Оглавление

[Введение 3](#_Toc390268878)

[Глава 1. Общие сведения 4](#_Toc390268879)

[1.1 Торговая марка «Электроника» 4](#_Toc390268880)

[1.2 Описание и правила игры 5](#_Toc390268881)

[1.3 Используемые технические средства 5](#_Toc390268882)

[Глава 2. Разработка программного продукта 6](#_Toc390268883)

[2.1 Выбор методов и средств реализации 6](#_Toc390268884)

[2.2 Реализация главного меню 7](#_Toc390268885)

[2.3 Создание игрового поля 8](#_Toc390268886)

[2.4 Реализация волка и корзины 9](#_Toc390268887)

[2.5 Реализация скатывающихся яиц 10](#_Toc390268888)

[Заключение 11](#_Toc390268889)

[Список литературы 12](#_Toc390268890)

[ПРИЛОЖЕНИЕ 1 13](#_Toc390268891)

# Введение

Данная курсовая работа посвящена самой первой и самой популярной игре из серии советских портативных электронных игр с жидкокристаллическим экраном,  производимых под торговой маркой «Электроника».

**«Ну, погоди!»** («Электроника ИМ-02») — электронная игра, являющаяся неофициальным клоном Nintendo EG-26 Egg из серии Nintendo Game & Watch 1981 года, где аббревиатура «ИМ» обозначает «игру микропроцессорную», производилась с 1984 года на территории СССР.

Актуальность исследования данной темы определяется тем, что создание игр является одним из самых популярных направлений в программировании, позволяющее изучить все графические возможности компьютера.

Целью данной курсовой работы является создание программного продукта при помощи стандартных графических модулей в С++. Для достижения данной цели были поставлены следующие задачи:

* изучить необходимые теоретические сведения для решения данной задачи;
* применить полученные знания для реализации программного продукта;
* создать пользовательский интерфейс;
* создать программный продукт, реализующий игру «Ну, погоди!»;
* провести тестирование и отладку программного продукта.

В первом разделе будут рассмотрены общие сведения, описание и правила игры, а второй раздел посвящен разработке программного продукта.

Заключение содержит в себе результаты проделанной работы при реализации игры «Ну, погоди!», а в приложении представлен листинг разработанного программного продукта.

# Глава 1. Общие сведения

## 1.1 Торговая марка «Электроника»

Практически все, кто родился и вырос в СССР, прекрасно помнят торговую марку «Электроника». Под этой маркой в советское время разными заводами выпускался широкий спектр бытовых электроприборов: телевизоры, компьютерные системы, калькуляторы, электронные часы, магнитофоны, видеомагнитофоны и другие изделия. Но не все догадывались, что многие изделия, выпущенные под этой маркой, были копиями образцов иностранной техники.

Первой неофициальной клонированной игрой и была «Ну, погоди!», – предел мечтаний многих советских детей. Розничная цена составляла 25 рублей, а среди детей бытовал миф, что при достижении 1000 очков игра показывает мультфильм «Ну погоди!».



Рисунок 1. Внешний вид игры «Ну, погоди!»

## 1.2 Описание и правила игры

Четыре курицы, сидящие на насестах, несут яйца, скатывающиеся вниз по четырём лоткам. Игроку, управляющему волком из мультфильма «Ну, Погоди!», который может занимать четыре позиции, требуется наловить как можно больше яиц в корзину. За пойманное яйцо игроку добавляется одно очко.

В случае падения яйца игроку добавляется штрафное очко. Если падение произошло в присутствии зайца, высунувшегося из окна домика в левом верхнем углу, то игроку добавляется половина штрафного очка. При наборе в корзину 200 и 500 яиц штрафные очки аннулируются. После получения трёх штрафных очков (от трёх до шести падений яиц) игра прекращается. При достижении 999 очков игра продолжается со счёта 0, а набранные штрафные очки не сбрасываются. Штрафные очки могут быть сброшены в дальнейшем при повторном достижении 200 очков.

## 1.3 Используемые технические средства

Для реализации данного программного продукта были использованы:

1. Входные данные:

* Нажатие клавиш 'Q', 'A', ‘D’, ‘E’ в режиме реального времени позволяют контролировать движение волка с корзиной в четырех положениях. Данная функция была реализована при помощи переключателя switch().

1. Отображаемые данные:

* фоновый рисунок;
* динамически генерируемые катящиеся яйца по лоткам;
* волк с корзиной, принимающий 4 различных положения;
* счёт, набранный игроком в процессе игры.

# Глава 2. Разработка программного продукта

## 2.1 Выбор методов и средств реализации

Основной задачей в программировании графики является графическое отображение объектов. Для построения изображений в среде Borland C используется ПДСК(Прямоугольная Декартова Система Координат) с началом координат в точке (0;0) в левом верхнем углу. Изменения по осям Ox и Oy аналогичны изменениям в ПДСК.

Для того, чтобы воспользоваться графическими методами, необходимо следующее:

1. Подключить библиотеку graphics.h, позволяющую использовать графику.
2. Подключить библиотеку dos.h, позволяющую использовать функцию *delay*(), которая отвечает за задержку информации на экране.
3. Объявить переменные типа int:

* gdriver – определяет графический драйвер по целочисленному значению.
* gmode – определяет разрешение экрана и цветовую гамму, в которых будет отображаться программа.

1. Для непосредственного включения графики, необходимо указать в функции *initgraph*():
   * + Ссылку на переменную &gdriver;
     + Ссылку на переменную &gmode;
     + Путь к графическим драйверам;
2. В самом конце программы (в конце функции *main*() ) выполнить зарезервированную функцию *closegraph*(), отвечающую за выключение графического режима.

## 2.2 Реализация главного меню

После запуска программы пользователь попадает в главное меню, в котором может выбрать любой из следующих пунктов:

* «PLAY», при выборе которого запускается игра;
* «HELP», при выборе которого пользователь может узнать управление и прочитать описание игры, а также сразу начать игру по нажатию клавиши «Enter» или выйти из программы по нажатию клавиши «Esc»;
* «QUIT», при выборе которого происходит выключение графического режима и последующий выход из программы .

Для реализации главного меню был использован принцип построения элементов по координатам, а навигация по элементам меню осуществляется при помощи стрелки «вниз». При нажатии стрелки «вниз» рамка-курсор, стоящая в положении «PLAY», сдвигается на одну позицию ниже. Происходит приблизительно следующее:

* задано начальное положение рамки-курсора;
* при нажатии стрелки «вниз» рамка-курсор, стоящая в положении «PLAY» закрашивается в цвет фона и заново прорисовывается на новом месте.

Все надписи выводятся на экран при помощи зарезервированной функции *outtextxy()*, отображающей заранее объявленные переменные строкового типа, содержащие названия пунктов меню, c определёнными координатами.

Если пользователь выберет при помощи рамки-курсора нужный ему пункт меню и нажмёт клавишу «Enter», то выполнится функция, соответствующая данному пункту меню. Проверка нажатия клавиш осуществляется при помощи зарезервированной функции *getch().* В данном случае стрелке «вниз» соответствует значение=80, а клавише «Enter»=13).

## 2.3 Создание игрового поля

Для создания игрового поля было использовано фоновое изображение в формате “1st.bmp”. После того, как пользователь в меню выберет пункт «PLAY» начнётся последовательная прорисовка каждого пикселя фонового рисунка в формате “1st.bmp”. Изображение прорисовывается несколько секунд, после чего следует выполнение функции *controls()*. Для того, чтобы загружать изображение в среде Borland C, необходимо сделать следующее:

* Подключить соответствующий графический драйвер, в данном случае –SVGA256.bgi, позволяющий использовать 256 цветов.
* Подключить соответствующие структуры и функцию, преобразовывающую цвета.

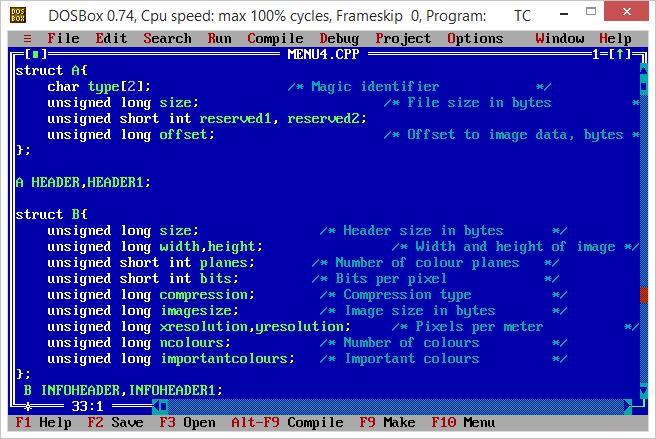


Рисунок 2. Структуры, позволяющие загружать изображение

## 2.4 Реализация волка и корзины

Для того, чтобы реализовать перемещение волка с корзиной использовался переключатель *switch(),* которому назначалось нажатие клавиш 'Q', 'A', ‘D’, ‘E’ в режиме реального времени.

После загрузки фонового изображения запускается функция *controls()*, которая и реализует перемещение волка с корзиной. В данной функции запущен цикл *while(1)*, который продолжается до тех пор, пока пользователь не нажмёт клавишу «Enter».

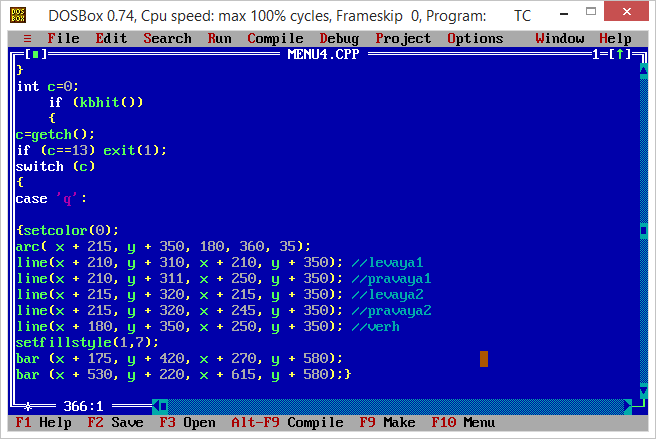


Рисунок 3. Пример из функции *controls(*), нажатие клавиши 'Q'

То есть, когда пользователь нажимает одну из четырех клавиш, то происходит прорисовка волка с корзиной в одном месте и зарисовка в фоновой цвет трёх других положений. За зарисовку в фоновой цвет отвечает функция *bar().*

## 2.5 Реализация скатывающихся яиц

Для реализации данного действия был также использован принцип построения элементов по координатам.

Внутри функции *controls()* были вставлены функции *egg1(), egg2(), egg3(), egg4(),* которые в коде объявлены выше самой функции *controls().* Также был реализован динамический вызов функций с помощью *randomize()* и *random().*

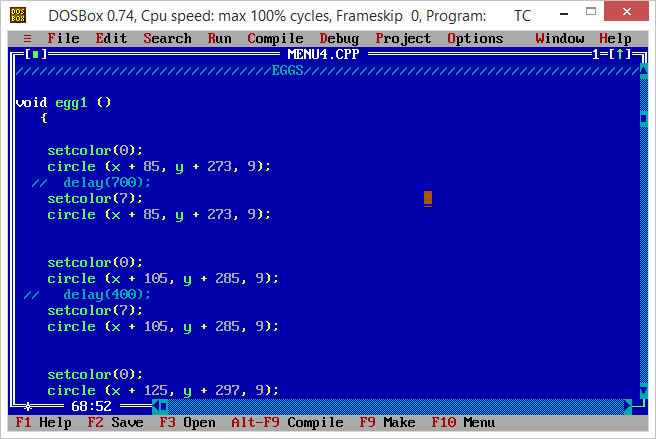


Рисунок 4. Пример прорисовки скатывающегося яйца

Но, учитывая тот факт, что *getch()* не пригоден для правильного выполнения данного программного продукта, функция заменилась функцией *kbhit(),* которая выполняет программу до тех пор, пока не будет достигнуто условие, детерминирующее выполнение программы, даже если нажата клавиша с предварительно зарезервированной функцией.

# Заключение

Хотелось бы отметить, что программирование графики и создание видеоигр достаточно трудоёмкий процесс, требующий полной отдачи и максимальной концентрации разработчиков, и, на первый взгляд, даже самая простая видеоигра может доставить немало неприятностей.

По итогам данной курсовой работы были выполнены следующие задачи:

* изучены необходимые теоретические сведения;
* полученные знания применены для реализации программного продукта;
* создан пользовательский интерфейс;
* создан программный продукт, реализующий игру «Ну, погоди!»;
* проведено тестирование и отладка программного продукта.

В ходе тестирования программного продукта были выявлены следующие недочёты:

* во время нажатия клавиш ‘Q’,’A’,’D’,‘E’ корзина прорисовывалась только после прорисовки последнего яйца, что было исправлено;
* не до конца реализован счёт и таблица рекордов.

В заключении хотелось бы сказать, что любой программный продукт имеет свои достоинства и недостатки, но работа уже сделана и должна быть оценена по достоинству.

# Список литературы

1. Шестопал О.В., Сташкова О.В. Программирование на языке высокого уровня СИ. Часть II: практикум — Тирасполь, 2010.
2. Стефан Р. Дэвис. C++ для чайников, 2003.
3. Джесс Либерти. Освой самостоятельно C++ за 21 день. — 3-е ред. — М.: «Вильямс», 2007.
4. Дональд Кнут. Искусство программирования, том 1. Основные алгоритмы. — 3-е изд. — М.: «Вильямс», 2006.
5. Керниган Б., Ритчи Д. Язык программирования Си. — 2-е изд. — М.: Вильямс, 2007.
6. Википедия. Свободная энциклопедия. — http://ru.wikipedia.org
7. Форум программистов и сисадминов. — http://www.cyberforum.ru
8. Гукин Д*.* Язык программирования Си для «чайников. — М.: Диалектика, 2006.

# ПРИЛОЖЕНИЕ

#include <graphics.h>

#include <iostream.h>

#include <conio.h>

#include <stdlib.h>

#include <dos.h>

#include <stdio.h>

#include <fstream.h>

#include <stdlib.h>

#include <string.h>

/////////////////////////////////Structures/////////////////////////////////////////

struct A{ //

char type[2]; /\* Magic identifier \*/ //

unsigned long size; /\* File size in bytes \*/ //

unsigned short int reserved1, reserved2; //

unsigned long offset; /\* Offset to image data, bytes \*/ //

}; //

//

A HEADER,HEADER1; //

//

struct B{ //

unsigned long size; /\* Header size in bytes \*/ //

unsigned long width,height; /\* Width and height of image \*/ //

unsigned short int planes; /\* Number of colour planes \*/ //

unsigned short int bits; /\* Bits per pixel \*/ //

unsigned long compression; /\* Compression type \*/ //

unsigned long imagesize; /\* Image size in bytes \*/ //

unsigned long xresolution,yresolution; /\* Pixels per meter \*/ //

unsigned long ncolours; /\* Number of colours \*/ //

unsigned long importantcolours; /\* Important colours \*/ //

}; //

B INFOHEADER,INFOHEADER1; //

////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////GLOBAL\_C//////////////////////////////////////////

int XCor=0, YCor=0;

int x=1, y=1;

int sc=0;

////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////BG///////////////////////////////////////////////

void bg()

{

cleardevice();

setbkcolor(0);

}

////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////EGGS////////////////////////////////////////////////

void egg1 ()

{

setcolor(0);

circle (x + 85, y + 273, 9);

delay(200);

setcolor(7);

circle (x + 85, y + 273, 9);

setcolor(0);

circle (x + 105, y + 285, 9);

delay(200);

setcolor(7);

circle (x + 105, y + 285, 9);

setcolor(0);

circle (x + 125, y + 297, 9);

delay (200);

setcolor(7);

circle (x + 125, y + 297, 9);

setcolor(0);

circle (x + 145, y + 309, 9);

delay (200);

setcolor(7);

circle (x + 145, y + 309, 9);

setcolor(0);

circle (x + 165, y + 321, 9);

delay (200);

setcolor(7);

circle (x + 165, y + 321, 9);

}

void egg2 ()

{

setcolor(0);

circle (x + 85, y + 447, 9);

delay (200);

setcolor(7);

circle (x + 85, y + 447, 9);

setcolor(0);

circle (x + 105, y + 459, 9);

delay (200);

setcolor(7);

circle (x + 105, y + 459, 9);

setcolor(0);

circle (x + 125, y + 471, 9);

delay (200);

setcolor(7);

circle (x + 125, y + 471, 9);

setcolor(0);

circle (x + 145, y + 483, 9);

delay (200);

setcolor(7);

circle (x + 145, y + 483, 9);

setcolor(0);

circle (x + 165, y + 495, 9);

delay (200);

setcolor(7);

circle (x + 165, y + 495, 9);

}

void egg3 ()

{

setcolor(0);

circle (x + 712, y + 273, 9);

delay (200);

setcolor(7);

circle (x + 712, y + 273, 9);

setcolor(0);

circle (x + 692, y + 285, 9);

delay (200);

setcolor(7);

circle (x + 692, y + 285, 9);

setcolor(0);

circle (x + 672, y + 297, 9);

delay (200);

setcolor(7);

circle (x + 672, y + 297, 9);

setcolor(0);

circle (x + 652, y + 309, 9);

delay (200);

setcolor(7);

circle (x + 652, y + 309, 9);

setcolor(0);

circle (x + 632, y + 321, 9);

delay (200);

setcolor(7);

circle (x + 632, y + 321, 9);

}

void egg4 ()

{

setcolor(0);

circle (x + 712, y + 450, 9);

delay (200);

setcolor(7);

circle (x + 712, y + 450, 9);

setcolor(0);

circle (x + 692, y + 462, 9);

delay (200);

setcolor(7);

circle (x + 692, y + 462, 9);

setcolor(0);

circle (x + 672, y + 474, 9);

delay (200);

setcolor(7);

circle (x + 672, y + 474, 9);

setcolor(0);

circle (x + 652, y + 486, 9);

delay (200);

setcolor(7);

circle (x + 652, y + 486, 9);

setcolor(0);

circle (x + 632, y + 498, 9);

delay (200);

setcolor(7);

circle (x + 632, y + 498, 9);

}

//////////////////////////////////////////////////////////////////////////////////

/////////////////////////////////Controls/////////////////////////////////////////

void controls ()

{

int e;

randomize();

while(1)

{

e=random(5);

{

if (e==4){egg4();}

if (e==3){egg3();}

if (e==2){egg2();}

if (e==1){egg1();}

}

int c=0;

if (kbhit())

{

c=getch();

if (c==13) exit(1);

switch (c)

{

case 'q':

{setcolor(0);

arc( x + 215, y + 350, 180, 360, 35);

line(x + 210, y + 310, x + 210, y + 350); //levaya1

line(x + 210, y + 311, x + 250, y + 350); //pravaya1

line(x + 215, y + 320, x + 215, y + 350); //levaya2

line(x + 215, y + 320, x + 245, y + 350); //pravaya2

line(x + 180, y + 350, x + 250, y + 350); //verh

setfillstyle(1,7);

bar (x + 175, y + 420, x + 270, y + 580);

bar (x + 530, y + 220, x + 615, y + 580);}

break;

case 'a':

{setcolor(0);

arc( x + 215, y + 520, 180, 360, 35);

line(x + 210, y + 480, x + 210, y + 520); //levaya1

line(x + 210, y + 481, x + 250, y + 520); //pravaya1

line(x + 215, y + 490, x + 215, y + 520); //levaya2

line(x + 215, y + 490, x + 245, y + 520); //pravaya2

line(x + 181, y + 520, x + 250, y + 520); //verh

setfillstyle(1,7);

bar (x + 175, y + 300, x + 250, y + 450);

bar (x + 530, y + 220, x + 615, y + 580);}

break;

case 'e':

{setcolor(0);

arc( x + 575, y + 350, 180, 360, 35);

line(x + 570, y + 310, x + 570, y + 350); //levaya1

line(x + 570, y + 311, x + 540, y + 350); //pravaya1

line(x + 565, y + 325, x + 545, y + 350);

line(x + 566, y + 325, x + 565, y + 350); //levaya2

line(x + 610, y + 350, x + 540, y + 350); //verh

setfillstyle(1,7);

bar (x + 615, y + 490, x + 535, y + 595);

bar (x + 175, y + 220, x + 255, y + 580);}

break;

case 'd':

{setcolor(0);

arc( x + 575, y + 530, 180, 360, 35);

line(x + 570, y + 490, x + 570, y + 530); //pr1

line(x + 570, y + 491, x + 540, y + 530); //lev1

line(x + 565, y + 505, x + 545, y + 530); //lev2

line(x + 565, y + 505, x + 565, y + 530); //pr2

line(x + 610, y + 530, x + 540, y + 530); //verh

setfillstyle(1,7);

bar (x + 530, y + 310, x + 610, y + 400);

bar (x + 175, y + 220, x + 255, y + 580);}

break;

}

if ((e==1)&&(c=='q'))

{

setcolor(0);

circle (x + 185, y + 333, 9);

delay (400);

setcolor(7);

circle (x + 185, y + 333, 9);

clrscr();

sc++;

cout<<"\n "<<sc<<" ";

}

if ((e==2)&&(c=='a'))

{

setcolor(0);

circle (x + 185, y + 507, 9);

delay (400);

setcolor(7);

circle (x + 185, y + 507, 9);

clrscr();

sc++;

setcolor(7);

cout<<"\n "<<sc<<" ";

}

if ((e==3)&&(c=='e'))

{

setcolor(0);

circle (x + 612, y + 333, 9);

delay (400);

setcolor(7);

circle (x + 612, y + 333, 9);

clrscr();

sc++;

setcolor(7);

cout<<"\n "<<sc<<" ";

}

if ((e==4)&&(c=='d'))

{

setcolor(0);

circle (x + 612, y + 510, 9);

delay (400);

setcolor(7);

circle (x + 612, y + 510, 9);

clrscr();

sc++;

setcolor(7);

bar (x -1, y + 4, x + 730, y + 4);

cout<<"\n "<<sc<<" ";

}

}

}

}

//////////////////////////////////////////////////////////////////////////////////

/////////////////////////////////Display Bitmap/////////////////////////////////////

huge DetectSvga()

{

return 3;

}

void Show()

{

fstream File;

//Here you have to define the path of the bitmap file. Like according to this example i have to open one Board1.bmp file. So write you bitmap file path here.

File.open("C:\\bitmaps\\1-4\\1st.bmp",ios::in);

unsigned char Ch;

File.read((char\*)&HEADER,14); //This is the header part of the Bitmap. It always looks like same. Don't change the content hear. The value remains 14 here.

File.read((char\*)&INFOHEADER,40); //This is another part of the bitmap, here also the value remains same like 40 here.

unsigned int i;

char ColorBytes[4];

char\*PaletteData;

PaletteData=new char[256\*3];

if(PaletteData)//if memory allocated successfully

{

//read color data

for(i=0;i<256;i++)

{

//Don't change the code here because i have done some shifting here. Its working fine.

File.read(ColorBytes,4);

PaletteData[(int)(i\*3+2)]=ColorBytes[0]>>2;

PaletteData[(int)(i\*3+1)]=ColorBytes[1]>>2;

PaletteData[(int)(i\*3+0)]=ColorBytes[2]>>2;

}

outp(0x03c8,0); //tell DAC that data is coming

for(i=0;i<256\*3;i++) //send data to SVGA DAC

{

outp(0x03c9,PaletteData[i]);

}

delete[]PaletteData;

}

for(i=0;i<INFOHEADER.height;i++) //This for loop is used to display the bitmap.

{

for(int j=0;j<INFOHEADER.width;)

{

File.read(&Ch,1); // Here Ch reads the color of your bitmap.

putpixel(XCor+j++,YCor+INFOHEADER.height-i-1,Ch); //XCor and YCor are the X and Y cordinates. It depends upon you.

}

}

File.close();

controls();

}

//////////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////HELP/////////////////////////////////////////

int help (int x, int y)

{

bg();

setcolor (4);

settextstyle (1, 0, 6);

outtextxy (x + 30, y-70, "Controls:");

setcolor (15);

settextstyle (1, 0, 4);

outtextxy (x + 30, y + 30, "q - left upper tray ");

outtextxy (x + 30, y + 130, "a - left lower tray ");

outtextxy (x + 30, y + 230, "e - right upper tray ");

outtextxy (x + 30, y + 330, "d - right lower tray");

setcolor(4);

line (x + 400, y-100, x + 400, y + 601);

setcolor (4);

settextstyle (1, 0, 6);

outtextxy (x + 530, y-70, "Rules:");

setcolor (15);

settextstyle (1, 0, 2);

outtextxy (x + 410, y + 30, "Four chicken sitting on perches,lay");

outtextxy (x + 410, y + 50, "eggs,which rolls down for four trays.");

outtextxy (x + 410, y + 90, "Controlling the basket,which can");

outtextxy (x + 410, y + 110, "take four positions required to catch");

outtextxy (x + 410, y + 130, "as many eggs as possible.");

outtextxy (x + 410, y + 170, "When player caught one egg,");

outtextxy (x + 410, y + 190, "one point will be add to scoreboard.");

setcolor (4);

settextstyle (1, 0, 3);

outtextxy (x + 415, y + 270, "Press 'Enter' to play ");

setcolor (4);

settextstyle (1, 0, 3);

outtextxy (x + 415, y + 300, "Press 'Esc' to exit");

char a;

a = getch ();

if (a == 13) Show();

if (a == 27) exit(1);

return 0;}

//////////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////MENU/////////////////////////////////////////

void Menu (int x, int y)

{

rectangle (x + 30, y + 30, x + 160, y + 80);

}

void paint\_menu (int x, int y, int count)

{

setcolor(3);

for (int i = 0; i < count; i++)

{

Menu (x + i \* 100, y - 100);

}

}

void paint\_menu\_name\_game (int x, int y, char \*name\_game[])

{

setcolor (15);

settextstyle (10, 0, 8);

for (int i = 0; i < 1; i++)

{

outtextxy (x + i \* 100 + 45, y + 20, name\_game[i]);

}

}

void paint\_menu\_text (int x, int y, char \*PLAY[], char \*HELP[], char \*QUIT[])

{

setcolor (4);

settextstyle (10, 0, 4);

for (int i = 0; i < 1; i++)

{

outtextxy (x + i \* 150 + 330, y + 250, PLAY[i]);

outtextxy (x + i \* 160 + 334, y + 350, HELP[i]);

outtextxy (x + i \* 170 + 337, y + 450, QUIT[i]);

}

}

int move\_menu (int x, int y, int count)

{

int count\_menu = 0;

char c = 0;

count--;

setcolor(7);

{Menu(x + 295, y + 235);

setcolor(0);}

while (c != 13)

{

c = getch();

if (!c) c += getch();

switch (c)

{

case 80:

{

if (count\_menu == count) count\_menu = 0;

else count\_menu++;

setcolor (7);

y = count\_menu \* 100 + 10;

Menu (x + 295, y + 226);

if (count\_menu > 0)

{

setcolor (0);

Menu (x + 295, y + 126);

}

else

{

setcolor (0);

Menu (x + 295, y + 426);

}

break;

} // move down

}

}

if ((c==13)&&(count\_menu==0)) Show();

if ((c==13)&&(count\_menu==1)) help(x,y);

if ((c==13)&&(count\_menu==2)) exit(1);

return count\_menu;

}

//////////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////MAIN/////////////////////////////////////////

int main()

{

int x = 1;

int y = 1;

int count = 3;

char \*PLAY[] = {"PLAY\0"};

char \*HELP[] = {"HELP\0"};

char \*EXIT[] = {"QUIT\0"};

char \*name\_game[] = {"NU,POGODI!\0"};

int gd = DETECT, md, a;

installuserdriver("SVGA256",&DetectSvga);

initgraph(&gd,&md,"C:\\TC\\BGI"); //Path may be different in your computer.

int count\_menu;

paint\_menu (x, y, count);

paint\_menu\_name\_game (x, y, name\_game);

paint\_menu\_text (x, y, PLAY, HELP, EXIT);

count\_menu = move\_menu (x, y, count);

getch();

closegraph();

return 0;

}

//////////////////////////////////////////////////////////////////////////////////