# 基于C#的贪吃蛇游戏

## 一、课程题目: 基于C#的贪吃蛇游戏

设计所开发基于C#的贪吃蛇小游戏，主要是利用Windows系统环境下Visual Stadio开发环境中各种空间的熟悉使用，主要通过分析游戏开始条件、暂停、游戏执行条件、游戏结束条件和游戏中一些变量的控制使用贪吃蛇小游戏的功能。

## 二、系统分析、设计与实现

**1.开发环境：**

**Visual Studio 2010**

**2.需求分析：**

### 1.功能剖析

1）游戏的目标。在不被撞死的前提下，吃掉方块增加自己的长度，来完成升级。

2）游戏中的物体。蛇，墙壁，方块。

3）动作。蛇移动，蛇吃奖子，蛇增加长度。

附：游戏中可以通过改变时间间隔，从而实现蛇的移动速度的改变

### 2.数据结构与算法分析

1）数据结构。简单起见，所有物体都用方块拼接。

        则蛇可以使用一个一维数组描述，数组的每个单元描述了蛇块的状态，如方向；可使用一个二维数组来描述地面情况，不可通过的地方为墙壁设置为1，可通过的地方设置为0；奖子，就是一个特殊的方块。

2）算法。

        注意观察蛇的特点。将蛇肢解为方块，则会发现每个方块的移动都依赖于它前面方块上一次的移动状态（第一个方块由玩家控制）。

蛇每次移动的时候，都只是头和尾发生了变化，那么，每次更新蛇的时候，我们只需要将尾部的蛇块移到头部相应的位置，不就更简单吗？ 算法的复杂度立即从O(n)变到了O(1)！而且我们还会发现，我们只需要记录一个方向就可以了，则空间复杂度也因此降低了。

3）地图。描述了地面信息。

        我们的贪吃蛇游戏地图信息很简单，总共有3类物体会站到地面上：墙壁，蛇，奖子。在每次更新的时候，我们将3类物体的信息按类别填充到地图中。如，墙壁的位置填1，奖子的位置填2，蛇的位置填3（每个蛇块都填），没有东西的地方填0。然后，将这个填满0,1,2,3的二维数组，交给渲染系统。

   到此，我们的幕后操作就算基本完成，剩下的就是些细节，等到编码的时候在详细处理。

### 3、项目逻辑分析

本游戏使用Bean、Snack、Block、Floor四个大类来分别存储游戏的四大模块

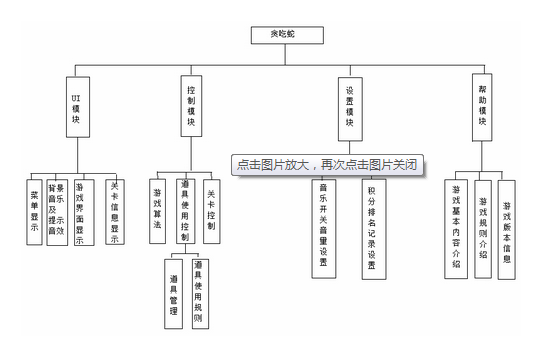
Bean: 豆类 指随机出现的豆豆

Floor: 空间类 指蛇运动的场所类

Block: 块类 蛇是由块组成

Snack: 蛇类 蛇的判断、升级条件

### 4.软件功能架构图

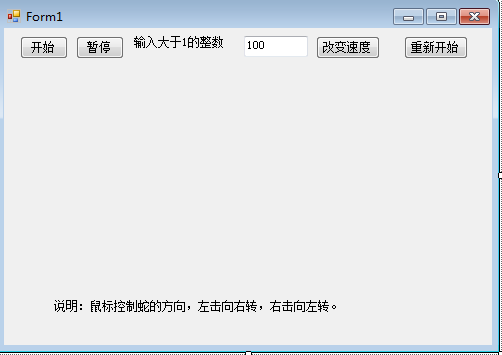


## 界面布局设计

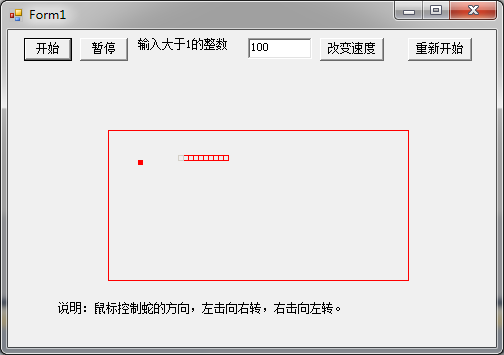


## 三、系统分析、设计与实现

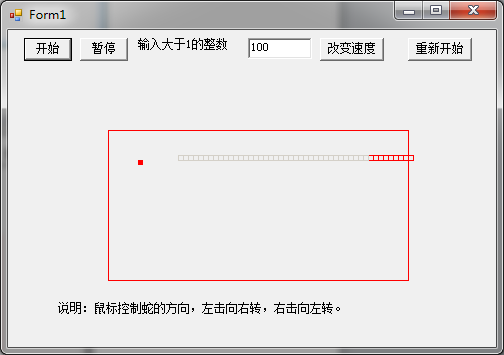
1.初始化游戏界面



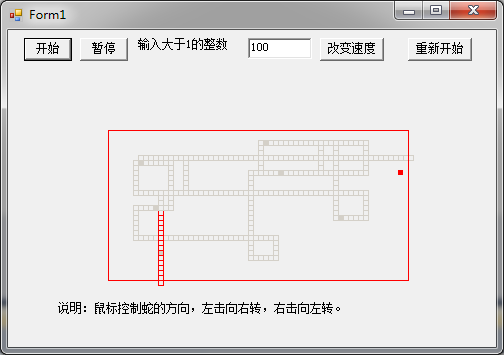
2.游戏开始运行情况（暂停）



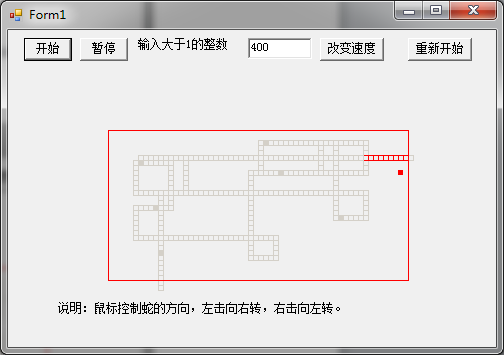
3.游戏死亡条件



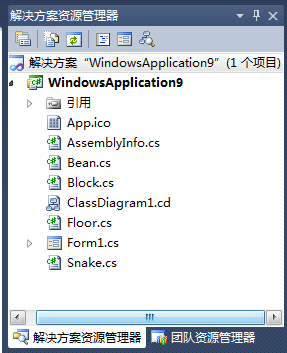
4.游戏升级条件



5.改变速度，重新开始游戏



6.实验资源解决方案



## 四、个人实验心得

本次贪吃蛇实验整整花了一个多星期的时间做完，从游戏的基本构想与网上的资源参考开始，根据已有的项目资源，先熟悉实验内容中要用到的一些控件、比如说Timer类和贪吃蛇中蛇的身体用的方块类，对于这些图形化界面的类首先进行构想，最基本知识熟悉与学习，能够学会简单实现类的方法。主要开发时间集中在星期六日（两个星期）游戏开发并没有太多林林总总的界面，本次实验内容只是基于单个窗口内容实现开发，是我第一次的游戏项目开发经验，游戏中逻辑结构相对一些简单的项目系统设计来说要不知道复杂多少倍，首先要从项目的整体进行宏观考虑，设定好开始条件，游戏升级条件，胜利条件、结束条件，不管是多么简单的一个游戏，图形化的开发必不可少，良好的界面与用户体验是每个游戏的关键核心，本次实验让我稍稍地接触了游戏开发的一些相关知识，主要通过阅读博客的方法参考收集资料，实验的周期分析主要是代码的逻辑实现，后期的实验报告只花了一小部分时间完成。试验中遇到的问题有很多，都是通过网上搜索和同学们各自熟悉的模块进行交流分享，问题得以解决，本次实验内容相对前面四次实验来说是个巨大的挑战，将这个学期所学到的C#的各个层次的知识基本用上，从面向过程设计到面向对象的设计，正如此次实验的主要使用了四个大类分别存储贪吃蛇游戏的四大模块，是对课本知识的良好总结，也是对理论知识的实践运用

附录：项目Snack类、Block类

using System;

using System.Drawing;

using System.Collections;

namespace WindowsApplication9

{

/// <summary>

/// Snake 的Ì?摘a要°a说¦Ì明¡Â。¡ê蛇¦?类¤¨¤。¡ê

/// </summary>

public delegate void SnakeDele();

public class Snake

{

public Snake()

{

//

// TODO: 在¨²此ä?处ä|添¬¨ª加¨®构1造¨¬函¡¥数ºy逻?辑-

//

}

//参?数ºyvertex是º?顶£¤点Ì?，ê?用®?于®¨²开a始º?时º¡À蛇¦?出?现?的Ì?位?置?，ê?count表À¨ª示º?开a始º?时º¡À蛇¦?是º?由®¨¦多¨¤少¦¨´块¨¦构1成¨¦的Ì?

public Snake(Point vertex, int count)

{

Block newB;

Point p = new Point(vertex.X + 25, vertex.Y + 25);

blockList = new ArrayList(count);

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

{

p.X = p.X + 5;

newB = new Block();

newB.Number = i + 1;

newB.Origin = p;

blockList.Add(newB);

//如¨?果?是º?蛇¦?头ª¡¤就¨ª把ã?顶£¤点Ì?赋3给?headPoint

if (i == count -1)

{

headPoint = newB.Origin;

}

}

headNumber = count;

}

//蛇¦?死¨¤亡ª?事º?件t

public event SnakeDele snakeDie;

public void SnakeDie()

{

if (snakeDie != null)

snakeDie();

}

//数ºy组Á¨¦列¢D表À¨ª，ê?用®?来¤¡ä保À¡ê存ä?块¨¦。¡ê

ArrayList blockList = new ArrayList();

//为a蛇¦?头ª¡¤的Ì?编À¨¤号?,也°2是º?蛇¦?的Ì?长¡è度¨¨

private int headNumber;

public int HeadNumber

{

get { return headNumber; }

set { headNumber = value; }

}

//蛇¦?头ª¡¤的Ì?位?置?，ê?用®?于®¨²判D断?蛇¦?是º?否¤?吃?了¢?豆1或¨°是º?撞Á2墙?了¢?

private Point headPoint;

public Point getHeadPoint

{

get { return headPoint; }

/\*get

{

IEnumerator myEnumerator = blockList.GetEnumerator();

try

{

while ( myEnumerator.MoveNext() )

{

Block b = (Block)myEnumerator.Current;

if (b.Number == headNumber)

{

return b.Origin;

}

}

}

catch(Exception e)

{

System.Console.WriteLine(e.ToString());

}

return new Point(0, 0);//理¤¨ª论?上¦?这a条¬?语®?句?是º?不?会¨¢执¡ä行D的Ì?。¡ê

}\*/

}

//蛇¦?是º?否¤?撞Á2到Ì?自Á?已°?身¦¨ª上¦?了¢?

public bool getHitSelf

{

get

{

IEnumerator myEnumerator = blockList.GetEnumerator();

try

{

while ( myEnumerator.MoveNext() )

{

Block b = (Block)myEnumerator.Current;

if (b.Number != headNumber && b.Origin.Equals(headPoint))

{

return true;

}

}

}

catch(Exception e)

{

System.Console.WriteLine(e.ToString());

}

return false;

}

}

//蛇¦?的Ì?得Ì?分¤?

private int score;

public int Score

{

get { return score; }

set { score = value; }

}

//当Ì¡À前¡ã运?动¡¥的Ì?方¤?向¨° 0为a向¨°上¦?，ê?1为a向¨°右®¨°，ê?2为a向¨°下?，ê?3为a向¨°左Á¨®

private int direction = 1;//默?认¨?是º?向¨°右®¨°

public int Direction

{

get { return direction; }

set { direction = value; }

}

//转Áa方¤?向¨°参?数ºypDirection有®D两¢?个?值¦Ì,0 表À¨ª示º?向¨°右®¨°转Áa（ê¡§即¡ä顺3时º¡À针?方¤?向¨°）ê?1表À¨ª示º?向¨°左Á¨®转Áa（ê¡§即¡ä逆?时º¡À针?方¤?向¨°）ê?

public void TurnDirection(int pDirection)

{

switch(direction)

{

case 0:

if (pDirection == 0)

direction = 1;

else if (pDirection == 1)

direction = 3;

break;

case 1:

if (pDirection == 0)

direction = 2;

else if (pDirection == 1)

direction = 0;

break;

case 2:

if (pDirection == 0)

direction = 3;

else if (pDirection == 1)

direction = 1;

break;

case 3:

if (pDirection == 0)

direction = 0;

else if (pDirection == 1)

direction = 2;

break;

}

}

//生¦¨²长¡è

public void Growth()

{

Block newB = new Block();

IEnumerator myEnumerator = blockList.GetEnumerator();

try

{

while ( myEnumerator.MoveNext() )

{

Block b = (Block)myEnumerator.Current;

if (b.Number == headNumber)

{

int x = b.Origin.X;

int y = b.Origin.Y;

switch(direction)

{

case 0:

y = y - 5;

break;

case 1:

x = x + 5;

break;

case 2:

y = y + 5;

break;

case 3:

x = x - 5;

break;

}

Point headP = new Point(x, y);

newB.Origin = headP;

newB.Number = b.Number + 1;

headNumber ++;

headPoint = headP;

blockList.Add(newB);

}

}

}

catch(Exception e)

{

System.Console.WriteLine(e.ToString());

}

}

//绘?制?自Á?身¦¨ª

public void Display(Graphics g)

{

try

{

Block newB = new Block();

IEnumerator myEnumerator = blockList.GetEnumerator();

while ( myEnumerator.MoveNext() )

{

Block b = (Block)myEnumerator.Current;

b.Number--;

if (b.Number < 1)

{

blockList.Remove(b);

b.UnDisplay(g);

continue;

}

if (b.Number == (headNumber - 1))

{

newB = new Block();

int x = b.Origin.X;

int y = b.Origin.Y;

switch(direction)

{

case 0:

y = y - 5;

break;

case 1:

x = x + 5;

break;

case 2:

y = y + 5;

break;

case 3:

x = x - 5;

break;

}

Point headP = new Point(x, y);

newB.Origin = headP;

newB.Number = headNumber;

newB.Display(g);

headPoint = newB.Origin;//重?新?指?定¡§蛇¦?头ª¡¤的Ì?点Ì?

}

b.Display(g);

}

blockList.Add(newB);

}

catch (Exception e)

{

System.Console.WriteLine(e.ToString());

}

}

//消?除y自Á?身¦¨ª

public void UnDisplay(Graphics g)

{

try

{

Block newB = new Block();

IEnumerator myEnumerator = blockList.GetEnumerator();

while ( myEnumerator.MoveNext() )

{

Block b = (Block)myEnumerator.Current;

b.UnDisplay(g);

}

}

catch(Exception e)

{

System.Console.WriteLine(e.ToString());

}

}

//重?设¦¨¨蛇¦?的Ì?位?置?，ê?以°?重?新?开a始º?游®?戏¡¤

public void Reset(Point vertex, int count)

{

Block newB;

Point p = new Point(vertex.X + 25, vertex.Y + 25);

blockList = new ArrayList(count);

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

{

p.X = p.X + 5;

newB = new Block();

newB.Number = i + 1;

newB.Origin = p;

blockList.Add(newB);

//如¨?果?是º?蛇¦?头ª¡¤就¨ª把ã?顶£¤点Ì?赋3给?headPoint

if (i == count -1)

{

headPoint = newB.Origin;

}

}

headNumber = count;

//方¤?向¨°

direction = 1;

}

}

}

using System;

using System.Drawing;

namespace WindowsApplication9

{

/// <summary>

/// Block 的Ì?摘a要°a说¦Ì明¡Â。¡ê块¨¦类¤¨¤，ê?蛇¦?是º?由®¨¦块¨¦组Á¨¦成¨¦的Ì?，ê?

/// </summary>

public class Block

{

public Block()

{

//

// TODO: 在¨²此ä?处ä|添¬¨ª加¨®构1造¨¬函¡¥数ºy逻?辑-

//

}

//编À¨¤号?

private int number;

public int Number

{

get { return number; }

set { number = value; }

}

//

private Point origin;

public Point Origin

{

get { return origin; }

set { origin = value; }

}

//绘?制?自Á?身¦¨ª

public void Display(Graphics g)

{

Pen p = new Pen(Color.Red);

g.DrawRectangle(p,origin.X,origin.Y,5,5);

}

//消?除y自Á?身¦¨ª

public void UnDisplay(Graphics g)

{

//使º1用®?背À3景¡ã色¦?

//Color.FromArgb(212, 208, 200)创ä¡ä建¡§了¢?背À3景¡ã色¦?,然¨?后¨®用®?这a种?颜?色¦?画-这a个?块¨¦就¨ª使º1得Ì?这a个?块¨¦消?失º¡ì了¢?。¡ê

Pen p = new Pen(Color.FromArgb(212, 208, 200));

g.DrawRectangle(p,origin.X, origin.Y,5,5);

}

}

}

using System;

using System.Drawing;

namespace WindowsApplication9

{

/// <summary>

/// Bean 的Ì?摘a要°a说¦Ì明¡Â。¡ê豆1类¤¨¤

/// </summary>

public class Bean

{

public Bean()

{

//

// TODO: 在¨²此ä?处ä|添¬¨ª加¨®构1造¨¬函¡¥数ºy逻?辑-

//

score = 5;//默?认¨?的Ì?分¤?数ºy值¦Ì为a5

}

//当Ì¡À蛇¦?吃?了¢?这a个?豆1后¨®能¨¹增?加¨®的Ì?分¤?数ºy

private int score;

public int Score

{

get { return score; }

set { score = value; }

}

//原-点Ì?

private Point origin;

public Point Origin

{

get { return origin; }

set { origin = value; }

}

//豆1的Ì?功|能¨¹,

public void Function(Snake s)

{

//蛇¦?长¡è度¨¨长¡è多¨¤一°?块¨¦

s.Growth();

s.Score += this.score;

}

//绘?制?自Á?身¦¨ª

public void Display(Graphics g)

{

SolidBrush b = new SolidBrush(Color.Red);

g.FillRectangle(b,origin.X,origin.Y,5,5);

}

//消?除y自Á?身¦¨ª

public void UnDisplay(Graphics g)

{

//使º1用®?背À3景¡ã色¦?

//Color.FromArgb(212, 208, 200)创ä¡ä建¡§了¢?背À3景¡ã色¦?,然¨?后¨®用®?这a种?颜?色¦?画-这a个?块¨¦就¨ª使º1得Ì?这a个?块¨¦消?失º¡ì了¢?。¡ê

SolidBrush b = new SolidBrush(Color.FromArgb(212, 208, 200));

g.FillRectangle(b,origin.X, origin.Y,5,5);

}

}

}