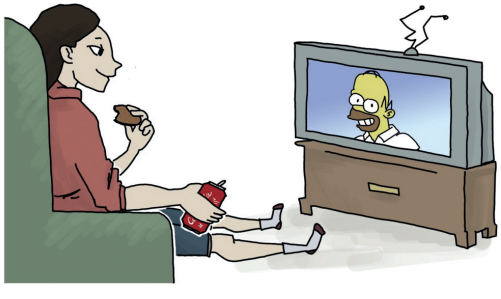
Информатика

Грацианова Татьяна Юрьевна,
Кандидат физико-математических наук,
преподаватель подготовительных курсов
факультета Вычислительной математики
и кибернетики МГУ им. М.В. Ломоносова

Волшебная точка

В прошлый раз мы на языке Паскаль писали программы, рисующие картинки с помощью графической процедуры SetPixel – точки[1]. Мы рисовали линии. Продолжим эксперименты с точкой.

Будем создавать мультфильмы. Как – мультфильмы? Настоящие? Мультфильм – движущееся изображение, так что вполне настоящие.



[1] Потенциал 2019 № 1

Впрочем, наши изображения уже и были движущимися, мы просто не успевали это увидеть. Вспомним, как с помощью процедуры изображения точки и оператора цикла нарисовать линию:

```
uses GraphABC;
var X : Integer;
Begin
  For X:=0 to 640 do
      SetPixel(X, 10,clRed);
End.
```

Такая линия появляется на экране потому, что последовательно рисуются точки, но они появляются так быстро, что мы не замечаем отдельных точек, видим сразу готовую линию. Как бы замедлить появление точек? Существует несколько способов, один из них – использование стандартной процедуры Delay. Эта процедура находится в библиотеке CRT, название библиотеки надо добавить в первую строку программы, теперь она будет выглядеть так:

```
uses Crt,GraphABC;
```

При работе такой программы у вас появится сразу два новых окна: в одном будет рисоваться прямая, а другое останется пустым.

Процедура Delay имеет один параметр – натуральное число (его надо ставить в скобках, как и все параметры процедур). Чем больше число, тем на большее время задержка. Поставим эту процедуру после SetPixel. Какое число будет в скобочках? А его можно подобрать – кому какое больше понравится. Поставите 2 – прямая будет рисоваться быстро, поставите 100 – очень медленно, так, что ждать надоест. Конкретное время задержки зависит от того, какой у вас компьютер, какая версия Паскаля.

Как – поставили 100, даже 1000, но она всё равно сразу появляется? А процедуру в правильном месте написали? Она ведь в цикле должна быть! Вспоминаем, как устроен оператор цикла FOR. Он выполняет ОДИН оператор, который записан у него после DO. Поэтому исправляйте ошибку, если в программе написано:

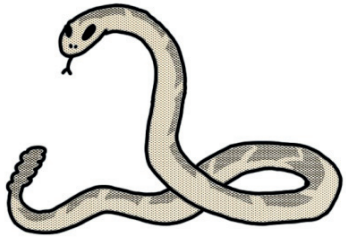
```
For X:=0 to 640 do
          SetPixel(X, 10,clRed);
          Delay(100)
```

Получается, что после DO стоит один оператор, так что сначала выполняется цикл, ставятся 640 точек, и только потом – задержка, которая уже ничего не даёт. Чтобы эффект задержки ощущался, она должна быть внутри цикла, чтобы программа приостанавливалась после рисования каждой точки. Для этого операторы надо заключить в операторные скобки Begin – End.

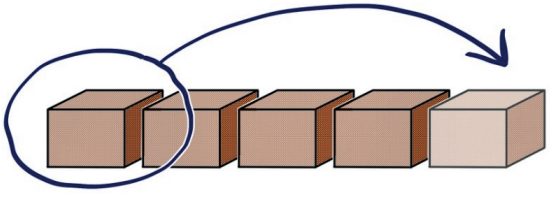
Приведём полную программу:

```
uses Crt,GraphABC;
var X : Integer;
Begin
  For X:=0 to 640 do
  Begin   SetPixel(X, 10,clRed);
          Delay(20)
  End
End.
```

А теперь хочу, чтобы по экрану полз червячок или змейка – кому как приятнее. Никто не нравится? Тогда скажем «по-научному»: пусть по экрану двигается отрезок. Как это сделать?



Предположим, что у нас есть красный отрезок на белом фоне. Пририсуем к правому концу отрезка красную точку, а левый конец (самую первую точку отрезка) «закрасим» в белый цвет (напомним – это цвет фона). Отрезок продвинется на одну точку вправо. Конечно, он при этом сначала немножко удлинится, потом опять станет прежней длины, но всё это произойдет настолько быстро, изменения длины столь незначительны, что глаз не успеет этого заметить, а вот движение прекрасно заметит. Мы как бы переставляем точку с левого конца отрезка на правый. На этом и основан эффект мультипликации и вообще кино: отдельные быстро сменяющиеся картинки человеческий глаз принимает за единый движущийся объект (конечно, если картинки хорошо подобраны). В цирке можно видеть такой фокус: клоун быстро переставляет кирпичики – самый левый вправо, и кажется, что лента из кирпичей плывет по воздуху.



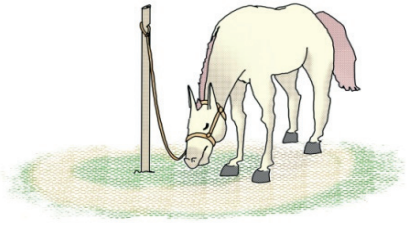
Программа будет выглядеть так:

```
uses Crt,GraphABC;
var X : Integer;
Begin
  For X:=10 to 640 do
  Begin   SetPixel(X, 100,clRed);{красная точка справа}
          Delay(10);
          SetPixel(X-10, 100,clWhite);{белая точка слева}
          Delay(10);
  End
End.
```

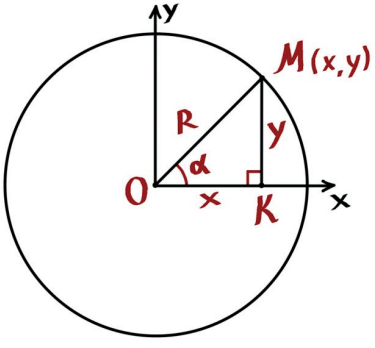
Координаты белой точки (X-10,100): 10 – это длина отрезка. Обратите внимание, мы начали цикл теперь не с 0, а с 10, иначе значение X-10 получается отрицательным, ставить точку с отрицательными координатами нельзя. Конечно, из-за этого у нас в начале на экране появляется точка, которая только через 10 шагов цикла превращается в нужный отрезок, но мы это заметить не успеваем. Если хочется добиться совсем правильного результата (чтобы в начале на экране сразу рисовался отрезок нужной длины и потом начинал двигаться), надо написать программу, в которой будет два оператора цикла: для рисования отрезка и для его движения.

Теперь вы можете сделать «змейку», которая двигается вдоль границ экрана. Для этого понадобится 4 оператора цикла: для движения вдоль каждой стороны. Для превращения нашей программы в игру «змейка» осталось только научиться реагировать на нажатие клавиш на клавиатуре. Как видите, со «змейкой» всё просто. Именно поэтому эта игра столь популярна у производителей мобильных телефонов.

А что это наша точка бегает только по прямой? Давайте заставим её бежать по кругу. Для этого придется вспомнить формулу окружности. Нет, $X^2+Y^2=R^2$ здесь не подойдет. Формула правильная, но для наших целей не очень удобная. Вспомним другую.



Начнём с определения. Окружность – это множество точек, расположенных на равном расстоянии от центра. Это расстояние называется радиус.



Начертим окружность с центром в начале координат. Под некоторым углом (обозначим его Alfa) проведём

радиус, пусть он пересекается с окружностью в точке M с координатами (x, y). Опустим перпендикуляр MK на ось OX. Треугольник OMK – прямоугольный. |OM|= R; |OK| = x; |MK| = y. Катеты прямоугольного треугольника можно выразить через гипотенузу (радиус) и угол. А катеты – это и есть координаты точки. Значит,

x =R*cos (Alfa) ; y=R*sin (Alfa)

Меняя угол Alfa ,будем получать разные точки окружности. Для использования этой формулы для рисования окружности на экране надо вспомнить:

- окружность у нас будет не в начале координат (ведь оно у нас в углу экрана);
- координаты могут быть только целыми, в формуле участвуют синус и косинус, значения которых нецелые, так что перед присваиванием результата координате, его надо округлить. Воспользуемся функцией округления ROUND. Формула приобретает следующий вид:

```
X=X0+ ROUND (R*cos(Alfa))) ;  Y=Y0+ ROUND (R*sin(Alfa))
```

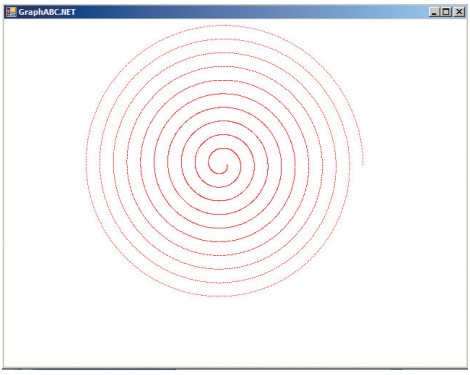
где X0, Y0 – координаты центра окружности.

Чтобы построить из точек окружность, надо, во-первых, пробежаться по ней целиком, «сделать полный круг», то есть угол должен меняться от 0 до 360°. В Паскале углы изменяются в радианах, так что от 0 до 2 π. И второе условие – расстояние между соседними точками должно быть небольшим, оно зависит от того, насколько будет изменяться угол, от приращения угла. Если это число сделать очень маленьким, будет нарисована окружность, причём рисоваться она будет достаточно медленно, и задержка не нужна: компьютер делает сложные вычисления, к тому же очень много раз. Если приращение взять побольше, станут видны отдельные точки на окружности.

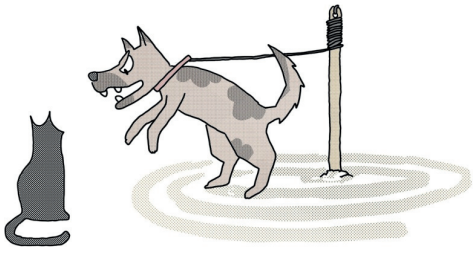
Программа такова:

```
uses Crt,GraphABC;
var X, Y, X0, Y0, R : Integer;
    Alfa, dAlfa:Real;
Begin
   X0:= 300; Y0:=200; R:=100;
  Alfa:=0;
  dAlfa:=0.01; {приращение угла}
  While Alfa<=2*Pi do
  Begin   X:=X0+ ROUND (R*cos(Alfa)) ;
                Y:=Y0+ ROUND (R*sin(Alfa));
                SetPixel(X, Y,clRed);
               Alfa:=Alfa+dAlfa;
  End
End.
```

А что будет, если изменять не только угол, но и радиус? Спираль. Посмотрим, какие изменения надо сделать в программе, чтобы получить такую картинку.



Во-первых, переменная для значения радиуса теперь будет не целого, а вещественного типа. Начальное значение радиуса можно взять поменьше, в цикле записать его изменение R:=R+0.3. Число может быть другое, это расстояние между соседними кружочками спирали. Ещё надо изменить условие цикла. Оно зависит от угла. При рисовании окружности угол должен был описать один полный круг (2 π). Сейчас нам нужно нарисовать несколько кругов, так что поставим, например, 20*Pi, то есть наша спираль будет состоять из 10 кружочков. Заметим, что здесь увлекаться не стоит, это число не должно быть слишком большим, а то спираль может «убежать» за границы экрана, то есть одна из координат станет отрицательной или больше допустимого максимального значения. Попытка нарисовать точку с такими координатами вызовет ошибку, так что вместо спирали мы увидим чёрный экран с сообщением о недопустимом значении параметра. Если добавить задержку, спираль будет появляться не сразу, а постепенно как бы раскручиваться.



Тело программы будет выглядеть так:

```
X0:= 300; Y0:=200; R0:=20;
Alfa:=0; dR:=0.03;
dAlfa:=0.01; {приращение угла}
While Alfa<=20*Pi do
Begin   X:=X0+ ROUND (R*cos(Alfa)) ;
        Y:=Y0+ ROUND (R*sin(Alfa));
        SetPixel(X, Y,clRed);
        Alfa:=Alfa+dAlfa;
        R:=R+dR;
       delay(5)
End
```

Но ведь обещали мультфильм! Надо, чтобы наш «червячок» теперь побежал по спирали. Делается это так же, как и с отрезком: следует в один конец «червячка» добавлять красные точки, в другой – белые. Только с отрезком было проще, у точек координата у не менялась, а х легко вычислялась. Здесь в формуле участвует угол, радиус, и оба значения меняются. Можно, конечно, запоминать их (например, в массиве):

вычисляем координаты красной точки, записываем их в массив, рисуем точку. А чтобы нарисовать белую точку (то есть стереть красную), берём координаты из массива, точку рисуем, а числа в массиве сдвигаем.

Есть и другой способ. Перепишем формулы, чтобы и угол, и радиус зависели от номера точки (обозначим его I). Вместо Alfa:=Alfa+dAlfa будет I*dAlfa, а вместо R:=R+dR получим R0+I*dR. Значения переменных можно не вычислять отдельно, а сразу подставить в оператор рисования точки. Получим:

```
SetPixel(X0+ ROUND (R0+I*dR*cos(I*dAlfa)),
Y0+ ROUND (R0+I*dR*sin(I*dAlfa)),clRed);
```

Таким образом, будет нарисована красная точка. Только цикл теперь будем использовать другой – FOR, ведь у нас всё зависит от переменной I, которая станет меняться с шагом 1. Считать координаты белой точки надо по той же формуле, только для I взять «старое» значение, то, для которого уже нарисована красная точка, то есть в формуле будем использовать вместо I другую переменную (например, J), на сколько-то меньшую I. Например, J:=I-20. 20 – это число, от которого зависит размер нашего «червячка» (к сожалению, назвать его длиной нельзя).

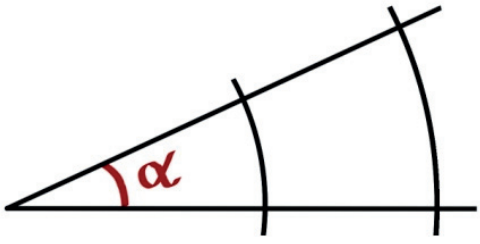
Итак, получаем:

```
uses Crt,GraphABC;
var X, Y, X0, Y0, I, j : Integer;
    Alfa, dAlfa, dR, R0:Real;
Begin
    X0:= 300; Y0:=200; R0:=20;
     dAlfa:=0.01; dR:=0.03;
  For I:=1 to 5000 do
  Begin  SetPixel(X0+ ROUND (R0+I*dR*cos(I*dAlfa)),
                     Y0+ ROUND (R0+I*dR*sin(I*dAlfa)),clRed);
               J:=I-20;
              delay(1);
            SetPixel(X0+ ROUND (R0+J*dR*cos(J*dAlfa)),
                 Y0+ ROUND (R0+J*dR*sin(J*dAlfa)),clWhite);
  End
End.
```

Запускаем... И довольно долго ничего не видим. Придётся набраться терпения. Через несколько секунд можно будет заметить в центре экрана маленькую точку, которая довольно медленно двигается по спирали.

Почему так медленно (даже задержку можно уменьшить)? Мы вычисляем все координаты по два раза, к тому же убрали дополнительные переменные и вместо значений переменных в формулах также используем вычисления. А вычисления сложные: умножение, тригонометрические функции... Вот и считает наш компьютер так долго, что это даже становится заметно.

Можно видеть, что вначале картинка рисуется очень медленно, потом всё быстрее, и к тому же длина «змейки» со временем увеличивается. Почему так происходит? Догадайтесь сами. Вот рисунок-подсказка:



Итак, мы получили с помощью точки линию, окружность, спираль и даже «змейку». А что же можно получить с помощью линии? Об этом в следующий раз.

Новости Новости Новости Новости

Команда МФТИ заняла первое место в соревновании Challenge Shield на чемпионате мира по робофутболую.

Со 2 по 8 июля в Сиднее прошел чемпионат мира по робофутболу RoboCup 2019. Команда МФТИ «Старкит» участвовала в лигах Standard Platform League (SPL) и Humanoid KidSize. В SPL, где команды роботов-гуманоидов NAO составом до пяти роботов играют в футбол в автономном режиме, команда заняла первое место в соревновании Challenge Shield.

Источник: https://mipt.ru/news