Прикладные программные интерфейсы САПР

Интерактивное взаимодействие пользователя с КОМПАС. Диалоговые окна и вариантные изображения

Александр Иванович Сергеев
д-р техн. наук, профессор

Рассматриваемые вопросы

1 Введение в интерактивное взаимодействие пользователя с КОМПАС

2 Стандартные диалоговые окна

3 Отображение вариантов

1 Введение в интерактивное взаимодействие пользователя с КОМПАС

Многие команды RTW-библиотек не удается сделать полностью автоматическими и при выполнении этих команд приходится запрашивать данные у пользователя.

Операции ввода данных можно разделить на три группы:
- операции со стандартными диалоговыми окнами;
- операции с окном КОМПАС;
- операции с библиотечными диалоговыми окнами.

Примерами стандартных диалоговых окон являются окна сообщений, окна сохранения и открытия файлов. Для вызова этих окон в КОМПАС-МАСТЕР есть специальные методы.

1 Введение в интерактивное взаимодействие пользователя с КОМПАС

Среди операций с окном КОМПАС выделяются операции указания точки и направления (угла) на листе чертежа. Пользователя можно попросить указать мышью на некоторую точку или задать значение угла. Эти данные удобно использовать для создания некоторой библиотечной детали в заданном месте чертежа и с заданной ориентацией. С операциями задания местоположения тесно связан вопрос вариантного (фантомного) отображения. Когда пользователь размещает деталь на чертеже, в процессе перемещения мыши желательно также перемещать на чертеже некоторое предварительное изображение детали - вариантное изображение, которое пользователь может принять или отказаться от него.

1 Введение в интерактивное взаимодействие пользователя с КОМПАС

RTW-библиотеки могут выводить собственные диалоговые окна. При построении стандартных деталей может потребоваться указать параметры этой детали, если, например, ее можно строить в различных видах или с различными параметрами по ГОСТу. Подобные параметры удобно вводить в специальных диалоговых окнах, а затем учитывать при построении детали.

В диалоговом окне (рисунок 1) присутствуют стандартные элементы управления Windows, а также слайд с изображением детали в соответствии с текущими параметрами. На слайде пользователь сразу видит примерное изображение детали. Для отображения слайдов детали можно пользоваться BMP-слайдами, но более удобным средством являются векторные слайды КОМПАС-МАСТЕР, предназначенные для компактного представления команд для формирования изображения слайда.

1 Введение в интерактивное взаимодействие пользователя с КОМПАС

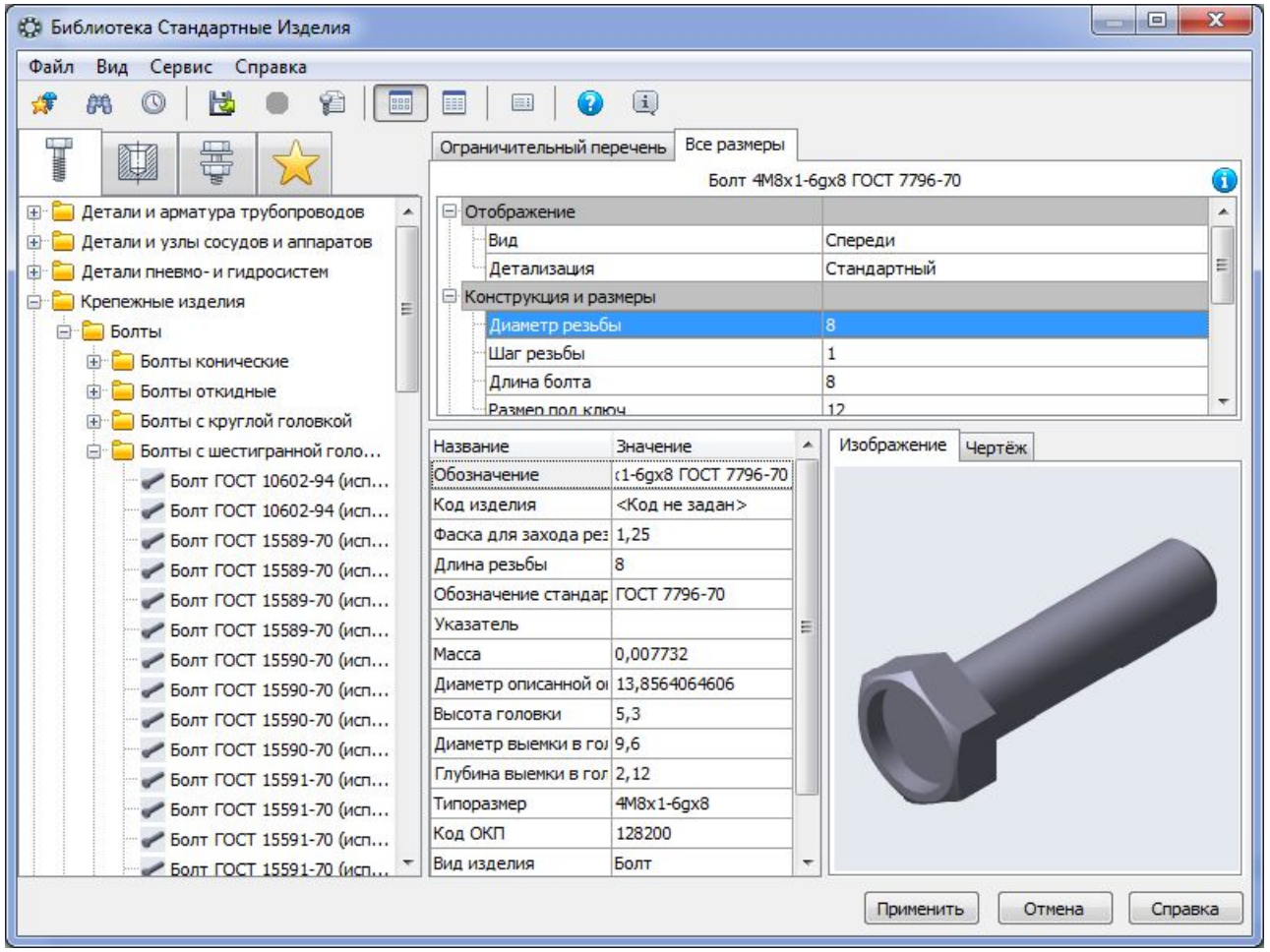


Рисунок 1 - Пример диалогового окна библиотеки для выбора параметров создания стандартной детали

1 Введение в интерактивное взаимодействие пользователя с КОМПАС

После того, как стандартная деталь нарисована и запомнена в модели чертежа, пользователю может потребоваться отредактировать ее. Чаще всего обеспечивается редактирование по двойному щелчку и редактирование с помощью характерных точек.

Двойным щелчком по детали вызывается библиотечное окно параметров детали (рисунок 1).

При однократном щелчке на нарисованной детали на ней могут появиться характерные точки (если библиотека их поддерживает и версия КОМПАС не ниже 5.11).

1 Введение в интерактивное взаимодействие пользователя с КОМПАС

Пользователь может перетаскивать характерные точки. С каждой из них связан один или несколько параметров, которые при перетаскивании автоматически изменяются и библиотека соответствующим образом перестраивает изображение детали.

Например, у винта (рисунок 2) есть 4 характерные точки: точка О - местоположение детали, А - угол наклона, L - длина винта и Dr - диаметр резьбы. Важно, что при перетаскивании точек, связанных с дискретными параметрами (такими, как длина винта или диаметр - они определены ГОСТом), эти параметры меняются именно дискретно, а не с шагом указателя мыши. Таким образом, характерные точки являются удобным механизмом для изменения параметров нарисованной библиотечной детали "по месту".

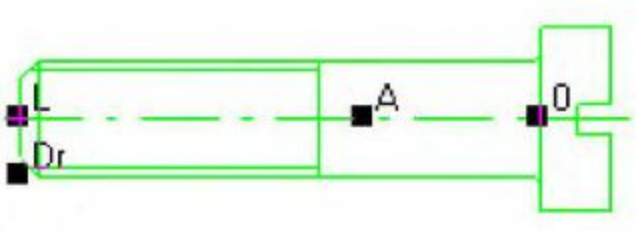


Рисунок 2 - Выделенное изображение библиотечной детали с 4-мя характерными точками

2 Стандартные диалоговые окна

Для выдачи информационных окон есть три метода **Message**, **MessageBoxResult** и **Error**.

Метод **YesNo** позволяет выдать сообщение и получить ответ как нажатие на кнопку "Да" или "Нет" - этот метод удобен для выдачи пользователю простых запросов на подтверждение некоторого действия.

Если в диалоговых окнах вводятся какие-нибудь данные, то результат ввода помещается либо в один из параметров метода, переданный по ссылке, либо передается в качестве возвращаемого значения.

2 Стандартные диалоговые окна

При отмене окна пользователем методы выбора имен файлов возвращают пустую строку. Типичный фрагмент программы для выбора файла выглядит так:

```
// Выбрать имя файла
void ReadFileName()
{

char fileName[128];
char buf[128];
if (ChoiceFile("*.cdw", 0, fileName, 128))
{
sprintf_s(buf, "Выбран файл с именем %s\n", fileName);
Message(buf);
}
}
```

2 Стандартные диалоговые окна

Часто применяются окна для ввода значения из заданного диапазона - методы с именами **Read**... . Например, **ReadInt** для ввода целого числа от 0 до 32000 со значением "по умолчанию" 100 можно вызвать так:

```
int X;
if (ReadIntT(_T("Введите координату X"), 100, 0, 32000, &X))
{
// действия с координатой X
}
```

2 Стандартные диалоговые окна

Таблица 1 - Методы для работы со стандартными диалоговыми окнами

Имя метода	Назначение
Message	Вывести произвольное текстовое сообщение
MessageBoxResult	Вывести сообщение с объяснением результата работы (кода ошибки) последнего вызванного метода КОМПАС-МАСТЕР
Error	Вывести текстовое сообщение об ошибке функций RTW-библиотеки
YesNo	Вывести строку сообщения в диалоговом окне и ожидать подтверждение или отказ от действия.
ksChoiceFile	Вызвать диалоговое окно выбора файла для чтения
ksChoiceFileAppointedDir	Вызвать диалог выбора файла для чтения с указанием каталога, с которого начинается выбор
ksChoiceFiles	Вызвать диалог выбора группы файлов для чтения (возвращает динамический массив интерфейсов ksChar255)
ksSaveFile	Вызвать диалог выбора файла для сохранения
ksInitFilePreviewFunc	Задать адрес пользовательской функции просмотра файла (для отображения в окне предварительного просмотра файлов, не являющихся документами КОМПАС)

2 Стандартные диалоговые окна

Таблица 1 - Методы для работы со стандартными диалоговыми окнами

Имя метода	Назначение
ReadString	Ввести символьную строку
ReadInt	Ввести целое число с контролем попадания значения в заданный интервал
ReadLong	Ввести длинное целое число с контролем попадания значения в заданный интервал
ReadDouble	Ввести вещественное число с контролем попадания значения в заданный интервал
ksMaterialDlg	Получить параметры материала из Справочника материалов

3 Отображение вариантов

Вариантом называется предварительное изображение библиотечной детали, выводимое тонкими линиями вместе с указателем мыши в процессе размещения детали пользователем в графическом документе. Получение от пользователя параметров местоположения (координат точки) выполняется с помощью метода **Cursor** или **Placement** (кроме координат точки позволяет задать значение угла наклона). Типичный вид окна КОМПАС-ГРАФИК в процессе размещения детали с помощью методов **Cursor/Placement** показан на рисунке 3.

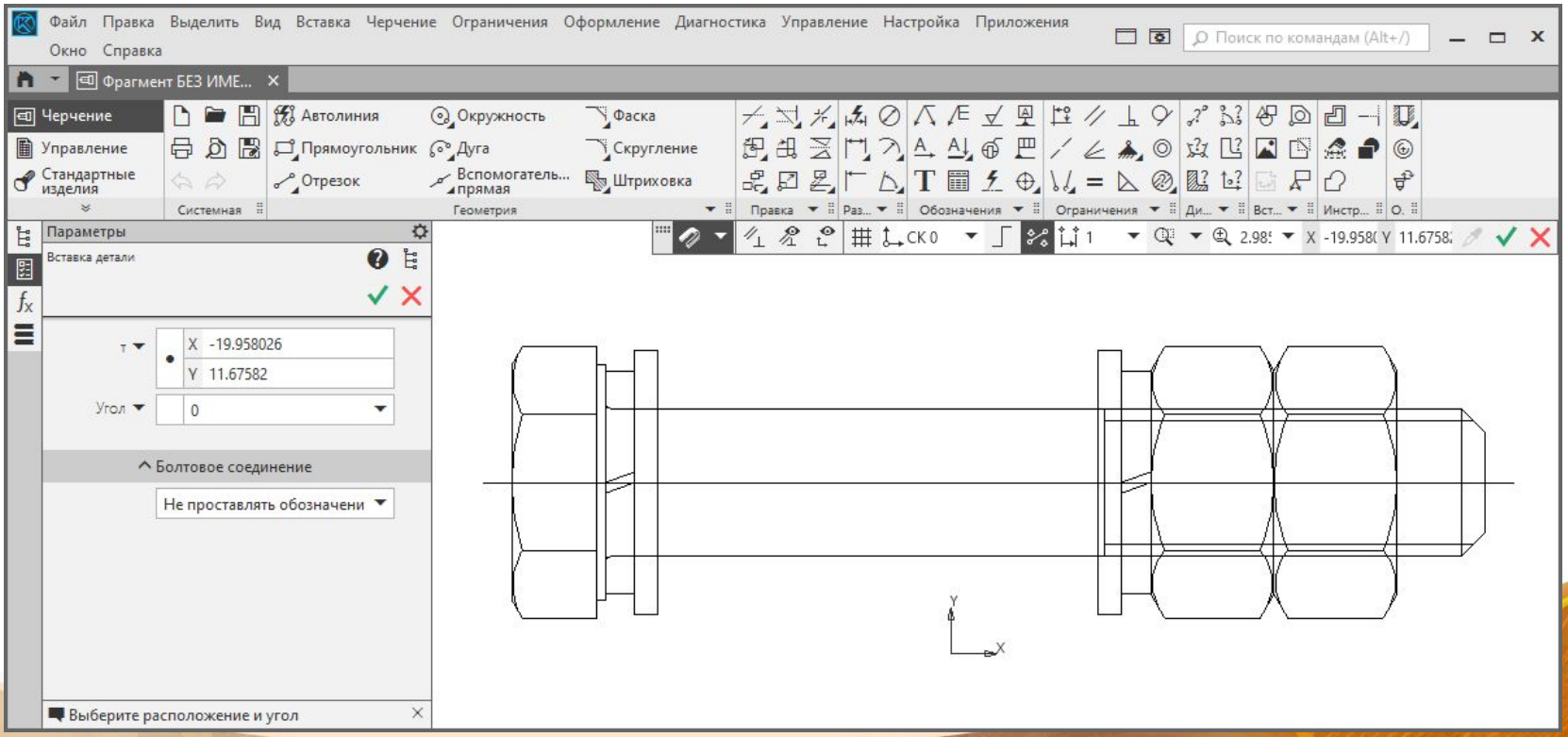


Рисунок 3 - Окно КОМПАС-ГРАФИК в процессе размещения библиотечной детали

3 Отображение вариантов

На рисунке 3 присутствует изображение варианта болтового соединения, которое перемещается вместе с указателем мыши. В строке сообщений выводится подсказка пользователю о том, что он должен указать базовую точку детали.

Для выбора параметров соединения может быть создано командное окно.

Команды командного окна можно выбирать не только в окне, но и в контекстном меню в процессе размещения детали на чертеже.

Перечисленные элементы интерфейса - командное окно, содержимое строки сообщения, вариантное изображение и параметры, получаемые от пользователя, задаются свойствами интерактивного запроса. Запрос выполняется методами **Cursor/Placement**. Эти методы имеют описание:

Указать положение объекта или определить вариант действия

```
int Cursor(RequestInfo *info, double *x, double *y, void * phantom);
```

Входные параметры :

`info` - указатель на структуру параметров запроса к системе,

`x, y` - координаты введенной точки,

`phantom` - указатель на структуру управления фантомом, определяющую тип движения курсора.

Возвращаемое значение :

`1` - в случае удачного завершения,

`0` - в случае неудачи.

3 Отображение вариантов

Задать точку и угол

```
int Placement(RequestInfo *info, double *x, double *y, double
*angle, void *phantom);
```

Входные параметры :

info - структуру параметров запроса к системе,

x, y - координаты введенной точки,

phantom - указатель на структуру управления фантомом, определяющую тип движения курсора,

angle - введенный угол.

Возвращаемое значение :

1 - в случае удачного завершения,

0 - в случае неудачи.

3 Отображение вариантов

Запрос к системе на получение точки

```
int CursorEx(RequestInfo *info, double *x, double *y, void *phantom,
LPUNKNOWN processParam);
```

Входные параметры:

info - указатель на структуру параметров запроса к системе RequestInfo,

x, y - координаты введенной точки,

phantom - указатель на структуру управления фантомом, определяющую тип движения курсора(аналог типа резиновой нити в версии 4),

processParam - указатель на интерфейс параметров процесса IProcessParam.

Выходные параметры:

x, y - возвращаемые координаты точки.

Возвращаемое значение :

-1 - если указана точка,

0 - отказ(Esc), идентификатор выбранной команды из командной строки или меню, определенный в файле ресурсов.

3 Отображение вариантов

Запрос к системе на получение точки и угла

```
int PlacementEx(RequestInfo *info, double *x, double *y,
double *angle, void *phantom, LPUNKNOWN processParam);
```

Входные параметры :

info - указатель на структуру параметров запроса к системе,

x, y - координаты введенной точки,

angle - введенный угол,

phantom - указатель на структуру управления фантомом, определяющую тип движения курсора(аналог типа резиновой нити в версии 4),

processParam - указатель на интерфейс параметры процесса ProcessParam.

Выходные параметры :

x, y - возвращаемые координаты точки,

angle - возвращаемое значение угла.

Возвращаемое значение :

1 - в случае удачного завершения,

0 - в случае неудачи.

3 Отображение вариантов

В структуре **RequestInfo** передаются параметры, определяющие способ взаимодействия пользователя с окном КОМПАС-ГРАФИК - описание командного окна и содержимого строки сообщений.

В интерфейсе **phantom** указываются параметры вариантного изображения.

В переменных **x, y** и **angle** возвращаются координаты точки и угол наклона.

В качестве возвращаемого значения может быть возвращен код:

0 - пользователь отменил выполнение команды;

- **1** - пользователь задал местоположение и ориентацию;

- **положительное значение** - пользователь выбрал в командном окне команду с соответствующим порядковым номером (или код команды, если содержимое командного окна было задано с помощью ресурса меню).

3 Отображение вариантов

Рассмотрим интерфейс параметров интерактивного запроса **RequestInfo**.

Основное назначение этого интерфейса - управление командным окном, которое показывается в немодальном диалоговом окне в процессе выполнения запроса. Пользователь может не только сдвинуть и повернуть вариантное изображение, но и выбрать двойным щелчком команду из этого окна.

Команды, помещаемые в окно, надо задать в виде строки в свойстве **RequestInfo.commands**. Даже если команд много, все они задаются одной строкой - перед именем каждой команды ставится восклицательный знак.

В свойстве **RequestInfo.title** задается заголовок командного окна, в свойстве **prompt** - содержимое строки - приглашения.

3 Отображение вариантов

Вариантное изображение описывается интерфейсом параметров **Phantom**. Вариантное изображение может быть нескольких типов, который надо указать в свойстве **phantom**. В соответствии со значением этого свойства, внутри интерфейса **Phantom** надо заполнить структуру параметров варианта именно этого типа (**Type1**, **Type2**, ... или **Type6**). Допустимые значения **phantom** и соответствующие подчиненные интерфейсы перечислены в таблице 2.

Таблица 2 - Допустимые типы вариантов

Тип варианта phantom	Назначение варианта	Подчиненный интерфейс
1	параметры сдвига группы	Type1
2	параметры фантома-отрезка	Type2
3	параметры фантома-прямоугольника	Type3
4	параметры фантома-отрезка с заданным углом	Type3
5	параметры фантома-половины прямоугольника	Type5
6	параметры пользовательского фантома	Type6
7	параметры фантома-окружности	Type2

3 Отображение вариантов

Наиболее универсальным вариантом является вариант 1 - в интерфейсе **Type1** можно задать дескриптор произвольной группы, которая будет отрисовываться с учетом смещения и поворота, которые задал пользователь с помощью мыши. Остальные типы позволяют, например, отобразить вариант в виде отрезка с зафиксированным концом или в виде прямоугольника с зафиксированным углом. Варианты типов 2-7 обычно применяют с методом **Cursor**. Часто варианты этих типов - отрезок, окружность и прямоугольник - называются типами резиновой нити.

Основные приемы работы с вариантным изображением и командным окном показаны в процедуре **Demo_Phantom**. Эта процедура позволяет нарисовать в текущем графическом документе окружность, треугольник, квадрат или отрезок фиксированного размера. Для треугольника, квадрата и отрезка можно указать местоположение и ориентацию, для окружности - только местоположение. Не выходя из этой процедуры, можно нарисовать произвольное количество любых фигур. Для этого выполняется анализ возвращаемого значения метода **Placement**.

3 Отображение вариантов

```
int g_type = 1; // Глобальная переменная - тип текущей фигуры
// Обновляется не только изображение но и меню для запроса
void Update_Cursor(reference & rGroup,  // Группа
RequestInfo & info) // Структура параметров запроса к системе
{
// Группа для фантома должна быть временная и обновляться при изменении вида отрисовки
if (rGroup)
DeleteObj(rGroup);
// Создание группы объектов, type - тип группы ( 0 - определяет модельный, 1 -
временный )
rGroup = NewGroup(1);
switch (g_type)
{
case 0:
// Состав гуппы - Квадрат
LineSeg(-10, 0, 10, 0, 1);
LineSeg(10, 0, 10, 20, 1);
LineSeg(10, 20, -10, 20, 1);
LineSeg(-10, 20, -10, 0, 1);
// Отображаемое меню
info.commands = ("!Окружность !Треугольник !Отрезок");
break;
```

3 Отображение вариантов

```
case 1:
// Состав гуппы - Окружность
Circle(0, 0, 20, 1);
// Отображаемое меню
info.commands = ("!Квадрат !Треугольник !Отрезок");
break;
case 2:
// Состав гуппы - Треугольник
LineSeg(-10, 0, 10, 0, 1);
LineSeg(10, 0, 0, 20, 1);
LineSeg(0, 20, -10, 0, 1);
// Отображаемое меню
info.commands = ("!Окружность !Квадрат !Отрезок");
break;
case 3:
// Состав гуппы - Отрезок
LineSeg(0, 0, 20, 0, 1);
// Отображаемое меню
info.commands = ("!Окружность !Треугольник !Квадрат");
break;
}
EndGroup(); // Завершить создание группы объектов
}
```

3 Отображение вариантов

```
void Demo_Phantom()
{
Phantom phantom; // Структура параметров фантома
phantom.phType = 1; // Тип фантома ( type1, type2, ... type7 )
phantom.type1.xBase = 0; // Координаты начальной точки группы
phantom.type1.yBase = 0;
phantom.type1.scale = 1; // Масштаб
phantom.type1.gr = 0; // Указатель на группу
phantom.type1.ang = 0; // Угол поворота группы
// Структура параметров запроса к системе
RequestInfo info;
memset(&info, 0, sizeof(info));
// Координата вставки
double x, y;
int comm = 1; // Пришедшая команда
while (comm)
{// Обновляется не только изображение, но и меню для запроса
Update_Cursor(phantom.type1.gr, info);
```

3 Отображение вариантов

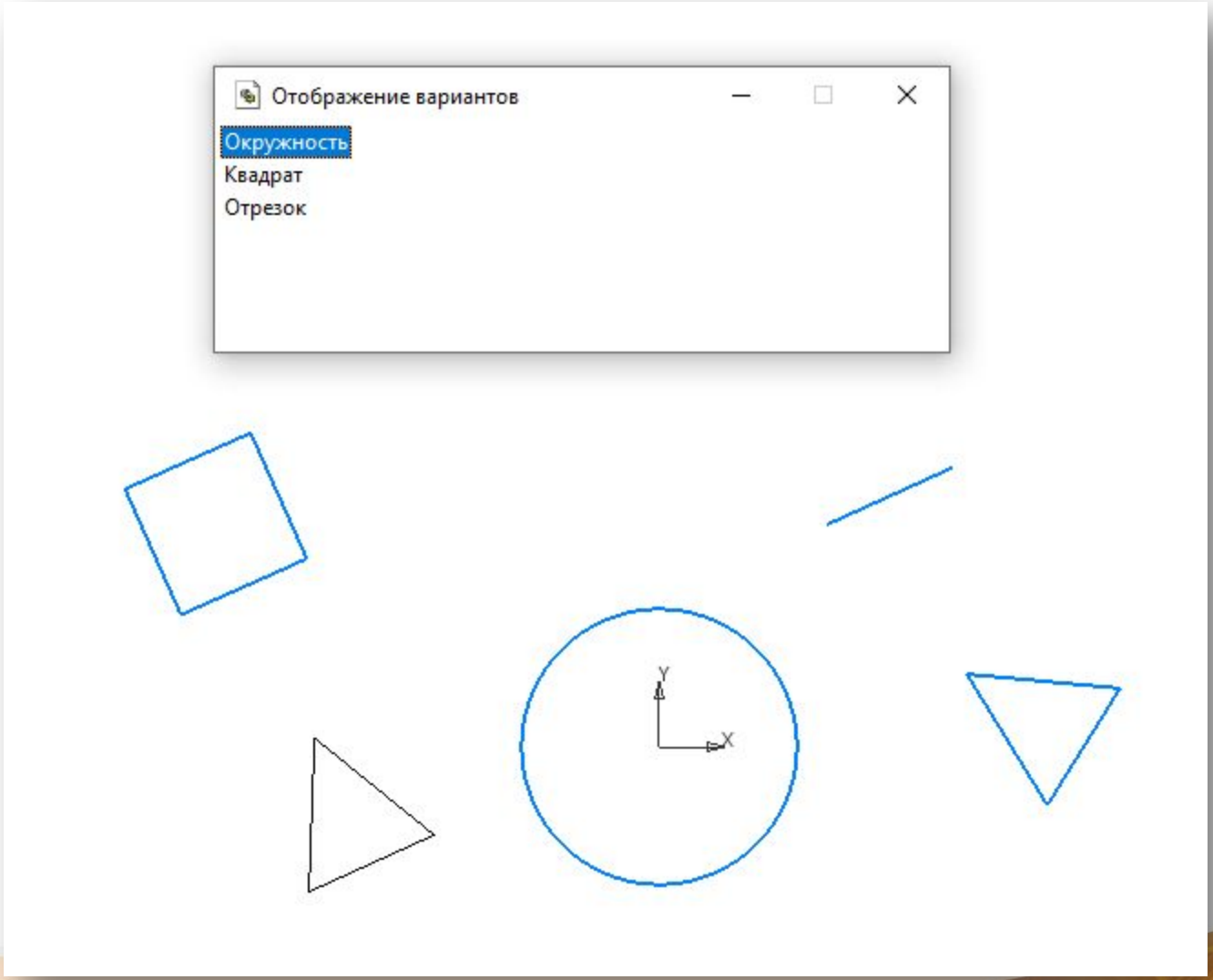
```
if (g_type == 1)
{// Пример с использованием Cursor
// Ввод точки или команды
comm = Cursor(&info,              // Указатель на структуру параметров запроса к системе
&x, &y,           // Координаты введенной точки
&phantom); // Указатель на структуру управления фантомом, определяющую тип движения курсора
}
else
{// Пример с использованием Placemant
// Задание точки, угла или команды
comm = Placement(&info,                // Указатель на структуру параметров запроса к системе
&x, &y,             // Координаты введенной точки
&phantom.type1.ang, // Введенный угол
&phantom);          // Указатель на структуру управления фантомом, определяющую типдвижения курсора
}
```

3 Отображение вариантов

```
// Выполнение команды
if (comm == -1) // Поставить в модель
{// Сдвинуть и повернуть группу
MoveObj(phantom.type1.gr, x, y);
if (fabs(phantom.type1.ang) > 0.001)
RotateObj(phantom.type1.gr, x, y, phantom.type1.ang);
// Поставить временную группу в вид
StoreTmpGroup(phantom.type1.gr);
// Очистить группу объектов
ClearGroup(phantom.type1.gr);
}
else // Сменить тип фигуры
{
if ((g_type == 1 && comm == 1) || (g_type == 2 && comm == 2) || (g_type == 3 && comm ==
3))
g_type = 0;
else
g_type = comm;
}
}
}
```

3 Отображение вариантов

Результат работы программы



3 Отображение вариантов

Процедура **Demo_Phantom** показывает способ использования **Cursor/Placement** без функции обратной связи. Процесс указания точки метод **Cursor/Placement** - многократно запускается во внешнем (по отношению к КОМПАС) цикле прикладной библиотеки и выбранная команда обрабатывается вне процесса.

Недостаток данного способа заключается в том, что командное окно заметно исчезает и появляется снова, так как создается при каждом вызове **Cursor/Placement**. При использовании функции обратной связи процесс указания точки запускается один раз. Обработка команды происходит в функции обратной связи. Процесс указания точки не завершается, пока функция обратной связи не вернет значение 0.

Команды в командном окне нумеруются, начиная с 1 . С учетом этого обстоятельства и перечисления команд в строке **RequestInfo: commands** в процедуре **Demo_Phantom** определена глобальная переменная - тип текущей фигуры **g_type**.

3 Отображение вариантов

Для меню команд:

```
info.commands = _T("!Окружность !Треугольник !Отрезок");
```

g_type=1 – соответствует Окружности,

g_type=2 – Треугольнику,

g_type=3 – Отрезку.

Для последующих фигур перечисление команд меняется таким образом, чтобы номер команды построения квадрата соответствовал номеру текущего типа фигуры. Это требование проверяется в соответствующем условии и при равенстве типа текущей фигуры и номера выбранной команды, переменной **g_type** присваивается 0, что соответствует фигуре Квадрат:

```
if ( ( g_type == 1 && comm == 1 ) || ( g_type == 2 && comm == 2 ) || (g_type == 3 && comm
== 3))
        g_type = 0;
      else
        g_type = comm;
```

В **Demo_Phantom** группа, описывающая вариантное изображение, создается и уничтожается на каждой итерации цикла. Если пользователь успешно разместил вариант, то перед уничтожением группа запоминается в документе.

Контрольные вопросы

1 Назовите методы для работы со стандартными диалоговыми окнами.

2 На какие три группы можно разделить операции ввода данных?

3 Как может выполняться редактирование уже созданной библиотечной детали?

4 Что такое вариантное изображение?

5 Объясните параметры метода Cursor?

6 Объясните параметры метода Placement?

7 Опишите структуру параметров RequestInfo?

8 Опишите структуру параметров Phantom?

Спасибо за внимание!