Уважаемые студенты групп!

Вашему вниманию представлена лабораторная работа на тему «МЕТОДЫ СОЗДАНИЯ ДВИЖУЩИХСЯ ГРАФИЧЕСКИХ ОБЪЕКТОВ СРЕДСТВАМИ ЯЗЫКА ПАСКАЛЬ»

Задание

1. Реализовать в системе Паскаль приведенный пример выполнения задания, протестировать программу и исправить ошибки.
2. Лабораторные работы оформляются в тетради в клеточку!
3. Дата предоставления фотоотчет до 28.04.2023
4. С уважением Ганзенко Ирина Владимировна

!!! Если возникнут вопросы обращаться по телефону 0721134803 (вацап), +79591134803 (телеграмм)
disobuch.ganzenko2020@mail.ru

МЕТОДЫ СОЗДАНИЯ ДВИЖУЩИХСЯ ГРАФИЧЕСКИХ ОБЪЕКТОВ СРЕДСТВАМИ ЯЗЫКА ПАСКАЛЬ

Цель работы: показать возможности создания движущихся графических объектов и простых мультипликационных изображений средствами языка паскаль на примерах решения задач, развить логическое мышление, творческие способности поддерживать стремление к усвоению новых знаний.

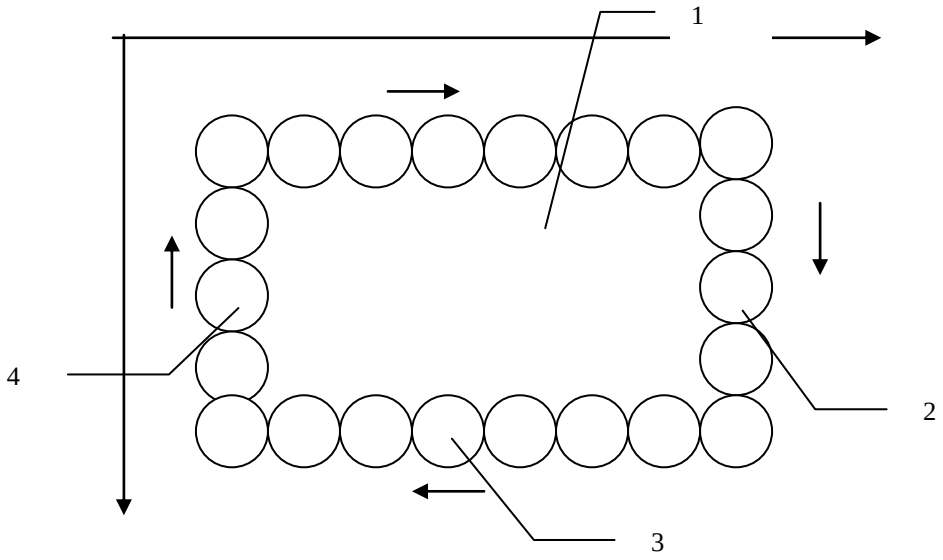
1 Теоретические положения

Чтобы начать работу над созданием простого "мультика" (движущегося изображения), сначала разберемся, как оно создается в реальных условиях. Все знают, что художник мультипликатор рисует серию изображений, в каждом из которых показывается один и тот же движение с едва заметными изменениями. Например так, как это показано на следующем рисунке.(рис.1)

Рисунок 1 – движение фигуры

Теперь, если изображение быстро менять одно за другим, мы не замечаем изменение рисунков, а видим движение этого персонажа (в данном случае Незнайка поднимает руку). Подумаем, как воспроизвести последовательность похожих объектов на экране монитора. Первое, что приходит в голову, это нарисовать изображение, задержать его немного на экране, вытереть изображения (очистить экран) и вывести новое изображение с едва заметными изменениями. При достаточно большой скорости рисования человек не заметит изменения рисунков и ей будет казаться, что объект движется. Решим таким методом следующую задачу.

Пример 1. Вывести на экран рисунок: движение круга по прямоугольнику.



1) Движение круга по верхней горизонтали

```
 for i:=2 to 9 do
begin
    setcolor(random(16));
    circle(50*i,100,50);
    delay(2000);
end;
```

2) Движение круга по правой вертикали

```
for i:=2 to 7 do
begin
    setcolor(random(16);
    circle(450,50*i,50);
    delay(2000);
end;
```

3) Движение круга по нижней горизонтали

```
for i:=9 downto 2 do
begin
    setcolor(random(16);
```

```
        circle(50*i,350,50);
        delay(2000);
    end;
```

4) Движение круга по левой вертикали

```
    for i:=7 downto 2 do
     begin
        setcolor(random(16);
        circle(100,50*i,50);
        delay(2000);
    end;
```

В данном фрагменте каждый круг выводится по - очереди на экран и не исчезает. Для того, чтобы создать иллюзию движения нужно после каждого **delay(2000)-** задержка изображения на экране, вставить процедуру чистки экрана - **CleaDevice.**

Пример 2. Вывести движение круга по диагонали экрана, используя уравнение прямой, проходящая через две точки: Y= k*x+b

0=k*0+b
480=k*640+b
k=480/640=0.75
y=0.75*x

```
uses crt,graph;
  var
     dv,md:integer;
     x,y:real;
begin
    dv:=detect;
    initgraph(dv,md,'');
       x:=25;
    y:=0.75*x;
   While x<= 615 do
      begin
        setcolor(random(25));
        circle(round(x),round(y),25);
        delay(1000);
        cleardevice
        y:=0.75*x;
        x:=x+1;
      end;
    readkey;
    closegraph;
end.
```

Пример 3. ***Условие*:** " Часы ". Смоделировать движение часовой и минутной стрелки .Если смоделировать работу часов в реальном времени, то наглядность программы будет небольшой, так как движение стрелок будет едва заметным. Поэтому сделаем имитацию работы часов, то есть минутная стрелка будет двигаться достаточно быстро, а движение часовой стрелки будет зависеть от минутной. В начале работы выясним, из каких элементов состоит часы. Во-первых, это круг с делениями, а, во-вторых, два отрезка разной длины, имитирующие стрелки (стрелки можно сделать и более сложными). Круг является недвижимым объектом, поэтому он рисуется статически с абсолютными координатами центра и радиусом, а стрелки двигаются, к тому же перемещается только один конец стрелки-отрезка, а второй тоже является статичным (центр круга).

Формулы, по которым вычисляются координаты подвижного конца стрелки - поворот точки на заданный угол относительно неподвижного центра с координатами $x0$, $y0$ известные по математике. Поэтому приводим их здесь без объяснений:

$x = x0 + L*\cos a$
$y = y0 + L*\sin a$
где L - расстояние, на котором находится точка от центра поворота,
a - угол, на который возвращается точка.

Обратите внимание только на то, что в программе вторая формула вместо знака "-" будет содержать знак "+", потому что экранные координаты имеют направленность осей, обратных к реальным декартовых координат (на экране значение координаты Y увеличивается в направлении сверху вниз).

Рисование делений на циферблате выполняется тоже с помощью приведенных выше формул. Программа, реализующая предложенный алгоритм, приведена ниже.

Обратите внимание, что в этой программе
L_min, L_time - длина минутной и часовой стрелок соответственно;
Color_min, Color_time - цвета минутной и часовой стрелок соответственно;
R - радиус циферблата часов;
x_centr, y_centr - координаты центра экрана (определяются в соответствии с текущей разрешения с помощью функций getmaxx и getmaxy;
x_min, y_min - координаты подвижного конца минутной стрелки;
x_time, y_time - координаты подвижного конца часовой стрелки;
Ang_min, Ang_time - углы поворота минутной и часовой стрелок соответствующие.

Движение стрелок по циферблату осуществляется за счет постоянной их перерисовки, то активным цветом рисования стрелки, то цветом фона ("затирание" изображения). Программа завершается после нажатия любой клавиши за счет использования цикла ***repeat until keypressed.***

```
Program Example_629;
Uses crt,graph;        {Подключение библиотек }
const L_min=174;
      L_time=145;
      Color_min=white;
      Color_time=white;
      R = 200;
var gd,gm:integer;
    S:string[2];
    x_centr, y_centr:integer;
    i,x_min,y_min:integer;
    x_time,y_time:integer;
    Ang_min,Ang_time:real;
begin
  {Инициализация графического режима }
  gd:=VGA; gm:=VGAHi;
  InitGraph (gd,gm,'egavga.bgi');
  {Определение центра экрана}
  x_centr := getmaxx div 2;
  y_centr := getmaxy div 2;
  {Рисование статической части рисунке}
  SetColor(brown);
  SetFillStyle(1,brown);
  {Рисование циферблата коричневого цвета}
  FillEllipse(x_centr,y_centr,R,R);
  Ang_time:=-90;
  {Установка цвета рисования, стиля и выравнивания текста}
  SetColor(yellow);
  SetTextJustify(CenterText, CenterText);
  SetTextStyle(DefaultFont, HorizDir, 2);
  {Рисование делений желтого цвета и цифр}
  for i:=1 to 12 do
  begin
    Ang_time:=Ang_time+30;
    x_time:=round(x_centr+185*cos(Ang_time*pi/180));
    y_time:=round(y_centr+185*sin(Ang_time*pi/180));
    str(i,S);
    OutTextXy(x_time,y_time,S);
  end;
  {Рисование хода часов}
  Ang_min:=-90;
  Ang_time:=-90;
  repeat
    x_time:=round(x_centr+L_time*cos(Ang_time*pi/180));
    y_time:=round(y_centr+L_time*sin(Ang_time*pi/180));
```

```
    SetColor(Color_min);
    Line(x_centr,y_centr,x_time,y_time);
    x_min:=round(x_centr+L_min*cos(Ang_min*pi/180));
    y_min:=round(y_centr+L_min*sin(Ang_min*pi/180));
    SetColor(Color_min);
    Line(x_centr,y_centr,x_min,y_min);
    Delay(10000);   {Задержка изображения на экране}
    SetColor(brown);
    Line(x_centr,y_centr,x_time,y_time);
    Line(x_centr,y_centr,x_min,y_min);
    Ang_min:=Ang_min+6;
    Ang_time:=Ang_time+0.5;
  until keypressed;
  readkey;
  CloseGraph;
end.
```

Предложенный метод построения мультипликационных объектов является простейшим, но если объект движется, имеет большие линейные размеры, чем в предложенной задачи, он будет существенно мелькать на экране. Поэтому существует другой подход к решению этой задачи. В этом случае предлагается следующий алгоритм:

а) нарисовать желаемый объект;

б) запомнить область экрана, с выведенным рисунком;

в) восстановить экран в месте, где был рисунок (то есть стереть рисунок);

г) вывести рисунок на новое место и т.д.

Этот подход очень похож на предыдущий вариант, но имеет существенные преимущества в том, что не требует многократного перерисовки рисунке. Объект создается один раз, сохраняется его копия, а затем выводится в нужном месте.

Для сохранения нарисованного фрагмента необходимо использовать оперативную память, причем так как мы не знаем размеры объекта в начале программы, память необходимо запрашивать в системе непосредственно во время работы программы. Это можно сделать, только используя динамическую память с помощью процедур.

Модуль Graph содержит две процедуры, которые используются для движения сложных окрашенных изображений независимо от того в каком месте экрана они выведены и независимо от других изображений.

1. getimage(x1,y1,x2,y2,im) - процедура заносит в память копию прямоугольного фрагмента графического изображения.

x1,y1,x2,y2 – координаты углов диагонально противоположные фрагмента, в котором находится графическое изображение.

im – переменная, куда будет записана копия видеопамяти с

фрагментом изображения, а затем выведена на экране в нужном месте.

(Записывает в массив цвета пикселей)

Эта переменная описывается в программе в разделе описания данных, как двумерный массив. Вспоминаем, как описывается массив?

Var ,

im: array [1..(X2-X1), 1..(Y2-Y1)] of byte

Количество элементов массива-количество пикселей по горизонтали и по вертикали. Объяснить, почему используется тип byte. (Для экономии оперативной памяти)

На отдельной доске рисунок. Учащимся предлагается записать на доске размеры двумерного массива, куда будет занесена копия прямоугольного фрагмента.

im: array [1..150, 1..125] of byte

2. putimage (x1,y1, im, Mode) – выводит в заданное место экрана копию фрагмента изображения, которая была помещена (занесена) в память процедурой getimage (или забирает).

X1,Y1 – координаты верхнего левого угла того места экрана, куда будет скопирован фрагмент изображения.

im – переменная откуда выводится изображение.

Mode – маска, способ копирования, режим вывода

(может принимать значения 1 или 0)

0 - если не было изображения - появилось, если было - то будет

1 - если не было изображения - появилось, если было - сотрется

2,3,4 - инверсия - не объяснять

(Можно еще так объяснить: параметр Mode определяет способ взаимодействия копии, снова размещается с изображением которое существует на экране **0**- замена существующего изображения на копию)

1 – применив к тому месту экрана, откуда получили копию, сотрет эту часть экрана. Если дважды использовать эту процедуру с **1**, то изображение не изменится.

Таким способом можно перемещать изображение на экране создавая иллюзию движения (при этом увеличивать или уменьшать координаты - в цикле).

Пример 4. Считая, что осуществлена в программе инициализация графического режима, описаны все переменные, кроме массива для сохранения прямоугольного фрагмента, рисунок выведен на экран с координатами - написать фрагмент программы для движения данного рисунка с верхнего левого угла к нижнему правому.

```
Var
    im: array [1..150, 1..125] of byte;
```

```
……
getimage(250,150,400,275, im);
putimage(250,150,im,1);
for i:=1 to 90 do
  begin
        putimage(i*5,i*3,im,0)
  end;
readkey;
```

2 Контрольные вопросы

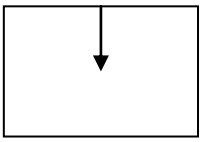
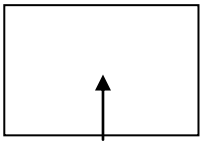
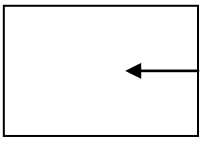
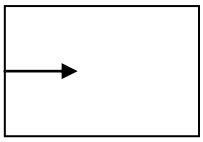
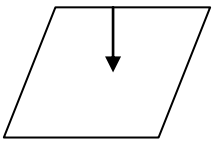
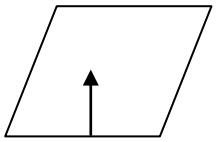
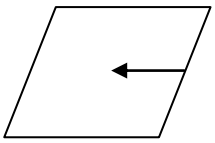
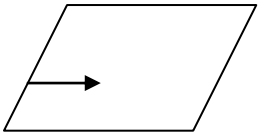
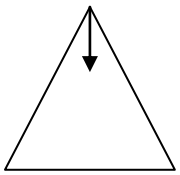
1. Как движущееся изображение создается в реальных условиях?
2. Словесный алгоритм построения движущихся изображений на ЕОМ.
3. Назовите процедуры модуля Graph которые используются для движения сложных окрашенных изображений независимо от того в каком месте экрана они выведены и независимо от других изображений.
4. Какая процедура заносит в память графическое изображение?
5. Как описать параметр im?
6. Какая процедура выводит из памяти на экран графическое изображение?
7. Чем отличаются различные значения маски Mode в процедуре putimage?
8. Составьте схему алгоритма и программу движения точки в середине экрана по вертикали.
9. Составьте схему алгоритма и программу движения точки в середине экрана по горизонтали.
10. Составьте схему алгоритма и программу движения точки по диагонали.
11. Составьте схему алгоритма и программу движения точки по треугольнику.
12. Объясните оператор ***getimage(x1,y1,x2,y2,im)***

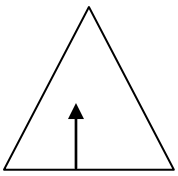
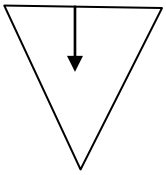
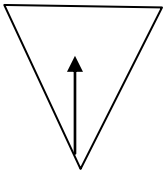
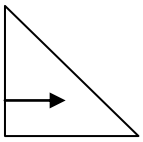
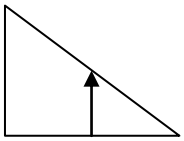
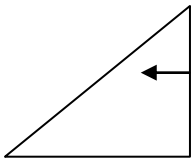
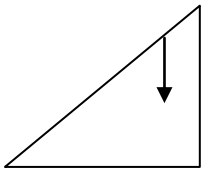
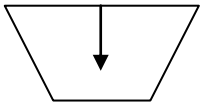
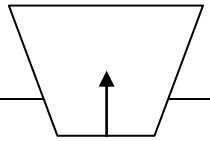
3 Задание

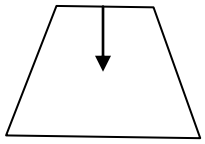
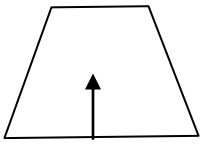
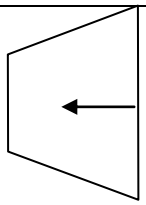
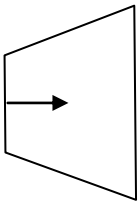
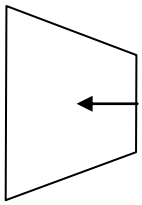
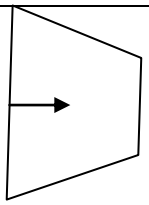
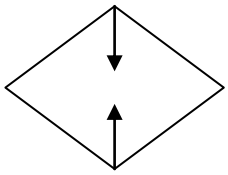
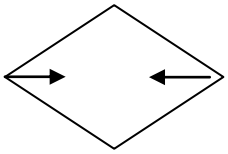
1. Согласно варианту, составить схему алгоритма и ТР-программ для построения на экране монитора подвижного объекта, который представлен в таблице №1.
2. В программе обеспечить окраски объекта в том направлении, которое указано в таблице №1 (с помощью ⟶).
4. Цветом объекта и экрана задаться самостоятельно.
3. Сделать выводы о проделанной работе.

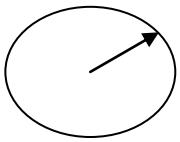
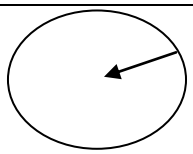
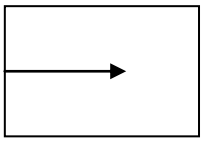
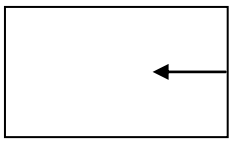
Таблица 1. ***Варианты заданий***

№ варианта	Объект	Траектория движения

1		По диагонали с верхнего левого угла
2		По диагонали из нижнего левого угла
3		По диагонали с верхнего правого угла
4		По диагонали из нижнего правого угла
5		По горизонтали слева направо
6		По горизонтали с право на левая
7		По вертикали сверху в низ
8		По вертикали снизу в верх
9		По горизонтали слева направо

10		По горизонтали с право на левая
11		По вертикали сверху в низ
12		По вертикали снизу в верх
13		По диагонали с верхнего левого угла
14		По диагонали из нижнего правого угла
15		По вертикали снизу вверх
16		По горизонтали слева направо
17		По вертикали снизу в верх
18		По горизонтали слева направо

19		По диагонали с верхнего левого угла
20		По вертикали сверху в низ
21		По горизонтали слева направо
22		По вертикали снизу в вверх
23		По горизонтали слева направо
24		По вертикали сверху в низ
25		По вертикали сверху в низ
26		По горизонтали с право на лево

27		По диагонали из нижнего левого угла
28		По диагонали с верхнего левого угла
29		По диагонали с верхнего левого угла
30		По диагонали из нижнего левого угла