**Практическое занятие 22**

**События компонентов. Создание процедур на основе событий.**

Цель работы:

1Перетаскивание графических объектов с помощью мыши.

2 Управление событиями компонентов.

**

Требования к работе:

1) На форме, в границах PictureBox, должны быть созданы три графических объекта: квадрат, эллипс и прямоугольник.

2) Реализована возможность перетаскивания этих объектов по форме.

3) Должно быть выполнено следующее условие: если один из объектов, будет перенесён на Label «Вид», то в третьем Lable «Информация», должны появиться данные о том какого цвета и что это за фигура.

4) При перетаскивании окружности или квадрата на Label «Форма», объект должен меняться на противоположный (т.е. если перенесли квадрат, то он становится окружностью, а окружность, в свою очередь, становится квадратом и наоборот).

Задание 1. Создание визуальной части приложения.

1) Расположите на форме PictureBox.

2) Добавьте три компонента Label

Задание 2. Программирование элементов.

1) Рисование фигур.

2) Реализация Drag’n’Drop.

Порядок действий:

1) Для того, чтобы в элементе PictureBox нарисовать фигуру необходимо выполнить следующие действия:

а) Создать три структуры Rectangle в шапке класса вашей формы:

public partial class Работа23: Form

{

Rectangle Rectangle = new Rectangle(10, 10, 200, 100);

Rectangle Circle = new Rectangle(220, 10, 150, 150);

RectangleSquare = newRectangle(380, 10, 150, 150);

*Примечание: первые два параметра - X и Y, вторые – ширина и высота соответственно.*

б) Создать обработчик события Paint для PictureBox, в котором, используя методы FillRectangle и FillEllipse класса Graphiscs для параметра обработчика рисования *PaintEventArgs*, «залейте» три необходимых фигуры:

e.Graphics.FillEllipse(Brushes.Red, Circle);

e.Graphics.FillRectangle(Brushes.Blue, Square);

e.Graphics.FillRectangle(Brushes.Yellow, Rectangle);

2) Для реализации Drag’n’Drop’ нужно:

а) Там же, где объявлялись структуры, создайте три переменные типаbool с начальным значением – false. Эти переменные нужны для проверки «кликнули» ли мы на какой-либо наш объект.

б) Так же создайте еще по две переменных типа intcо значением равным 0 для каждого объекта (т.е. 6 переменных). Данные переменные будут фиксировать как изменяются координаты объектов во время перетаскивания.

в) Теперь, в обработчике события MouseDown вашего PictureBox, вам нужно фиксировать изменение координат для ваших объектов, а так же устанавливать флаг, что тот или иной объект выбран.

Пример для прямоугольника, где Rectangle – сам объект, e – параметр для событий мыши (*в данном примере, с помощью него мы получаем координаты указателя мыши*), RectangleX/RectangleY – те самые int-овские переменные, в которые мы и заносим текущие полученные координаты:

if ((e.X<Rectangle.X + Rectangle.Width) && (e.X>Rectangle.X))

{

if ((e.Y<Rectangle.Y + Rectangle.Height) && (e.Y>Rectangle.Y))

{

RectangleClicked = true;

RectangleX = e.X - Rectangle.X;

RectangleY = e.Y - Rectangle.Y;

}

}

*Для эллипса и квадрата всё делается аналогично.*

г) Создайте обработчик события MouseUp, для PictureBox, в котором просто всем трём переменным типа bool задайте false. Т.е. когда кнопка мыши отпускается, мы устанавливаем всем трём объектам флаги, что ни один из них не «кликнут».

д) Создайте обработчик события MouseMove, всё так же, для PictureBox. Здесь вам нужно проверить какой из объектов в данный момент перетаскивается и нужному присвоить координаты, которые мы считывали в событии MouseDown.

Пример для окружности:

if (CircleClicked)

{

Circle.X = e.X - CircleX;

Circle.Y = e.Y - CircleY;

}

После того, как вы сделаете это для остальных двух объектов, добавьте строку:

pictureBox1.Invalidate();

В ней вызывается перерисовка PictureBox’a.

3) Выполнение остальной части задания.

Чтобы при перетаскивании объекта в определённую область, у нас происходили какие-то события, нам нужно просто считывать координаты объекта и сравнивать их с координатами нужной нам области (в нашем случае это Label).

а) В событии MouseMove, нам необходимо сверить попадает ли какой-либо из наших объектов в границы Label «Вид». Алгоритм подобен тому, что мы делали в обработчике MouseDown ранее.

Примердляквадрата:

if ((label1.Location.X <Square.X + Square.Width) && (label1.Location.X >Square.X))

{

if ((label1.Location.Y <Square.Y + Square.Height) && (label1.Location.Y >Square.Y))

{

label3.Text = "Синий квадрат";

}

}

б) Создайте ещё пять переменных типа int:

int X, Y, dX, dY;

intLastClicked = 0;

А теперь в обработчике события MouseUp, вам необходимо, как и в прошлом пункте определять положение вашего объекта и сверять его с Label «Форма». Но теперь, предварительно нам нужно знать какая именно из фигур была перенесена. Для этого и нужна переменная LastClicked, в которой содержится информация об этом. (значение этой переменной вы должны задать в MouseUp-событии. Например: 1 – прямоугольник, 2 – круг и 3 – окружность). Переменные X, Y, dX и dY нужны для сохранения координат одной из фигур, дабы передать их другой фигуре.

Пример для изменения формы круга на форму квадрата:

if (LastClicked == 2)

{

if ((label2.Location.X <Circle.X + Circle.Width) && (label2.Location.X >Circle.X))

{

if ((label2.Location.Y <Circle.Y + Circle.Height) && (label2.Location.Y >Circle.Y))

{

X = Circle.X;

Y = Circle.Y;

dX = CircleX;

dY = CircleY;

Circle.X = Square.X;

Circle.Y = Square.Y;

CircleX = SquareX;

CircleY = SquareY;

Square.X = X;

Square.Y = Y;

SquareX = dX;

SquareY = dY;

}

}

}

Для обратного изменения (квадрат становится кругом) всё делается аналогично.

Дополнительное задание: попробуйте сделать так, чтобы при «клике» на определённую фигуру, она становилась на передний план.

*Подсказка: в событии Paint, фигуры рисуются в прямом порядке, в котором вы их установили. Т.е. если сначала вы «залили» прямоугольник, а потом круг, то круг будет на переднем плане. Значит, для выполнения, этого задания, вам нужно проверять, как фигура выбрана, а далее просто менять порядок прорисовки.*