**Лабораторная работа №12**

**Анимация**

**1. Цель работы: Программирование движения графического объекта**

**2. Основные сведения**

***Свободное движение объекта.***

Рассмотрим объект, который движется по экрану. Пусть это будет круг с радиусом 20 пикселей. При движении его координаты будут меняться. Для перемещения зададим следующий алгоритм:

1) рисуем фигуру на экране;

2) делаем небольшую задержку (обычно 10-20 мс);

3) стираем фигуру;

4) меняем ее координаты;

5) переходим к шагу 1.

Эти действия повторяются до тех пор, пока не будет получена команда «закончить движение» (например, нажата клавиша Esc или объект вышел за правую границу экрана).

**#include <conio.h>**

**#include <graphics.h>**

**void Draw ( int x, int y, int color1,int color2 ) // функция рисования круга**

**// color1 цвет линий, color2 цвет заливки**

**{ setcolor(color1);**

**setfillstyle ( 1, color2 ); // сплошная заливка, цвет color2**

**circle(x,y,20); // рисуем окружность радиусом 20 цвет color1**

**floodfill(x,y,color1); // заполняем цветом color2 до границы color1**

**}**

**main()**

**{**

**int x, y; // координаты центра окружности**

**initwindow (400, 400); // открыть окно для графики размером 400х400**

**setfillstyle(1, 0); // сплошная заливка, чёрный цвет (для экрана)**

**bar (0, 0, 399, 399); // заливаем фон чёрным**

**x = 20; y = 200; // начальные координаты**

**/\* анимация \*/**

**while ( x + 20 < 400 ) // пока не коснулся границы окна**

**{**

**if ( kbhit() ) // если нажата клавиша…**

**if ( getch() == 27 ) // если нажата Esc**

**break; // выход из цикла**

**Draw ( x, y, 15,4 ); // рисуем белый круг, залитый красным**

**delay ( 20 ); // задержка 20 мс**

**Draw ( x, y, 0, 0); // стираем, перерисовывая всё цветом фона**

**x ++; // перемещаем**

**}**

**closegraph(); // закрываем графическое окно**

**}**

Пусть движение круга происходит на черном фоне. Тогда самый быстрый и простой способ стереть его – это нарисовать его же, но всё - черным цветом. Поэтому удобно написать функцию, параметрами которой являются координаты x и y, а также цвета color1 – цвет окружности и color2 – цвет заливки. Все действия, которые входят в алгоритм, надо выполнить много раз, поэтому применяем цикл. Кроме того, мы заранее не знаем, сколько раз должен выполниться этот цикл, поэтому применяем цикл while . Условие окончания цикла – выход фигуры за границы экрана или нажатие на клавишу

Esc. Мы будем использовать окно размером 400 на 400 пикселей. При этом координата x может меняться от 0 до 399, поэтому нужное нам условие продолжения цикла выглядит так:

x + 20 < 400 // 20 – радиус окружности, x – координата центра окружности;

Когда это условие нарушается, круг «уехал» за границу окна и нужно закончить выполнение программы. Надо также обеспечить выход по клавише Esc. При этом объект должен двигаться и нельзя просто ждать нажатия на клавишу с помощью функции getch(), которая приостановит выполнение программы. В этом случае проверяем, нажата ли какая-нибудь клавиша; это делает функция kbhit, которая возвращает результат 0, если никакая клавиша не нажата, и ненулевое значение, если нажали любую клавишу. В программе проверка выполнена с помощью условного оператора

if ( kbhit() ) { ... }

Если клавиша нажата, то определяем код этой клавиши, вызывая функцию getch. Если полученный код равен коду клавиши Esc (27), то выходим из цикла.

***Управление движением клавишами-стрелками.***

Принцип работы программы следующий: получив код клавиши, надо сдвинуть объект в соответствующую сторону. Если обозначить изменения координат фигуры x и y за 1 шаг цикла

через dx и dy, для движения в четырех направлениях получаем:

движение влево dx > 0, dy = 0

движение вправо dx < 0, dy = 0

движение вверх dx = 0, dy < 0

движение вниз dx = 0, dy > 0

Это значит, что надо сделать выбор одного из четырех вариантов в зависимости от кода нажатой клавиши. Для этого можно использовать оператор switch, который позволяет легко организовать выбор из нескольких

вариантов. Еще одна проблема связана с тем, что клавиши управления курсором (стрелки) – не совсем обычные клавиши. Они относятся к группе функциональных клавиш, у которых нет кодов в таблице символов. Когда нажата одна из специальных клавиш, система реагирует на нее как ***на***

***2 нажатия***, причем для первого код символа ***всегда равен нулю***, а для второго мы получим специальный код (так называемый скан-код, номер клавиши на клавиатуре). Мы будем использовать упрощенный подход, когда анализируется только этот второй код:

влево 75

вверх 72

вправо 77

вниз 80

При этом объект будет также реагировать на нажатие клавиш с кодами 75, 77, 72 и 80 в таблице символов, то есть на заглавные латинские буквы K,

M, H и P.

Рассмотрим алгоритм, при выполнении которого фигура будет двигаться только тогда, когда мы нажмем на клавишу-стрелку. В цикле мы сначала рисуем фигуру, ждем нажатия на клавишу и принимаем ее код с помощью функции getch() . После этого стираем фигуру в том же месте (пока не изменились координаты) и, в зависимости от этого кода, меняем координаты

фигуры нужным образом. Здесь используется бесконечный цикл while(1). Выйти из него можно только одним способом – через оператор break (досрочный выход из цикла).

**while ( 1 ) // бесконечный цикл**

**{**

**Draw ( x, y, color1,color2 ); // рисуем**

**code = getch(); // ждем нажатия клавиши**

**if ( code == 27 ) break; // если Esc, то выход**

**Draw ( x, y, color3,color4 ); // стираем квадрат**

**switch ( code )**

**{ // выбор направления**

**case 75: x --; break; // влево**

**case 77: x ++; break; // вправо**

**case 72: y --; break; // вверх**

**case 80: y ++; // вниз**

**}**

}

В операторе switch значения координат меняются на единицу, хотя можно использовать любой шаг dx, dy.

**3. Выполнение работы**

Изобразить на экране «снежную карусель». Координаты снежинок изменяются случайным образом. Для этого заполнить 2 массива с координатами, например, x[100], y[100] с помощью датчика случайных чисел, таким образом, чтобы значения элементов не выходили за границы графического окна. Снежинки окрашиваются в различные цвета, появляются и исчезают в случайных координатах окна с задержкой в несколько десятков миллисекунд. Одновременное количество снежинок на экране – 100. Для стирания снежинки можно ее перерисовать цветом фона.

**4. Контрольные вопросы**

4.1. Каков простейший алгоритм перемещения фигуры на экране?

4.2. Как осуществить выполнения программы до нажатия любой клавиши?

4.3. Как определить код функциональной клавиши?

4.4. Как стереть изображение?

4.5. Как задать направление движения объекта?

4.6. Как задать скорость движения объекта?