《软件开发综合实验》报告三

——代码编写报告

|  |  |
| --- | --- |
| **小组编号** | **第十三组** |
| **小组成员** | **张钦贤、李策、杜梅** |

**目录**

[一、 界面表示 3](#_Toc1564)

[程序界面构成 3](#_Toc31671)

[地图的绘制 4](#_Toc27343)

[物种的表示 4](#_Toc17535)

[二、 类的设计 5](#_Toc1279)

[三、 控制类（Biz）的实现 6](#_Toc2789)

[随机布种 6](#_Toc1016)

[核心函数NextGeneration 7](#_Toc5533)

[四、 规则类（Rules）的实现 8](#_Toc21121)

[五、 实验难点代码实现 10](#_Toc19727)

[圆形区域的绘制 10](#_Toc30119)

[程序的自动运行 11](#_Toc28490)

[六、本周小组分工 11](#_Toc13196)

1. 界面表示
2. 程序界面构成

程序界面由如下控件构成：

private System.Windows.Forms.Button btnStartStop;

private System.Windows.Forms.Button btnClear;

private System.Windows.Forms.Button btnRndSeed;

private System.Windows.Forms.GroupBox groupBox1;

private System.Windows.Forms.TrackBar trbSpeed;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Timer tmrTime;

private System.Windows.Forms.Button btnStep;

private System.Windows.Forms.ToolTip toolTip;

private System.Windows.Forms.SaveFileDialog dlgSave;

private System.Windows.Forms.OpenFileDialog dlgOpen;

private System.Windows.Forms.FolderBrowserDialog dlgDir;

private System.Windows.Forms.PictureBox picCanvas;

public ListViewNF listView1;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.TextBox txtDisXmax;

private System.Windows.Forms.TextBox txtDisXmin;

private System.Windows.Forms.Label lbDisX;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.TextBox txtDisYmax;

private System.Windows.Forms.TextBox txtDisYmin;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.Button btnPlaceSlice;

private System.Windows.Forms.Button btnPlaceRound;

private System.Windows.Forms.Button btnPlaceChunk;

private System.Windows.Forms.Button btnPlaceKuang;

private System.Windows.Forms.Button btnSaveState;

private System.Windows.Forms.Button btnChange;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.Button btnPlaceRand;

private System.Windows.Forms.Button btnChangePoint;

private System.Windows.Forms.CheckBox ckbSaveGif;

private System.Windows.Forms.Label label8;

private System.Windows.Forms.Label label7;

private System.Windows.Forms.Button btnSaveBigImage;

private System.Windows.Forms.TextBox txtChangeY;

private System.Windows.Forms.TextBox txtChangeX;

private System.Windows.Forms.Label label6;

private System.Windows.Forms.Button btnPlaceRemove;

程序总体界面如图1.1所示：

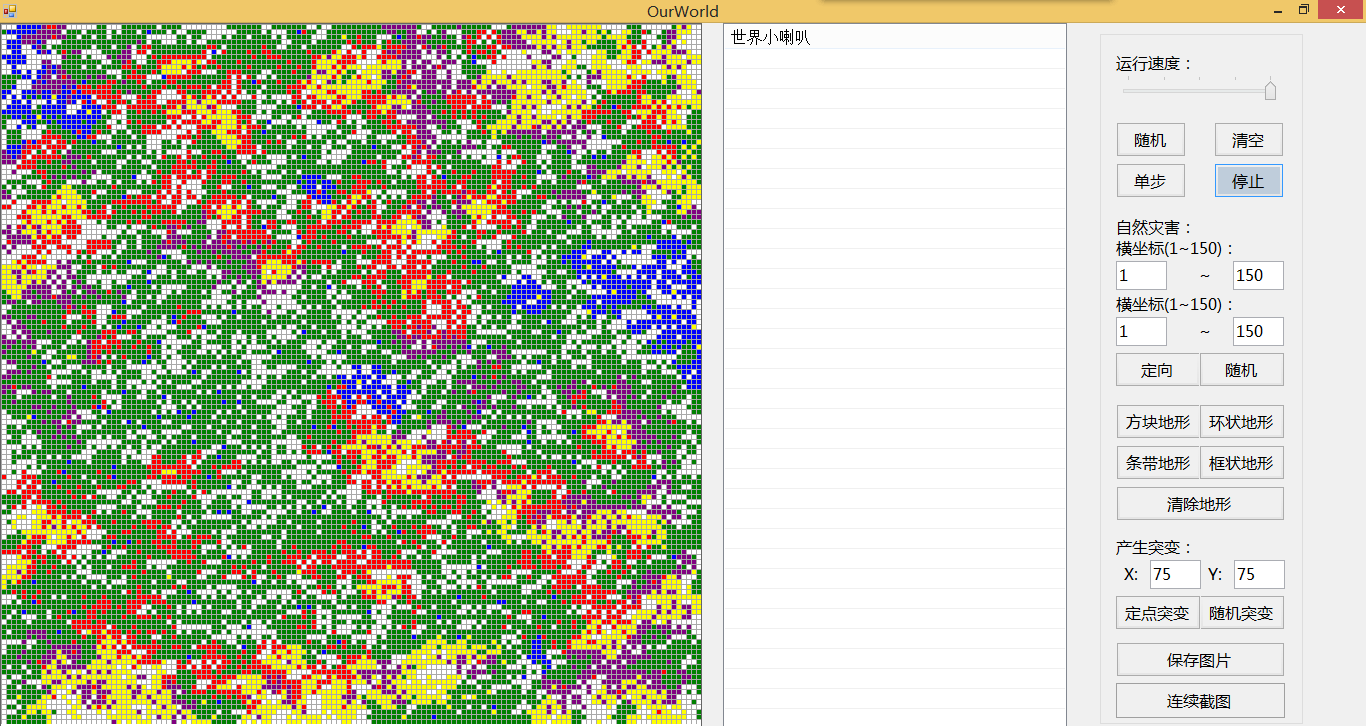


图1.1 程序总体界面

1. 地图的绘制

地图采用150\*150的方格表示，使用C#中Pen类的DrawLine方法绘制：

//获得画布大小，用白色填充

Bitmap img = new Bitmap(Constant.CANVAS\_WIDTH, Constant.CANVAS\_WIDTH);

SolidBrush whiteBrush = new SolidBrush(Color.White);

Graphics g = Graphics.FromImage(img);

g.FillRectangle(whiteBrush, 0, 0, img.Width, img.Height);

//初始化绘制直线的画笔，利用DrawLine方法绘制表格

Pen p = new Pen(Color.DarkGray);

for (int i = 0; i < Constant.GRID\_COUNT; i++)

{

g.DrawLine(p, 0, i \* Constant.GRID\_WIDTH, Constant.CANVAS\_WIDTH, i \* Constant.GRID\_WIDTH);

g.DrawLine(p, i \* Constant.GRID\_WIDTH, 0, i \* Constant.GRID\_WIDTH, Constant.CANVAS\_WIDTH);

}

1. 物种的表示

使用不同的颜色来表示物种，对应规则为：生产者-->绿色，食草动物-->蓝色低级食肉动物-->黄色，高级食肉动物-->红色，人类-->紫色，突变种-->黑色。使用C#中Graphics类的FillRectangle方法来绘制：

//根据二维数组中的颜色种类绘制不同的颜色

switch (Biz.World[i, j].colorKind)

{

//白色画笔

case Constant.NOLIFE: g.FillRectangle(whiteBrush, x, y, Constant.LIFE\_WIDTH, Constant.LIFE\_WIDTH); break;

//绿色画笔

case Constant.GRASS: g.FillRectangle(greenBrush, x, y, Constant.LIFE\_WIDTH, Constant.LIFE\_WIDTH); break;

//黑色画笔

case Constant.GRASS\_ANIMAL: g.FillRectangle(blueBrush, x, y, Constant.LIFE\_WIDTH, Constant.LIFE\_WIDTH); break;

//黄色画笔

case Constant.LOW\_ANIMAL: g.FillRectangle(yellowBrush, x, y, Constant.LIFE\_WIDTH, Constant.LIFE\_WIDTH); break;

//红色画笔

case Constant.HIGH\_ANIMAL: g.FillRectangle(redBrush, x, y, Constant.LIFE\_WIDTH, Constant.LIFE\_WIDTH); break;

//黑色画笔

case Constant.PEOPLE\_ANIMAL: g.FillRectangle(blackBrush, x, y, Constant.LIFE\_WIDTH, Constant.LIFE\_WIDTH); break;

//紫色画笔

case Constant.BOSS: g.FillRectangle(purpleBrush, x, y, Constant.LIFE\_WIDTH, Constant.LIFE\_WIDTH); break;

}

1. 类的设计
2. 常量类(Constants)：封装上文规定的所有常量。
3. 窗体类(FormMain)：表示界面窗体，容纳地图及各类控件。
4. 网格类(Node)：表示世界中的每个方格，有三个属性：colorKind表示生物种类，lifeCount：表示生物存活了几代，placeMode表示生物所处地形。整个世界由150\*150个Node对象组成的对象数组构成。
5. 规则类(Rules)：封装了《需求分析》规定的全部12条规则。
6. 控制类(Biz)：封装了世界运行所需的各类方法，例如绘制地图、物种繁衍、设置地形等。

主要类的UML类图如图4.1所示：（虚线为依赖关系）：

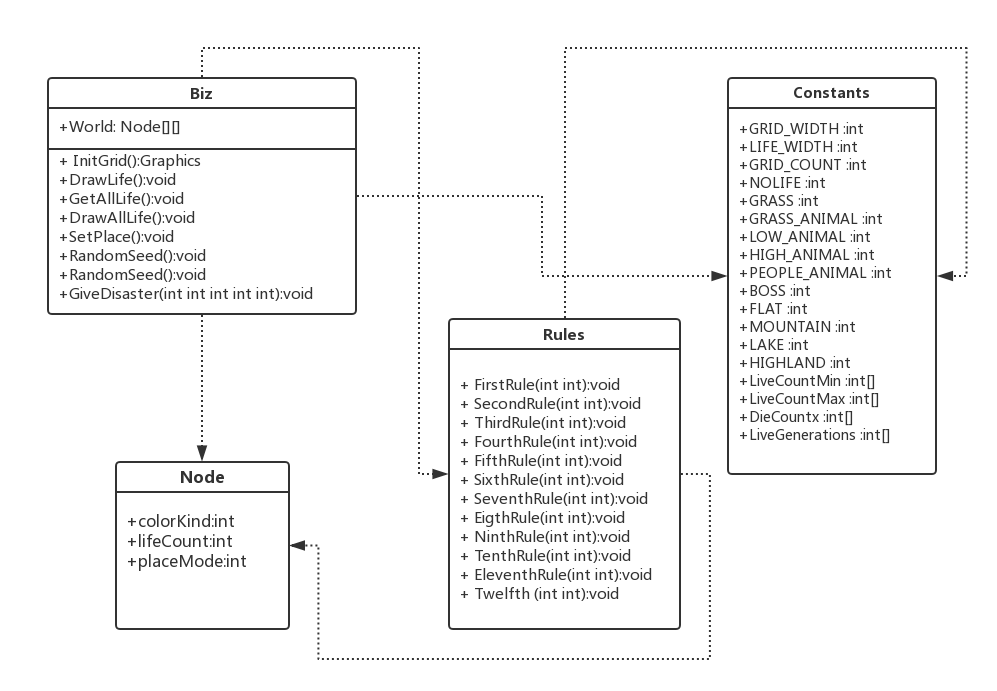


图 2.1UML类图

1. 控制类（Biz）的实现
2. 随机布种

实现原理：使用C#的Random类的NextBytes方法用随机数填充一个二维数组，随机数的范围是0~255，然后根据Constants类中规定的CREAT\_LIFE值来确定某个方格是有生命还是无生命。核心代码如下：

//填充随机数组

Random rnd = new Random();

byte[] values = new byte[Constant.GRID\_COUNT \* Constant.GRID\_COUNT];

rnd.NextBytes(values);

for (int i = 0; i < Constant.GRID\_COUNT; i++)

{

for (int j = 0; j < Constant.GRID\_COUNT; j++)

{

//有生命

if (values[i \* Constant.GRID\_COUNT + j] < Constant.CREAT\_LIFE)

{

Biz.World[i, j].colorKind = Constant.GRASS;

}

//无生命

else

{

Biz.World[i, j].colorKind = Constant.NOLIFE;

}

}

}

1. 核心函数NextGeneration

实现原理：遍历世界数组World，然后逐一执行《需求分析》中规定的1~12个判定条件，根据判定条件改变辅助数组HiddenWorld的值，等判定条件全部结束后，再把HiddenWorld中的值全部复制到World中，这样可以避免前面的判定条件改变World导致后面判定不准确。核心代码如下:

for (int i = 0; i < Constant.GRID\_COUNT; i++)

{

for (int j = 0; j < Constant.GRID\_COUNT; j++)

{

if (Biz.World[i, j].colorKind != Constant.BOSS)

{

//第一到第九判定条件

Rule.FirstJudeg(i, j);

Rule.SecondJudeg(i, j);

Rule.ThirdJudge(i, j);

Rule.FourthJudeg(i, j);

Rule.FirstJudeg(i, j);

Rule.SixthJudeg(i, j);

Rule.SenventhJudeg(i, j);

Rule.eigthJudeg(i, j);

Rule.ninthJudeg(i, j);

//如果有生命，则生命寿命加一

if (Biz.World[i, j].colorKind != Constant.NOLIFE)

{

Biz.HiddenWorld[i, j].lifeCount++;

}

}

//突变种执行单独的判定条件

else

{

Rule.bossJudeg(i, j);

}

}

}

//如果选择了地形因素，则执行地形判定条件

if(FormMain.form.isPlace)

Rule.tenthJudeg();

//如果选择了自然灾害因素，则执行自然灾害判定条件

if(FormMain.form.isDisaster)

Rule.eleventhJudeg();

//判定条件全部执行完毕，将辅助数组中的内容全部复制到世界数组World中

Node[,] tmp = Biz.HiddenWorld;

Biz.HiddenWorld = Biz.World;

Biz.World = tmp;

//重新绘制地图

Biz.DrawAllLife();

}

1. 规则类（Rules）的实现

规则类（Rules）封装了《需求分析》中规定的全部12条规则，其实现原理是给每个函数传入两个参数i和j，代表世界二维数组中的一个Node对象，然后根据此对象的colorKind、placeMode、lifeCount等属性来执行判定规则，改变辅助数组HiddenWorld中的值。12条判定规则原理相近，在这里展示第一到第三判定规则：

/// <summary>

/// 第一判定条件：生存规则

/// </summary>

/// <param name="i"></param>

/// <param name="j"></param>

public static void FirstJudeg(int i, int j)

{

//获取对象颜色属性

int colorKind = Biz.World[i, j].colorKind;

if (colorKind != Constant.NOLIFE && (colorKind != Constant.GRASS))

{

//GetLifeCount函数此处用于获取此对象周围8个对象中低级生物的数量

int lifeCount = GetLifeCount(i, j, colorKind, 1);

//如果周围8个对象中低级生物数量不足以位置它的生存，则该对象死亡

if (lifeCount < Constant.LiveCountMin[colorKind])

{

Biz.HiddenWorld[i, j].colorKind = Constant.NOLIFE;

Biz.HiddenWorld[i, j].lifeCount = 0;

}

}

}

/// <summary>

/// 第二判定条件，捕食规则

/// </summary>

/// <param name="i"></param>

/// <param name="j"></param>

public static void SecondJudeg(int i, int j)

{

//获取对象颜色属性

int colorKind = Biz.World[i, j].colorKind;

if (colorKind != Constant.NOLIFE)

{

//GetLifeCount函数此处用于获取此对象周围8个对象中高级生物的数量

int DieCount = GetLifeCount(i, j, colorKind, 2);

//如果周围8个对象中高级生物数量多于一定条件，则该对象被捕食

if (DieCount >= Constant.DieCountx[colorKind])

{

Biz.HiddenWorld[i, j].colorKind = Constant.NOLIFE;

Biz.HiddenWorld[i, j].lifeCount = 0;

}

}

}

/// <summary>

/// 第四判定条件，死亡规则

/// </summary>

/// <param name="i"></param>

/// <param name="j"></param>

public static void FourthJudeg(int i, int j)

{

//获取对象颜色属性

int colorKind = Biz.World[i, j].colorKind;

if (colorKind != Constant.NOLIFE)

{

//此对象的寿命已经超过LiveGenerations中规定的上限，则该对象死亡

if (Biz.World[i, j].lifeCount >= Constant.LiveGenerations[colorKind])

{

Biz.HiddenWorld[i, j].colorKind = Constant.NOLIFE;

//死亡之后寿命归0

Biz.HiddenWorld[i, j].lifeCount = 0;

}

}

}

1. 实验难点代码实现
2. 圆形区域的绘制

运行过程中需要绘制圆形区域，在遍历数组过程中，利用圆的方程便可以得到圆形数组区域。核心代码如下：

for (int i = 0; i < Constant.GRID\_COUNT; i++)

{

for (int j = 0; j < Constant.GRID\_COUNT; j++)

{

//得到的值，然后执行GetPlaceType函数，得到地形种类

Double num = (i - Constant.GRID\_COUNT / 2) \* (i - Constant.GRID\_COUNT / 2)

+ (j - Constant.GRID\_COUNT / 2) \* (j - Constant.GRID\_COUNT / 2);

Biz.World[i, j].placeMode = GetPlaceType(num);

}

}

/// <summary>

/// 得到地形种类

/// </summary>

/// <param name="num"></param>

/// <returns></returns>

private static int GetPlaceType(Double num)

{

//最下半径

Double Radius = Constant.GRID\_COUNT / 8;

//大于4倍半径，高原

if (num > 4 \* Radius \* 4 \* Radius)

{

return Constant.HIGHLAND;

}

//大于3倍半径，山区

if (num > 3 \* Radius \* 3 \* Radius)

{

return Constant.MOUNTAIN;

}

//大于2倍半径，湖泊

if (num > 2 \* Radius \* 2 \* Radius)

{

return Constant.LAKE;

}

//2倍半径以内，平原

else

{

return Constant.FLAT;

}

}

1. 程序的自动运行

为了实现世界的自动繁衍，使用C#的Timer控件，在Tick函数中调用NextGeneration函数，并设置好运行速度，世界便可以自动繁衍了。核心代码如下：

/// <summary>

/// 定时器Tick事件

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void tmrTime\_Tick(object sender, EventArgs e)

{

//重复执行NextGeneration函数

Biz.NextGeneration();

}

六、本周小组分工

张钦贤：常量类(Constants)、控制类(Biz)的代码编写，核心函数NextGeneration的代码编写，人员分工，进度安排。

李策：规则类(Rules)的代码编写，圆形区域的实现，运行逻辑的实现，文档编写。

杜梅：界面设计、地图设计、物种表示、UML类图绘制，定时器设置与实现。