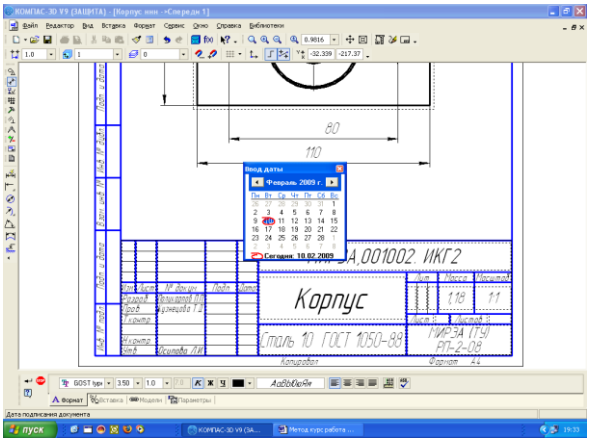


Рис. 3.43. Ввод даты

- В поле **Обозначение** введите код документа.
- В поле **Наименование изделия** вставьте его название.
- В поле **Обозначение материала детали** можно ввести обозначение материала с клавиатуры или выбрать из библиотеки.
- В поле **Наименование предприятия** введите **ВФ МАДИ** и номер группы.
- Нажмите кнопку **Создать объект** на **Панели свойств.**

После выполнения некоторых команд система может обнаружить несогласованность изображений, и габаритный прямоугольник будет перечеркнут штриховой линией. В этом случае выполните **Вид** │ **Перестроить**. На запрос системы на перестроение чертежа также необходимо давать положительный ответ.

Рабочий чертёж **Корпуса** (рис. 3.44) имеет незначительное отклонение от требований ЕСКД. Требуется отредактировать чертеж:

- необходимо скорректировать изображение резьбы в районе проточки;
- необходимо при помощи выносного элемента показать форму и размеры проточки для выхода резьбонарезного инструмента;
- в соответствии с ГОСТ 2.311-68 надо удалить резьбовую фаску на виде сверху.

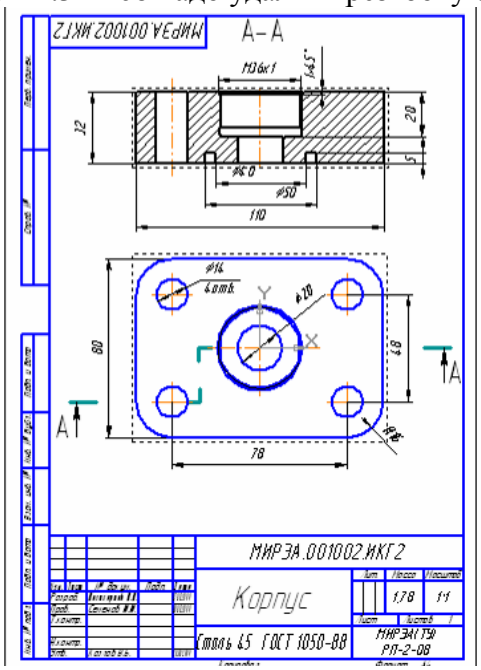


Рис. 3.44 Ассоциативный чертеж **Корпуса**

Для того, чтобы отредактировать чертёж в соответствии с требованиями ЕСКД, сохраните файл под другим именем (например, как «**Корпус разр**.») и разрушьте все ассоциативные связи. Для этого выполните: **Вид** │ **Дерево построения** │ **Проекционный вид 2.** Правой кнопкой мыши вызовите контекстное меню вида**.** Щелкните на команде **Разрушить вид.** В появившемся диалоге **Разрушить вид** подтвердите нажатием кнопки **ОК**. (рис. 3.44.1). Повторите команду для каждого вида.

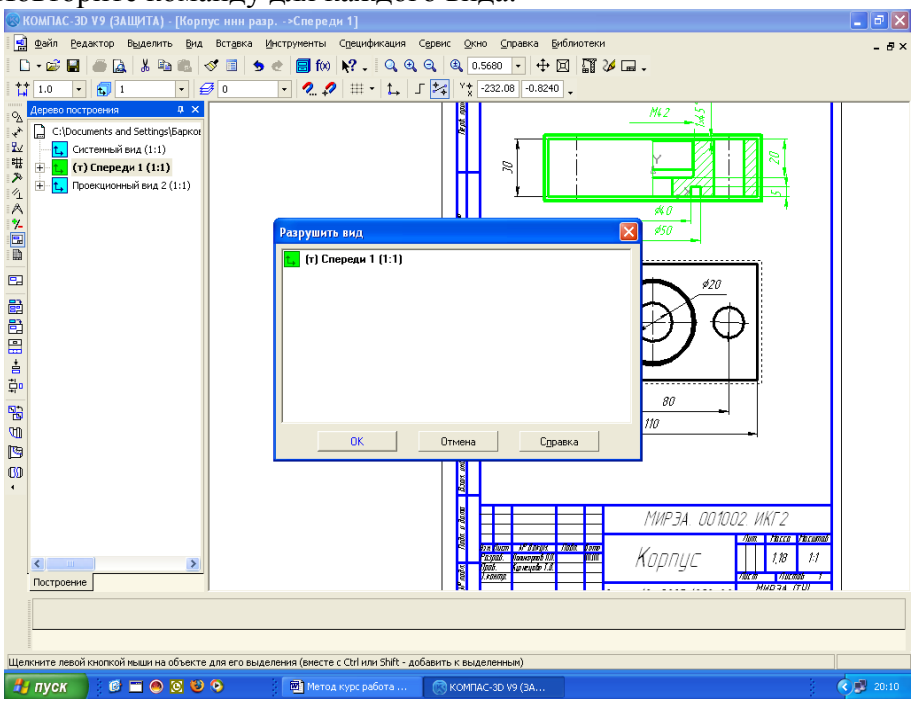


Рис. 3.44.1. Диалоговое окно **Разрушить вид**

Удалите изображение резьбовой фаски на **Виде сверху**, выделив окружность щелчком мыши и нажав **Delete.** Должно остаться изображение резьбы – три четверти дуги окружности тонкой линией.

Скорректируйте изображение резьбы (рис. 3.45).

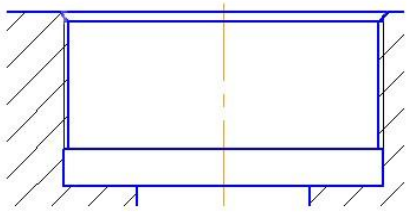


Рис. 3.45. Отредактированное изображение резьбы

Следует иметь в виду, что проточки на основных изображениях показывают упрощенно. Выполните выносной элемент, раскрывающий форму и размеры проточки для выхода резьбонарезного инструмента.

Выносной элемент выполняется следующим образом: **Менеджер библиотек** │ **Машиностроение** │ **Конструкторская библиотека** │ **Конструктивные элементы** │ **Проточки** │ **Для внутренней метрической резьбы** (рис. 3.46).

Нажмите кнопку **Менеджер библиотек** на Стандартной панели. Раскройте раздел **Машиностроение** и в правой части диалога дважды щелкните пункт **Конструкторская библиотека.** Также двойным щелчком активизируйте вкладку **Конструктивные элементы** │ **Проточки** │ **Для внутренней метрической резьбы.**

Размеры канавки выбираются в соответствии с ГОСТ 10549-80. Выбираем **Сечение с размерами.** Сам выносной элемент располагают как можно ближе к соответствующему месту предмета. Над изображением выносного элемента указывают обозначение **Инструменты** │ **Обозначения** │ **Выносной элемент** и в круглых скобках масштаб, в котором он выполнен.

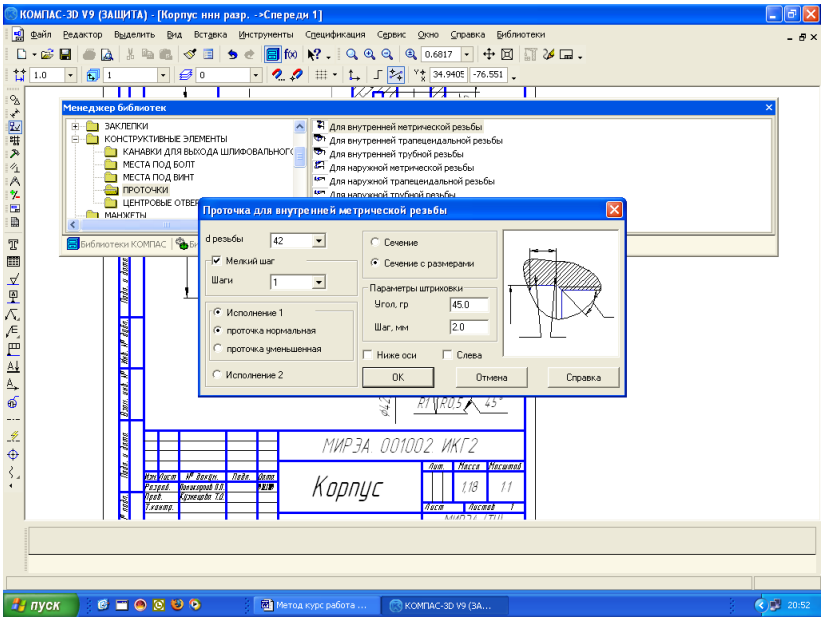


Рис. 3.46. Построение выносного элемента.

Чертеж полностью готов. Убедитесь в этом, нажав кнопку **Показать все** (рис. 3.47).

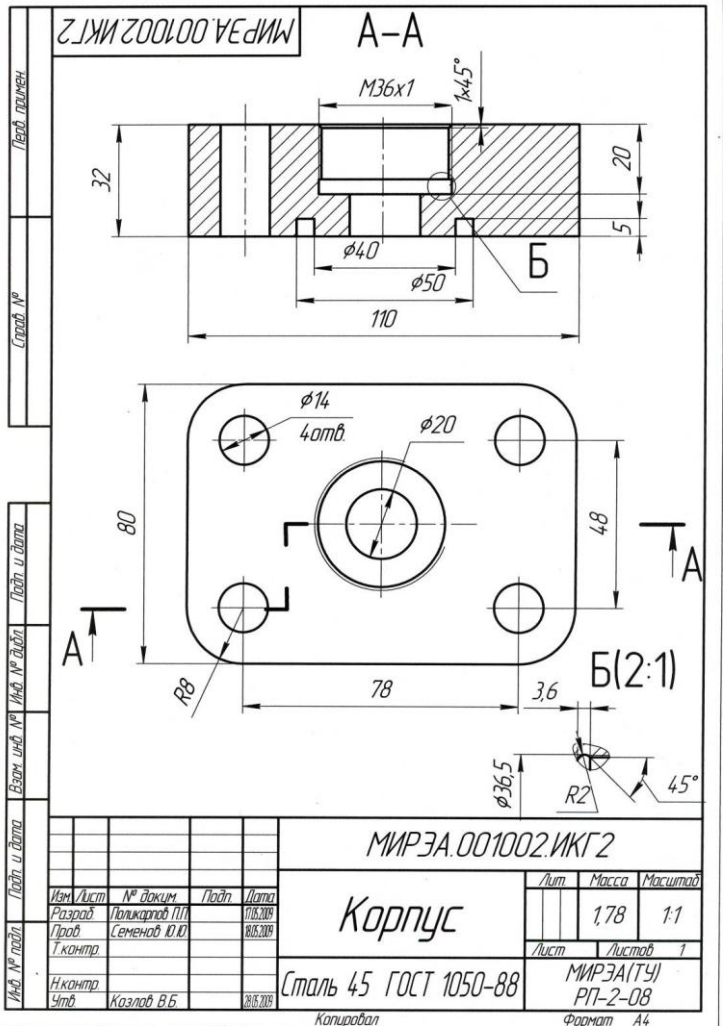
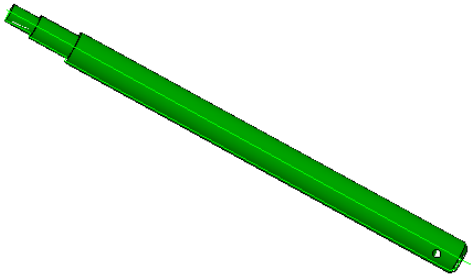
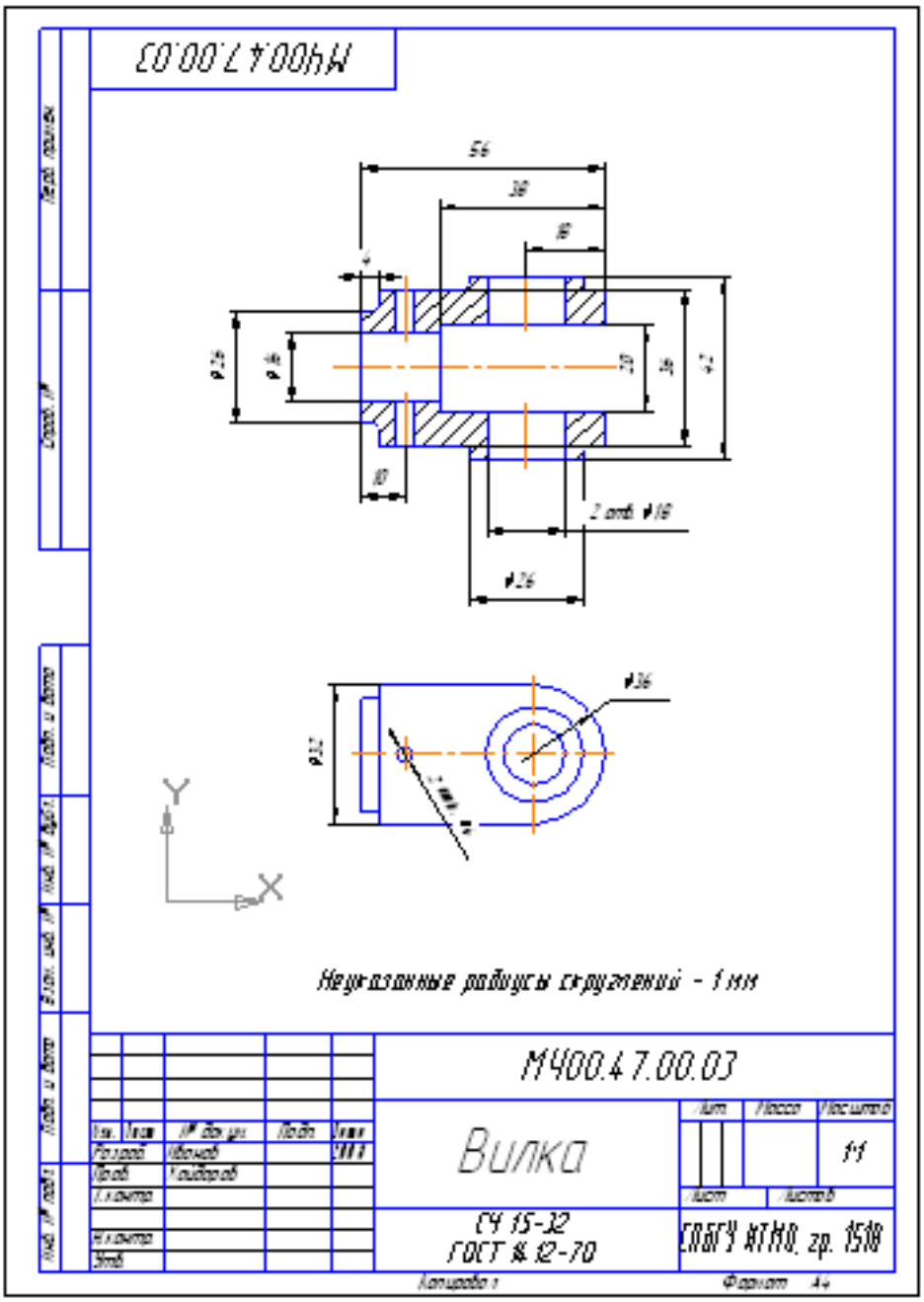


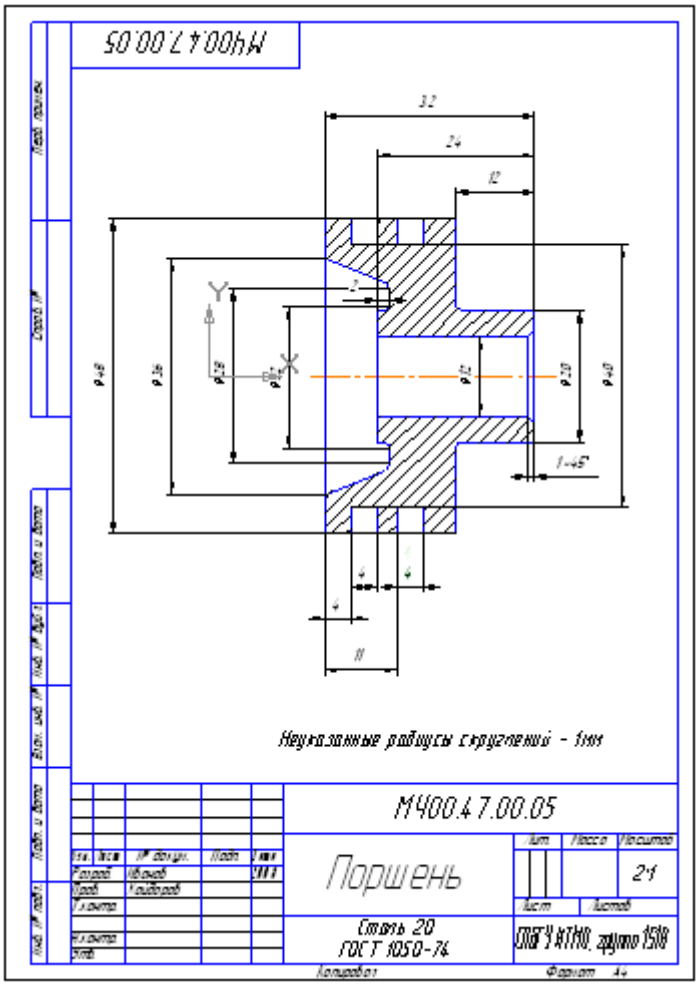
Рис. 3.47. Рабочий чертеж **Корпуса**

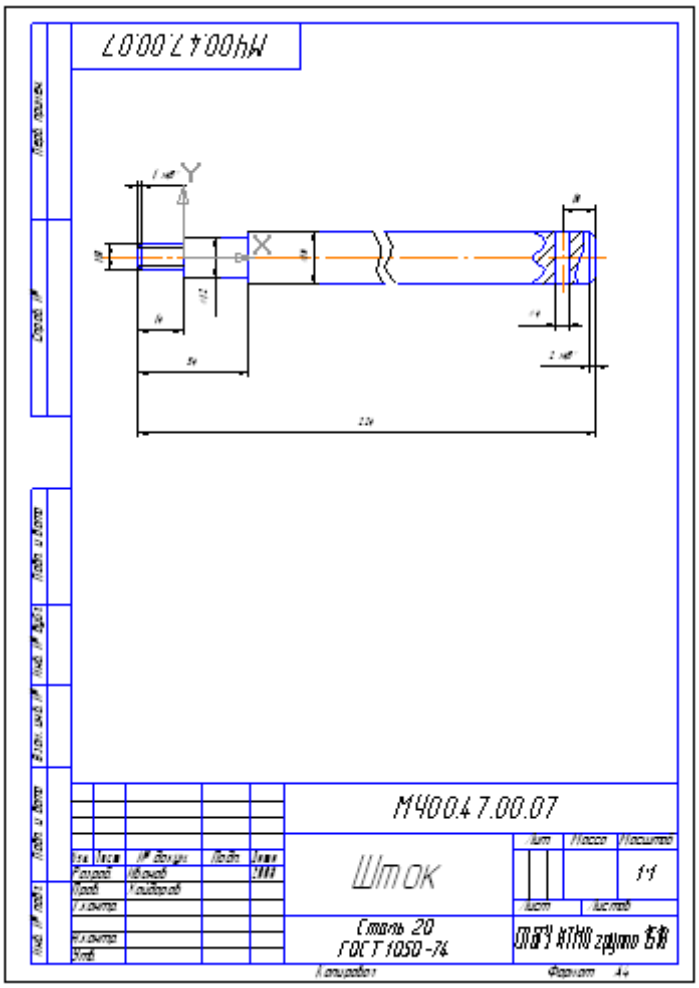
Сохраните документ.

Ниже приведены примеры выполнения лабораторной работы






М400.4.7.00.05
32
24
12
Неуказанные радиусы скруглений - 1мм
М400.4.7.00.05
Поршень
Сталь 20
ГОСТ 1050-74
2:1


М400.4.7.00.07
М400.4.7.00.07
Шток
Сталь 20
ГОСТ 1050-74
1:1

Лабораторная работа № 4 Создание сборки из трехмерных моделей, спецификации и ассоциативного сборочного чертежа в «Компас-3D»

Сборочный чертеж – документ, содержащий изображение сборочной единицы и другие данные, необходимые для ее сборки и контроля. Каждый сборочный чертеж сопровождается спецификацией.

Сборочный чертеж должен содержать:

1) изображение сборочной единицы, дающее представление о расположении и взаимной связи составных частей, соединяемых по данному чертежу;

2) сведения, обеспечивающие возможность контроля сборки;

3) указания о способе выполнения неразъемных соединений;

4) номера позиций составных частей, входящих в изделие;

5) габаритные размеры, определяющие предельные внешние очертания изделия;

6) установочные размеры, по которым изделие устанавливается на место монтажа;

7) присоединительные размеры, по которым изделие присоединяется к другим изделиям.

При выполнении сборочного чертежа обычно применяются разрезы и сечения, раскрывающие форму и расположение деталей, входящих в изделие. Правила выполнения видов, разрезов, сечений на сборочных чертежах те же, что и для обычных чертежей. В основной надписи сборочного чертежа к шифру добавляется «СБ», а ниже названия узла добавляется текст - «Сборочный чертеж».

На сборочном чертеже все составные части узла нумеруются. Номера позиций наносят на линиях полок-выносок, проводимых от изображений составных частей. Линии-выноски пересекают контур изображения и заканчиваются точкой. Линии-выноски не должны пересекаться между собой, не должны быть параллельны штриховке, не пересекать размерные линии чертежа.

Сборка в системе КОМПАС-3D – это трехмерная модель, объединяющая модели деталей, входящих в узел. Конструктор собирает узел, добавляя в него новые компоненты или удаляя существующие. В качестве примера рассмотрим построение сборки, состоящей из двух деталей: Вала (рис. 4.1) и Гайки (рис. 4.2), трехмерные модели которых были созданы заранее и сохранены в памяти компьютера.

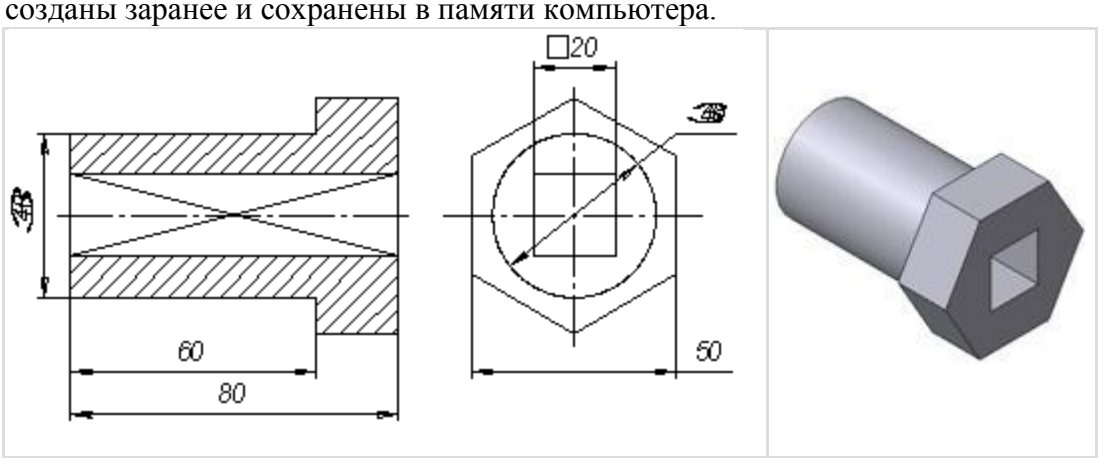


Рис. 4.1. Чертеж и модель вала

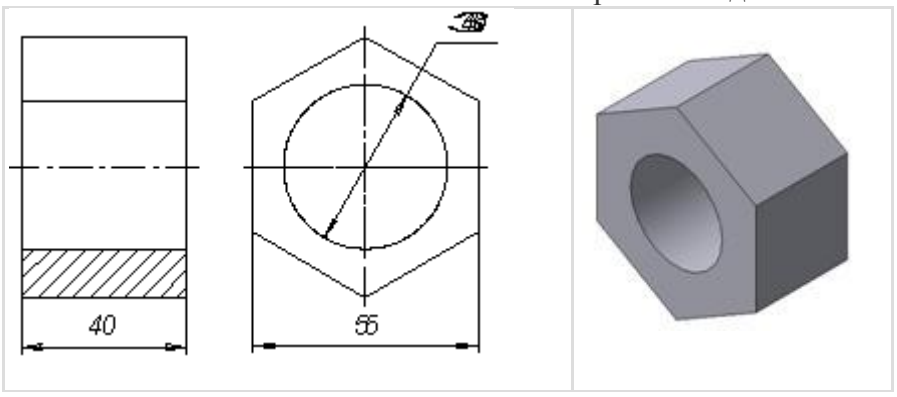


Рис. 4.2. Чертеж и модель гайки

Для того чтобы начать работу, нужно нажать кнопку «Новая сборка» на панели управления Новый документ (рис.4.3).

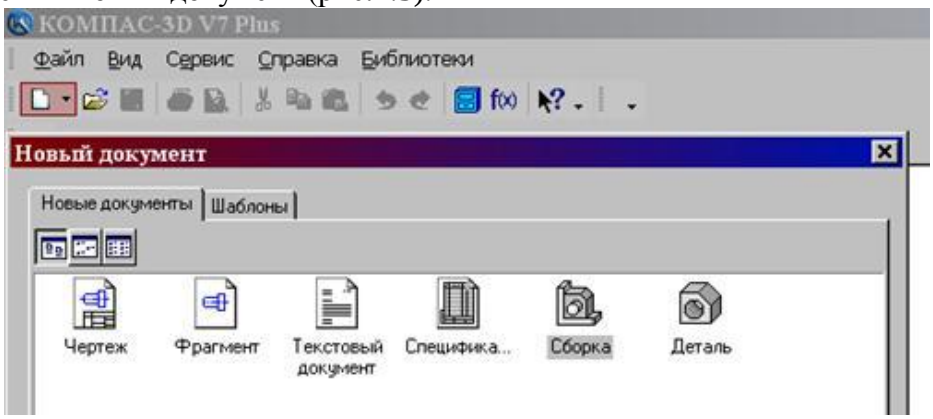


Рис. 4.3. Кнопка Новая сборка

На экране откроется окно нового документа – сборки. В окне сборки находится Дерево построения с системой координат и плоскостями проекций. На инструментальной панели появятся кнопки, управляющие процессом сборки (рис.4.4).

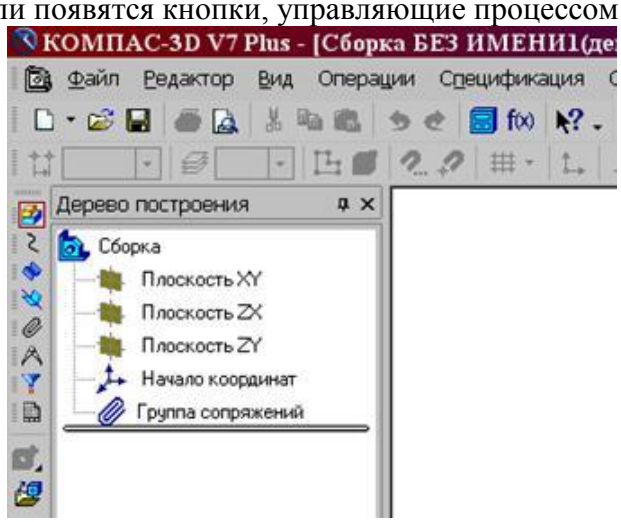


Рис. 4.4 Окно построения сборки

Добавление детали из файла

1. Чтобы вывести на экран первую деталь – вал, созданную заранее и сохраненную в памяти компьютера, нужно нажать кнопку Редактирование сборки, а затем выбрать кнопку Добавить из файла (рис.4.4).

2. В диалоге выбора файлов для открытия выберите файл Вал и нажмите кнопку Открыть (рис. 4.5).

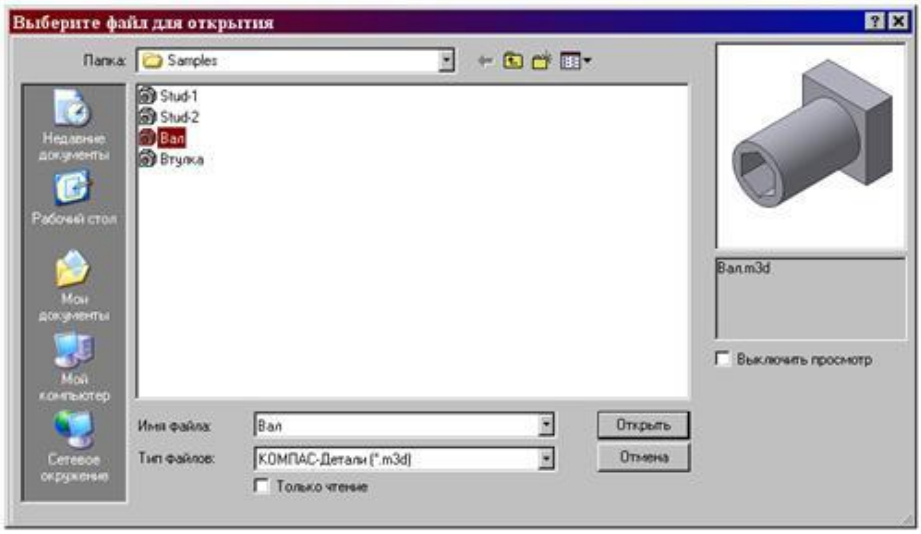


Рис.4.5. Открытие файла детали Вал

На экране появится мерцающее, свободно перемещающееся изображение вала – его фантом. Щелчком мыши закрепите вал в точке начала координат.

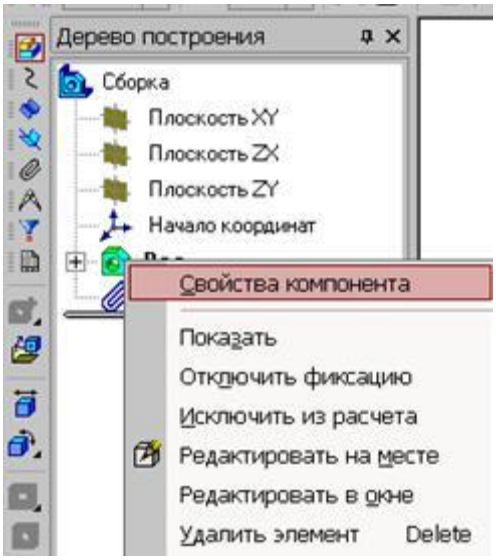


Рис.4.6 Контекстное меню детали Вал.

При необходимости вал можно расфиксировать и затем перемещать по экрану. Переключатель 

, управляющий фиксацией компонента, расположен на панели Свойств компонента (рис. 4.7), которая вызывается из контекстного меню детали Вал. Контекстное меню появляется на экране, если щелкнуть правой кнопкой мыши по имени детали - Вал в дереве построений (рис. 4.6).

Рис. 4.7 Панель свойств детали Вал

Незафиксированную деталь можно поворачивать (кнопка  Повернуть) и перемещать (кнопка  Переместить).

Затем выведите на экран вторую деталь – Втулка (рис. 4.8). Обе детали можно на экране расположить в соответствии с принятыми видами в инженерной графике (спереди, слева, сверху, в изометрии и т.д.). Для этого нужно использовать кнопку Ориентация (рис. 4.8).

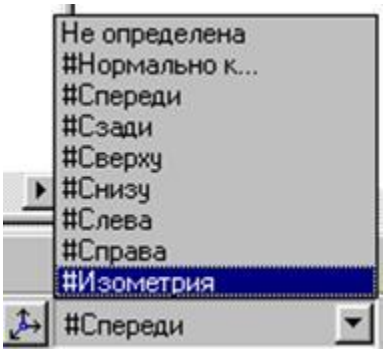


Рис. 4.8. Выбор ориентации деталей

Процессом сборки управляют кнопки, расположенные на Компактной панели Сопряжения (рис. 4.6 и 4.9). В данной лабораторной работе для осуществления сборки вал нужно вставить в отверстие гайки до соприкосновения торцевыми (боковыми) поверхностями. Вначале вал и гайку следует расположить так, чтобы их оси находились на одной прямой, т.е. детали были соосны.

Для установления соосности вала и втулки необходимо на странице Сопряжения (рис. 4.9) включить кнопку  Соосность и последовательно показать цилиндрические

поверхности вала и втулки (рис. 4.10). Втулка переместится и расположится на одной оси с валом.

Рис.4.9. Компактная панель Сопряжения

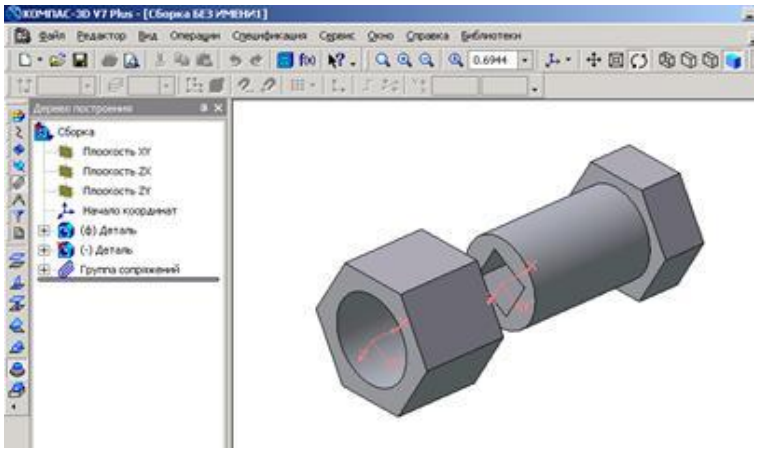
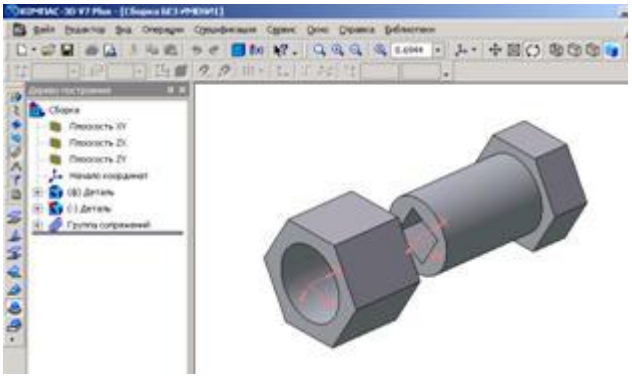


Рис. 4.10. Установление соосности деталей

Для совпадения торцевой поверхности втулки с боковой плоскостью головки вала используем кнопку Совпадение объектов (рис. 4.9) и, поворачивая деталь, последовательно указываем курсором названные плоскости (рис. 4.11) - детали займут необходимое положение (рис. 4.12).



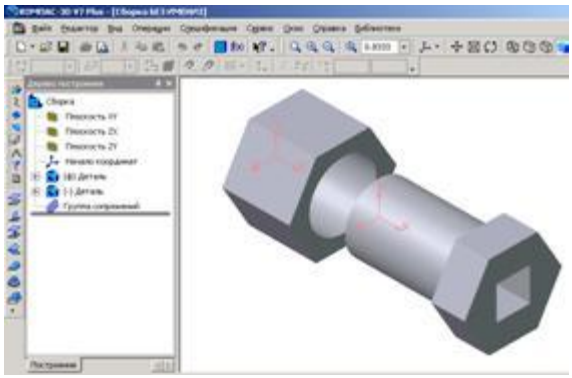


Рис. 4.11 Операция Совпадение объектов

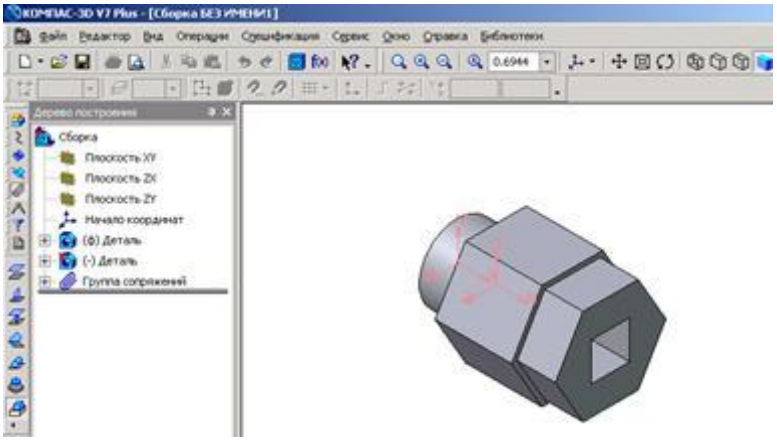


Рис.4.12 Твердотельная модель сборки

Сохраним твердотельную модель сборки под именем Вал в сборе. Обратите внимание, что файлы сборок имеют расширение *.a3d, которое система автоматически добавляет к имени документа.

Создание ассоциативного чертежа сборки

По сохраненной в памяти компьютера твердотельной модели сборки создадим ассоциативный сборочный чертеж. Ориентацию сборки, в которой она будет изображена на главном виде, можно выбирать из раскрывающегося списка Ориентация главного вида (рис. 4.13) на вкладке Параметры Панели свойств. Список содержит ориентации, соответствующие стандартным видам - спереди, сзади, сверху, снизу, слева, справа, изометрия.

Рис. 4.13 Выбор ориентации сборки на главном виде

Выберем ориентацию главного вида – слева, с помощью кнопки Схема видов оставим только главный вид (рис. 4.14).

Ассоциативный разрез

Чтобы показать внутреннее строение деталей сборки, выполним ассоциативный разрез:

1. Установив привязку Выравнивание, зададим траекторию разреза А-А (рис. 4.14);

2. Включив кнопку Разрез/Сечение на странице Ассоциативный вид, укажем курсором линию разреза А-А. На экране появится перемещающийся фантом разреза в виде прямоугольника, зафиксируйте его в нужном месте (рис.4.15).

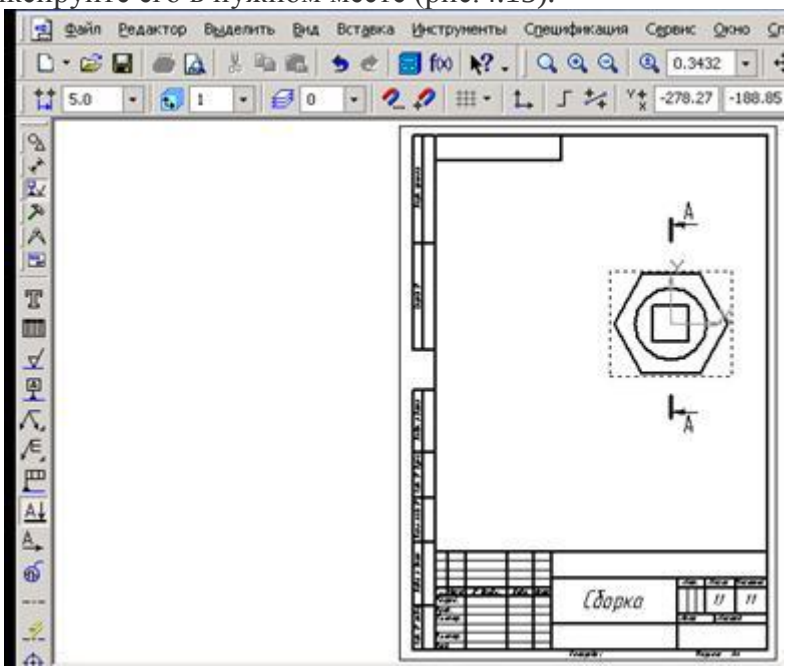


Рис. 4.14 Положение секущей плоскости на виде слева

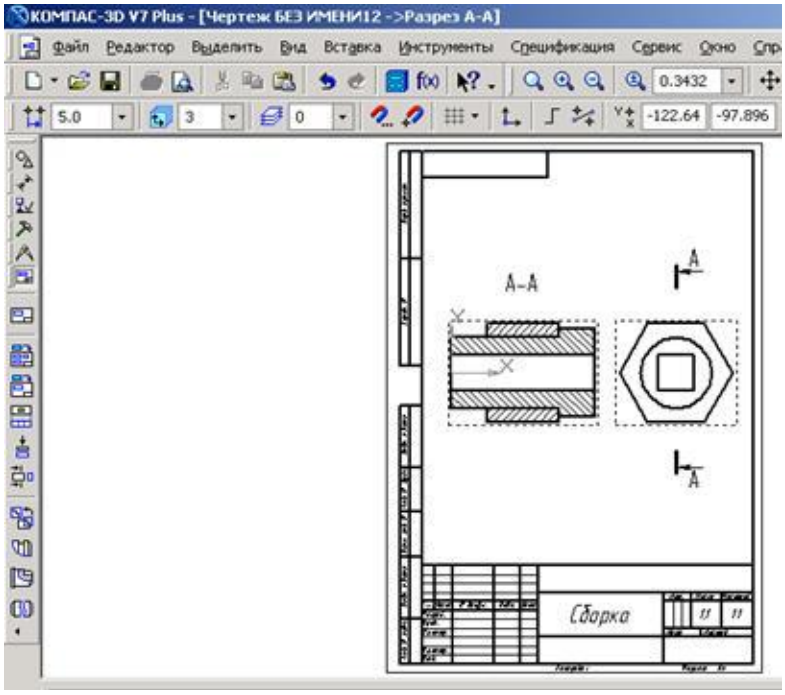


Рис. 4.15. Построение ассоциативного разреза

В соответствии с ГОСТ 2.307-68 разрез, образованный секущей плоскостью, совпадающей с плоскостью симметрии детали (когда отсекается половина или четверть детали), на чертеже не обозначается. В построенном разрезе на рис.158 обозначение разреза следует удалить.

Удаление обозначения разреза

В ассоциативном чертеже вид и разрез связаны между собой, если стереть обозначение разреза, то сотрется и сам разрез, поэтому нужно разрушить связь между различными элементами в изображении разреза и видом спереди:

1. Выделить построенный разрез, щелкнув по рамке вида левой кнопкой мыши;
2. Правой кнопкой мыши вызвать контекстное меню, в котором включить пункт Разрушить вид;
3. Стереть обозначение разреза.

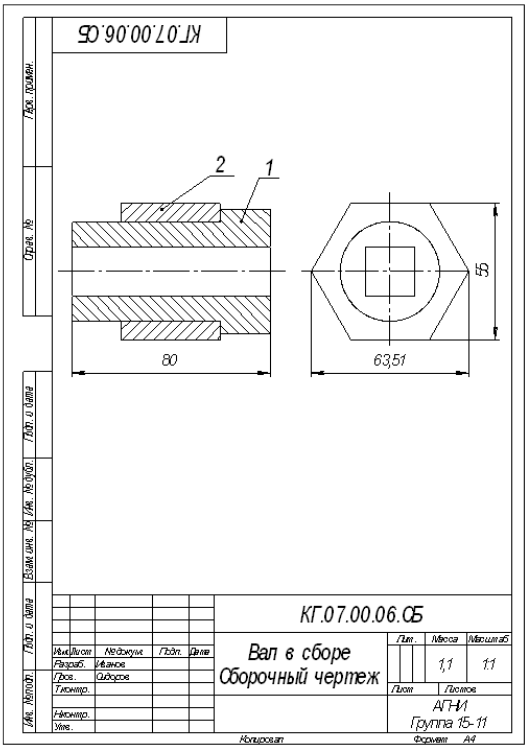


Рис. 4.16 Сборочный чертеж

Построенный ассоциативный сборочный чертеж (рис. 4.16) следует оформить:

- провести необходимые оси симметрии;

- нанести габаритные размеры;
- обозначить позиции;
- заполнить основную надпись.

Завершается работа по созданию сборочного чертежа оформлением спецификации (рис. 4.17).

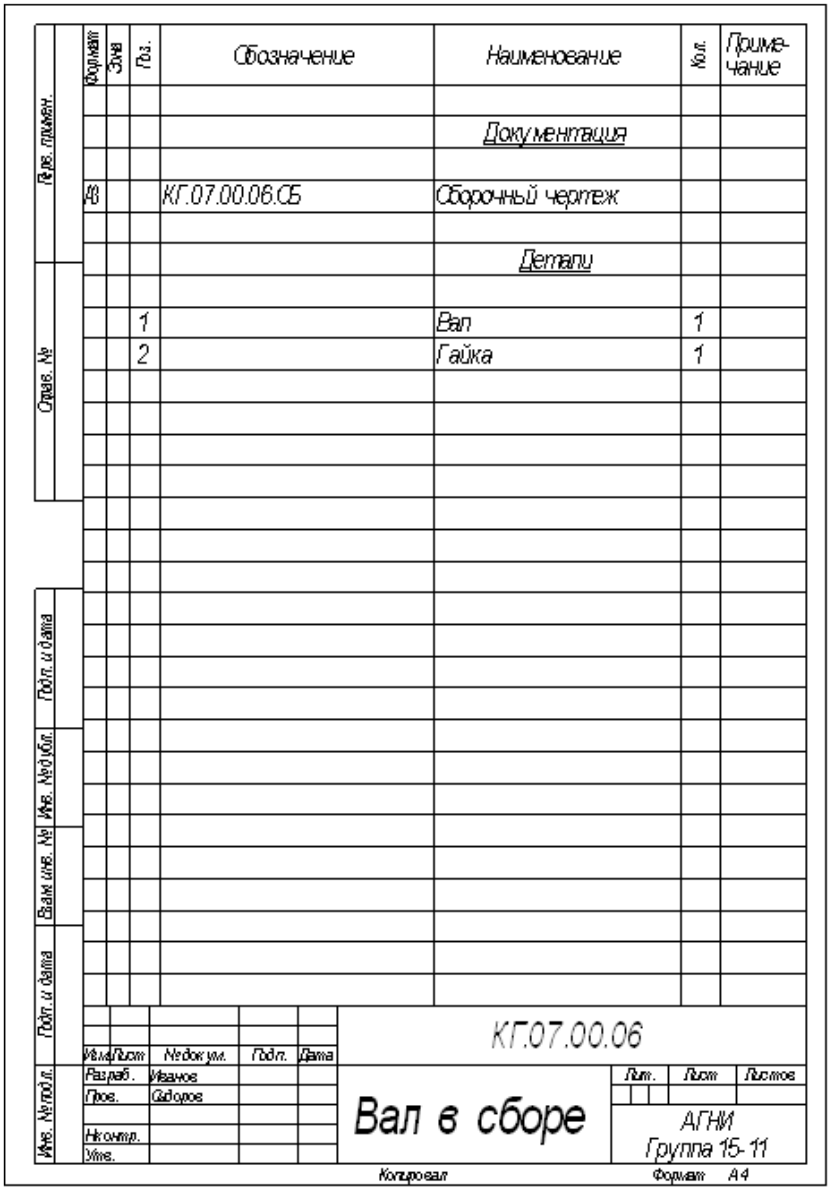
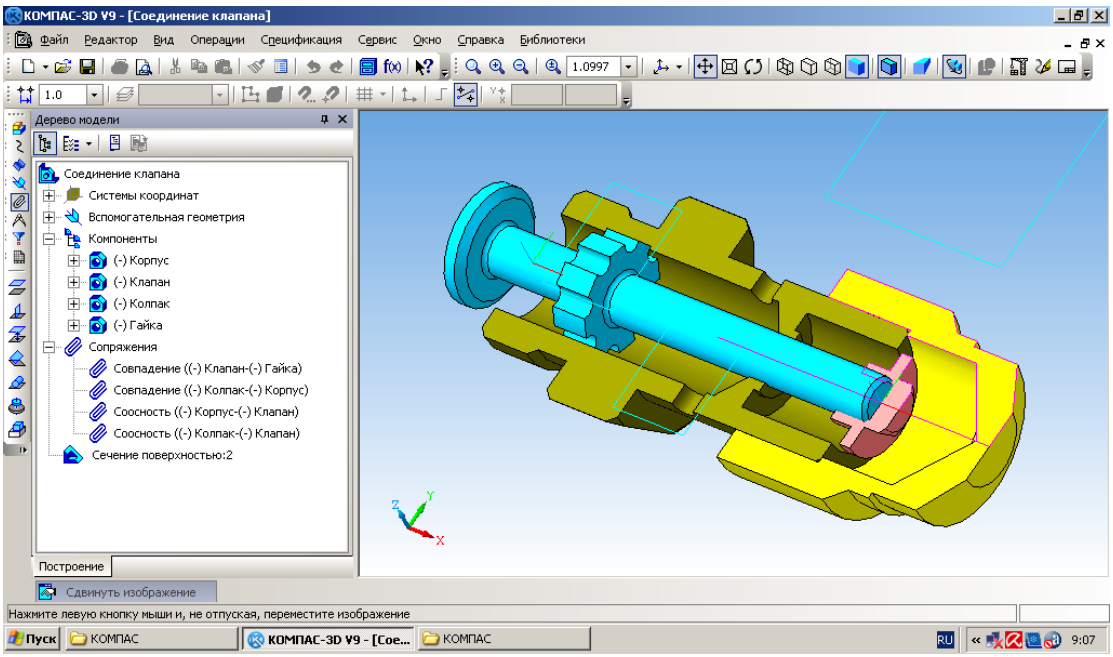
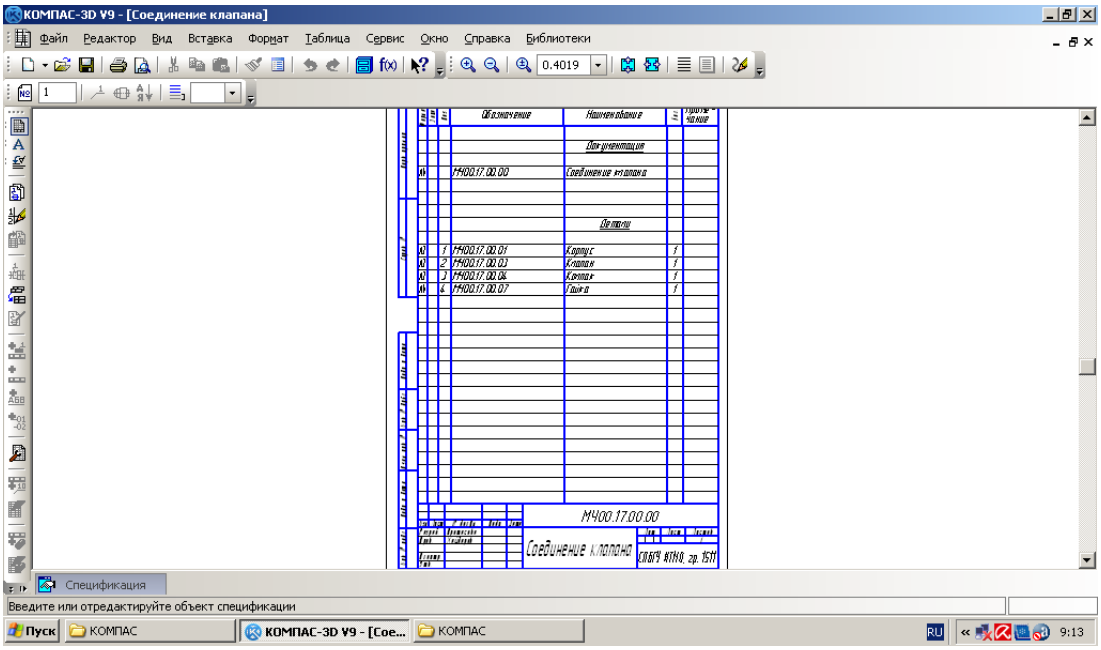
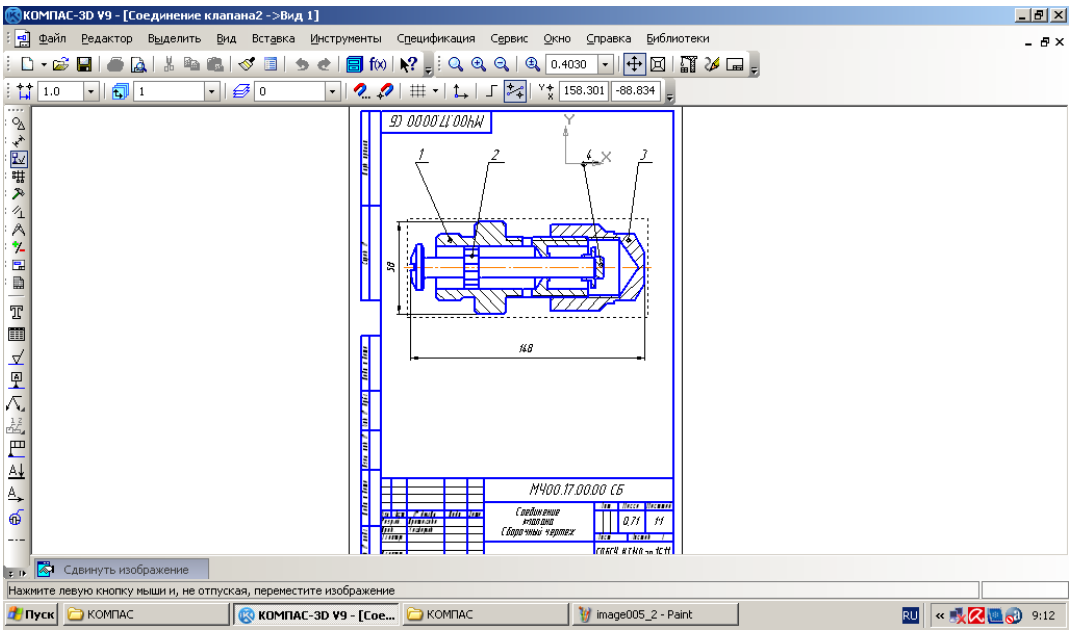


Рис. 4.17 Спецификация к сборочному чертежу

Ниже приведен вариант выполнения лабораторной работы







Лабораторная работа № 5. Создание анимации на языке C#
Сделать анимацию
Программа:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private int dx;
      private int i;//номер картинки
      //картинки
      private Bitmap myBitmap2;
      private Bitmap myBitmap3;
      private Bitmap myBitmap4;
      private Bitmap myBitmap5;
      private Bitmap myBitmap6;
      private Bitmap myBitmap7;
      private Bitmap myBitmap8;
      private Bitmap myBitmap9;
      private Bitmap myBitmap10;
      private Bitmap myBitmap11;

      private void button3_Click(object sender, EventArgs e)
      {
         timer1.Enabled = false;
      }

      private void button2_Click(object sender, EventArgs e)
      {
         timer1.Enabled = true;
         dx = 10;//приращение
         i = 0;
         pictureBox2.Visible = true;//вывод изображения


      }

      private void timer1_Tick(object sender, EventArgs e)
      {

         i += 1;
```

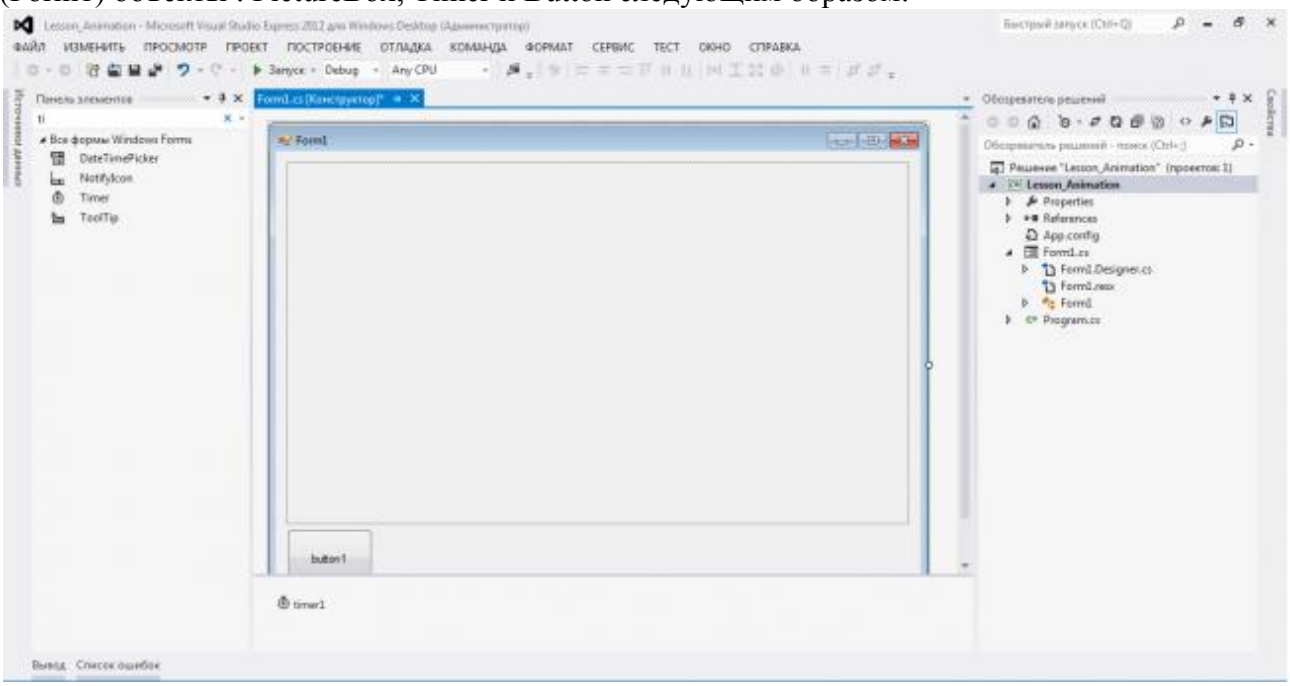
```
            if (dx > 0)//если вправо
            {
                if (i == 5) i = 1;
                if (pictureBox2.Left >= pictureBox1.Width) pictureBox2.Left = -100;//выход с
другой стороны экрана
                //вывод картинок
                if (i == 1) pictureBox2.Image = myBitmap2;
                if (i == 2) pictureBox2.Image = myBitmap3;
                if (i == 3) pictureBox2.Image = myBitmap4;
                if (i == 4) pictureBox2.Image = myBitmap5;
                if (i == 5) pictureBox2.Image = myBitmap6;

                pictureBox2.Left += dx;//движение по горизонтали
            }
            if (dx < 0)//если влево
            {
                if (i == 10) i = 6;
                if (pictureBox2.Left <= -100) pictureBox2.Left = pictureBox1.Width;//выход с
другой стороны экрана
                //вывод картинок
                if (i == 6) pictureBox2.Image = myBitmap7;
                if (i == 7) pictureBox2.Image = myBitmap8;
                if (i == 8) pictureBox2.Image = myBitmap9;
                if (i == 9) pictureBox2.Image = myBitmap10;
                pictureBox2.Left += dx;//движение по горизонтали
            }

        }

        private void Form1_Load(object sender, EventArgs e)
        {

            pictureBox1.Refresh();
            //загрузка картинок
            myBitmap2 = new Bitmap(System.IO.Directory.GetCurrentDirectory() + "\\1.png");
            myBitmap3 = new Bitmap(System.IO.Directory.GetCurrentDirectory() + "\\2.png");
            myBitmap4 = new Bitmap(System.IO.Directory.GetCurrentDirectory() + "\\3.png");
            myBitmap5 = new Bitmap(System.IO.Directory.GetCurrentDirectory() + "\\4.png");
            myBitmap6 = new Bitmap(System.IO.Directory.GetCurrentDirectory() + "\\5.png");
            myBitmap7 = new Bitmap(System.IO.Directory.GetCurrentDirectory() + "\\6.png");
            myBitmap8 = new Bitmap(System.IO.Directory.GetCurrentDirectory() + "\\7.png");
            myBitmap9 = new Bitmap(System.IO.Directory.GetCurrentDirectory() + "\\8.png");
            myBitmap10 = new Bitmap(System.IO.Directory.GetCurrentDirectory() +
"\\9.png");
            myBitmap11 = new Bitmap(System.IO.Directory.GetCurrentDirectory() +
"\\10.png");
            //начальное положение
            pictureBox2.Top = pictureBox1.Height / 5 + 10;
            pictureBox2.Left = pictureBox1.Width / 10;
            trackBar1.Value = 3;

        }

        private void button1_Click(object sender, EventArgs e)
        {
            timer1.Enabled = true;
```

```
            dx = -10;
            i = 5;
            pictureBox2.Visible = true;

        }

        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            switch (trackBar1.Value)
            {
                case 0: timer1.Interval = 350; break;
                case 1: timer1.Interval = 300; break;
                case 2: timer1.Interval = 250; break;
                case 3: timer1.Interval = 200; break;
                case 4: timer1.Interval = 150; break;
                case 5: timer1.Interval = 100; break;
                case 6: timer1.Interval = 75; break;
            }

        }

        private void button4_Click(object sender, EventArgs e)
        {
            Form2 myForm = new Form2();
            myForm.Show();

        }

        private void button5_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Результат:

В данном уроке используется среда программирования Microsoft Visual studio 2012. Алгоритм работы аналогичен и для других сред программирования. Перед тем, как приступить к данному уроку, следует ознакомиться с видеоуроком : Как начать работать с графикой в Microsoft C#

Сначала создаем свой проект, в котором и будем работать. Разместим на форме (Form1) объекты : PictureBox, Timer и Button следующим образом:

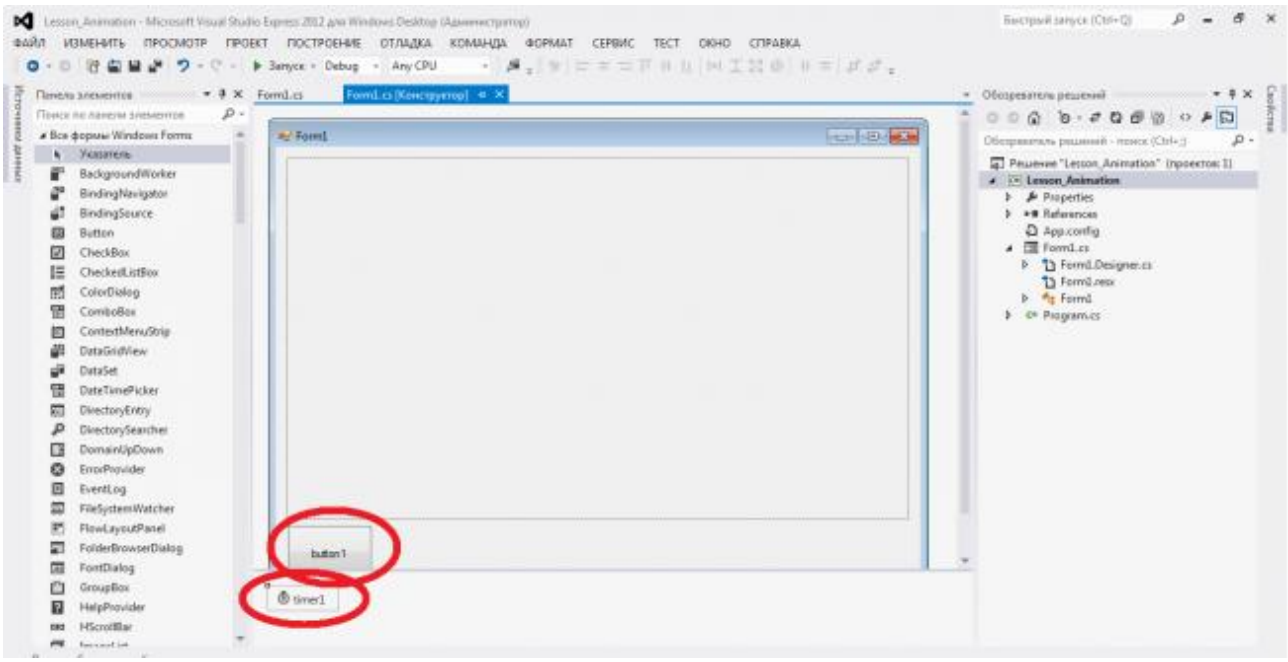


Теперь попробуем создать какую-нибудь примитивную анимацию, при помощи PictureBox и Timer, которая запустится после нажатия на кнопку (Button). Следовательно для этого будем обрабатывать событие нажатия на кнопку и событие "срабатывания" таймера. Также заведем все нужные для рисования объекты и переменные. Далее приведен код программы и скриншоты, которые содержат все необходимое пояснение реализации анимации.

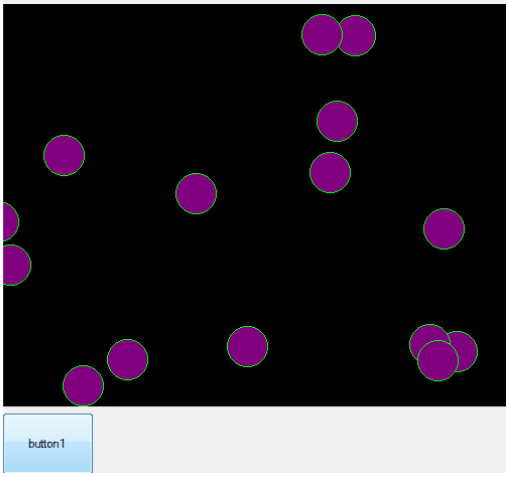
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lesson_Animation
{
    public partial class Form1 : Form
    {
        Graphics gr;            //объявляем объект - графику, на
которой будем рисовать
        Pen p;              //объявляем объект - карандаш, которым
будем рисовать контур
        SolidBrush fon;         //объявляем объект - заливки, для
заливки соответственно фона
        SolidBrush fig;     //и внутренности рисуемой фигуры


        int rad;     // переменная для хранения радиуса рисуемых
кругов
        Random rand;       // объект, для получения случайных чисел
```

```
        public Form1()
        {
            InitializeComponent();
        }

        //опишем функцию, которая будет рисовать круг по
координатам его центра
        void DrawCircle(int x, int y)
        {
            int xc, yc;
            xc = x - rad;
            yc = y - rad;
            gr.FillEllipse(fig, xc, yc, rad, rad);

            gr.DrawEllipse(p, xc, yc, rad, rad);

        }

        // для перехода к данной функции сделайте двойной щелчок
по кнопке (Button)
        // добавленной на форму. См. на фото, после кода
        private void button1_Click(object sender, EventArgs e)
        {
            gr  =  pictureBox1.CreateGraphics();    //инициализируем
объект типа графики
                                                    //   привязав     к
PictureBox

            p = new Pen(Color.Lime);             // задали цвет для
карандаша
            fon = new SolidBrush(Color.Black); // и для заливки
            fig = new SolidBrush(Color.Purple);

            rad = 40;                           //задали радиус для
круга
            rand = new Random();                  //инициализируем
объект для рандомных числе

            gr.FillRectangle(fon,    0,    0,    pictureBox1.Width,
pictureBox1.Height); // закрасим черным

// нашу область рисования

            // вызываем написанную нами функцию, для прорисовки
круга
            // случайным образом выбрав перед этим координаты
центра
            int x,  y;

            for (int i = 0; i < 15; i++)
            {
                x = rand.Next(pictureBox1.Width);
                y = rand.Next(pictureBox1.Height);
                DrawCircle(x, y);
            }
```

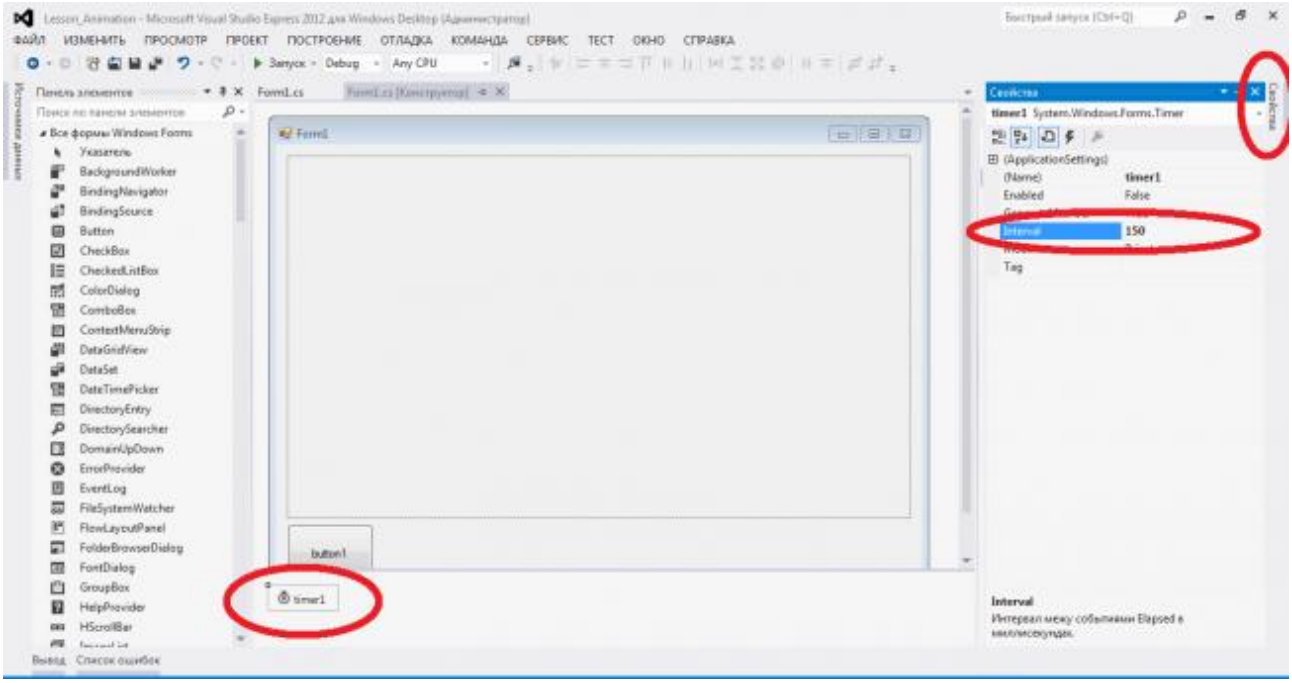
```
            timer1.Enabled = true;  //включим в работу наш таймер,
            // то есть теперь будет происходить событие Tick и его
будет обрабатывать функция On_Tick (по умолчанию)

        }

        // для получения данной функции перейдите к конструктору
формы
        // и сделайте двойной щелчок по таймеру, добавленному на
форму. См. на фото после кода
        private void timer1_Tick(object sender, EventArgs e)
        {

            //сначала будем очищать область рисования цветом фона
            gr.FillRectangle(fon,    0,    0,    pictureBox1.Width,
pictureBox1.Height);


            // затем опять случайным образом выбираем координаты
центров кругов
            // и рисуем их при помощи описанной нами функции
            int x, y;

            for (int i = 0; i < 15; i++)
            {
                x = rand.Next(pictureBox1.Width);
                y = rand.Next(pictureBox1.Height);
                DrawCircle(x, y);
            }

        }
    }
}
```



После этого, можно запустить программу и после нажатия на кнопку увидите простую анимацию – случайное перемещение кругов по черному фону, как показано ниже:



Для того, чтобы анимация соответствовала требованиям иногда необходимо менять так называемый тик таймера, т.е. промежуток выполнения очередного шага анимации. Это выполняется в Инспекторе объектов. Нужно выбрать элемент Timer, нажать на кнопку Свойства и там выбрать и изменить параметр Interval (выражается в миллисекундах) В данном примере Interval равен 150 мс.



Лабораторная работа № 6. Создание трехмерной модели на управляемой трехмерной сцене на языке C#

Сейчас трёхмерные изображения можно увидеть везде, начиная от компьютерных игр и заканчивая системами моделирования в реальном времени. Раньше, когда трёхмерная графика существовала только на суперкомпьютерах, не существовало единого стандарта в области графики. Все программы писались с «нуля» или с использованием накопленного опыта, но в каждой программе реализовывались свои методы для отображения графической информации. С приходом мощных процессоров и графических ускорителей трёхмерная графика стала реальностью для персональных компьютеров. Но в тоже время производители программного обеспечения столкнулись с серьёзной проблемой – это отсутствие каких-либо стандартов, которые позволяли писать программы, независимые от оборудования и операционной системы. Одним из первых таких стандартов, существующий и по сей день является OpenGL.

OpenGL – это графический стандарт в области компьютерной графики. На данный момент он является одним из самых популярных графических стандартов во всём мире. Ещё в 1982 г. в Стенфордском университете была разработана концепция графической машины, на основе которой фирма Silicon Graphics в своей рабочей станции Silicon IRIS реализовала конвейер рендеринга. Таким образом была разработана графическая библиотека IRIS GL. На основе библиотеки IRIS GL, в 1992 году был разработан и утверждён графический стандарт OpenGL. Разработчики OpenGL – это крупнейшие фирмы разработчики как оборудования так и программного обеспечения: Silicon Graphics, Inc., Microsoft, IBM Corporation, Sun Microsystems, Inc., Digital Equipment Corporation (DEC), Evans & Sutherland, Hewlett-Packard Corporation, Intel Corporation и Intergraph Corporation.

OpenGL переводится как Открытая Графическая Библиотека (Open Graphics Library), это означает, что OpenGL – это открытый и мобильный стандарт. Программы, написанные с помощью OpenGL можно переносить практически на любые платформы, получая при этом одинаковый результат, будь это графическая станция или суперкомпьютер. OpenGL освобождает программиста от написания программ для конкретного оборудования. Если устройство поддерживает какую-то функцию, то эта функция выполняется аппаратно, если нет, то библиотека выполняет её программно.

Что же представляет из себя OpenGL? С точки зрения программиста OpenGL – это программный интерфейс для графических устройств, таких как графические ускорители. Он включает в себя около 150 различных команд, с помощью которых программист может определять различные объекты и производить рендеринг. Говоря более простым языком, вы определяете объекты, задаёте их местоположение в трёхмерном пространстве, определяете другие параметры (поворот, масштаб,…), задаёте свойства объектов (цвет, текстура, материал,…), положение наблюдателя, а библиотека OpenGL позаботится о том чтобы отобразить всё это на экране. Поэтому можно сказать, что библиотека OpenGL является только воспроизводящей (Rendering), и занимается только отображением 3D обьектов, она не работает с устройствами ввода (клавиатуры, мыши). Также она не поддерживает менеджер окон.

OpenGL имеет хорошо продуманную внутреннюю структуру и довольно простой процедурный интерфейс. Несмотря на это с помощью OpenGL можно создавать сложные и мощные программные комплексы, затрачивая при этом минимальное время по сравнению с другими графическими библиотеками.

В некоторых библиотеках OpenGL (например под X Windows) имеется возможность изображать результат не только на локальной машине, но также и по сети. Приложение, которое вырабатывает команды OpenGL называется клиентом, а приложение, которое получает эти команды и отображает результат – сервером. Таким образом, можно строить очень мощные воспроизводящие комплексы на основе нескольких рабочих станций или серверов, соединённых сетью.

Программирование с использованием библиотеки OpenGl

Основные возможности

Геометрические и растровые примитивы. На основе геометрических и растровых примитивов строятся все объекты. Из геометрических примитивов библиотека предоставляет: точки, линии, полигоны. Из растровых: битовый массив(bitmap) и образ(image)

Использование B-сплайнов. B-сплайны используются для рисования кривых по опорным точкам.

Видовые и модельные преобразования. С помощью этих преобразований можно располагать объекты в пространстве, вращать их, изменять форму, а также изменять положение камеры из которой ведётся наблюдение.

Работа с цветом. OpenGL предоставляет программисту возможность работы с цветом в режиме RGBA (красный-зелёный-синий-альфа) или используя индексный режим, где цвет выбирается из палитры.

Удаление невидимых линий и поверхностей. Z-буферизация.

Двойная буферизация. OpenGL предоставляет как одинарную так и двойную буферизацию. Двойная буферизация используется для того, чтобы устранить мерцание при мультипликации, т.е. изображение каждого кадра сначала рисуется во втором(невидимом) буфере, а потом, когда кадр полностью нарисован, весь буфер отображается на экране.

Наложение текстуры. Позволяет придавать объектам реалистичность. На объект, например шар, накладывается текстура (просто какое-то изображение), в результате чего наш объект теперь выглядит не просто как шар, а как разноцветный мячик.

Сглаживание. Сглаживание позволяет скрыть ступенчатость, свойственную растровым дисплеям. Сглаживание изменяет интенсивность и цвет пикселей около линии, при этом линия смотрится на экране без всяких зигзагов.

Освещение. Позволяет задавать источники света, их расположение, интенсивность, и т.д.

Атмосферные эффекты. Например туман, дым. Всё это также позволяет придать объектам или сцене реалистичность, а также «почувствовать» трёхмерное изображение.

Прозрачность объектов.

Использование списков изображений.

Работа с матрицами

Для задания различных преобразований объектов сцены в OpenGL используются операции над матрицами, при этом различают три типа матриц: модельно-видовая, матрица проекций и матрица текстуры. Все они имеют размер 4x4. Видовая матрица определяет преобразования объекта в мировых координатах, такие как параллельный перенос, изменение масштаба и поворот. Матрица проекций определяет, как будут проецироваться трехмерные объекты на плоскость экрана (в оконные координаты), а матрица текстуры определяет наложение текстуры на объект.

Умножение координат на матрицы происходит в момент вызова соответствующей команды OpenGL, определяющей координату (как правило, это команда glVertex*.

Для того чтобы выбрать, какую матрицу надо изменить, используется команда: void glMatrixMode (GLenum mode), вызов которой, со значением параметра «mode» равным GL_MODELVIEW, GL_PROJECTION, или GL_TEXTURE включает режим работы с модельно-видовой матрицей, матрицей проекций, или матрицей текстуры соответственно. Для вызова команд, задающих матрицы того или иного типа, необходимо сначала установить соответствующий режим.

Для определения элементов матрицы текущего типа вызывается команда void glLoadMatrix [f d] (GLtype *m), где «m» указывает на массив из 16 элементов типа float или double в соответствии с названием команды, при этом сначала в нем должен быть записан первый столбец матрицы, затем второй, третий и четвертый. Еще раз следует обратить внимание, в массиве «m» матрица записана по столбцам.

Команда void glLoadIdentity(void) заменяет текущую матрицу на единичную.

Синтаксис команд

Определения команд GL находятся в файле gl.h, для включения которого нужно написать#include <gl/gl.h>

Для работы с библиотекой GLU нужно аналогично включить файл glu.h. Версии этих библиотек, как правило, включаются в дистрибутивы систем программирования, например Microsoft Visual C++ или Borland C++ 5.02.

В отличие от стандартных библиотек, пакет GLUT нужно инсталлировать и подключать отдельно. Подробная информация о настройке сред программирования для работы с OpenGL дана в Приложении С.

Все команды (процедуры и функции) библиотеки GL начинаются с префикса gl, все константы – с префикса GL_. Соответствующие команды и константы библиотек GLU и GLUT аналогично имеют префиксы glu (GLU_) и glut (GLUT_)

Кроме того, в имена команд входят суффиксы, несущие информацию о числе и типе передаваемых параметров. В OpenGL полное имя команды имеет вид: type glCommand_name [1 2 3 4] [b s i f d ub us ui] [v] (type1 arg1,…, typeN argN)

Таким образом, имя состоит из нескольких частей: Gl это имя библиотеки, в которой описана эта функция: для базовых функций OpenGL, функций из библиотек GLU, GLUT, GLAUX это gl, glu, glut, glaux соответственно

Command_name имя команды

[1 2 3 4] число аргументов команды

[b s i f d ub us ui] тип аргумента: символ b означает тип GLbyte (аналог char в C\C++), символ f – тип GLfloat (аналог float), символ i – тип GLint (аналог int) и так далее. Полный список типов и их описание можно посмотреть в файле gl.h

[v] наличие этого символа показывает, что в качестве параметров функции используется указатель на массив значений

Символы в квадратных скобках в некоторых названиях не используются. Например, команда glVertex2i() описана как базовая в библиотеке OpenGL, и использует в качестве параметров два целых числа, а команда glColor3fv() использует в качестве параметра указатель на массив из трех вещественных чисел.

Использования нескольких вариантов каждой команды можно частично избежать, применяя перегрузку функций языка C++. Но интерфейс OpenGL не рассчитан на конкретный язык программирования, и, следовательно, должен быть максимально универсален.

Освещение

В OpenGL используется модель освещения, в соответствии с которой цвет точки определяется несколькими факторами: свойствами материала и текстуры, величиной нормали в этой точке, а также положением источника света и наблюдателя. Для корректного расчета освещенности в точке надо использовать единичные нормали, однако команды: типа glScale*(), могут изменять длину нормалей. Чтобы это учитывать, нужно использовать режим нормализации векторов нормалей, который включается вызовом команды glEnable (GL_NORMALIZE).

Для задания глобальных параметров освещения используются команды void glLightModel [i, f] (GLenum pname, GLenum param) и void glLightModel [i f] v (GLenum pname, const GLtype *params).

Аргумент «pname» определяет, какой параметр модели освещения будет настраиваться и может принимать следующие значения: GL_LIGHT_MODEL_LOCAL_VIEWER, параметр «param» должен быть булевым и задает положение наблюдателя. Если он равен GL_FALSE, то направление обзора считается параллельным оси z, вне зависимости от положения в видовых координатах. Если же он равен GL_TRUE, то наблюдатель находится в начале видовой системы координат. Это может улучшить качество освещения, но усложняет его расчет. Значение по умолчанию – GL_FALSE.

GL_LIGHT_MODEL_TWO_SIDE параметр «param» должен быть булевым и управляет режимом расчета освещенности, как для лицевых, так и для обратных граней. Если он равен GL_FALSE, то освещенность рассчитывается только для лицевых граней.

Если же он равен GL_TRUE, расчет проводится и для обратных граней. Значение по умолчанию – GL_FALSE.

GL_LIGHT_MODEL_AMBIENT параметр «params» должен содержать четыре целых или вещественных числа, которые определяют цвет фонового освещения даже в случае отсутствия определенных источников света. Значение по умолчанию – (0.2, 0.2, 0. 2,1.0).

Спецификация материалов

Для задания параметров текущего материала используются команды void glMaterial [i f] (GLenum face, GLenum pname, GLtype param) void glMaterial [i f] v (GLenum face, GLenum pname, GLtype *params).

С их помощью можно определить рассеянный, диффузный и зеркальный цвета материала, а также степень зеркального отражения и интенсивность излучения света, если объект должен светиться. Какой именно параметр будет определяться значением «param», зависит от значения pname:

– GL_AMBIENT параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют рассеянный цвет материала (цвет материала в тени). Значение по умолчанию – (0.2, 0.2, 0.2, 1.0);

– GL_DIFFUSE параметр «params» должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют диффузный цвет материала. Значение по умолчанию – (0.8, 0.8, 0.8, 1.0);

– GL_SPECULAR параметр «params» должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют зеркальный цвет материала. Значение по умолчанию – (0.0, 0.0, 0.0, 1.0);

– GL_SHININESS параметр params должен содержать одно целое или вещественное значение в диапазоне от 0 до 128, которое определяет степень зеркального отражения материала. Значение по умолчанию – 0;

– GL_EMISSION параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют интенсивность излучаемого света материала. Значение по умолчанию: (0.0, 0.0, 0.0, 1.0);

– GL_AMBIENT_AND_DIFFUSE эквивалентно двум вызовам команды: glMaterial*() со значением «pname» GL_AMBIENT и GL_DIFFUSE и одинаковыми значениями «params».

Из этого следует, что вызов команды: glMaterial [i f]() возможен только для установки степени зеркального отражения материала. Команда glMaterial [i f] v() используется для задания остальных параметров.

Параметр «face» определяет тип граней, для которых задается этот материал и может принимать значения GL_FRONT, GL_BACK или GL_FRONT_AND_BACK.

Если в сцене материалы объектов различаются лишь одним параметром, рекомендуется сначала установить нужный режим, вызвав glEnable() с параметром GL_COLOR_MATERIAL, а затем использовать команду void glColorMaterial (GLenum face, GLenum pname), где параметр «face» имеет аналогичный смысл, а параметр «pname» может принимать все перечисленные значения. После этого значения выбранного с помощью «pname» свойства материала для конкретного объекта (или вершины) устанавливаются вызовом команды glColor*(), что позволяет избежать вызовов более ресурсоемкой команды glMaterial*() и повышает эффективность программы.

Создание эффекта тумана

Одна из интересных и часто используемых возможность OpenGL – создание эффекта тумана. Легкое затуманивание сцены создает реалистичный эффект, а частенько может и скрыть некоторые артефакты, которые появляются, когда в сцене присутствуют отдаленные объекты.

Туман в OpenGL реализуется путем изменения цвета объектов в сцене в зависимости от их глубины, т.е. расстояния до точки наблюдения. Изменение цвета происходит либо для вершин примитивов, либо для каждого пикселя на этапе растеризации в зависимости от реализации OpenGL. Этим процессом можно частично управлять.

Для включения эффекта затуманивания необходимо вызвать команду glEnable (GL_FOG).

Метод вычисления интенсивности тумана в вершине можно определить с помощью команд void glFog[if] (enum pname, T param); void glFog[if] v (enum pname, T params);

Аргумент «pname» может принимать следующие значения:

GL_FOG_MODE аргумент «param» определяет формулу, по которой будет вычисляться интенсивность тумана в точке. В этом случае «param» может принимать значения:

- GL_EXP интенсивность вычисляется по формуле f=exp (-d*z);
- GL_EXP2 интенсивность вычисляется по формуле f=exp(– (d*z) 2);
- GL_LINEAR интенсивность вычисляется по формуле f=e-z/e-s,

где z – расстояние от вершины, в которой вычисляется интенсивность тумана, до точки наблюдения.

Коэффициенты d, e, s задаются с помощью следующих значений аргумента pname:

- GL_FOG_DENSITY param определяет коээфициент d;
- GL_FOG_START param определяет коэффициент s;
- GL_FOG_END param определяет коэффициент e.

Цвет тумана задается с помощью аргумента pname, равного GL_FOG_COLOR в этом случае params – указатель на массив из 4-х компонент цвета.

Разработка приложения для построения динамического изображения трехмерной модели объекта «Мотоцикл»

Разработка процедуры визуализации трехмерной сцены

Прорисовка в рабочей области начинается с метода void CLab1View: OnDraw (CDC* pDC), в котором вызывается функция usr_RenderScene (). Она отвечает за прорисовку функциональных частей и за некоторые важные расчёты, которые связаны с расположением некоторых отдельных деталей.

Рисование 3D объекта начинается с установления формата пикселей и области вывода в методах BOOL CLab1View:usr_setuppixelformat() и void CLab1View:usr_Resize (int x, int y, int width, int heidht) соответственно. Рассмотрим подробнее метод usr_RenderScene (), код которого представлен в приложении Б.

В этом методе начинается прорисовка объекта. Модель рисуем с переднего колеса состоящего из нескольких деталей – gluCylinder (quadric, 0.5, 0.5, 0.12, 50,40), gluCylinder (quadric, 0.45, 0.45, 0.12, 50,40), Cylinder (quadric, 0.5, 0.45, 0.0, 50,40), Cylinder (quadric, 0.5, 0.45, 0.0, 50,40), Cylinder (quadObj, 0.05, 0.46, 0.0, 25, 15), Cylinder (quadric, 0.05, 0.05, 0.18, 50,40) и 2-х SolidSphere (0.05). Затем рисуем корпус предварительно развернув сцену на 80 градусов при помощи функции glRotatef, при помощи функций gluCylinder (quadric, 0.3, 0.3, 0.4, 50,40). Следующий шаг – стойки состоящие из цилиндров, но прежде настраиваем сцену при помощи функций (glScalef (0. 4,1,1); glRotatef (90, 1, 0, 0); glRotatef (-10, 0, 1, 0);). Теперь рисуем фару также из цилиндра предварительно настроив сцену glColor3f (155.0f, 5.0f, 0.0f), glTranslatef (-0. 55,0. 2,0.07), glScalef (0. 4,1,1), glRotatef (90, 0, 1, 0). Рисуем руль при помощи цилиндров и сфер и glRotatef, glScalef, glTranslatef. Теперь рисуем заднее колесо, задние стойки аналогично передним. Рисуем сидение при помощи auxSolidBox((GLfloat) 0. 2,0. 3,0.38). Перед прорисовкой каждой детали использовалась функция glTranslatef для переноса начала координат. На этом прорисовка модели закончена.

Весь код метода usr_RenderScene() представлен в приложении Б.

Разработка интерфейса пользователя

Для данной программы разработан интерфейс, позволяющий:

- включать и выключать эффект тумана;
- выбрать цвет тумана;
- выбирать тип тумана;
- вращать Мотоцикл;
- задавать цвет деталей Мотоцикла;
- выбирать тип полигонов;

- выбирать несколько источников света;
- выбирать тип и задавать параметры перспективы;
- вращать и поворачивать сцену цифровой клавиатурой;
- приближать и удалять объект с помощью мышки.

В главном меню добавлены пункты:

- «Настройки» – различные настройки всей сцены.

Приложение имеет интуитивно понятный интерфейс, с которым удобно работать.

Разработка подсистемы управления событиями

Любое windows-положение основано на событиях и их обработке, другими словами поведение программы управляется событиями. Данный проект тоже является windows приложением, и, следовательно, обработка событий занимает важное место. К основным событиям, играющим важную, для корректной работы программы, роль относятся следующие:

- WM_DESTROY – освобождение занятых ресурсов;
- WM_SIZE – изменения сцены относительно размеров окна;
- WM_ERASEBKGND – предотвращения мерцания;
- WM_TIMER – используется для создания таймера;
- WM_MOUSEWHEEL – обработка вращения колеса мышки;
- WM_KEYDOWN – обработка нажатия клавиши;
- COMMAND (ID_PERSP, CKarkasView: OnPersp ()) – обработка события при вызове окна настройки перспективы;
- COMMAND (ID_OPTIONS, CKarkasView: OnOptions ()) – обработка события при вызове окна настройки типа вращения и скорости движения объекта;
- COMMAND (ID_VIEW_1, CKarkasView: OnView1 ()) – обработка события выбора типа тумана;
- COMMAND (ID_VIEW_SBROS, CKarkasView: OnVewSbros()) – обработка события нажатия кнопки «Убрать туман».

Информационное и программное обеспечение

Общие сведения о программе

Программа называется «Мотоцикл». При работе с данной программой у пользователя есть возможность работать с визуальной моделью данного объекта. Вращать ее относительно осей, включать и выключать эффект тумана, выбирать цвет тумана, выбирать тип полигонов, выбирать несколько источников света, выбирать цвет деталей Мотоцикла, также задавать тип тумана, приближать и удалять сцену с помощью колеса мышки, включать и выключать вращение модели и устанавливать скорость вращения. Программное обеспечение, на котором разработана приложение – Microsoft Visual C++ 6.0.

Функциональное назначение

Данная программа предназначается для представления трехмерной модели Мотоцикла. Приложение дает следующие возможности:

- наблюдать модель
- включать и выключать эффект тумана;
- выбрать цвет тумана;
- выбирать тип тумана;
- вращать Мотоцикл;
- задавать цвет деталей Мотоцикла;
- выбирать тип полигонов;
- выбирать несколько источников света;
- выбирать тип и задавать параметры перспективы;
- вращать и поворачивать сцену цифровой клавиатурой;
- приближать и удалять объект с помощью мышки.

Логическая структура и функциональная декомпозиция проекта

Инициализация OpenGL происходит в несколько этапов.

1. Выбираем и устанавливаем формат пикселей. В информации о формате пикселей указывается такая информация как глубина цвета, различные флаги поверхности. Вся эта структура представляется в специальной структуре PIXELFORMATDESCRIPTOR. Далее передаем на рассмотрение операционной системе, выбранный формат пикселей. После того, как система просмотрит его, она выберет наиболее совпадающий формат с тем, который поддерживается в контексте устройства. Функцией, осуществляющей такого рода проверку, является ChoosePixelFormat(). После выбора формата пикселей функция SetPixelFormat() устанавливает его в контексте устройства.
2. Создаем контекст вывода для библиотеки OpenGL. Данный контекст создается с помощью функции wglCreateContext(), далее функция wglMakeCurrent() устанавливает текущий контекст. Функция wglGetCurrentDC() необходима для корректного завершения приложения, а wglGetCurrentDC() – для удаления контекста воспроизведения.

Одним из важных методов является usr_ReSize(), который устанавливает перспективу и область вывода.

За отображение сцены отвечает метод usr_RenderScene(), который вызывает в свою очередь функции рисования компьютера.

Функции, вызываемые методом usr_RenderScene() были подробно рассмотрены в разделе «Разработка процедуры визуализации трехмерной сцены», а важные для логического понимания структуры события программы рассмотрены в разделе «Разработка подсистемы управления событиями».

Для наглядности приведем таблицу наиболее важных методов – таблица 6.1.

Таблица 6.1 – Основные методы и функции

№	Метод	Назначение	Назначение параметров
1	PreCreateWindow (CREATESTRUCT& cs)	Инициализация окна	cs – объект структуры CREATESTRUCT. Производится изменение значений, присвоенных переменным-членам структуры CREATESTRUCT для изменения режима открытия окна и его параметров.
2	usr_bSetupPixelFormat ()	Установка формата пикселей	
3	usr_bInitOpenGL()	Инициализация OpenGL	
4	user_DestoryOpenGL()	Освобождение ресурсов (из-под OpenGL)	
5	usr_ReSize (int x, int y, int width, int height)	Корректирует вывод сцены на экран при изменении размера окна	x и y определяют координаты левого нижнего угла вывода, width и height – ширину и высоту области вывода
6	usr_RenderScene()	Рисует Мотоцикл целиком	

В таблице 6.2 приведены спецификации разработанных классов.

Таблица 6.2 – Спецификации классов

Название	Назначение
CMainframe	Класс главного окна приложения. Используется для управления главным окном
CKarkasApp	Главный класс приложения. Управляет работой всего приложения. Методы этого класса выполняют инициализацию приложения, обработку цикла сообщений и вызываются при завершении приложения.
CKarkasDoc	Класс документа приложения.
CKarkasView	Класс окна просмотра документа. Служит для отображения в клиентской области класса документа приложения в нашем случае нашей 3D модели.
CAboutDlg	Класс справочной информации о программе
DlgPers	Класс диалогового окна. Служит для настройки и смены перспективы
DlgOptions	Класс диалогового окна. Служит для включения различных настроек сцены.

Требования к техническому и программному обеспечению

Для успешной эксплуатации программного продукта необходим персональный компьютер со следующими характеристиками: процессор Intel Pentium с тактовой частотой 800 МГц и выше, оперативная память – не менее 64 Мбайт, свободное дисковое пространство – не менее 200 Мбайт, устройство для чтения компакт-дисков, монитор типа Super VGA (число цветов – 256). Программное обеспечение: операционная система WINDOWS 2000/XP и выше.

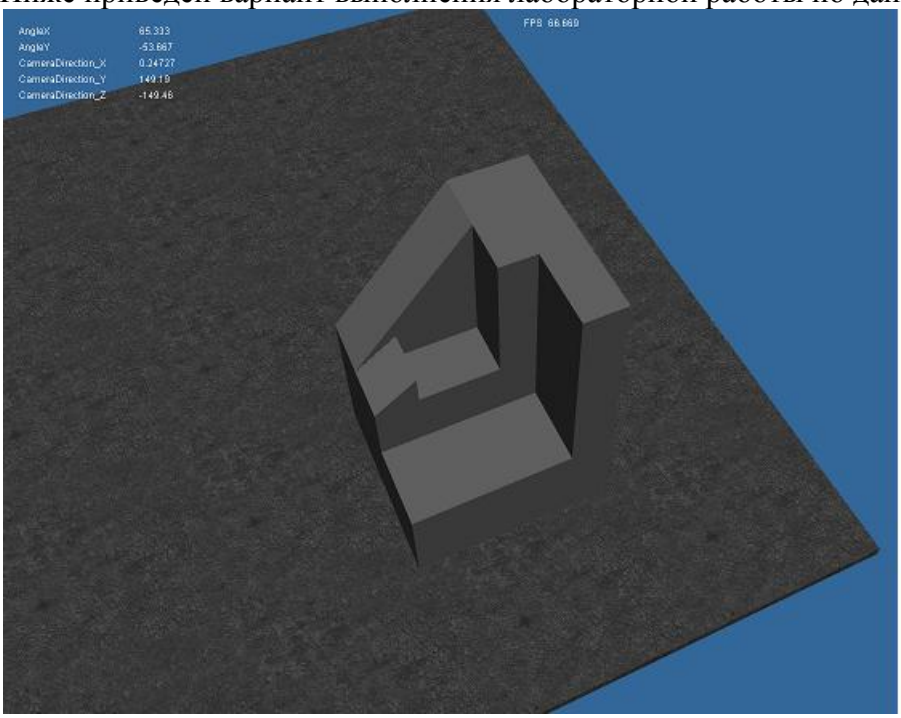
Руководство пользователя

Для установки приложения требуется скопировать с установочного диска, прилагаемого к работе файл «Мотоцикл.exe» в любую директорию на жестком диске. Для запуска программы нужно два раза нажать на левую клавишу мыши.

Разработанное приложение имеет интуитивно понятный интерфейс, который схож с другими Windows – приложениями. После запуска программы пользователь, может вращать сцену и поворачивать её с помощью цифровой клавиатуры (8 – вверх, 2 – вниз, 4 – влево, 6 – вправо, 7 и 9 – вращение по оси, 1 и 3 – вращение по другой оси). Также имеется возможность приближать и удалять модель, это можно сделать, задействовав колесо мыши.

Существует поддержка различных графических эффектов. Для их выбора нажмите на кнопку Настройки, после нажатия откроется окно где можно выбрать различные опции, для подтверждения нужно нажать кнопку Ок.

Ниже приведен вариант выполнения лабораторной работы по данной тематике.



Лабораторная работа № 7. Создание трехмерной модели детали средствами библиотеки OpenGL

Данная лабораторная работа нужна студенту для преобретения навыков работы в среде программирования C++ Builder с использованием библиотеки OpenGL.

Цели работы

Познакомиться с конкретным примером выполнения программы в среде программирования

C++ Builder с использованием библиотеки OpenGL.

Выполнить задание по указанию преподавателя, оформить файл отчета, сдать исходные и исполняемый файл.

Задание

По заданию своего номера варианта создать трехмерную каркасную модель детали средствами графики C++ Builder. Написать текст программы. Предусмотреть параметризацию размеров детали. Привести рисунок копии экрана для своего варианта.

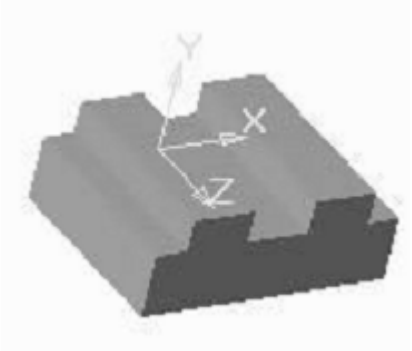


Рисунок задания

Пояснение

Прежде чем начать выполнять задание рассмотрим такие графические понятия, как вершины и примитивы библиотеки OpenGL.

Под вершиной понимается точка в трехмерном пространстве, координаты которой можно задавать следующим образом:

void glVertex[2 3 4][s i f d](type coords)

void glVertex[2 3 4][s i f d]v(type *coords)

Координаты точки задаются максимум четырьмя значениями: x, y, z, w, при этом можно указывать два (x,y) или три (x,y,z) значения, а для остальных переменных в этих случаях используются значения по умолчанию: z=0, w=1. Как уже было сказано выше, число в названии команды соответствует числу явно задаваемых значений, а последующий символ – их типу.

Координатные оси расположены так, что точка (0,0) находится в левом нижнем углу экрана, ось x направлена влево, ось y- вверх, а ось z- из экрана. Это расположение осей мировой системы координат, в которой задаются координаты вершин объекта, другие системы координат будут рассмотрены ниже.

Однако чтобы задать какую-нибудь фигуру одних координат вершин недостаточно, и эти вершины надо объединить в одно целое, определив необходимые свойства. Для этого в OpenGL используется понятие примитивов, к которым относятся точки, линии, связанные или замкнутые линии, треугольники и так далее. Задание примитива происходит внутри командных скобок:

void glBegin(GLenum mode)

void glEnd(void)

Параметр mode определяет тип примитива, который задается внутри и может принимать следующие значения:

GL_POINTS каждая вершина задает координаты некоторой точки.

GL_LINES каждая отдельная пара вершин определяет отрезок; если задано нечетное число вершин, то последняя вершина игнорируется.

GL_LINE_STRIP каждая следующая вершина задает отрезок вместе с предыдущей.

GL_LINE_LOOP отличие от предыдущего примитива только в том, что последний отрезок определяется последней и первой вершиной, образуя замкнутую ломаную.

GL_TRIANGLES каждая отдельная тройка вершин определяет треугольник; если задано не кратное трем число вершин, то последние вершины игнорируются.

GL_TRIANGLE_STRIP каждая следующая вершина задает треугольник вместе с двумя предыдущими.

GL_TRIANGLE_FAN треугольники задаются первой и каждой следующей парой вершин (пары не пересекаются).

GL_QUADS каждая отдельная четверка вершин определяет четырехугольник; если задано не кратное четырем число вершин, то последние вершины игнорируются.

GL_QUAD_STRIP четырехугольник с номером n определяется вершинами с номерами 2n-1, 2n, 2n+2, 2n+1.

GL_POLYGON последовательно задаются вершины выпуклого многоугольника.

Для задания текущего цвета вершины используются команды

void glColor[3 4][b s i f](GLtype components)

void glColor[3 4][b s i f]v(GLtype components)

Первые три параметра задают R, G, B компоненты цвета, а последний параметр определяет alpha-компоненту, которая задает уровень прозрачности объекта. Если в названии команды указан тип ‘f’ (float), то значения всех параметров должны принадлежать отрезку [0,1], при этом по умолчанию значение alpha-компоненты устанавливается равным 1.0, что соответствует полной непрозрачности. Если указан тип ‘ub’ (unsigned byte), то значения должны лежать в отрезке [0,255].

Разным вершинам можно назначать различные цвета и тогда будет проводиться линейная интерполяция цветов по поверхности примитива.

Для управления режимом интерполяции цветов используется команда void glShadeModel(GLenummode) вызов которой с параметром GL_SMOOTH включает интерполяцию (установка по умолчанию), а с GL_FLAT отключает.

Например, чтобы нарисовать треугольник с разными цветами в вершинах, достаточно написать:

```
GLfloat BlueCol[3]={0,0,1};
glBegin(GL_TRIANGLE);
glColor3f(1.0, 0.0, 0.0); //красный
glVertex3f(0.0, 0.0, 0.0);
glColor3ub(0,255,0); //зеленый
glVertex3f(1.0, 0.0, 0.0);
glColor3fv(BlueCol); //синий
glVertex3f(1.0, 1.0, 0.0);
glEnd();
```

Для задания цвета фона используется команда void glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha). Значения должны находиться в отрезке [0,1] и по умолчанию равны нулю. После этого вызов команды void glClear(GLbitfield mask) с параметром GL_COLOR_BUFFER_BIT устанавливает цвет фона во все буфера, доступные для записи цвета (иногда удобно использовать несколько буферов цвета).

Решение

Для решения данной задачи достаточно использовать только один примитив – многоугольник (GL_POLYGON). Особое внимание необходимо обратить на соблюдение обязательного условия выпуклости многоугольника. Для создания многоугольника необходимы трехмерные значения координат вершин, заданных в примитиве вершина (например, glVertex3f). При необходимости используется задание цвета многоугольника (например, glColor3f(0.9,0.4,0.1).

Таблица 1 Значения координат вершин граней детали

Номер грани	Координаты вершин для грани			
	Вершина	X	Y	Z
1	1	0.0	0.2	0.1
	2	0.0	0.6	0.1
	3	0.9	0.6	0.1
	4	0.9	0.2	0.1
…				
13bok_3	1	0.2	0.6	0.3
	2	0.2	0.6	0.5
	3	0.3	0.6	0.5
	4	0.3	0.6	0.3

Текст программы

Для упрощения работы над заданием студентам выдается электронный вариант выполненного задания для кубика (с координатами для восьми граней). От студента требуется почесть задание и вставить в функцию OpenGLPanel1Paint свои координаты.

Листинг 1 Текст программы создания трехмерной модели.

```
/* Выполнил студент группы 1511 Петров С.Ю., вариант 03, дата
09.03.2004, файл Ug151103S2.cpp */
#include <vcl.h>
#pragma hdrstop
#include " Ug151103S2.h"
//-------------------------
#pragma package(smart_init)
#pragma link "OpenGLPanel"
#pragma resource "*.dfm"
TForm1 *Form1;
//-------------------------
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{  // Повороты детали вокруг осей X и Y
 RotX=(float)TrackBar1->Position;
 RotY=(float)TrackBar2->Position;}
//--------------------------
void __fastcall TForm1::OpenGLPanel1Init(TObject *Sender)
{// Инициализация OpenGL графики в среде C++ Builder
glViewport(0,0,(GLsizei)OpenGLPanel1->Width,
(GLsizei)OpenGLPanel1->Height);
glMatrixMode(GL_PROJECTION); glLoadIdentity();
   if ( OpenGLPanel1->Height==0)
gluPerspective(45, (GLdouble)OpenGLPanel1->Width, 1.0, 2000.0);
   else
gluPerspective(45, (GLdouble)OpenGLPanel1->Width/
(GLdouble)OpenGLPanel1->Height,1.0, 2000.0);
glMatrixMode(GL_MODELVIEW); glLoadIdentity();
glEnable(GL_DEPTH_TEST); glClearColor(0.0,0.0,0.0,1.0);}
//--------------------------------
void __fastcall TForm1::OpenGLPanel1Resize(TObject *Sender)
{ /* Перерисовка изображения при измененииии размеров окна
приложения */
 glViewport(0,0,(GLsizei)OpenGLPanel1->Width,
```

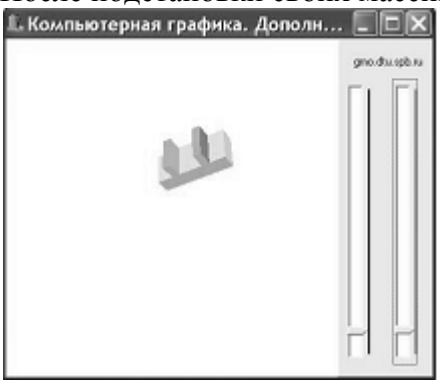
```
(GLsizei)OpenGLPanel1->Height);
 glMatrixMode(GL_PROJECTION);  glLoadIdentity();
 if ( OpenGLPanel1->Height==0)
gluPerspective(45, (GLdouble)OpenGLPanel1->Width, 1.0, 2000.0);
 else
gluPerspective(45, (GLdouble)OpenGLPanel1->Width/
(GLdouble)OpenGLPanel1->Height,1.0, 2000.0);
 glMatrixMode(GL_MODELVIEW); glLoadIdentity();}
//-------------------------------------
void __fastcall TForm1::OpenGLPanel1Paint(TObject *Sender)
{/* Создание поверхностной модели детали из многоугольников
GL_POLYGON */
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glPushMatrix();  glTranslated(0.0, 0.0, -5.0);
 glRotatef(RotX, 1.0, 0.0, 0.0); glRotatef(RotY, 0.0, 1.0, 0.0);
////////////////////////////-1-
glBegin(GL_POLYGON); glColor3f(0.9,0.4,0.1);
glVertex3f(  0.0,  0.2,  0.1); glVertex3f(  0.0,  0.6,  0.1);
glVertex3f(  0.9,  0.6,  0.1); glVertex3f(  0.9,  0.2,  0.1);
 glEnd();
////////////////////////////-2-
 glBegin(GL_POLYGON); glColor3f(0.8,0.8,0.8);
glVertex3f(  0.9, 0.2,  0.1); glVertex3f(  0.9, 0.6,  0.1);
glVertex3f(  0.9, 0.6,  0.3); glVertex3f(  0.9, 0.2,  0.3);
 glEnd();
////////////////////////////-3-
glBegin(GL_POLYGON); glColor3f(1.0,0.8,1.0);
glVertex3f(  0.9, 0.2,  0.3); glVertex3f(  0.9, 0.6,  0.3);
glVertex3f(  0.7, 0.6,  0.3); glVertex3f(  0.7, 0.2,  0.3);
 glEnd();
////////////////////////////-4-
glBegin(GL_POLYGON); glColor3f(0.5,0.5,0.5);
glVertex3f(  0.7, 0.2,  0.5); glVertex3f(  0.7, 0.6,  0.5);
glVertex3f(  0.7, 0.6,  0.3); glVertex3f(  0.7, 0.2,  0.3);
 glEnd();
////////////////////////////-5-
glBegin(GL_POLYGON); glColor3f(1.0,0.5,1.0);
glVertex3f(  0.7, 0.2,  0.5); glVertex3f(  0.7, 0.6,  0.5);
glVertex3f(  0.6, 0.6,  0.5); glVertex3f(  0.6, 0.2,  0.5);
 glEnd();
////////////////////////////-6-
glBegin(GL_POLYGON); glColor3f(0.0,1.0,1.0);
glVertex3f(  0.6, 0.2,  0.5); glVertex3f(  0.6, 0.6,  0.5);
glVertex3f(  0.6, 0.6,  0.3); glVertex3f(  0.6, 0.2,  0.3);
 glEnd();
////////////////////////////-7-
glBegin(GL_POLYGON); glColor3f(1.0,0.8,1.0);
glVertex3f(  0.6, 0.2,  0.3); glVertex3f(  0.6, 0.6,  0.3);
glVertex3f(  0.3, 0.6,  0.3); glVertex3f(  0.3, 0.2,  0.3);
 glEnd();
////////////////////////////-8-
glBegin(GL_POLYGON); glColor3f(0.0,1.0,1.0);
glVertex3f(  0.3, 0.2,  0.3); glVertex3f(  0.3, 0.6,  0.3);
glVertex3f(  0.3, 0.6,  0.5); glVertex3f(  0.3, 0.2,  0.5);
 glEnd();
////////////////////////////-9-
```

```
glBegin(GL_POLYGON); glColor3f(1.0,0.5,1.0);
glVertex3f(  0.2, 0.2,  0.5); glVertex3f(  0.2, 0.6,  0.5);
glVertex3f(  0.3, 0.6,  0.5); glVertex3f(  0.3, 0.2,  0.5);
 glEnd();
//////////////////////////-10-
glBegin(GL_POLYGON); glColor3f(0.2,0.2,1.0);
glVertex3f(  0.2, 0.2,  0.5); glVertex3f(  0.2, 0.6,  0.5);
glVertex3f(  0.2, 0.6,  0.3); glVertex3f(  0.2, 0.2,  0.3);
 glEnd();
//////////////////////////-11-
glBegin(GL_POLYGON); glColor3f(1.0,0.8,1.0);
glVertex3f(  0.0, 0.2,  0.3); glVertex3f(  0.0, 0.6,  0.3);
glVertex3f(  0.2, 0.6,  0.3); glVertex3f(  0.2, 0.2,  0.3);
 glEnd();
//////////////////////////-12-
glBegin(GL_POLYGON); glColor3f(0.2,0.2,0.8);
glVertex3f(  0.0, 0.2,  0.3); glVertex3f(  0.0, 0.6,  0.3);
glVertex3f(  0.0, 0.6,  0.1); glVertex3f(  0.0, 0.2,  0.1);
 glEnd();
//////////////////////////-12- bok_1
 glBegin(GL_POLYGON); glColor3f(0.2,1.0,0.1);
glVertex3f(  0.6,  0.2,  0.1); glVertex3f(  0.9,  0.2,  0.1);
glVertex3f(  0.9,  0.2,  0.3); glVertex3f(  0.7,  0.2,  0.3);
glVertex3f(  0.7,  0.2,  0.5); glVertex3f(  0.6,  0.2,  0.5);
glEnd();
 //////////////////////////-12- bok_1
 glBegin(GL_POLYGON); glColor3f(0.2,1.0,0.1);
glVertex3f(  0.0,  0.2,  0.1); glVertex3f(  0.9,  0.2,  0.1);
glVertex3f(  0.9,  0.2,  0.3); glVertex3f(  0.0,  0.2,  0.3);
 glEnd();
 //////////////////////////-12- bok_2
 glBegin(GL_POLYGON); glColor3f(0.2,1.0,0.1);
glVertex3f(  0.6,  0.2,  0.3); glVertex3f(  0.6,  0.2,  0.5);
glVertex3f(  0.7,  0.2,  0.5); glVertex3f(  0.7,  0.2,  0.3);
  glEnd();
  //////////////////////////-12- bok_3
 glBegin(GL_POLYGON); glColor3f(0.2,1.0,0.1);
glVertex3f(  0.2,  0.2,  0.3); glVertex3f(  0.2,  0.2,  0.5);
glVertex3f(  0.3,  0.2,  0.5); glVertex3f(  0.3,  0.2,  0.3);
 glEnd();
//////////////////////////-13- bok_1
 glBegin(GL_POLYGON); glColor3f(1.0,1.0,0.0);
glVertex3f(  0.0,  0.6,  0.1); glVertex3f(  0.9,  0.6,  0.1);
glVertex3f(  0.9,  0.6,  0.3); glVertex3f(  0.0,  0.6,  0.3);
 glEnd();
 //////////////////////////-13- bok_2
 glBegin(GL_POLYGON); glColor3f(1.0,1.0,0.0);
glVertex3f(  0.6,  0.6,  0.3); glVertex3f(  0.6,  0.6,  0.5);
glVertex3f(  0.7,  0.6,  0.5); glVertex3f(  0.7,  0.6,  0.3);
  glEnd();
  //////////////////////////-13- bok_3
 glBegin(GL_POLYGON); glColor3f(1.0,1.0,0.0);
glVertex3f(  0.2,  0.6,  0.3); glVertex3f(  0.2,  0.6,  0.5);
glVertex3f(  0.3,  0.6,  0.5); glVertex3f(  0.3,  0.6,  0.3);
 glEnd();
glPopMatrix();}
```

```
//-------------------------------------
void __fastcall TForm1::TrackBar1Change(TObject *Sender)
{ // Перерисовка детали после поворота вокруг оси X
RotX=(float)TrackBar1->Position;
 OpenGLPanel1->Repaint();}
//-------------------------------------
void __fastcall TForm1::TrackBar2Change(TObject *Sender)
{// Перерисовка детали после поворота вокруг оси Y
 RotY=(float)TrackBar2->Position;
 OpenGLPanel1->Repaint();}
//-------------------------------------
```

Копия экрана

После подстановки своих массивов координат в программу получаем решение.



Экранная копия работы программы

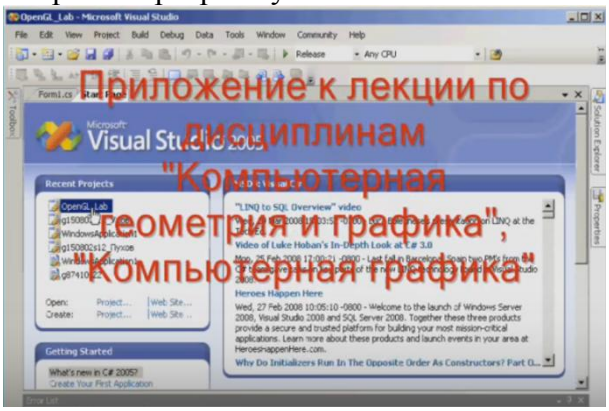
Оформление результатов работы

Результаты работы оформляются в виде отчета по выдаваемому в электронном виде образцу. В отчет входят следующие разделы: задание, решение, текст программы, копия экрана, литература и интернет- ссылки. К отчету прилагается исходные и исполняемые файлы программы. Нумерация файлов и папки осуществляется по личному коду, выдаваемому преподавателем. Все материалы сдаются в электронном виде. Допускается распечатка отчетных материалов.

Например, имя папки с файлами: g151103S2, где g- студенческая группа, 1511- номер группы, 03- номер студента по списку, L- лабораторная работа, 2-порядковый номер работы. В папке находятся следующие файлы: g151103L2.doc-файл отчета по работе (образец выдается преподавателем), исходные файлы (Unit) - Ug151103S2.cpp, Ug151103S2.h, Ug151103S2.dfm, файл проекта (Project)- g151103S2.bpr, g151103S2.cpp, g151103S2.res, исполняемый файл g151103S2.exe.

Ниже приведен пример выполнения работы.

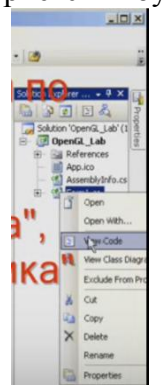
Откройте программу Microsoft Visual Studio C# (либо Visual Studio C# Express).



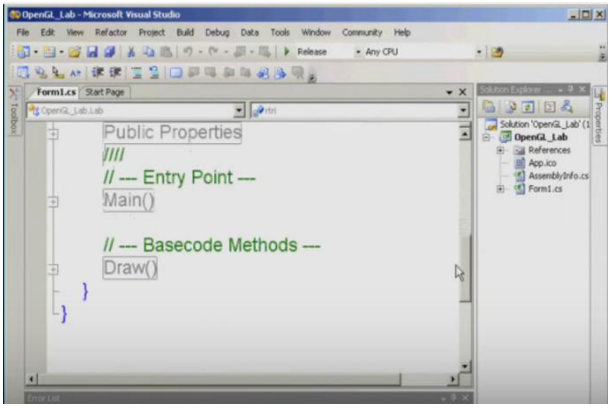
На правой боковой панели открыть вкладку «Solution Explorer»

В верхней части открытой вкладки есть серая панель. Нажмите на значок  .

После этого панель станет активной и будет гореть синим цветом. Значок перевернется и будет вертикальным.

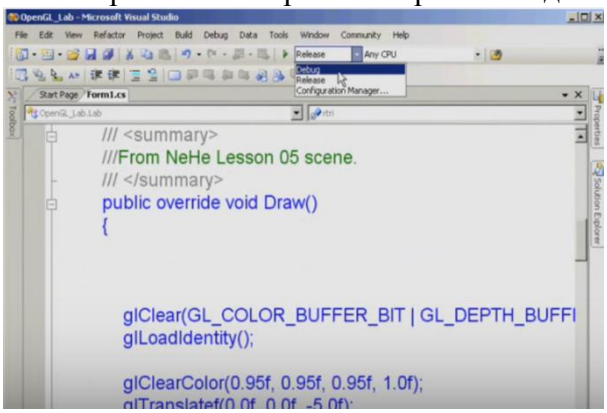


Выберите документ «Form1.cs». Выберите команду «View Code».

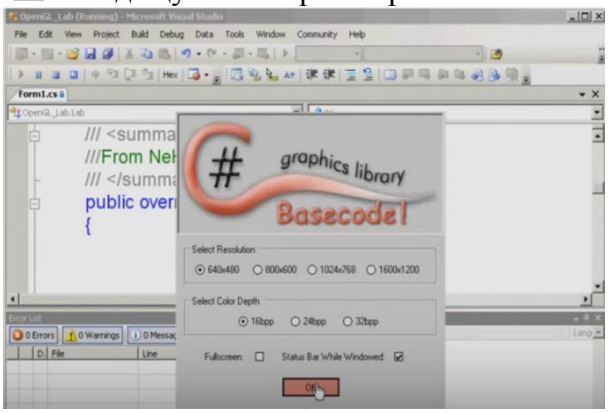


Получим следующее.

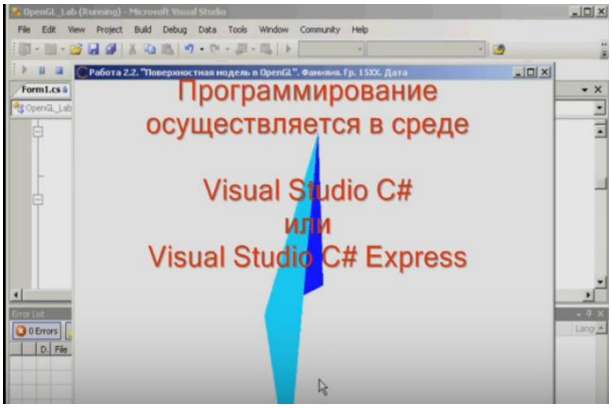
Рассмотрим самый простой вариант создания трехмерной полигональной модели.



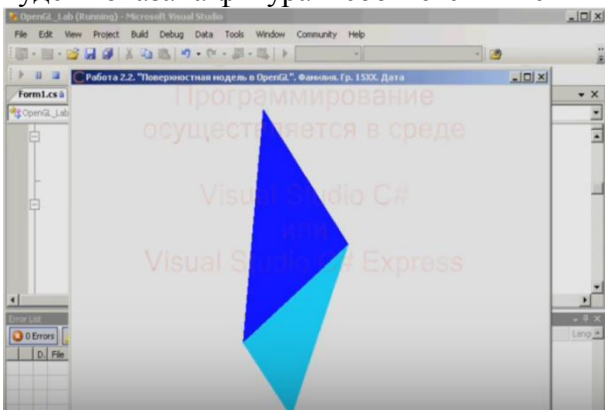
В верхней панели выбрать команду «Detug». Сохраните работу. Далее нажмите кнопку  находящуюся в верхней рабочей панели.

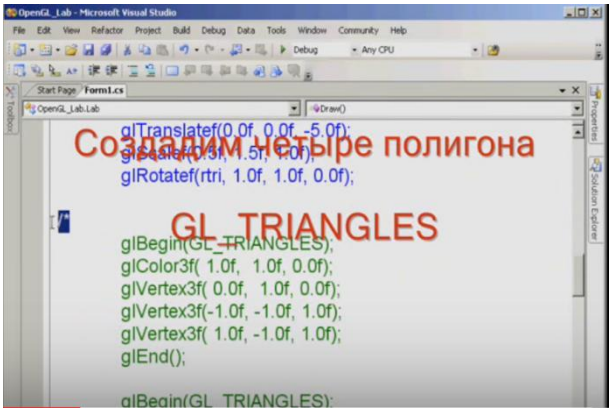


Выйдет окно графической библиотеки. Нажмите кнопку «Ок».

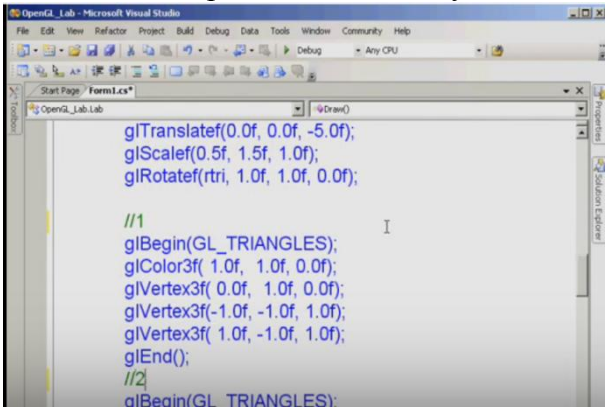


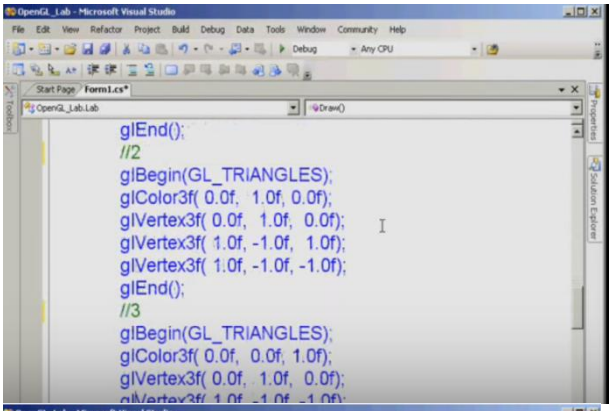
Будет показана фигура в соответствии с выполняемым вами заданием.

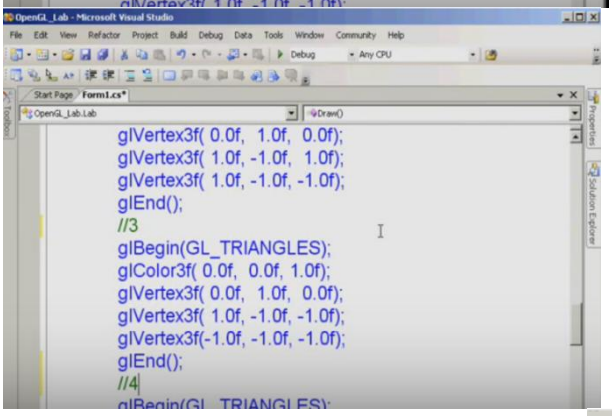




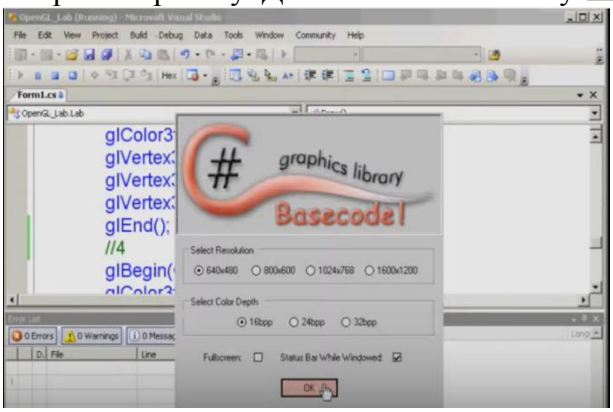
Создадим четыре полигона в документе «Form1.cs».



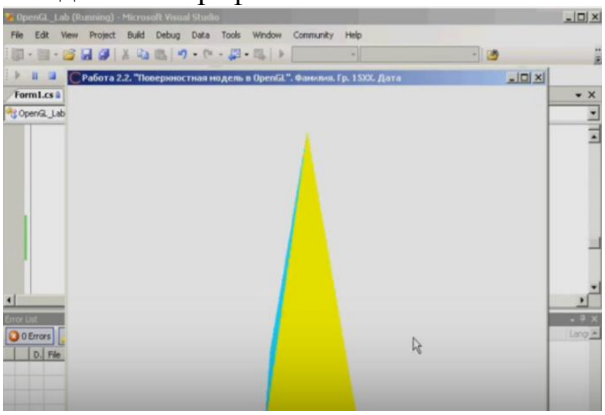




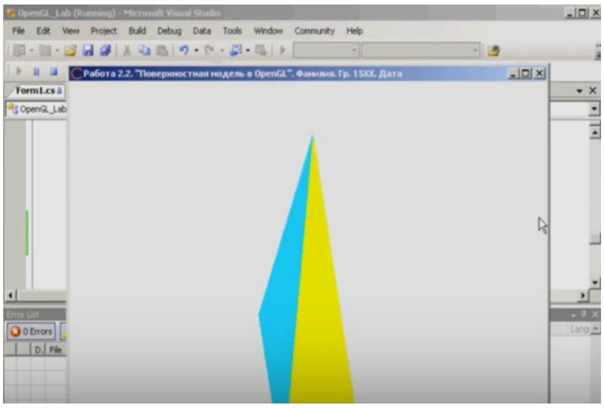
Сохраните работу. Далее нажмите кнопку  находящуюся в верхней рабочей панели.

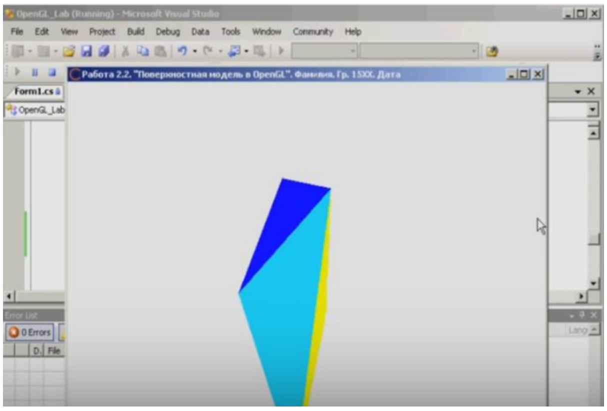


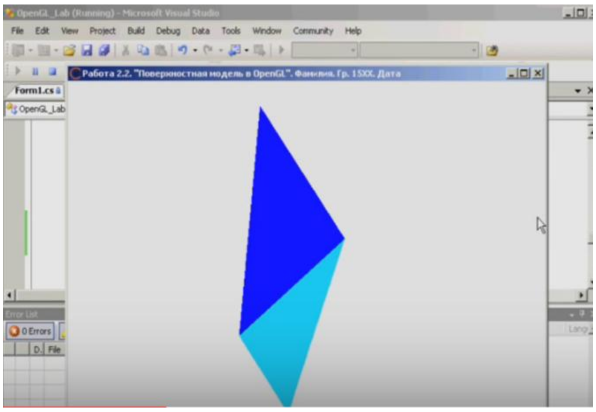
Выйдет окно графической библиотеки. Нажмите кнопку «Ок».

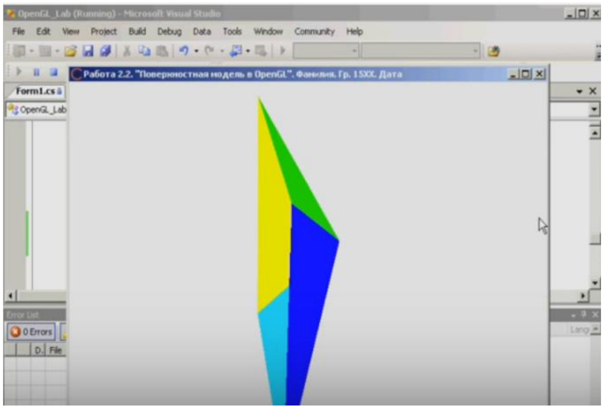


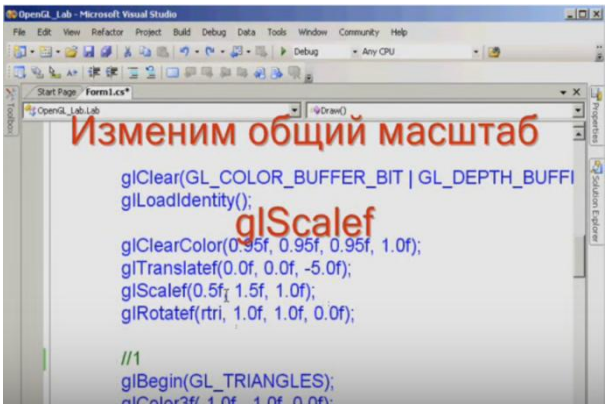
Появившаяся фигура будет поворачиваться вокруг собственной оси.


OpenGL_Lab (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Работа 2.2. "Поверхностная модель в OpenGL". Фамилия. Гр. 15XX. Дата
Form1.cs
OpenGL_Lab
0 Errors

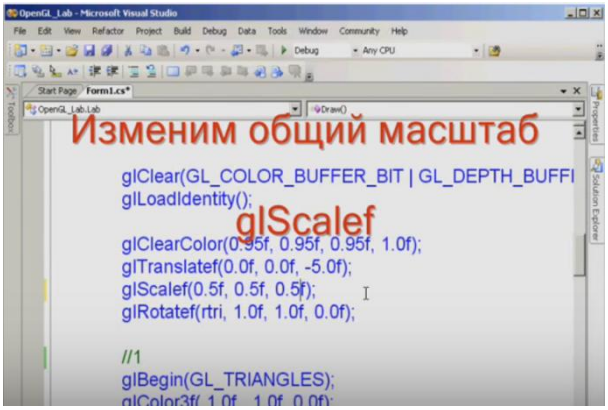

OpenGL_Lab (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Работа 2.2. "Поверхностная модель в OpenGL". Фамилия. Гр. 15XX. Дата
Form1.cs
OpenGL_Lab
0 Errors


OpenGL_Lab (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Работа 2.2. "Поверхностная модель в OpenGL". Фамилия. Гр. 15XX. Дата
Form1.cs
OpenGL_Lab
0 Errors

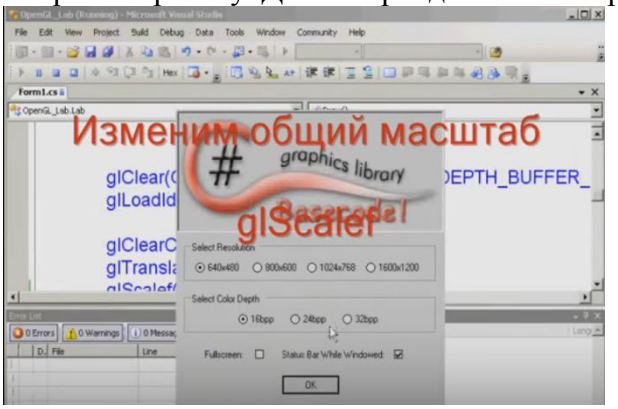

OpenGL_Lab (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Работа 2.2. "Поверхностная модель в OpenGL". Фамилия. Гр. 15XX. Дата
Form1.cs
OpenGL_Lab
0 Errors

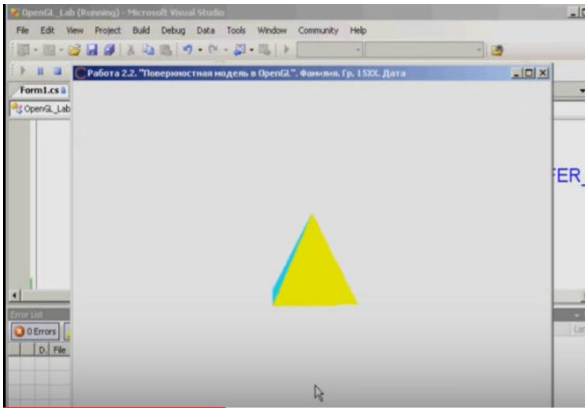


Проведем изменение общего масштаба glScalef. Смотри третью строчку, второго абзаца.

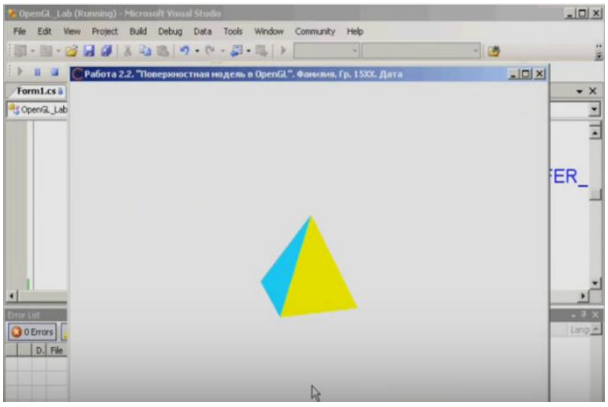


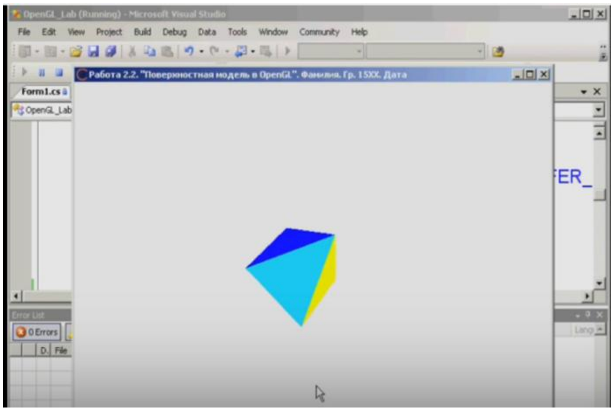
Сохраните работу. Далее перейдите в окно графической библиотеки нажав кнопку  .

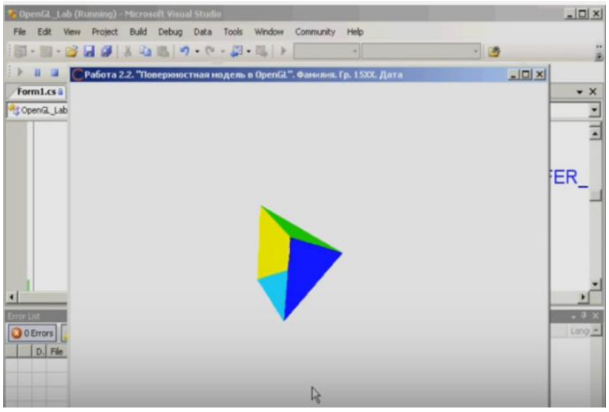


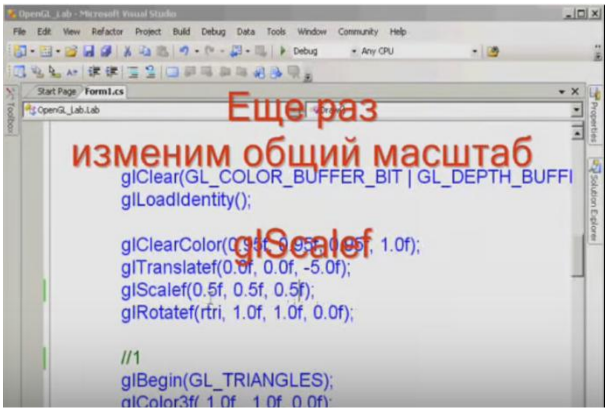


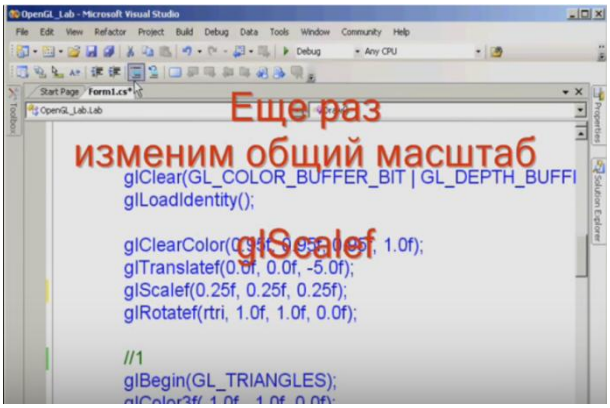
Получим следующее.



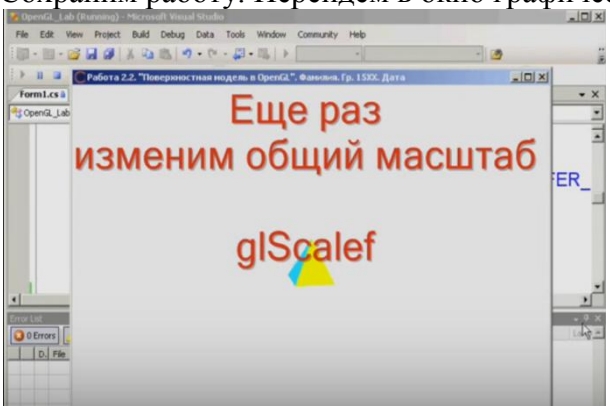


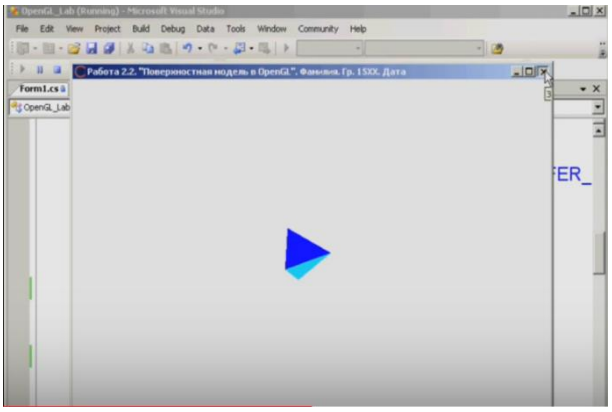



Еще раз
изменим общий масштаб
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFF
glLoadIdentity();
glScalef
glTranslatef(0.0f, 0.0f, -5.0f);
glScalef(0.5f, 0.5f, 0.5f);
glRotatef(rtri, 1.0f, 1.0f, 0.0f);
//1
glBegin(GL_TRIANGLES);

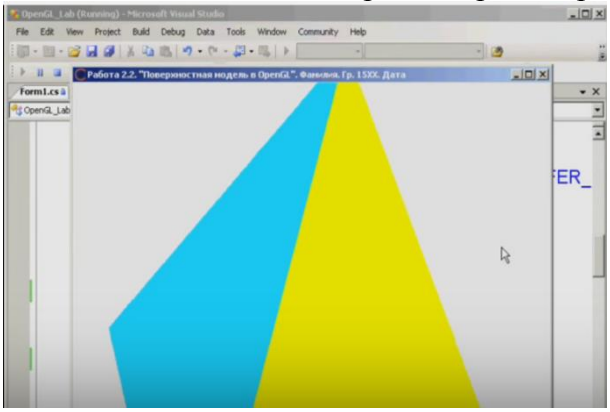


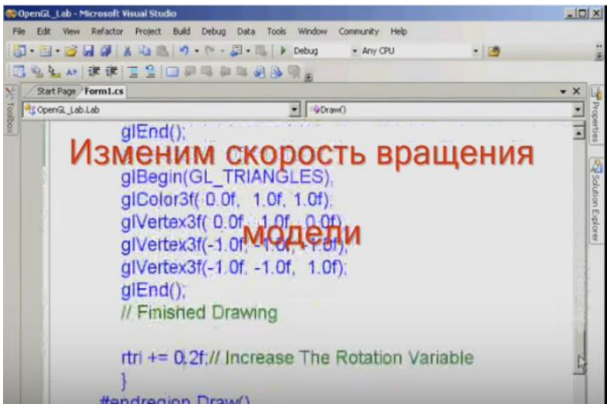
Сохраним работу. Перейдем в окно графической библиотеки.

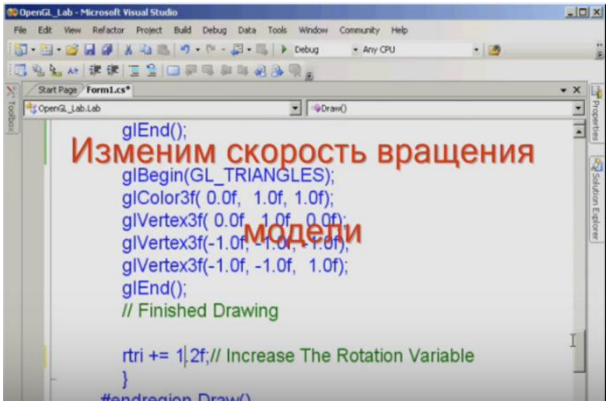




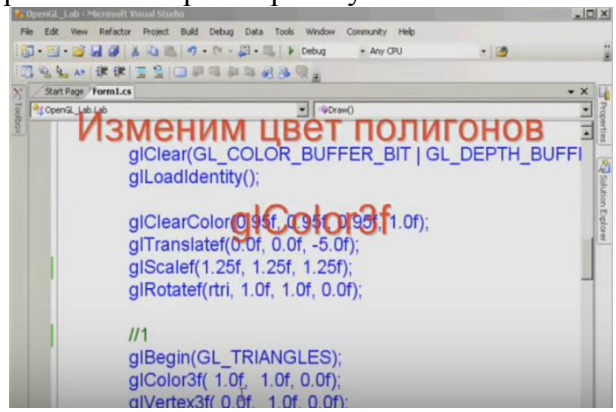
Увеличим масштаб. Повторим все ранее приведенные для этого операции.



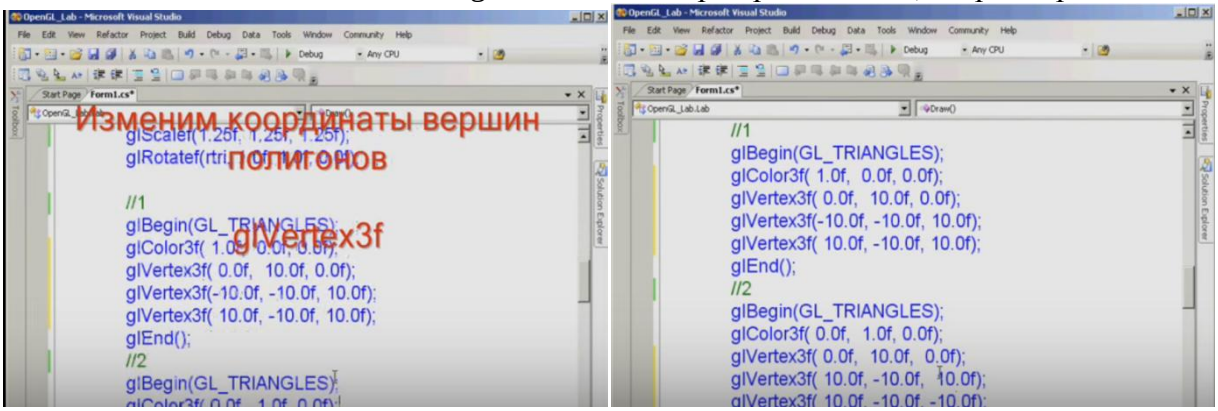


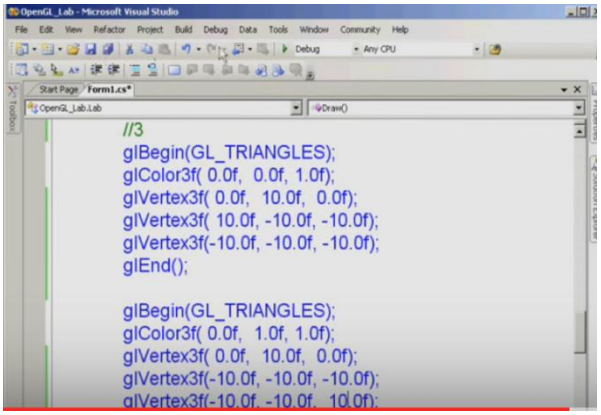


Изменим скорость вращения модели «Increase The Rotation Variable». Фигура будет вращаться быстрее. Для просмотра перейдите в окно графической библиотеки, предварительно сохранив работу.

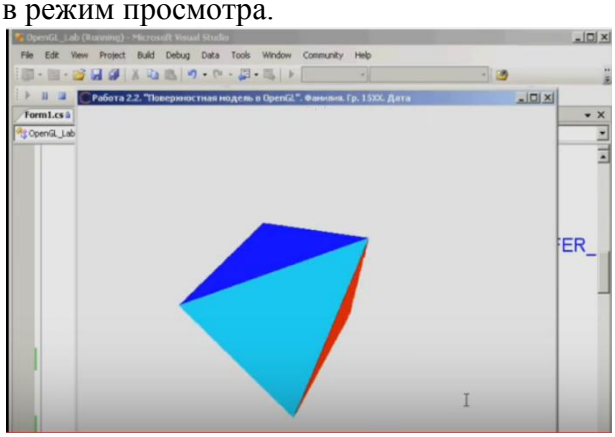


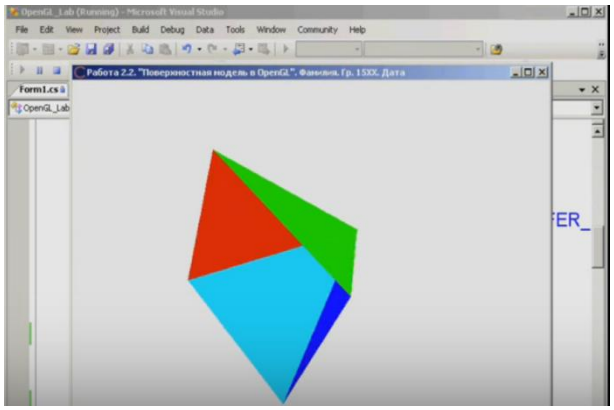
Изменим цвет полигонов «glColor3f». Смотри третий абзац, вторая строка.



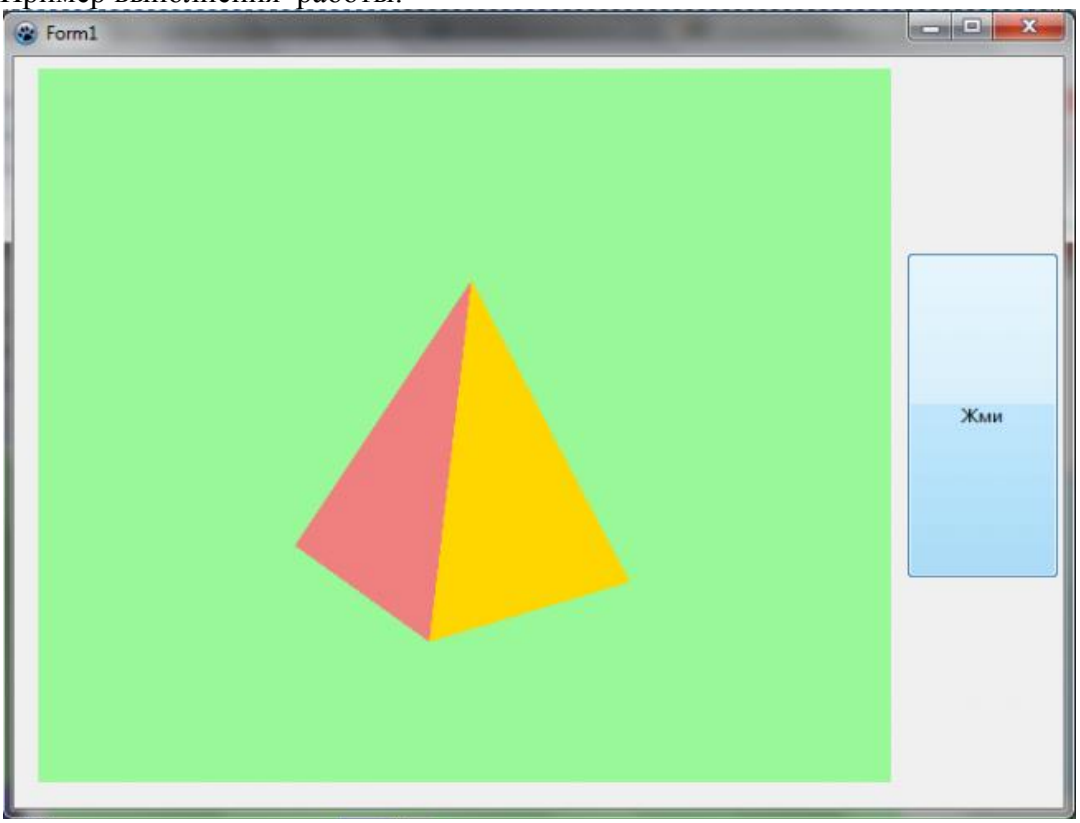


Для изменения координат вершин полигонов воспользуемся «glVertex3f». Введите данные указанные на этих рисунках. Сохраните работу. Измените масштаб на 0,1. Перейдите в режим просмотра.





Пример выполнения работы:



Среда программирования:

Lazarus 1.0.14 win 64

Статья по теме:

Вращение фигуры в 3-х мерном пространстве

Программа демонстрирует вращение четырехугольной пирамиды в OpenGL.

glTranlsatef() производит перенос объекта, прибавляя к координатам его вершин значения своих параметров. **glRotatef()** производит поворот объекта против часовой стрелки на угол angle (измеряется в градусах) вокруг вектора (x,y,z).

Код программы:

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls,
  ExtCtrls, OpenGLContext, GL, GLU;

type

  { TForm1 }

  TForm1 = class(TForm)
    Button1: TButton;
    OpenGLControl1: TOpenGLControl;
    Timer1: TTimer;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
    tri_rotationx: GLFloat;
    tri_rotationy: GLFloat;
    tri_rotationz: GLFloat;
    Speed:          Double;
  end;

var
  Form1: TForm1;

implementation

{$R *.lfm}

{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject);
begin
  Timer1.Enabled := true;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Timer1.Enabled  := false;
  Timer1.Interval := 100;
  Speed := 1;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  glClearColor(0.6, 0.98, 0.6, 1.0);       // сделали зеленоватый фон
```

```
  glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT);
  glEnable(GL_DEPTH_TEST);

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(45.0, double(width) / height, 0.1, 100.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();

  glTranslatef(0.0, 0.0,-6.0);
  glRotatef(tri_rotationx, tri_rotationy, tri_rotationz, 0.0);

  glBegin(GL_TRIANGLES);
          glColor3f(0.52,0.44,1.0);// Сделали боковую сторону фиолетовой

          glVertex3f( 1.0, -1.0,-1.0);
          glVertex3f(1.0, -1.0,1.0);
          glVertex3f(0.0, 1.0, 0.0);
  glEnd();

  glBegin(GL_TRIANGLES);
          glColor3f(1.0,0.84,0.0);  // Сделали боковую сторону желтой

          glVertex3f( 1.0,-1.0, 1.0);
          glVertex3f(-1.0,-1.0, 1.0);
          glVertex3f(0.0,1.0,0.0);
  glEnd();

  glBegin(GL_TRIANGLES);
         glColor3f(0.94,0.5,0.5);// Сделали сторону  розовой

          glVertex3f(-1.0,-1.0,1.0);
          glVertex3f(-1.0, -1.0,-1.0);
          glVertex3f(0.0,1.0,0.0);
     glEnd();

  glBegin(GL_TRIANGLES);
          glColor3f(0.0,1.0,0.0);  // Сделали сторону  светло зеленой

          glVertex3f(-1.0,-1.0,-1.0);
          glVertex3f(1.0,-1.0,-1.0);
          glVertex3f(0.0, 1.0,0.0);
  glEnd();

  glBegin(GL_QUADS);// основание пирамиды
          glColor3f(1.0,0.51,0.28); // сделали основание рыжим

          glVertex3f( 1.0,-1.0, 1.0);
          glVertex3f(-1.0,-1.0, 1.0);
          glVertex3f(-1.0,-1.0,-1.0);
          glVertex3f( 1.0,-1.0,-1.0);
  glEnd();

  tri_rotationx += 5.15 * Speed;
  tri_rotationy += 5.15 * Speed;
  tri_rotationz += 20.0 * Speed;

  OpenGLControl1.SwapBuffers;
end;

end.
```

Лабораторная работа № 8. Создание трехмерной модели на языке VRML

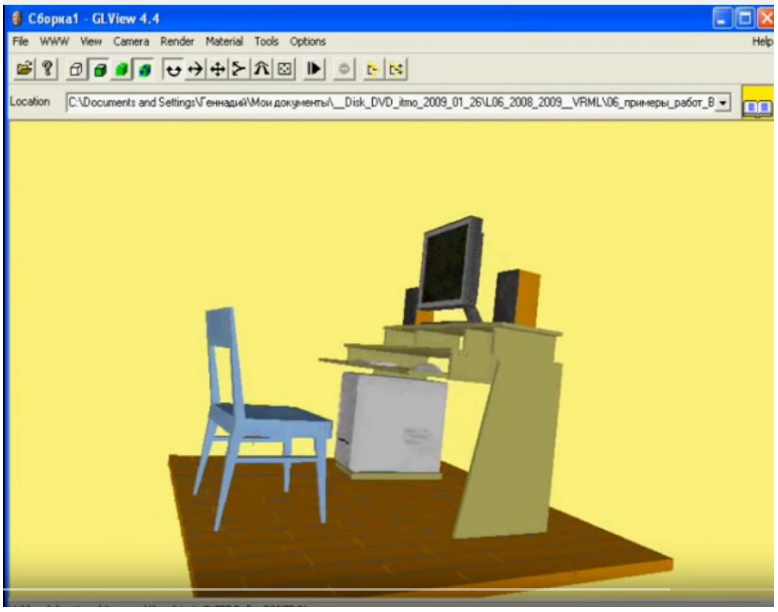
Данная лабораторная работа нужна студенту для преобретения навыков работы с трехмерной графикой.

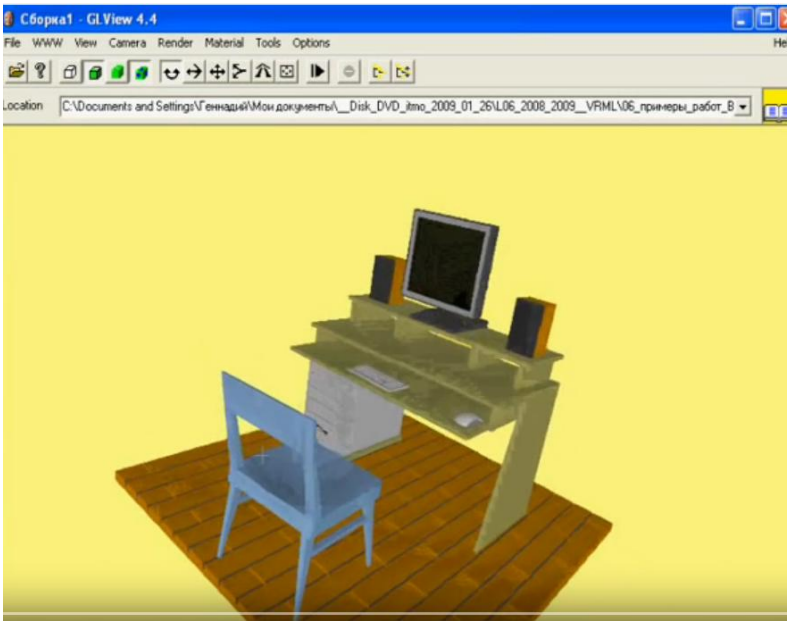
Цели работы

- Познакомиться с конкретным примером выполнения программы средствами VRML.
- Выполнить задание по указанию преподавателя, оформить файл отчета, сдать исходные и исполняемый файл.

Задание

По заданию своего номера варианта создать параметрическую трехмерную модель детали средствами языка VRML. Написать текст программы. Привести рисунок копии экрана для своего варианта.





Пояснение

Прежде чем начать выполнять задание рассмотрим основные понятия языка программирования VRML.

Язык VRML (Virtual Realty Modelling Languagy) предназначен для описания трехмерных изображений и оперирует объектами, описывающими геометрические фигуры и их расположение в пространстве. Vrml-файл представляет собой обычный текстовый файл, интерпретируемый браузером. Поскольку большинство браузеров не имеет встроенных средств поддержки vrml, для просмотра Vrml-документов необходимо подключить вспомогательную программу - Vrml-браузер, например, Live3D или Cosmo Player. Как и в случае с HTML, один и тот же vrml-документ может выглядеть по-разному в

разных VRML-браузерах. Кроме того, многие разработчики VRML-браузеров добавляют нестандартные расширения VRML в свой браузер.

Существует немало VRML-редакторов, делающих удобней и быстрее процесс создания VRML-документов, однако несложные модели можно создать при помощи самого простого текстового редактора.

В VRML приняты следующие единицы измерения: расстояние и размер: метры, углы: радианы, остальные значения: выражаются, как часть от 1. Координаты берутся в трехмерной декартовой системе координат.

Для того, чтобы VRML-браузер распознал файл с VRML-кодом, в начале файла ставится специальный заголовок - file header:

```
#VRML V1.0 ascii
```

Такой заголовок обязательно должен находиться в первой строке файла, кроме того, перед знаком диеза не должно быть пробелов.

В VRML определены четыре базовые примитивные фигуры: куб (верней не куб, а прямоугольный параллепипед), сфера, цилиндр и конус. Эти фигуры называются примитивами (primitives). Набор примитивов невелик, однако комбинируя их, можно строить достаточно сложные трехмерные изображения. Рассмотрим поподробней каждый из примитивов.

Куб. Возможные параметры: width - ширина, height - высота, depth - глубина.

```
Cube {
  width  2   # ширина
  height 3   # высота
  depth  1   # глубина
   }
```

Сфер. Параметр у сферы только один, это radius.

```
Sphere {
  radius 1   # радиус
    }
```

Конус. Возможные параметры: bottomRadius - радиус основания, height - высота, parts - определяет, какие части конуса будут видны. Параметр parts может принимать значения ALL, SIDES или BOTTOM.

```
Cone {
  parts        ALL  #видны и основание, и боковая поверхность конуса
  bottomRadius 1    #радиус основания
  height       2    #высота
 }
```

Цилиндр. Для цилиндра можно задать параметры radius и height. Кроме того, с помощью параметра parts для цилиндра можно определить будут ли отображаться основания цилиндра и его боковая поверхность. Параметр parts может принимать значения ALL, SIDES, BOTTOM или TOP.

```
Cylinder {
   parts  ALL     #видны все части цилиндра
   radius 1       #радиус основания
   height 2       #высота цилиндра
     }
```

Кроме описанных выше примитивных фигур формы строятся с помощью точек, линий и граней. Использование точек (Points) , линий (Lines) и граней(Faces) позволяет создавать более сложные формы, чем примитивы и генерировать более реальные VRML миры. Формы, созданные с помощью точек, линий и граней имеют больше функциональных возможностей, чем примитивы.

Рассматриваемый принцип построения форм характеризуется тем, что координаты точек и связь между ними задаются отдельно. Данный метод описания форм характеризуется большой гибкостью. Описание геометрии форм происходит в два этапа:

1.Описание координат точек

2.Соеденение точек

Координаты точек описываются с помощью узла Coordinate:

Coordinate {
point [1.0 2.0 3.0 ,
4.0 1.5 5.3 ,
. . .]
}

Три типа узлов описывают геометрию форм, основанную на точках и связях между ними:

PointSet
IndexedLineSet
IndexedFaceSet

Все они имеют поле coord, значением которого является узел Coordinate . Рассмотрим эти

типы узлов более подробно.

Узел PointSet. Этот узел определяет массив точек в трехмерном пространстве в локальной

системе координат :

PointSet {
coord Coordinate { point [. . .] }
color . . .
}

Каждой точке может быть присвоен свой цвет . Он задается в поле color, которое мы

рассмотрим позже. Если поле color не определено, то используется цвет, заданный в поле

emissiveColor узла Material. Размер точек постоянный и его нельзя изменять.

Узел IndexedLineSet. Этот узел определяет множество ломанных линий в трехмерном

пространстве, соединяющих точки, описанные в поле coord :

IndexedLineSet {
coord Coordinate { point [. . .] }
coordIndex [1 , 0 , 2 , -1 , . . .]
. . .
}

Поле coordIndex описывает соединенения между точками, описанными в поле coord . Числа в поле coordIndex являются индексами точек в поле coord :

coordIndex [1 , 0 , 3 , -1 , . . .]

При этом справедливы следующие условия:

порядок точек произвольный; точки индексируются, начиная с нуля; ломанная линия может

содержать несколько точек; конец ломанной определяется числом -1; можно описывать множество ломанных.

Узел IndexedFaceSet. Этот узел определяет форму в трехмерном пространстве, созданную из граней, полученных путем соединения по периметру грани точек, описанных в поле coord.

IndexedFaceSet {
coord Coordinate { point [. . .] }
coordIndex [1 , 0 , 2 , -1 , . . .]
. . .
}

Описание форм с помощью точек и связей между ними позволяет создавать сложные формы.

Узлы PointSet, IndexedLineSet и IndexedFaceSet описывают геометрию форм с помощью точек.

Решение

Для решения данной задачи достаточно использовать только два примитива: массив точек и массив граней. Особое внимание необходимо обратить на соблюдение обязательного условия выпуклости многоугольника. Для создания многоугольника

необходимы трехмерные значения массива координат вершин, заданных в примитиве вершина (point []) и массива индексов точек, из которых состоят грани(coordIndex []).

Таблица 1 Значения координат вершин детали

Номер вершины	X	Y	Z
0	0.0	0.0	0.0
1	0.0	20.0	10.0
2	90.0	20.0	10.0
3	90.0	20.0	30.0
4	70.0	20.0	30.0
…			
24	0.0	60.0	30.0

Таблица 2 Индексы граней

Номер вершины, входящий в грань				
Номер грани	№1	№2	№3	№4
1	1	2	14	13
2	2	3	15	14
3	3	4	16	15
4	4	5	17	16
5	5	6	18	17
…				
18	20	21	22	23

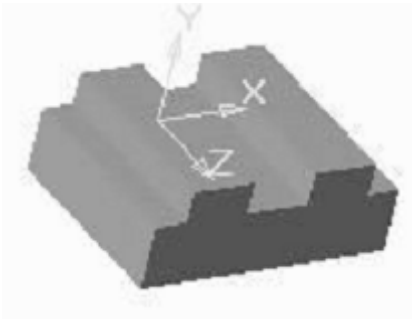


Рисунок задания

Текст программы

Для упрощения работы над заданием студентам выдается электронный вариант выполненного задания для кубика (с координатами для восьми граней). От студента требуется почесть задание и вставить свои координаты.

Листинг 1 Текст программы создания трехмерной модели.

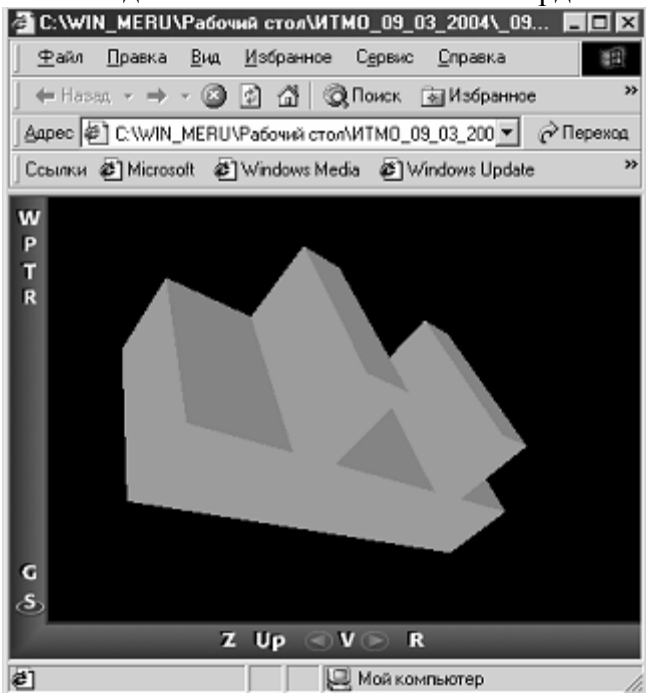
```
#VRML V2.0 utf8     ##############################
# Выполнил студент 1511 группы Петров С.Ю.,, вариант 03
# дата 09.03.2004, файл g151103S3.wrl
Group {       children [
 Shape {
    appearance Appearance {
     material DEF _DefMat Material {
     }
  }
    geometry IndexedFaceSet {
```

```
      coord Coordinate {
       point [
# Массив точек #############
 0.0  0.0 0.0,
  0.0  20.0  10.0,
 90.0  20.0  10.0,
 90.0  20.0  30.0,
 70.0  20.0  30.0,
 70.0  20.0  50.0,
 60.0  20.0  50.0,
 60.0  20.0  30.0,
 30.0  20.0  30.0,
30.0  20.0  50.0,
 20.0  20.0  50.0,
 20.0  20.0  30.0,
  0.0  20.0  30.0,
###################
  0.0  60.0  10.0,
 90.0  60.0  10.0,
 90.0  60.0  30.0,
 70.0  60.0  30.0,
 70.0  60.0  50.0,
 60.0  60.0  50.0,
 60.0  60.0  30.0,
 30.0  60.0  30.0,
 30.0  60.0  50.0,
 20.0  60.0  50.0,
 20.0  60.0  30.0,
  0.0  60.0  30.0
############################
 ]
   }
      solid FALSE
      creaseAngle 0.5
      coordIndex [
# Массив граней ########
 1, 2, 14, 13, -1,
 2, 3, 15, 14, -1,
3, 4, 16, 15, -1,
 4, 5, 17, 16, -1,
 5, 6, 18, 17, -1,
 6, 7, 19, 18, -1,
 7, 8, 20, 19, -1,
 8, 9, 21, 20, -1,
 9,10, 22, 21, -1,
10,11, 23, 22, -1,
11,12, 24, 23, -1,
12, 1, 13, 24, -1,
###################
1, 2,  3, 12, -1,
 4, 5,  6,  7, -1,
 8, 9, 10, 11, -1,
13,14, 15, 24, -1,
16,17, 18, 19, -1,
20,21, 22, 23, -1
#########################
```

```
]
  }
 }
   ]
}
```

Копия экрана

После подстановки своих массивов координат в программу получаем решение.



Экранная копия работы программы

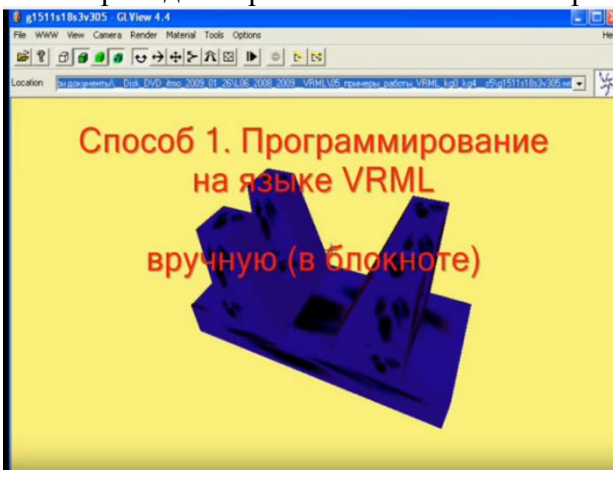
Оформление результатов работы

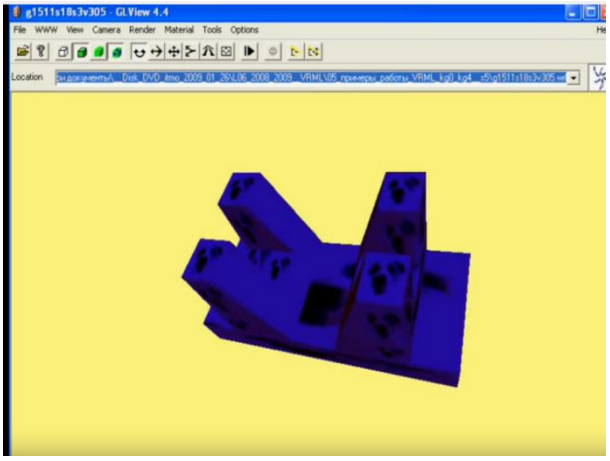
Результаты работы оформляются в виде отчета по выдаваемому в электронном виде образцу.

В отчет входят следующие разделы: задание, решение, текст программы, копия экрана, литература и интернет- ссылки. К отчету прилагается исходные и исполняемые файлы программы. Нумерация файлов и папки осуществляется по личному коду выдаваемому преподавателем. Все материалы сдаются в электронном виде. Допускается распечатка отчетных материалов.

Например, имя папки с файлами: g151103S3, где g- студенческая группа, 1511- номер группы, 03- номер студента по списку, S- самостоятельная работа, 3-порядковый номер работы. В папке находятся следующие файлы: g151103L3.doc-файл отчета по работе (образец выдается преподавателем), исходный файл -Ug151103S3.wrl.

Ниже приведен вариант выполнения лабораторной работы.

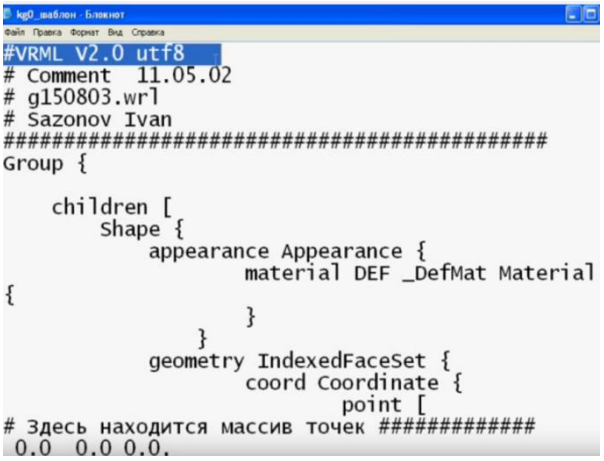




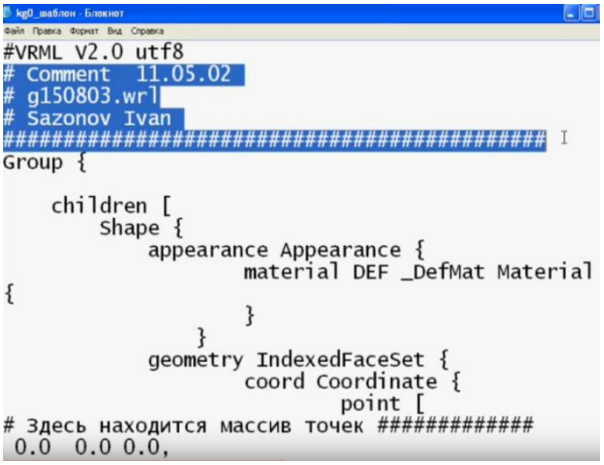

```
#VRML V2.0 utf8
# Comment  11.05.02
# g150803.wrl
# Sazonov Ivan
##########################################
Group {

    children [
        Shape {
            appearance Appearance {
                    material DEF _DefMat Material
{
                    }
                }
            geometry IndexedFaceSet {
                    coord Coordinate {
                            point [
# Здесь находится массив точек #############
 0.0  0.0 0.0,
```

```
# Здесь находится массив точек #############
 0.0  0.0 0.0,
 50.0  10.0  10.0,
 10.0  10.0  10.0,
 10.0  50.0  10.0,
 50.0  50.0  10.0,
 40.0  20.0  20.0,
 20.0  20.0  20.0,
 20.0  40.0  20.0,
 40.0  40.0  20.0,
###################

 ]
                        }
                    solid FALSE
                    creaseAngle 0.5
                    coordIndex [
# Здесь находится массив граней ########
# Здесь находится массив граней ########
```

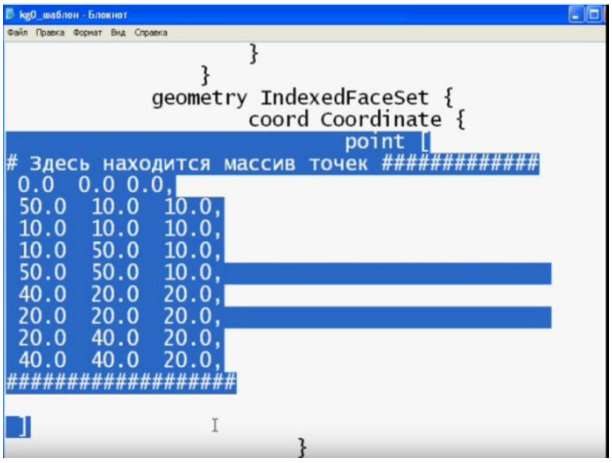
```
 ]
                        }
                    solid FALSE
                    creaseAngle 0.5
                    coordIndex [
# Здесь находится массив граней ########
 1, 2, 3, 4, -1,
 5, 6, 7, 8, -1,
 1, 2, 6, 5, -1,
 2, 6, 7, 3, -1,
 3, 7, 8, 4, -1,
 1, 5, 8, 4, -1,
#########################
]
                }
        }
    ]
}
```



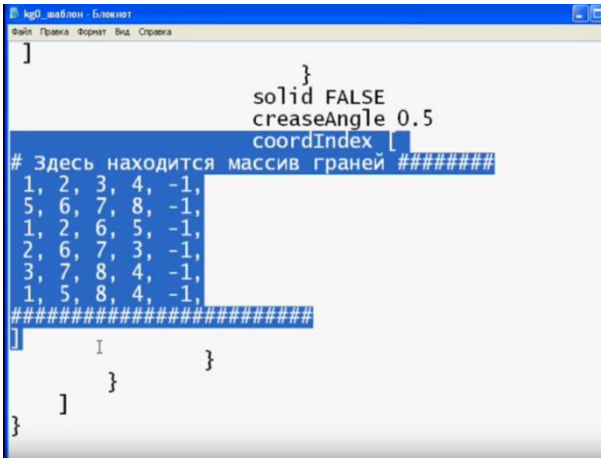
```
#VRML V2.0 utf8
# Comment  11.05.02
# g150803.wrl
# Sazonov Ivan
##########################################
Group {

    children [
        Shape {
            appearance Appearance {
                    material DEF _DefMat Material
{
                    }
                }
            geometry IndexedFaceSet {
                    coord Coordinate {
                            point [
# Здесь находится массив точек #############
 0.0  0.0 0.0,
```



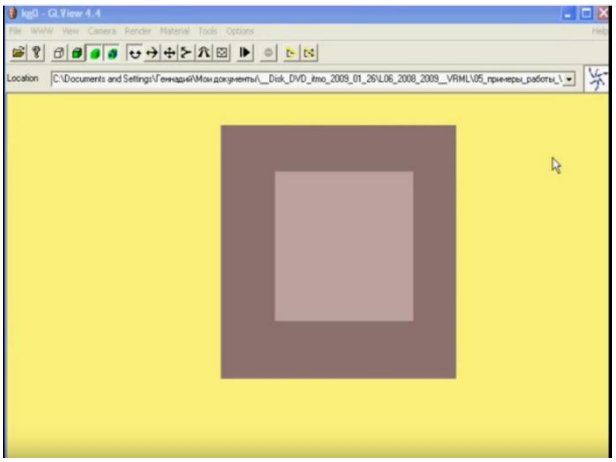
```
#VRML V2.0 utf8
# Comment  11.05.02
# g150803.wrl
# Sazonov Ivan
##########################################
Group {

    children [
        Shape {
            appearance Appearance {
                    material DEF _DefMat Material
{
                    }
                }
            geometry IndexedFaceSet {
                    coord Coordinate {
                            point [
# Здесь находится массив точек #############
 0.0  0.0 0.0,
```

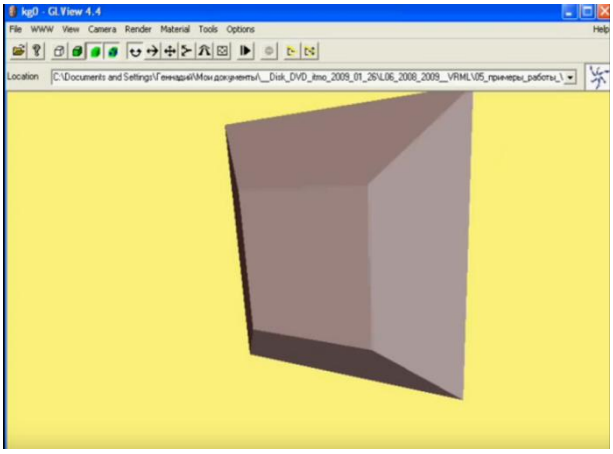


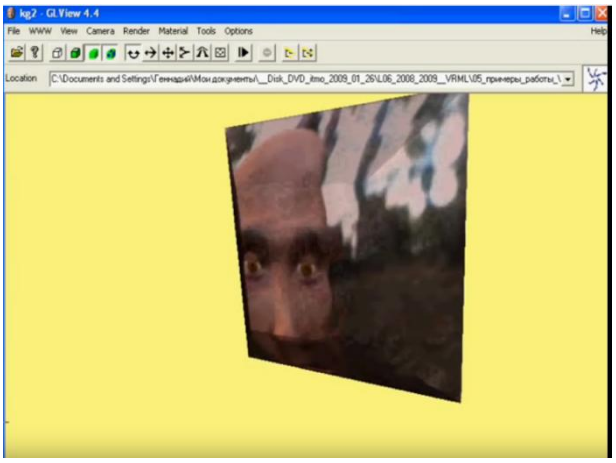
```
                    }
                }
            geometry IndexedFaceSet {
                    coord Coordinate {
                            point [
# Здесь находится массив точек #############
 0.0  0.0 0.0,
 50.0  10.0  10.0,
 10.0  10.0  10.0,
 10.0  50.0  10.0,
 50.0  50.0  10.0,
 40.0  20.0  20.0,
 20.0  20.0  20.0,
 20.0  40.0  20.0,
 40.0  40.0  20.0,
###################

]
                    }
```

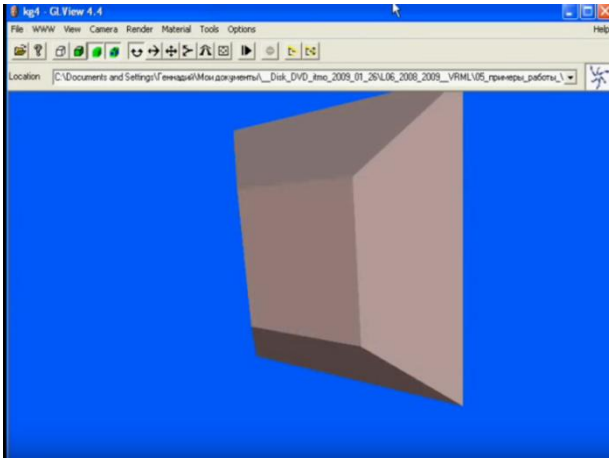


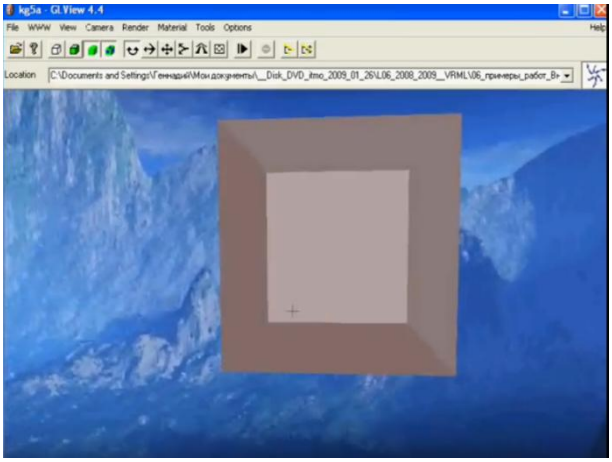
```
 ]
                    }
                solid FALSE
                creaseAngle 0.5
                coordIndex [
# Здесь находится массив граней ########
 1, 2, 3, 4, -1,
 5, 6, 7, 8, -1,
 1, 2, 6, 5, -1,
 2, 6, 7, 3, -1,
 3, 7, 8, 4, -1,
 1, 5, 8, 4, -1,
#########################
]
            }
        }
    ]
}
```


kg0 - GLView 4.4
File WWW View Camera Render Material Tools Options Help
Location C:\Documents and Settings\Геннадий\Мои документы__Disk_DVD_itmo_2009_01_26\L06_2008_2009__VRML\05_примеры_работы_\

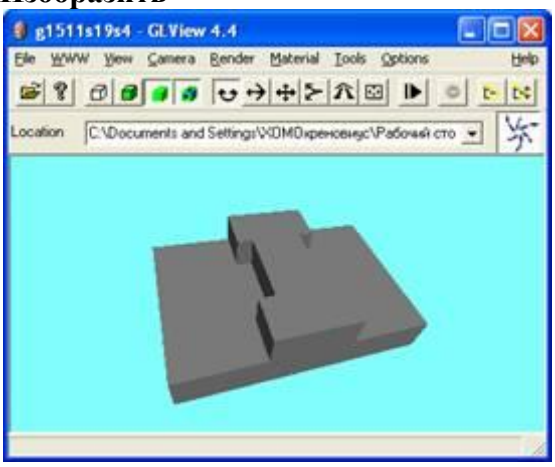

kg0 - GLView 4.4
File WWW View Camera Render Material Tools Options Help
Location C:\Documents and Settings\Геннадий\Мои документы__Disk_DVD_itmo_2009_01_26\L06_2008_2009__VRML\05_примеры_работы_\


kg2 - GLView 4.4
File WWW View Camera Render Material Tools Options Help
Location C:\Documents and Settings\Геннадий\Мои документы__Disk_DVD_itmo_2009_01_26\L06_2008_2009__VRML\05_примеры_работы_\


kg4 - GLView 4.4
File WWW View Camera Render Material Tools Options Help
Location C:\Documents and Settings\Геннадий\Мои документы__Disk_DVD_itmo_2009_01_26\L06_2008_2009__VRML\05_примеры_работы_\


kg5a - GLView 4.4
File WWW View Camera Render Material Tools Options
Help
Location

Изобразить

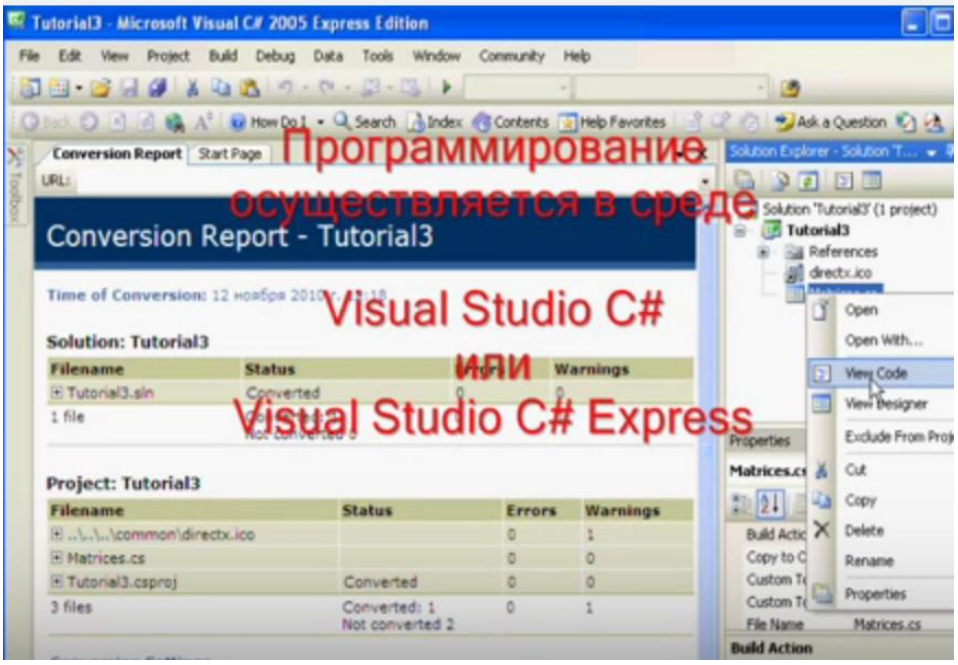

g1511s19s4 - GLView 4.4
File WWW View Camera Render Material Tools Options
Help
Location
C:\Documents and Settings\VDM0хренсвинус\Рабочий сто

Лабораторная работа № 9. Создание трехмерной модели с применением DirectX
Открыть DirectX Simple Browser.

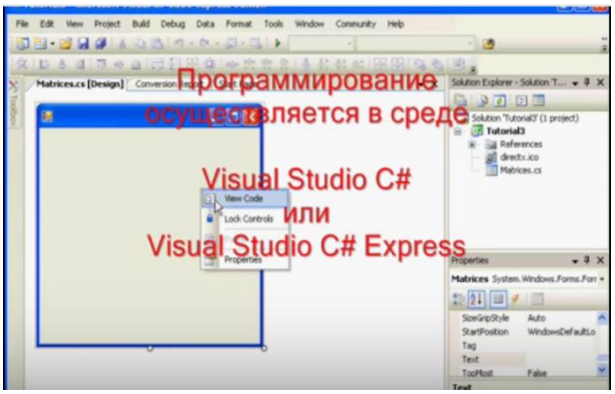
На левой боковой панели поставить галочки рядом со словами:
C#
Tutorials

В основном окне переместиться до надписи «Projekt File». Нажать на нее.

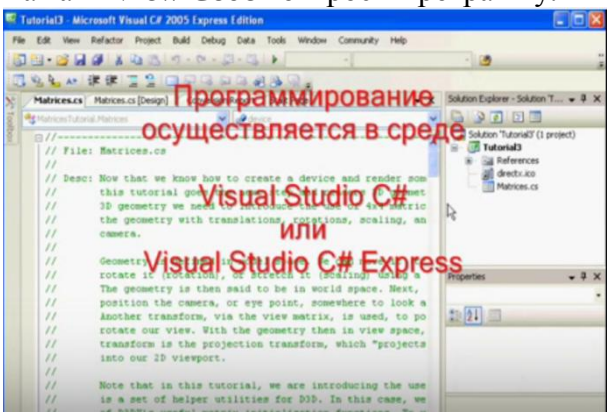
Выйдет окно проводника. Из него необходимо выбрать файл «Tut03_Matrices».



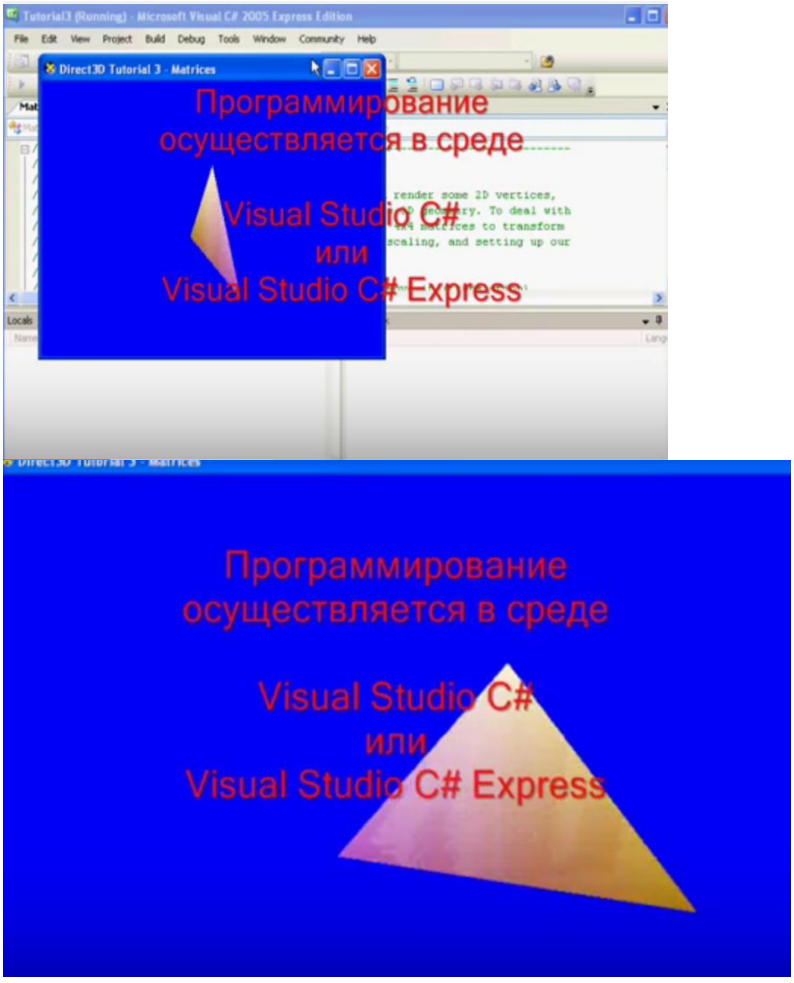
Выбрать на вкладке «View Disign».Появится новый документ, с которым необходимо работать.



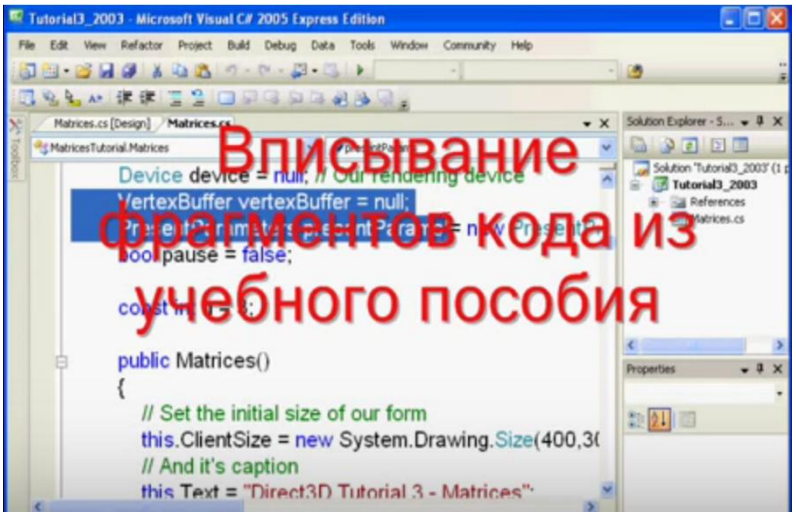
Нажав «View Code» откроем программу.

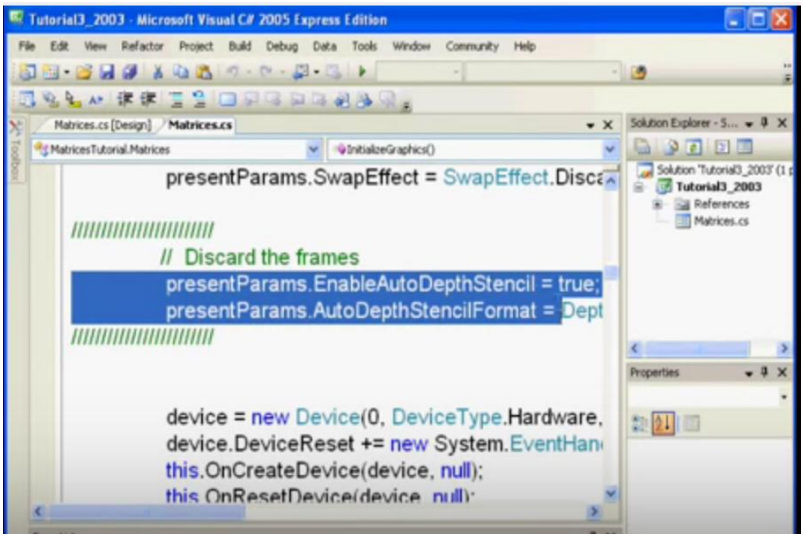


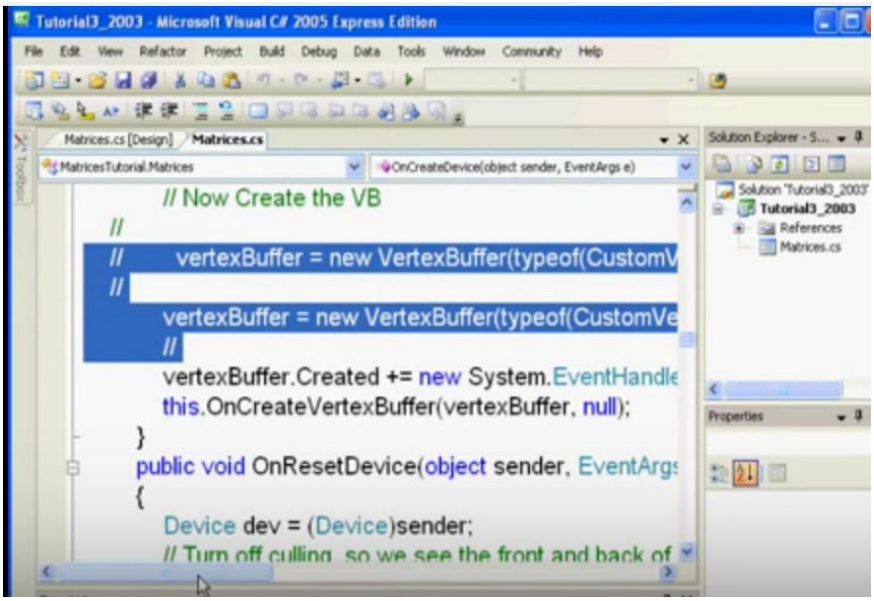
Сохраните работу. Далее перейдите в окно графической библиотеки нажав кнопку  .

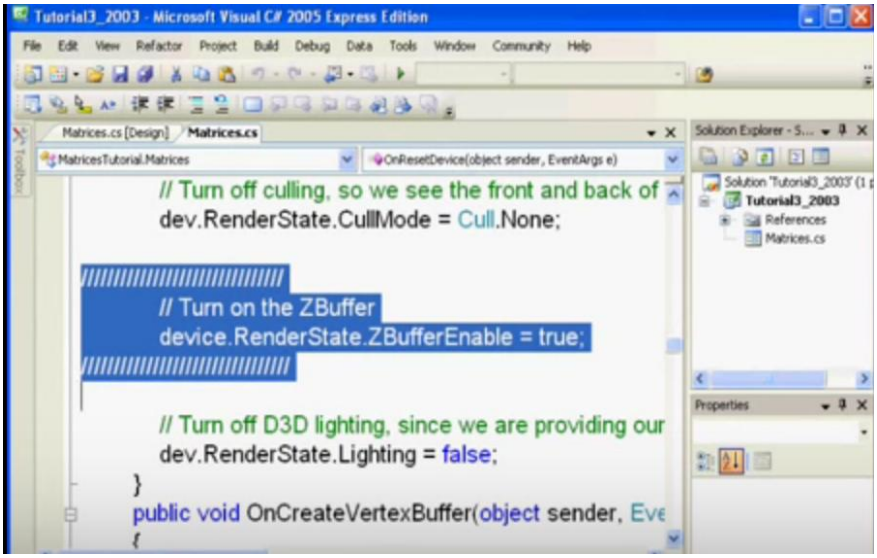


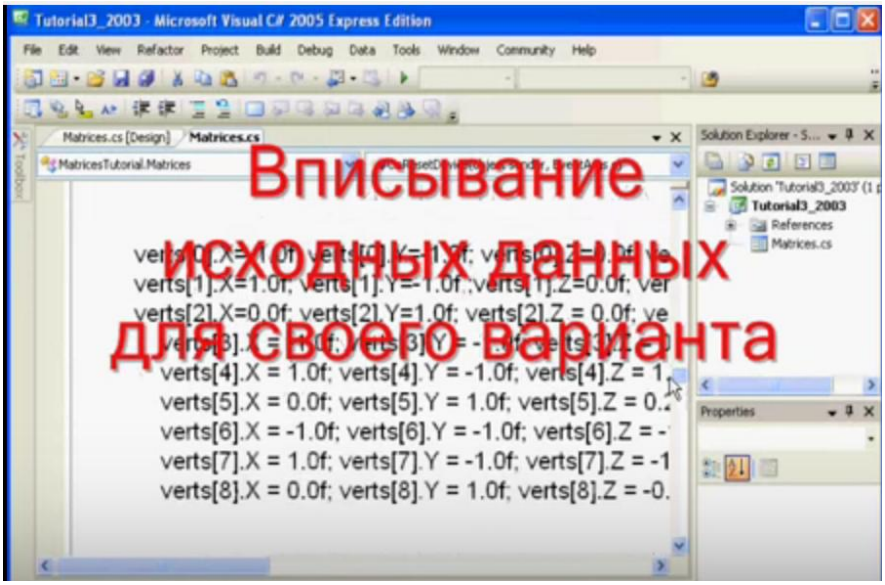
Фигура будет вращаться вокруг собственной оси.
Далее осуществим вписывание кода из курса лекций

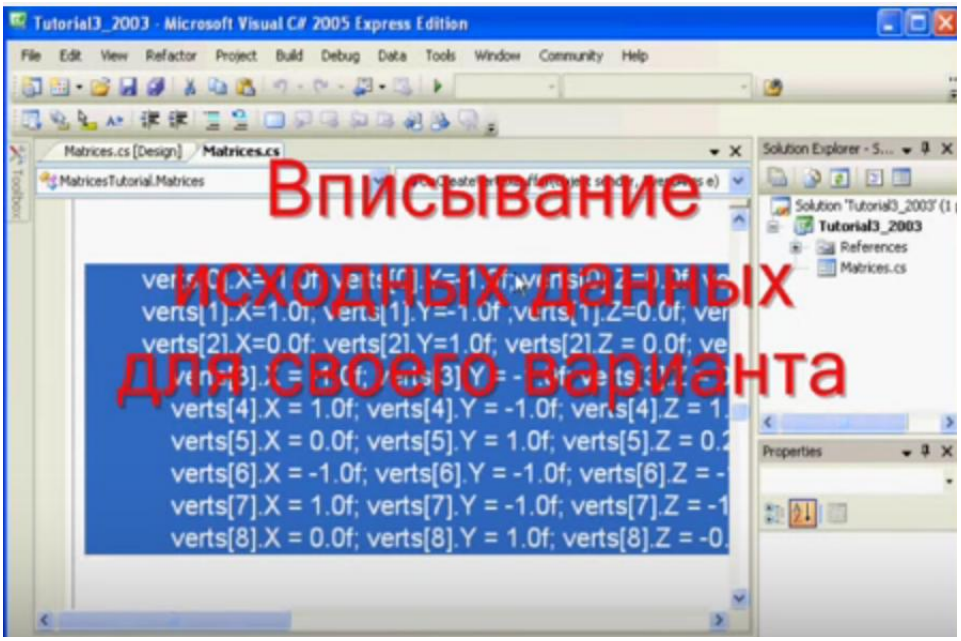

Tutorial3_2003 - Microsoft Visual C# 2005 Express Edition
Вписывание фрагментов кода из учебного пособия
Device device = null; // Our rendering device
VertexBuffer vertexBuffer = null;
bool pause = false;
public Matrices()
{
// Set the initial size of our form
this.ClientSize = new System.Drawing.Size(400,3
// And it's caption
this.Text = "Direct3D Tutorial 3 - Matrices";


Tutorial3_2003 - Microsoft Visual C# 2005 Express Edition
presentParams.SwapEffect = SwapEffect.Disca
// Discard the frames
presentParams.EnableAutoDepthStencil = true;
presentParams.AutoDepthStencilFormat = Dept
device = new Device(0, DeviceType.Hardware,
device.DeviceReset += new System.EventHan
this.OnCreateDevice(device, null);
this.OnResetDevice(device, null);

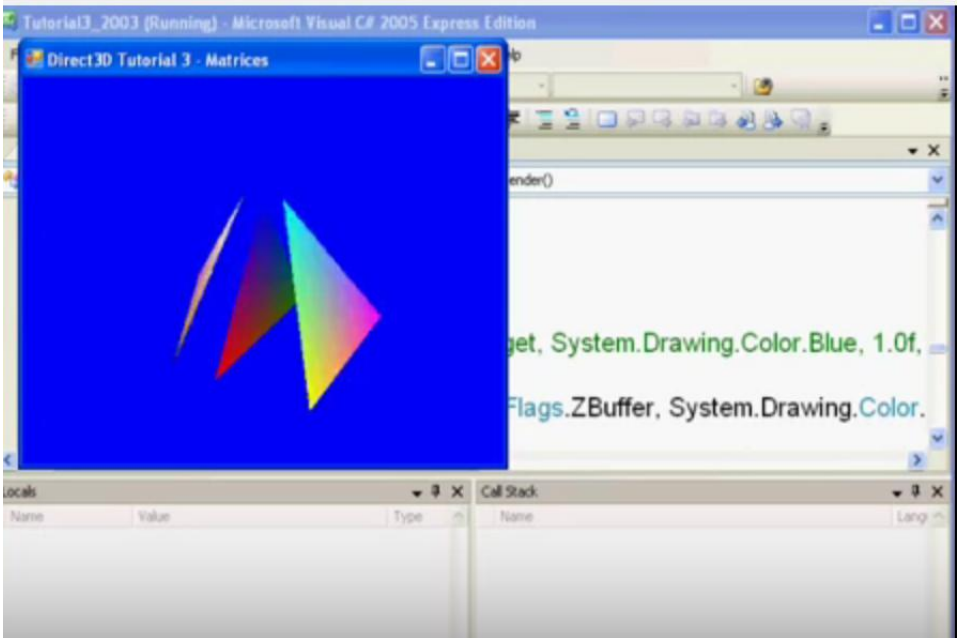

Tutorial3_2003 - Microsoft Visual C# 2005 Express Edition
// Now Create the VB
vertexBuffer = new VertexBuffer(typeof(CustomV
vertexBuffer = new VertexBuffer(typeof(CustomVe
vertexBuffer.Created += new System.EventHandle
this.OnCreateVertexBuffer(vertexBuffer, null);
}
public void OnResetDevice(object sender, EventArgs
{
Device dev = (Device)sender;
// Turn off culling, so we see the front and back of

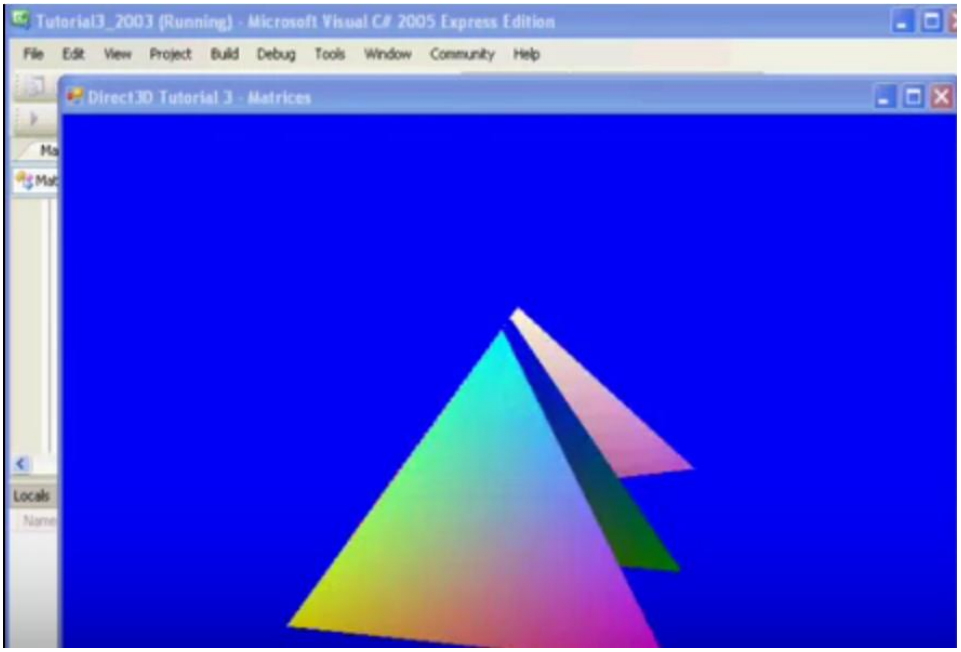




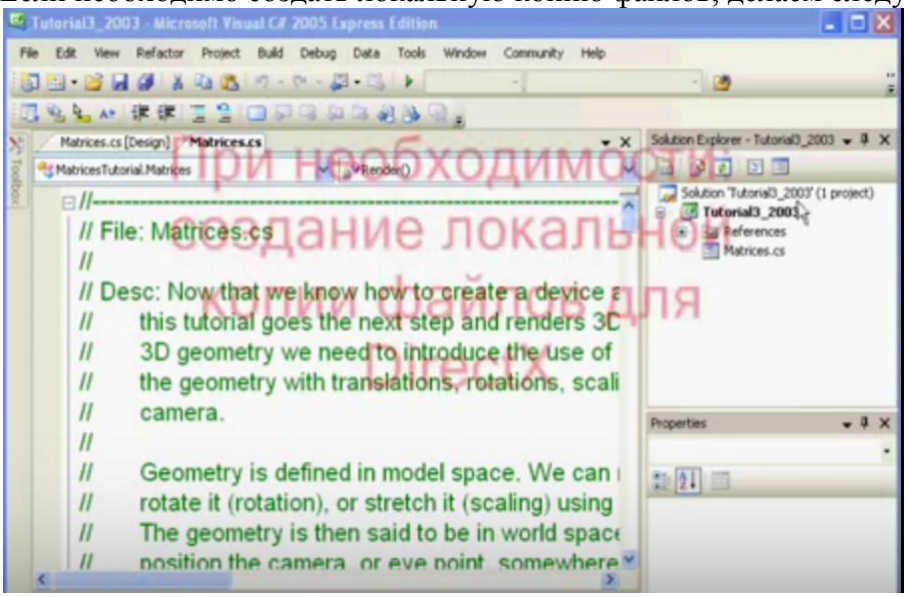


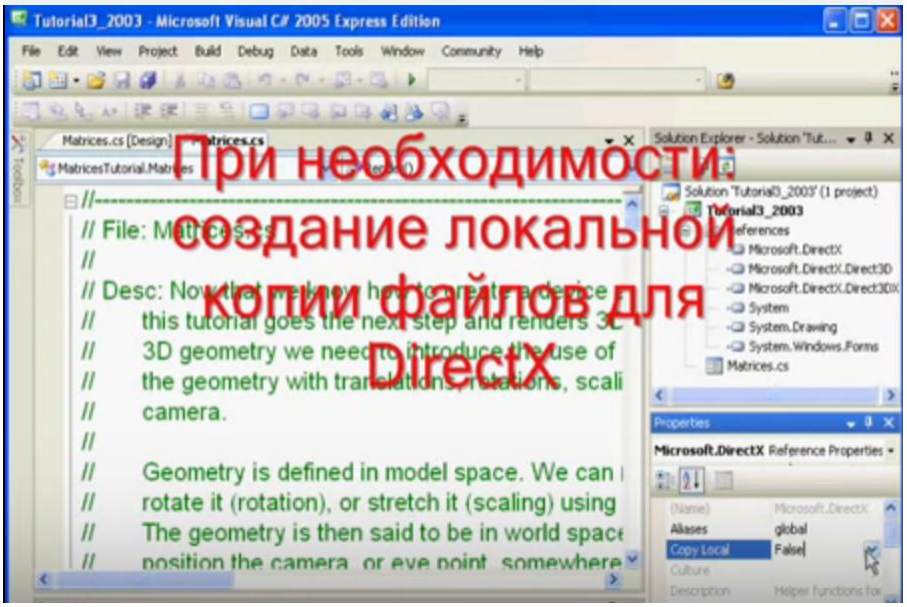
Сохраните работу. Далее перейдите в окно графической библиотеки нажав кнопку  .

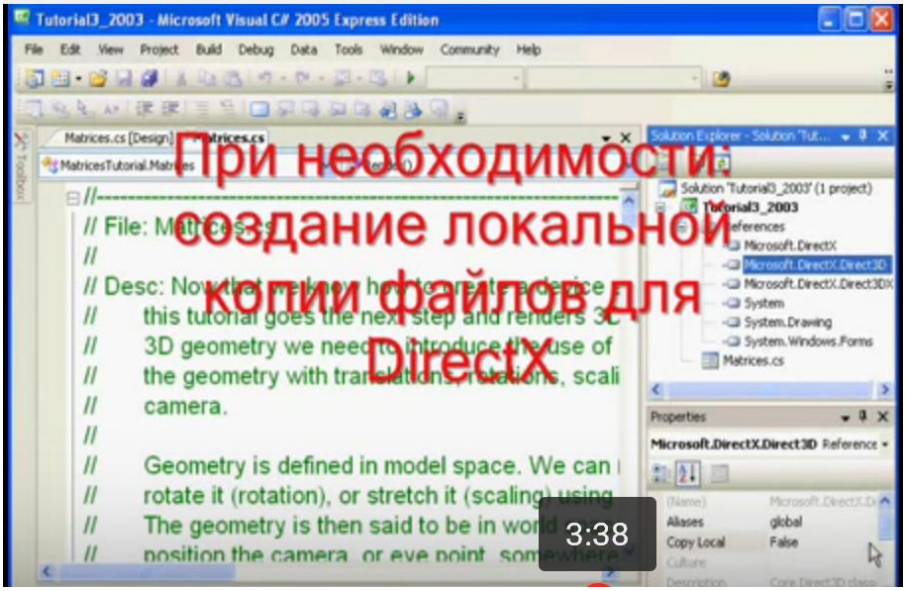


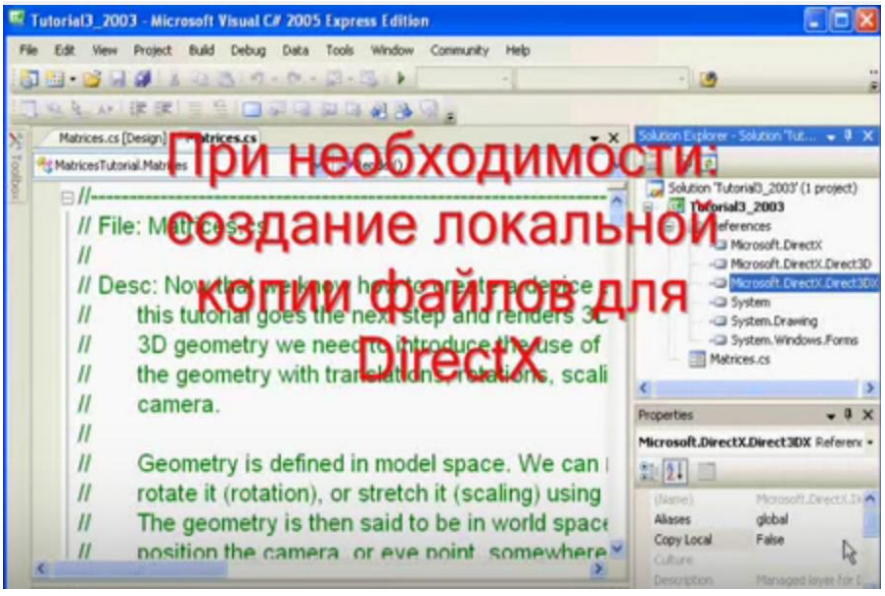


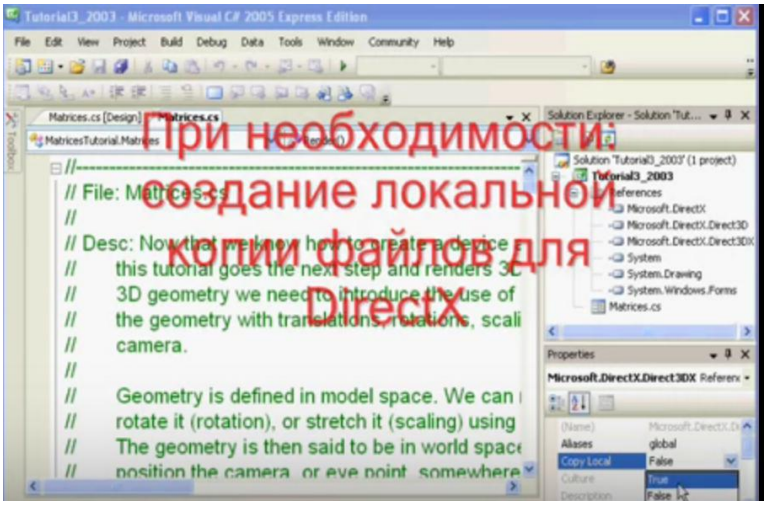
Если необходимо создать локальную копию файлов, делаем следующие действия.



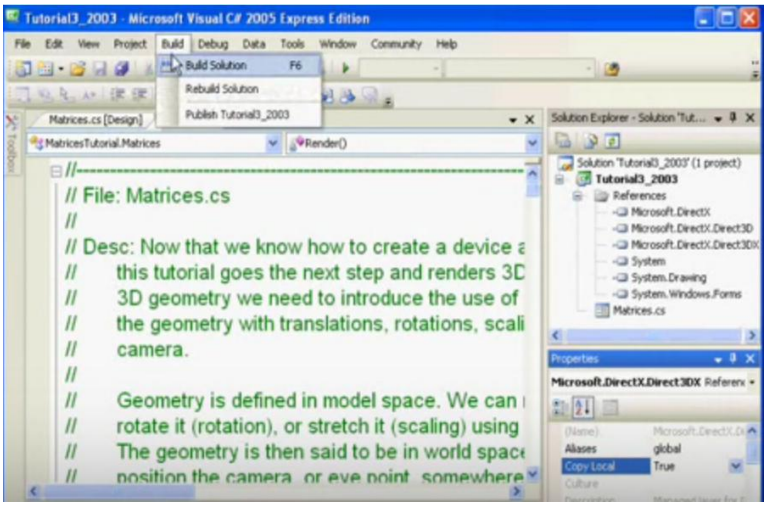

Tutorial3_2003 - Microsoft Visual C# 2005 Express Edition
При необходимости:
создание локальной
копии файлов для
DirectX


Tutorial3_2003 - Microsoft Visual C# 2005 Express Edition
При необходимости:
создание локальной
копии файлов для
DirectX
3:38

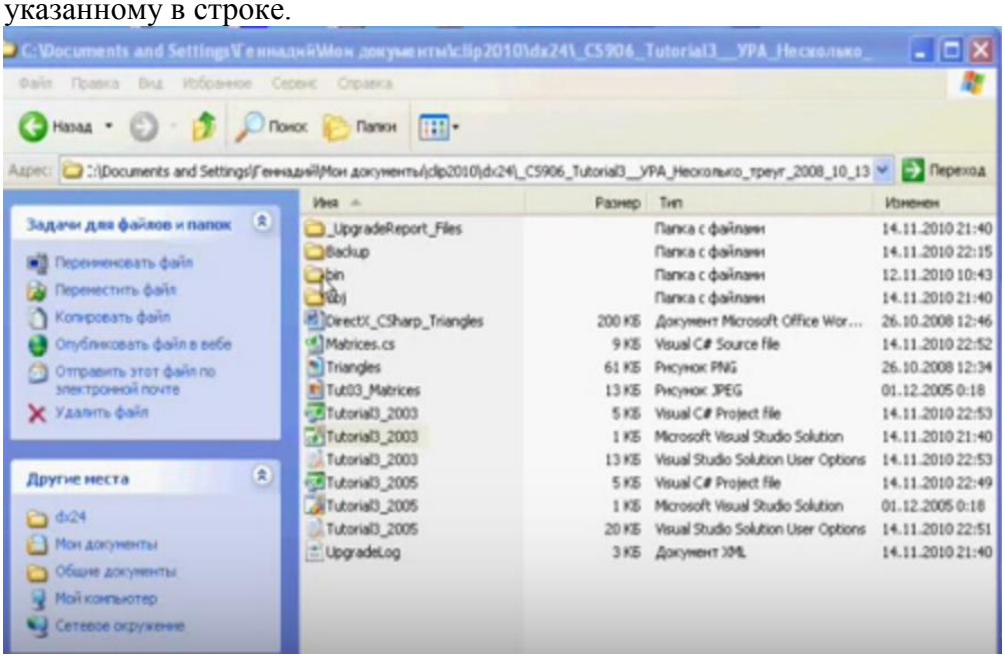

Tutorial3_2003 - Microsoft Visual C# 2005 Express Edition
При необходимости:
создание локальной
копии файлов для
DirectX



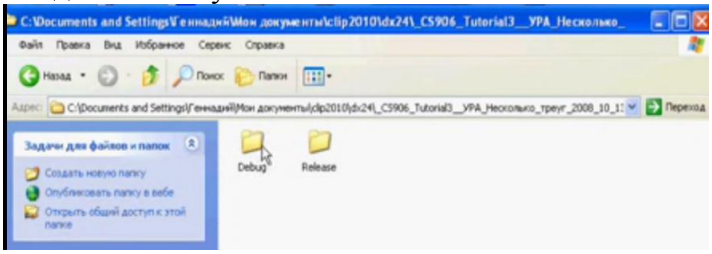
Сохраните работу.



В верхней панели выберите «Buld» - «Rebuld Solution». Сверните окно. Перейдите по адресу указанному в строке.

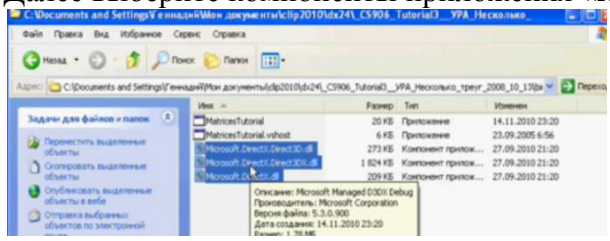


Войдите в папку «bin».

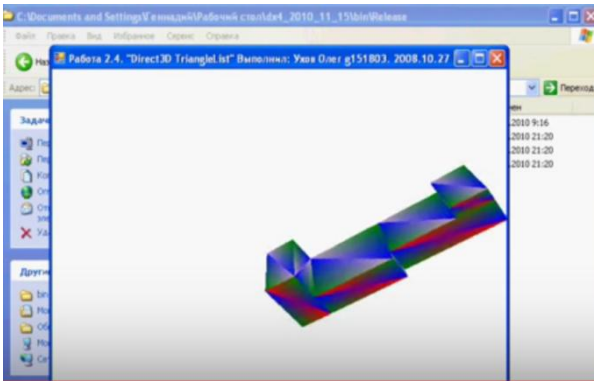


Далее войдите в папку «Release».

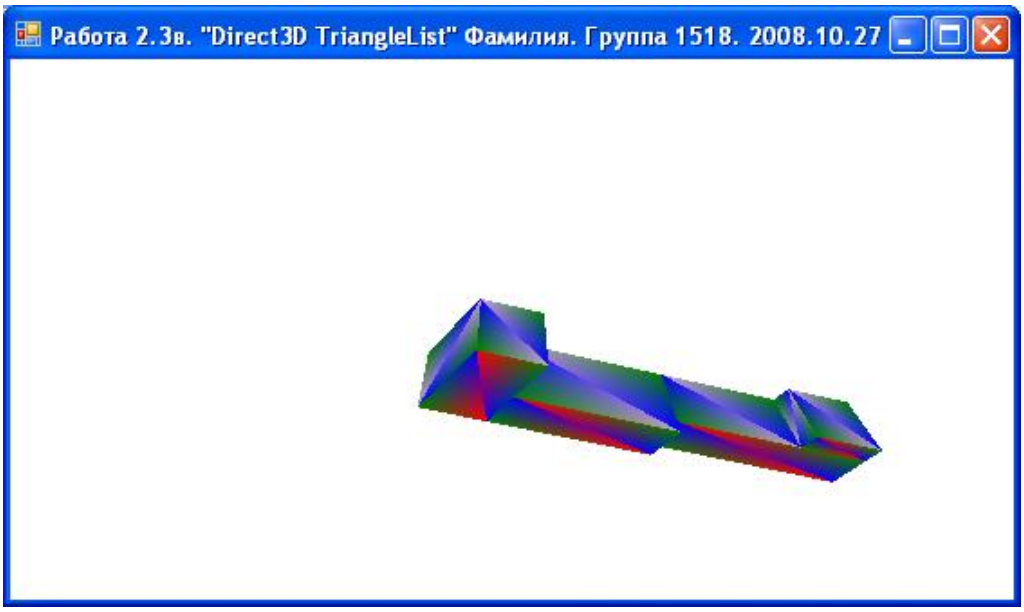
Далее выберите компоненты приложения «Matrics Tutorial».



Нажав на первый не выделенный файл получим следующее изображение.



Изображение будет вращаться вокруг собственной оси.



4. Формы контроля лабораторных работ

Подробно рассмотрены в фонде оценочных средств по дисциплине.

5. Учебно-методическое и информационное обеспечение дисциплины

а) основная литература

1. Гаврилов, М.В. Информатика и информационные технологии: учебник для бакалавров/ М.В. Гаврилов, В.А. Климов. – 3-е изд., перераб. и доп. – М.: Издательство Юрайт, 2013. – 378 с.

2. Дьяконов В.П. Maple 10/11/12/13/14 в математических расчетах[Электронный ресурс].- М.:ДМК Пресс, 2011. – 800 с. – Режим доступа http://e.lanbook.com/books/element.php?pl1_id=3034.

3. Дьяконов В.П. MATLAB R2007/2008/2009 для радиоинженеров[Электронный ресурс].- М.:ДМК Пресс, 2010. – 976 с. – Режим доступа http://e.lanbook.com/books/element.php?pl1_id=1180.

4. Илюшечкин В.М. Основы использования и проектирования базы данных: Учеб. пособие / В.М. Илюшечкин. -М.: Изд-во Юрайт, ИД Юрайт, 2010. - 213 с.

5. Поршнев С.В. Компьютерное моделирование физических процессов в пакете MATLAB: учебное пособие. - СПб.: Изд-во "Лань", 2011. - 736с.: ил. (+CD).

6. Прахов А.А. Blender: 3D-моделирование и анимация. Руководство для начинающих / А.А. Прахов. - СПб.: БХВ-Петербург, 2009. - 272с.: ил. + CD-ROM.

б) дополнительная литература:

1. Герасимов, А.А. Самоучитель КОМПАС-3D V12 / А.А. Герасимов. – СПб.: БХВ-Петербург, 2011. – 464с.: ил.

2. Дьяконов В.П. MATHEMATICA 5.1/5.2/6 в математических и научно-технических расчетах[Электронный ресурс].- М.:Солон- Пресс, 2008. – 744 с. – Режим доступа http://e.lanbook.com/books/element.php?pl1_id=13775.

3. Дьяконов В.П. Mathcad 8—12 для студентов [Электронный ресурс].- М.:Солон-Пресс, 2005. – 632 с. – Режим доступа http://e.lanbook.com/books/element.php?pl1_id=13711

4. ДьяконовВ.П. VisSim+Mathcad+MATLAB. Визуальное математическое моделирование [Электронный ресурс].- М.:Солон- Пресс, 2008. – 384 с. – Режим доступа http://e.lanbook.com/books/element.php?pl1_id=13679.

5. Илюшечкин В.Н. Основы использования и проектирования баз данных: учебное пособие / В.М. Илюшечкин. - М.: Изд-во Юрайт, 2011. - 213с.

6. Информатика: учебник для бакалавров / под ред. В.В. Трофимова. – 2-е изд., испр. и доп. – М.: Издательств Юрайт; ИД Юрайт, 2013. – 917 с.

7. Информатика: учебник для бакалавров / В.В. Трофимова. – М.: Изд-во «Юрайт», 2013. – 917с.

в) ресурсы сети «Интернет», программное обеспечение и информационно-справочные системы:

1. http://window.edu.ru/ - Каталог образовательных Internet- ресурсов;
2. http://www.cnews.ru - сетевое издание о высоких технологиях;
3. http://www.internet-technologies.ru/news – новости интернет-технологий;
4. http://www.compulenta.ru – интернет-издание. Новости из мира компьютеров;
5. http://www.google.ru – самая популярная в мире поисковая система;
6. http://www.yandex.ru – самая популярная российская поисковая система;
7. http://life-prog.ru – сайт для начинающих программистов;
8. http://kappasoft.narod.ru/info/3d/3d.htm - введение в 3D-программирование (курс лекций);
9. http://mihriutka2004.narod.ru/3DGraf.htm - программирование. Visual Basic. NET. Программирование.
10. http://vbzero.narod.ru – Visual Basic с нуля.
11. http://citforum.ru/programming/unix/ - программирование в среде Unix/Linux.
12. http://www.williamspublishing.com/PDF/5-8459-0549-4/part.PDF - лекция по трехмерному компьютерному моделированию.
13. http://www.programmersforum.ru – клуб программистов.
14. http://sernam.ru/book_java.php- Сухов, С.А. Электронный учебник по основам программирования на Java [Электронный ресурс].
15. http://www.frolov-lib.ru/programming/javasamples/vol8/vol8_4/index.html - библиотека примеров и приложений Java.
16. http://pmg.org.ru/nehe/ - уроки по OpenGL.
17. http://po28.on.ufanet.ru/10.html - экзамен САПР. Средства трехмерного моделирования.
18. https://www.youtube.com/watch?v=ECpAPCmjZ5U – приложения к лекциям по компьютерной графике и геометрии.

6. Материально-техническое обеспечение учебной дисциплины

№ п./п.	Наименование оборудованных учебных кабинетов, лабораторий	Перечень оборудования и технических средств обучения
1.	Аудитория 449	Сетевые персональные компьютеры в количестве 11 шт., мультимедийный проектор, интерактивная доска, принтер
1	Аудитория 451	Сетевые персональные компьютеры в количестве 12 шт., телевизор