**主要思想：**

众所周知，贪吃蛇游戏是一款经典的益智游戏，有PC和手机等多平台版本，既简单又耐玩。该游戏通过控制蛇头方向吃食物，从而使得蛇变得越来越长，蛇不能撞墙，也不能装到自己，否则游戏结束。玩过贪吃蛇的朋友都知道这个小游戏有一圈“墙”、还有食物以及蛇，还有吃了食物之后的分数。所以通过C语言实现贪吃蛇，可以分为以下几个模块来实现：

a)编程实现“墙”

　　b)实现随机食物的生成

　　c)蛇的构建

　　d)以上三部分都实现了之后，实现游戏运行的部分，包括蛇的移动、吃食物、判断是否撞墙或者撞到自己。

　　e)游戏结束时的相关操作

**一、代码核心的三个函数：**

menu(); setup(); draw();

menu用来设置菜单，也就是我们一运行看到的那个。setup用来设置参数，我们需要设置高度和宽度，还有分数，食物的位置。draw也就是画，也就是画整个画面。

还有一个枚举类型 这个结构体用来设置蛇的几个状态，我觉得这个也是面向对象编程的一个思想，把蛇的状态都封装成一个枚举类型。

typedef enum

{

STOP = 0,

LEFT,

RIGHT,

UP,

DOWN

}Direction;

1. **贪吃蛇长度判断**

/\*判断贪吃蛇的长度\*/

void logic()

这个函数，这个函数应该是整个贪吃蛇的精髓了，要理解代码怎么把蛇给连接起来。用了点巧妙的东西。

来看这里面的关键代码：

/\*把上一个位置记下\*/

int lastX = tailX[0];

int lastY = tailY[0];

int last2X, last2Y;

/\*重新获取当前的位置\*/

tailX[0]=x;

tailY[0]=y;

int i=0;

/\*遍历整条蛇的长度 把 0 的位置空出来，其余蛇的位置往后面的空间移动\*/

for(i=1; i<ntail;i++)

{

last2X = tailX[i];

last2Y = tailY[i];

tailX[i]=lastX;

tailY[i]=lastY;

lastX = last2X;

lastY = last2Y;

}

lastX lastY 用来存上一次的蛇头的位置。后面的 for 循环，通过tail 蛇的长度，把蛇上个状态给保存到数组tailX tailY里面。

1. **代码部分**
2. **初步源代码**

完整代码：

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <Windows.h>

#include <time.h>

#include <conio.h> /\*键盘输入获取\*/

bool gameOver;

bool stop = false;

bool hit = false;

/\*游戏的边框大小\*/

const int width = 50;

const int height = 20;

/\*蛇的坐标，食物的坐标还有分数\*/

int x,y,fruitX,fruitY,score;

/\*蛇每个点的坐标\*/

int tailX[200],tailY[200];

/\*蛇的默认长度\*/

int ntail=3;

typedef enum

{

STOP = 0,

LEFT,

RIGHT,

UP,

DOWN

}Direction;

Direction Dir;

/\*开始菜单\*/

void menu()

{

int a;

printf("------------------------------------------------------------------\n");

printf("| 贪吃蛇游戏 |\n");

printf("| 1) 新游戏 |\n");

printf("| 2) 开始边界 |\n");

printf("| 3) 退出游戏 |\n");

printf("------------------------------------------------------------------\n");

printf("---->请输入你的选择:");

scanf("%d", &a);

}

/\*初始化状态\*/

void setup()

{

gameOver = false;

/\*根据当前时间设置“随机数种子”\*/

srand(time(NULL));

Dir = STOP;

/\*贪吃蛇的位置,固定在中间\*/

x= width/2;

y= height/2;

/\*食物的位置，位置是随机的\*/

fruitX = rand()%width;

fruitY = rand()%height;

score = 0;

}

/\*绘制界面\*/

void draw()

{

if(stop == true)

{

return;

}

system("cls");/\*清除屏幕\*/

printf("分数:%d",score);

printf("\n");

/\*第一行\*/

int i;

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

{

printf("-");

}

printf("\n");

/\*画中间的画面\*/

int p;

for(p= 0 ;p<height;p++)/\*高度\*/

{

int q;

for(q= 0 ;q<width;q++)/\*宽度\*/

{

/\*第一行最后已给字符\*/

if(q==0 || q==width-1)

{

printf("|");

}

if(p == fruitY && q == fruitX)/\*食物的随机坐标\*/

{

printf("O");

}

else

{

int k=0;

bool print = false;

/\*贪吃蛇的长度 默认长度是 3\*/

for(k=0;k<ntail;k++)

{

if(tailX[k]==q && tailY[k]==p)

{

printf("\*");

print = true;

}

}

/\*如果这个位置打印了 \* 就不要打印空格了\*/

if(!print)

{

printf(" ");

}

}

}

printf("\n");

}

/\*最后一行\*/

int j;

for(j= 0 ;j<width+1;j++)

{

printf("-");

}

}

/\*按键输入控制\*/

void input()

{

if(\_kbhit())

{

/\*获取键盘的输入字符\*/

switch(\_getch())

{

case '4':

case 75:/\*左键\*/

Dir = LEFT;

hit= true;

break;

case '8':

case 72:/\*上键\*/

Dir = UP;

hit= true;

break;

case '6':

case 77:/\*右键\*/

Dir = RIGHT;

hit= true;

break;

case '2':

case 80:/\*向下键盘键 \*/

Dir = DOWN;

hit= true;

break;

case 'x':

case 27:/\*ESE\*/

gameOver = true;

break;

case 32:/\*空格 暂停键\*/

stop = !stop;

break;

}

}

else if(!hit && stop == false)/\*如果没有改变方向\*/

{

x++;

}

}

/\*判断贪吃蛇的长度\*/

void logic()

{

if(stop == true)

{

return;

}

/\*把上一个位置记下\*/

int lastX = tailX[0];

int lastY = tailY[0];

int last2X, last2Y;

/\*重新获取当前的位置\*/

tailX[0]=x;

tailY[0]=y;

int i=0;

/\*遍历整条蛇的长度 把 0 的位置空出来，其余蛇的位置往后面的空间移动\*/

for(i=1; i<ntail;i++)

{

last2X = tailX[i];

last2Y = tailY[i];

tailX[i]=lastX;

tailY[i]=lastY;

lastX = last2X;

lastY = last2Y;

}

/\*根据方向来改变x y 的值\*/

switch(Dir)

{

case UP:

y--;

break;

case DOWN:

y++;

break;

case LEFT:

x--;

break;

case RIGHT:

x++;

break;

}

if(x<0 || width<x || y<0 || height<y)

{

gameOver = true;

/\*清除屏幕\*/

system("cls");

printf("------------------------------------------------------------------\n");

printf("| |\n");

printf("| |\n");

printf("| 游戏结束 |\n");

printf("| |\n");

printf("| |\n");

printf("------------------------------------------------------------------\n");

}

if(x==fruitX && y==fruitY)

{

/\*吃了一个食物，蛇的长度增加1\*/

ntail++;

score+=10;

/\*更新下一个食物的位置\*/

fruitX = rand()%width;

fruitY = rand()%height;

}

}

int main()

{

#if 0

while(1)

{

printf("%d\n",\_getch());

}

#endif

menu();

setup();

draw();

/\*循环画贪吃蛇的界面\*/

while(!gameOver)

{

draw();

input();

logic();

Sleep(70);

}

return 0;

}

上面这段代码直接在Dev C++上面是可以运行的。

1. **初步代码试玩体验**

在运行过程中，会发现整个闪屏的现象（即：屏幕刷新）这个东西，如果只是用时间刷新就会闪屏。**见：附件1：初步代码运行效果.mp4**

1. **改正源代码**

#include <stdio.h>

#include <stdlib.h>

#include <Windows.h>//windows编程头文件

#include <time.h>

#include <conio.h>//控制台输入输出头文件

#ifndef \_\_cplusplus

typedef char bool;

#define false 0

#define true 1

#endif

//将光标移动到控制台的(x,y)坐标点处

void gotoxy(int x, int y)

{

COORD coord;

coord.X = x;

coord.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord);

}

#define SNAKESIZE 100//蛇的身体最大节数

#define MAPWIDTH 78//宽度

#define MAPHEIGHT 24//高度

//食物的坐标

struct {

int x;

int y;

}food;

//蛇的相关属性

struct {

int speed;//蛇移动的速度

int len;//蛇的长度

int x[SNAKESIZE];//组成蛇身的每一个小方块中x的坐标

int y[SNAKESIZE];//组成蛇身的每一个小方块中y的坐标

}snake;

//绘制游戏边框

void drawMap();

//随机生成食物

void createFood();

//按键操作

void keyDown();

//蛇的状态

bool snakeStatus();

//从控制台移动光标

void gotoxy(int x, int y);

int key = 72;//表示蛇移动的方向，72为按下“↑”所代表的数字

//用来判断蛇是否吃掉了食物，这一步很重要，涉及到是否会有蛇身移动的效果以及蛇身增长的效果

int changeFlag = 0;

int sorce = 0;//记录玩家的得分

int i;

void drawMap()

{

//打印上下边框

for (i = 0; i <= MAPWIDTH; i += 2)//i+=2是因为横向占用的是两个位置

{

//将光标移动依次到(i,0)处打印上边框

gotoxy(i, 0);

printf("■");

//将光标移动依次到(i,MAPHEIGHT)处打印下边框

gotoxy(i, MAPHEIGHT);

printf("■");

}

//打印左右边框

for (i = 1; i < MAPHEIGHT; i++)

{

//将光标移动依次到(0,i)处打印左边框

gotoxy(0, i);

printf("■");

//将光标移动依次到(MAPWIDTH, i)处打印左边框

gotoxy(MAPWIDTH, i);

printf("■");

}

//随机生成初试食物

while (1)

{

srand((unsigned int)time(NULL));

food.x = rand() % (MAPWIDTH - 4) + 2;

food.y = rand() % (MAPHEIGHT - 2) + 1;

//生成的食物横坐标的奇偶必须和初试时蛇头所在坐标的奇偶一致，因为一个字符占两个字节位置，若不一致

//会导致吃食物的时候只吃到一半

if (food.x % 2 == 0)

break;

}

//将光标移到食物的坐标处打印食物

gotoxy(food.x, food.y);

printf("\*");

//初始化蛇的属性

snake.len = 3;

snake.speed = 200;

//在屏幕中间生成蛇头

snake.x[0] = MAPWIDTH / 2 + 1;//x坐标为偶数

snake.y[0] = MAPHEIGHT / 2;

//打印蛇头

gotoxy(snake.x[0], snake.y[0]);

printf("■");

//生成初试的蛇身

for (i = 1; i < snake.len; i++)

{

//蛇身的打印，纵坐标不变，横坐标为上一节蛇身的坐标值+2

snake.x[i] = snake.x[i - 1] + 2;

snake.y[i] = snake.y[i - 1];

gotoxy(snake.x[i], snake.y[i]);

printf("■");

}

//打印完蛇身后将光标移到屏幕最上方，避免光标在蛇身处一直闪烁

gotoxy(MAPWIDTH - 2, 0);

return;

}

void keyDown()

{

int pre\_key = key;//记录前一个按键的方向

if (\_kbhit())//如果用户按下了键盘中的某个键

{

fflush(stdin);//清空缓冲区的字符

//getch()读取方向键的时候，会返回两次，第一次调用返回0或者224，第二次调用返回的才是实际值

key = \_getch();//第一次调用返回的不是实际值

key = \_getch();//第二次调用返回实际值

}

/\*

\*蛇移动时候先擦去蛇尾的一节

\*changeFlag为0表明此时没有吃到食物，因此每走一步就要擦除掉蛇尾，以此营造一个移动的效果

\*为1表明吃到了食物，就不需要擦除蛇尾，以此营造一个蛇身增长的效果

\*/

if (changeFlag == 0)

{

gotoxy(snake.x[snake.len - 1], snake.y[snake.len - 1]);

printf(" ");//在蛇尾处输出空格即擦去蛇尾

}

//将蛇的每一节依次向前移动一节（蛇头除外）

for (i = snake.len - 1; i > 0; i--)

{

snake.x[i] = snake.x[i - 1];

snake.y[i] = snake.y[i - 1];

}

//蛇当前移动的方向不能和前一次的方向相反，比如蛇往左走的时候不能直接按右键往右走

//如果当前移动方向和前一次方向相反的话，把当前移动的方向改为前一次的方向

if (pre\_key == 72 && key == 80)

key = 72;

if (pre\_key == 80 && key == 72)

key = 80;

if (pre\_key == 75 && key == 77)

key = 75;

if (pre\_key == 77 && key == 75)

key = 77;

/\*\*

\*控制台按键所代表的数字

\*“↑”：72

\*“↓”：80

\*“←”：75

\*“→”：77

\*/

//判断蛇头应该往哪个方向移动

switch (key)

{

case 75:

snake.x[0] -= 2;//往左

break;

case 77:

snake.x[0] += 2;//往右

break;

case 72:

snake.y[0]--;//往上

break;

case 80:

snake.y[0]++;//往下

break;

}

//打印出蛇头

gotoxy(snake.x[0], snake.y[0]);

printf("■");

gotoxy(MAPWIDTH - 2, 0);

//由于目前没有吃到食物，changFlag值为0

changeFlag = 0;

return;

}

void createFood()

{

if (snake.x[0] == food.x && snake.y[0] == food.y)//蛇头碰到食物

{

//蛇头碰到食物即为要吃掉这个食物了，因此需要再次生成一个食物

while (1)

{

int flag = 1;

srand((unsigned int)time(NULL));

food.x = rand() % (MAPWIDTH - 4) + 2;

food.y = rand() % (MAPHEIGHT - 2) + 1;

//随机生成的食物不能在蛇的身体上

for (i = 0; i < snake.len; i++)

{

if (snake.x[i] == food.x && snake.y[i] == food.y)

{

flag = 0;

break;

}

}

//随机生成的食物不能横坐标为奇数，也不能在蛇身，否则重新生成

if (flag && food.x % 2 == 0)

break;

}

//绘制食物

gotoxy(food.x, food.y);

printf("\*");

snake.len++;//吃到食物，蛇身长度加1

sorce += 10;//每个食物得10分

snake.speed -= 5;//随着吃的食物越来越多，速度会越来越快

changeFlag = 1;//很重要，因为吃到了食物，就不用再擦除蛇尾的那一节，以此来造成蛇身体增长的效果

}

return;

}

bool snakeStatus()

{

//蛇头碰到上下边界，游戏结束

if (snake.y[0] == 0 || snake.y[0] == MAPHEIGHT)

return false;

//蛇头碰到左右边界，游戏结束

if (snake.x[0] == 0 || snake.x[0] == MAPWIDTH)

return false;

//蛇头碰到蛇身，游戏结束

for (i = 1; i < snake.len; i++)

{

if (snake.x[i] == snake.x[0] && snake.y[i] == snake.y[0])

return false;

}

return true;

}

int main()

{

drawMap();

while (1)

{

keyDown();

if (!snakeStatus())

break;

createFood();

Sleep(snake.speed);

}

gotoxy(MAPWIDTH / 2, MAPHEIGHT / 2);

printf("Game Over!\n");

gotoxy(MAPWIDTH / 2, MAPHEIGHT / 2 + 1);

printf("本次游戏得分为：%d\n", sorce);

Sleep(5000);

return 0;

}

1. **改正代码试玩体验**

效果比初步代码好得多，闪屏消失。**见：附件2：改正代码运行效果.mp4**