1.

В программе надо использовать библиотеку графических функций – GraphABC. Для подключения библиотек в Паскале используется специальная инструкция **Uses**:

**Program** GraphicProbe\_01;

**Uses**

GraphABC;

...

2.

**Графические примитивы**, в компьютерной геометрии, – это основные геометрические фигуры, для которых написаны эффективные алгоритмы рисования.

Для программиста, графические примитивы – это процедуры и функции, с помощью которых выполняется отрисовка (рисование) геометрических фигур в графическом окне программы.

В PascalABC.NET имеются следующие графические примитивы:

Точки,

Линии (отрезки) и Ломаные линии (состоят из набора связанных отрезков),

Прямоугольники, со сторонами, параллельными осям,

Круги и эллипсы, у которых оси симметрии параллельны осям,

Прямоугольники со скруглёнными углами,

Кривые Безье и сцепленные

(т.е. Вы можете нарисовать любую относятся: линии

3.

Рисование осуществляется текущим пером (линии), текущей кистью (заливка замкнутых областей) и текущим шрифтом (вывод строк).

3.

Сводный список графических примитивов (процедур и функций)

**procedure SetPixel(x,y: integer; c: Color);**Закрашивает пиксел с координатами (x,y) цветом **c**

**procedure PutPixel(x,y: integer; c: Color);**Закрашивает пиксел с координатами (x,y) цветом **c**

**function GetPixel(x,y: integer): Color;**Возвращает цвет пиксела с координатами (x,y)

**procedure MoveTo(x,y: integer);**Устанавливает текущую позицию рисования в точку (x,y)

**procedure LineTo(x,y: integer);**Рисует отрезок от текущей позиции до точки (x,y). Текущая позиция переносится в точку (x,y)

**procedure LineTo(x,y: integer; c: Color);**Рисует отрезок от текущей позиции до точки (x,y) цветом c. Текущая позиция переносится в точку (x,y)

**procedure Line(x1,y1,x2,y2: integer);**Рисует отрезок от точки (x1,y1) до точки (x2,y2)

**procedure Line(x1,y1,x2,y2: integer; c: Color);**Рисует отрезок от точки (x1,y1) до точки (x2,y2) цветом c

**procedure FillCircle(x,y,r: integer);**Заполняет внутренность окружности с центром (x,y) и радиусом r

**procedure DrawCircle(x,y,r: integer);**Рисует окружность с центром (x,y) и радиусом r

**procedure FillEllipse(x1,y1,x2,y2: integer);**Заполняет внутренность эллипса, ограниченного прямоугольником, заданным координатами противоположных вершин (x1,y1) и (x2,y2)

**procedure DrawEllipse(x1,y1,x2,y2: integer);**Рисует границу эллипса, ограниченного прямоугольником, заданным координатами противоположных вершин (x1,y1) и (x2,y2)

**procedure FillRectangle(x1,y1,x2,y2: integer);**Заполняет внутренность прямоугольника, заданного координатами противоположных вершин (x1,y1) и (x2,y2)

**procedure FillRect(x1,y1,x2,y2: integer);**Заполняет внутренность прямоугольника, заданного координатами противоположных вершин (x1,y1) и (x2,y2)

**procedure DrawRectangle(x1,y1,x2,y2: integer);**Рисует границу прямоугольника, заданного координатами противоположных вершин (x1,y1) и (x2,y2)

**procedure FillRoundRect(x1,y1,x2,y2,w,h: integer);**Заполняет внутренность прямоугольника со скругленными краями; (x1,y1) и (x2,y2) задают пару противоположных вершин, а w и h – ширину и высоту эллипса, используемого для скругления краев

**procedure DrawRoundRect(x1,y1,x2,y2,w,h: integer);**Рисует границу прямоугольника со скругленными краями; (x1,y1) и (x2,y2) задают пару противоположных вершин, а w и h – ширину и высоту эллипса, используемого для скругления краев

**procedure Circle(x,y,r: integer);**Рисует заполненную окружность с центром (x,y) и радиусом r

**procedure Ellipse(x1,y1,x2,y2: integer);**Рисует заполненный эллипс, ограниченный прямоугольником, заданным координатами противоположных вершин (x1,y1) и (x2,y2)

**procedure Rectangle(x1,y1,x2,y2: integer);**Рисует заполненный прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2)

**procedure RoundRect(x1,y1,x2,y2,w,h: integer);**Рисует заполненный прямоугольник со скругленными краями; (x1,y1) и (x2,y2) задают пару противоположных вершин, а w и h – ширину и высоту эллипса, используемого для скругления краев

**procedure Arc(x,y,r,a1,a2: integer);**Рисует дугу окружности с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки)

**procedure FillPie(x,y,r,a1,a2: integer);**Заполняет внутренность сектора окружности, ограниченного дугой с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки)

**procedure DrawPie(x,y,r,a1,a2: integer);**Рисует сектор окружности, ограниченный дугой с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки)

**procedure Pie(x,y,r,a1,a2: integer);**Рисует заполненный сектор окружности, ограниченный дугой с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки)

**procedure DrawPolygon(points: array of Point);**Рисует замкнутую ломаную по точкам, координаты которых заданы в массиве points

**procedure FillPolygon(points: array of Point);**Заполняет многоугольник, координаты вершин которого заданы в массиве points

**procedure Polygon(points: array of Point);**Рисует заполненный многоугольник, координаты вершин которого заданы в массиве points

**procedure Polyline(points: array of Point);**Рисует ломаную по точкам, координаты которых заданы в массиве points

**procedure Curve(points: array of Point);**Рисует кривую по точкам, координаты которых заданы в массиве points

**procedure DrawClosedCurve(points: array of Point);**Рисует замкнутую кривую по точкам, координаты которых заданы в массиве points

**procedure FillClosedCurve(points: array of Point);**Заполняет замкнутую кривую по точкам, координаты которых заданы в массиве points

**procedure ClosedCurve(points: array of Point);**Рисует заполненную замкнутую кривую по точкам, координаты которых заданы в массиве points

**procedure TextOut(x,y: integer; s: string);**Выводит строку s в прямоугольник к координатами левого верхнего угла (x,y)

**procedure TextOut(x,y: integer; n: integer);**Выводит целое n в прямоугольник к координатами левого верхнего угла (x,y)

**procedure TextOut(x,y: integer; r: real);**Выводит вещественное r в прямоугольник к координатами левого верхнего угла (x,y)

**procedure DrawTextCentered(x,y,x1,y1: integer; s: string);**Выводит строку s, отцентрированную в прямоугольнике с координатами (x,y,x1,y1)

**procedure DrawTextCentered(x,y,x1,y1: integer; n: integer);**Выводит целое значение n, отцентрированное в прямоугольнике с координатами (x,y,x1,y1)

**procedure DrawTextCentered(x,y,x1,y1: integer; r: real);**Выводит вещественное значение r, отцентрированное в прямоугольнике с координатами (x,y,x1,y1)

**procedure FloodFill(x,y: integer; c: Color);**Заливает область одного цвета цветом c, начиная с точки (x,y).

*Пример программы, которая загружает 1 картинку из файла и рисует её.*

**Program** TryPic;

**Uses**

GraphABC;

**Var**

V: Picture;

i: integer;

x,y: integer;

**BEGIN**

// задаём размеры графического окна

SetWindowSize(500, 400);

// очистка окна, закраска бирюзовым цветом

ClearWindow(clCyan);

// загрузка картинки из файла

V := Picture.Create('frame1.png');

// выбор "прозрачного" цвета

V.TransparentColor := clWhite;

// включить режим "прозрачности"

V.Transparent := true;

// нарисовать картинку,

// левый верхний угол в точке х=150, y=100

V.Draw(150, 100);

**END**.

*Пример программы, которая загружает несколько картинок и «анимирует» их.*

**Program** TryPic;

**Uses**

GraphABC;

**Var**

V: **array**[0..2] **of** Picture;

i: integer;

x,y: integer;

s: integer;

k: integer;

**BEGIN**

// создать окно, задать его размеры

SetWindowSize(500, 400);

// очистка - закрасить всё окно одним цветом

ClearWindow(clCyan);

// загрузить несколько картинок

**for** i:=0 **to** 2 **do begin**

V[i] := Picture.Create('frame'+IntToStr(i+1)+'.png');

V[i].TransparentColor := clWhite;

V[i].Transparent := true;

**end**;

// s - это номер кадра, он будет циклически изменяться от 0 до 2

s:=0;

// котёнок Гав бежит налево

x:=450; y:=100; // начальное положение котёнка

**while** x>0 **do begin**

ClearWindow(clCyan); // очистка экрана

V[s].Draw(x,y); // рисуем текущий кадр

s:=(s+1) **mod** 3; // выбираем следующий кадр

x:=x-30; // сдвиг котёнка

Sleep(100); // задержка времени - чтобы человек увидел кадр

**end**;

// "отзеркалить" все картинки в горизонтальном направлении

**for** i:=0 **to** 2 **do begin**

V[i].FlipHorizontal;

**end**;

// котёнок Гав бежит направо

**while** x+V[0].Width<WindowWidth **do begin**

ClearWindow(clCyan);

V[s].Draw(x,y);

s:=(s+1) **mod** 3;

x:=x+30;

Sleep(100);

**end**;

// "отзеркалить" все картинки в горизонтальном направлении

**for** i:=0 **to** 2 **do begin**

V[i].FlipHorizontal;

**end**;

**END**.