ЛАБОРАТОРНАЯ РАБОТА № 11. РАЗРАБОТКА ЗАНИМАТЕЛЬНОЙ ПРОГРАММЫ

Цель: ознакомиться с возможностью разработки игровых программ.

Задание 1. Напишите программу для реализации игры «Угадай число».

Постановка задачи

Машина задумывает случайное число от 1 до 100. Пользователю предоставляется возможность ввести с клавиатуры число, чтобы угадать задуманное. В зависимости от ситуации выводится сообщение о том, что число угадано, или сообщение «меньше» или «больше». После того как число угадано, пользователю предоставляется возможность новой игры (после нажатия на кнопку «новое число»).

Задание 2. Разработайте проект «Морской бой».

Постановка задачи

Создать игровую программу, представляющую из себя игру «Морской бой». Необходимо попасть торпедой в корабль, который движется.

План разработки программы

1. Открываем новый проект.
2. Добавляем компонент ***Panel*** (***Standart***). Изменяем свойство ***align*** (выравнивание) для панели, делаем ***alTop*** (выравнивание по верху), высоту панели делаем 50—60 пикселей, убираем параметр Caption, цвет изменяем на светло-голубой. Свойство ***WindowState*** для формы делаем ***wsMaximized***.
3. На свободное место формы (не на панель!) добавляем компонент ***Image*** (стр. ***Additional***), загружаем в ***Image*** рисунок из файла ***fon.jpg.*** Свойство ***align*** (выравнивание) для картинки сделать ***alClient*** (выравнивание по всей области), свойство ***AutoSize*** (автоматический размер) изменяем на ***True***.
4. Добавляем еще три объекта ***Image.*** Загружаем три картинки — ***lodka.bmp***, ***ships.bmp***, ***torpeda.bmp***. Для удобства работы с компонентами изменяем имена (***name***) объектов ***Image —*** назовем их соответственно ***Lodka***, ***Ships***, ***Torpeda***. Называть надо обязательно английскими буквами, так как в именах компонентов (также как и в именах переменных) нельзя использовать буквы национальных языков. Для всех этих трех компонентов необходимо свойство ***AutoSize*** (автоматический размер) изменить на ***True***.

В результате у нас должно получиться примерно следующее расположение компонентов (Рис. 11.1):

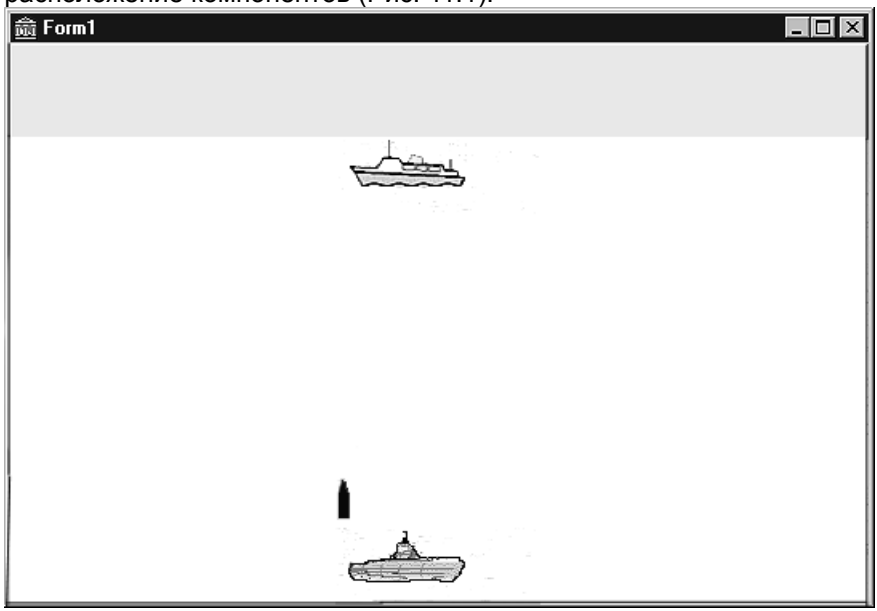

Рис. 11.1

5. Из палитры ***System*** берем компонент ***Timer*** и помещаем его куда угодно на форму. Он будет отвечать в этой программе за движение корабля. Интервал таймера сделаем равным 150, в обработчике изменения времени таймера (***Timer1.OnTimer***) напишем движение вражеского корабля (компонент ***Ships***) справа налево:

   Ships.left:= Ships.left-1;

   Запускаем программу и смотрим на движение объекта. Вы увидите явный недостаток — при достижении кораблем края формы, он исчезает.

   Чтобы корабль появился с противоположного края формы, необходимо добавить в обработчик событий строчку:

   If Ships.left<0 then Ships.left:=form1.width-60.

6. Организуем движение торпеды (объект ***Torpeda***). В начальный момент времени она не видна (измените свойство ***Visible***) и неподвижна (добавляем второй таймер и свойство ***Enabled*** устанавливаем в ***True***). Затем при нажатии на клавишу пробел должен совершиться выстрел. Для этого делаем следующее:

В обработчике формы нажатия на клавишу (***Form1.OnKeyPress***) пишем следующие строки:

if key=' ' then begin // если нажата клавиша пробел
Timer2.enabled:=true; // включаем таймер движения торпеды
Torpeda.visible:=true; // показываем торпеду
end;

В обработчике событий для таймера движения торпеды (***Timer2.OnTimer***) записываем движение торпеды снизу вверх, по аналогии с движением нашего корабля справа налево, с тем отличием, что торпеда летит быстрее нашего корабля, поэтому интервал таймера делаем равным 50, а за каждый интервал наша торпеда будет перемещаться не на один а на 5 пикселей. При достижении верха нашего рисунка фона (т.е. при условии ***Torpeda.top<Image1.top***) она становится опять невидимой и перемещается вниз:

if Torpeda.top<Image1.top // если торпеда дошла до верха
прячем торпеду (изменяем свойство видимости)
перемещаем вниз (изменяем свойство верх)
останавливаем таймер движения торпеды
end;

7. Добавляем возможность управления подводной лодкой с клавиатуры. Для этого вызовем обработчик нажатия на клавишу (***Form1.OnKeyDown***). В заголовке процедуры описана переменная ***Key,*** которую будем использовать. Нам понадобятся следующие значения этой переменной:

Vk_left — стрелка влево
Vk_right — стрелка вправо.

При нажатии на клавишу «стрелка влево» (***Vk_left***), мы будем перемещать нашу лодку на 2 пикселя влево (при этом должна перемещаться влево и наша торпеда):

if key= Vk_left then begin // если мы нажали на клавишу влево
Lodka.left:= Lodka.left-2; // передвигаем лодку на 2 пикселя влево
Torpeda.left:= Torpeda.left-2; // передвигаем торпеду
end;

Аналогично описываем нажатие на клавишу вправо.

8. Далее необходимо дать понять программе, при каком условии торпеда попадет в корабль. А это произойдет тогда, когда координаты торпеды будут равны координатам цели, то есть условие должно быть следующим:

If ((Torpeda.Top > Ships.Top) and	Если вертикальная координата торпеды больше верхнего,
(Torpeda.Top < Ships.Top+ Ships.height)	но меньше нижнего значения координат корабля
And	и
(Torpeda.left > Ships.left) and	Если горизонтальная координата торпеды находится
(Torpeda.left < Ships.left+ Ships.Width))	в промежутке между правым и левым бортом корабля,
Then Begin	то
Timer2.Enabled:= false;	Останавливаем движение торпеды,
Timer1.Enabled:= false;	Останавливаем движение корабля
End;	

Эти строки можно добавить как в обработчик движения корабля (***Timer1.OnTimer***), так и в обработчик движения торпеды (***Timer2.OnTimer***).

9. Запускаем программу, тестируем.

Задания для самостоятельной работы (к проекту «Морской бой»).

1. Добавить на панель кнопку, на которой написать «Новая игра». При нажатии на кнопку игра должна приводиться к первоначальному положению.
2. Добавить эффект затопления корабля. При попадании в него торпеды, он должен начинать медленно опускаться на дно.

Задание 3. Игра «Быки и коровы». Вот ее правила. Необходимо угадать случайное четырехзначное число, в котором нет повторяющихся цифр. Чтобы угадать задуманное число, играющий вводит свое четырехзначное число, в котором также нет повторяющихся цифр. Если в числе игрока есть цифра, которая есть и в задуманном числе в той же позиции, то засчитывается один «бык». Если в числе игрока есть цифра, которая есть и в задуманном числе, но ее позиция не совпадает с позицией этой цифры в задуманном числе, то засчитывается одна «корова». Игроку после ввода своего числа выдается сообщение об общем числе «быков» и «коров». Анализируя все сообщения, играющий отгадывает задуманное

число (рис. 11.2).

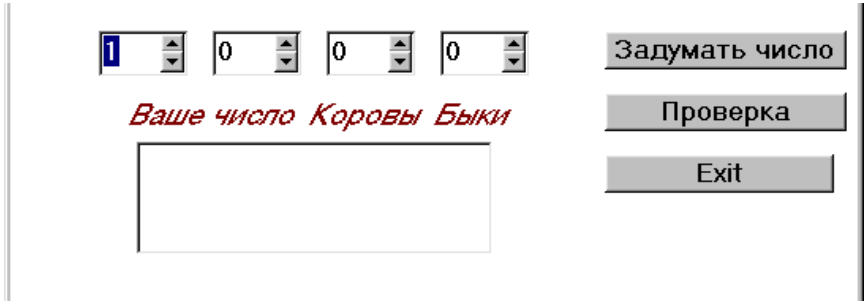

Рис. 11.2.

```
unit Unit1;
interface
uses Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
StdCtrls, Spin;
type
TForm1 = class(TForm)
SpinEdit1: TSpinEdit;
SpinEdit2: TSpinEdit;
SpinEdit3: TSpinEdit;
SpinEdit4: TSpinEdit;
Button1: TButton;
Button2: TButton;
Button3: TButton;
ListBox1: TListBox;
Label1: TLabel;
procedure Button3Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
a,b,c,d,a1,b1,c1,d1,k,kor,bik:integer;
s1,s2:string;
```

```
implementation
{$R *.DFM}
procedure TForm1.Button3Click(Sender: TObject);
begin
        close;
end;
// задумывание машиной случайного четырехзначного числа, в
// котором нет повторяющихся цифр
procedure TForm1.Button1Click(Sender: TObject);
begin
        randomize;
        a:=random(9)+1; // первая цифра
        repeat
                b:=random(10);   // вторая цифра
        until b<>a;
        repeat
                c:=random(10);  // третья цифра
        until (c<>a) and (c<>b);
        repeat
                d:=random(10); // четвертая цифра
        until (d<>a) and (d<>b) and (d<>c);

        Button1.Visible:=False;
        listbox1.Items.Clear;
end;
// проверка числа, введенного игроком
procedure TForm1.Button2Click(Sender: TObject);
begin
    kor:=0;
    bik:=0;
    k:=0;
    a1:=SpinEdit1.Value;
    b1:=SpinEdit2.Value;
    c1:=SpinEdit3.Value;
    d1:=SpinEdit4.Value;
    if (a1=b1) or (a1=c1) or (a1=d1) then k:=1;
    if (b1=a1) or (b1=c1) or (b1=d1) then k:=1;
    if (c1=a1) or (c1=b1) or (a1=d1) then k:=1;
    if (d1=a1) or (d1=b1) or (d1=c1) then k:=1;
    if k=1 then showmessage('Цифры повторяются')
    else begin
```

```
    if (a=a1) and (b=b1) and (c=c1) and (d=d1) then begin
    showmessage('Число угадано!');
    Button1.Visible:=true;
  end
  else begin
    if a=a1 then bik:=bik+1;
    if b=b1 then bik:=bik+1;
    if c=c1 then bik:=bik+1;
    if d=d1 then bik:=bik+1;
    if (a1=b) or (a1=c) or (a1=d) then kor:=kor+1;
    if (b1=a) or (b1=c) or (b1=d) then kor:=kor+1;
    if (c1=a) or (c1=b) or (c1=d) then kor:=kor+1;
    if (d1=a) or (d1=b) or (d1=c) then kor:=kor+1;
    s1:=inttostr(a1)+inttostr(b1)+inttostr(c1)+inttostr(d1);
    s2:=s1+'      '+inttostr(kor)+'      '+inttostr(bik);
    Listbox1.Items.Add(s2);
  end;
  end;
end; {проверки числа}
END.
```

Задание 4. Игра «15». Вот ее правила. В прямоугольной коробочке находятся 15 фишек, на которых написаны числа от 1 до 15. Размер коробочки — 4x4, таким образом, в коробочке есть одна пустая ячейка. В начале игры фишки перемешаны. Задача игрока состоит в том, чтобы, не вынимая фишки из коробочки, выстроить фишки в правильном порядке (рис. 11.3) [4].

Пояснение. При реализации проекта для данного задания не используются визуальные компоненты. Прямоугольная коробочка и фишки с цифрами прорисовываются на поверхности формы.

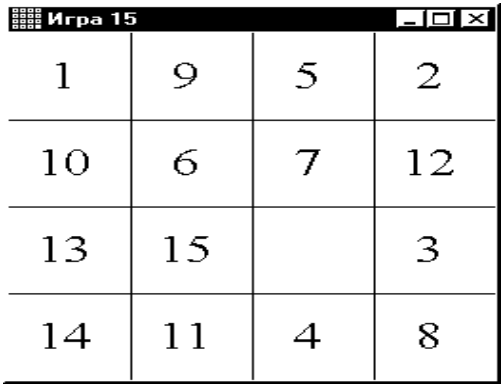

Рис. 11.3

```
type
TForm1 = class(TForm)
procedure FormCreate(Sender: TObject);
procedure FormMouseDown(Sender: TObject; Button:
TMouseButton;
        Shift: TShiftState; X, Y: Integer);
procedure FormPaint(Sender: TObject);

// эти объявления вставлены сюда вручную
procedure ShowPole;
procedure Mixer;

private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.dfm}
const
H = 4;   W = 4;   // размер поля — 4x4
CH = 64; CW = 64; // размер клеток — 16x16
```

```
var
// правильное расположение фишек
stp : array[1..H, 1..W] of byte =
        (( 1, 2, 3, 4),
        ( 5, 6, 7, 8),
        ( 9,10,11,12),
        (13,14,15, 0));
// игровое поле
pole: array[1..H, 1..W] of byte;
ex,ey: integer; // координаты пустой клетки

// новая игра
procedure NewGame;
var i,j: integer;
begin
        // исходное (правильное) положение
        for i:=0 to H+1 do
        for j:=0 to W+1 do
                pole[i,j] := stp[i,j];
        Form1.Mixer;    // перемешать фишки
        Form1.ShowPole; // отобразить поле
end;

// проверяет, расположены ли
// фишки в нужном порядке
function Finish: boolean;
var
        row,col: integer;
        i: integer;
begin
        row :=1; col :=1;
        Finish := True; // пусть фишки в нужном порядке
        for i:=1 to 15 do begin
                if pole[row,col] <> i then begin
                        Finish:= False;
                        break;
                end;
                // к следующей клетке
                if col < 4 then inc(col)
                else begin
```

```
                        col :=1;
                        inc(row);
                end;
        end;
end;

// «перемещает» фишку в соседнюю пустую клетку,
// если она есть
procedure Move(cx,cy: integer);
// cx,cy — клетка, в которой игрок сделал щелчок
var
        r: integer;       // выбор игрока
begin
        // проверим, возможен ли обмен
        if not (( abs(cx-ex) = 1) and (cy-ey = 0) or
        ( abs(cy-ey) = 1) and (cx-ex = 0))
        then exit;
        // Обмен. Переместим фишку из x,y в ex,ey
        Pole[ey,ex] := Pole[cy,cx];
        Pole[cy,cx] := 0;
        ex:=cx;
        ey:=cy;
        // отрисовать поле
        Form1.ShowPole;
        if Finish then begin
                r := MessageDlg('Цель достигнута!'+ #13+
                'Еще раз?',mtInformation,[mbYes,mbNo],0);
                if r = mrNo then Form1.Close; // завершить работу
        end;
end;

// щелчок в клетке
procedure TForm1.FormMouseDown(Sender: TObject; Button:
TMouseButton;
Shift: TShiftState; X, Y: Integer);
var
        cx,cy: integer; // координаты клетки
begin
        // преобразуем координаты мыши в координаты клетки
        cx := Trunc(X / CW) + 1;
        cy := Trunc(Y / CH) + 1;
```

```
        Move(cx,cy);
end;

// выводит игровое поле
procedure TForm1.ShowPole;
var
        i,j: integer;
        x,y: integer; // x,y — координаты вывода текста в клетке
begin
        // сетка: вертикальные линии
        for i:= 1 to W - 1 do
        begin
                Canvas.MoveTo(i*CW,0);
                Canvas.LineTo(i*CW,ClientHeight);
        end;
        // сетка: горизонтальные линии
        for i:= 1 to H - 1 do
        begin
                Canvas.MoveTo(0,i*CH);
                Canvas.LineTo(ClientWidth,i*CH);
        end;

        // содержимое клеток
        // x,y — координаты вывода текста
        for i:= 1 to H do
        begin
          y:=(i-1)*CH + 15;
          for j:=1 to W do begin
              x:= (j-1)*CW + 15;
              case Pole[i,j] of
              0:      Canvas.TextOut(x,y,'    ');
              1..9:       Canvas.TextOut(x,y,' '+IntToStr(Pole[i,j])+'  ');
              10..15:   Canvas.TextOut(x,y,IntToStr(Pole[i,j]));
              end;
        end;
    end;
end;

// «перемешивает» фишки
procedure TForm1.Mixer;
var
```

```
        x1,y1: integer; // пустая клетка
        x2,y2: integer; // эту переместить в пустую
        d: integer;     // направление, относительно пустой
        i: integer;
begin
        x1:=4;
        y1:=4;
        randomize;
        for i:= 1 to 150 do
        begin
                repeat
                        x2:=x1;
                        y2:=y1;
                        d:=random(4)+1;
                        case d of
                        1:  dec(x2);
                        2:  inc(x2);
                        3:  dec(y2);
                        4:  inc(y2);
                        end;
                until (x2>=1) and (x2<=4) and (y2>=1) and (y2<=4);
                // здесь определили фишку, которую
                // надо переместить в пустую клетку
                Pole[y1,x1] := Pole[y2,x2];
                Pole[y2,x2] := 0;
                x1:=x2;
                y1:=y2;
             end;
        // запомним координаты пустой клетки
        ex:= x1;
        ey:= y1;
      end;

// обработка события OnCreate
procedure TForm1.FormCreate(Sender: TObject);
begin
        ClientWidth := CW * W;
        ClientHeight := CH * H;
        Canvas.Font.Name := 'Times New Roman';
        Canvas.Font.Size := 22;
        NewGame;
```

```
end;

// обработка события OnPaint
procedure TForm1.FormPaint(Sender: TObject);
begin
        Form1.ShowPole;
end;
END.
```