# 生命游戏

## 一、游戏介绍

生命游戏其实是一个零玩家游戏，它包括一个二维矩形世界，这个世界中的每个方格居住着一个活着的或死了的细胞。一个细胞在下一个时刻生死取决于相邻八个方格中活着的或死了的细胞的数量。如果相邻方格活着的细胞数量过多，这个细胞会因为资源匮乏而在下一个时刻死去；相反，如果周围活细胞过少，这个细胞会因太孤单而死去。实际中，你可以设定周围活细胞的数目怎样时才适宜该细胞的生存。如果这个数目设定过高，世界中的大部分细胞会因为找不到太多的活的邻居而死去，直到整个世界都没有生命；如果这个数目设定过低，世界中又会被生命充满而没有什么变化。实际中，这个数目一般选取2或者3；这样整个生命世界才不至于太过荒凉或拥挤，而是一种动态的平衡。这样的话，游戏的规则就是：当一个方格周围有2或3个活细胞时，方格中的活细胞在下一个时刻继续存活；即使这个时刻方格中没有活细胞，在下一个时刻也会“诞生”活细胞。在这个游戏中，还可以设定一些更加复杂的规则，例如当前方格的状况不仅由父一代决定，而且还考虑祖父一代的情况。你还可以作为这个世界的上帝，随意设定某个方格细胞的死活，以观察对世界的影响。

在游戏的进行中，杂乱无序的细胞会逐渐演化出各种精致、有形的结构；这些结构往往有很好的对称性，而且每一代都在变化形状。一些形状已经锁定，不会逐代变化。有时，一些已经成形的结构会因为一些无序细胞的“入侵”而被破坏。但是形状和秩序经常能从杂乱中产生出来。

生命游戏的原理是细胞自动机（又称元胞自动机），名字虽然很深奥，但是它的行为却是非常美妙的。所有这些怎样实现的呢？我们可以把计算机中的宇宙想象成是一堆方格子构成的封闭空间，尺寸为N的空间就有N\*N个格子。而每一个格子都可以看成是一个生命体，每个生命都有**生**和**死**两种状态，如果该格子生就显示蓝色，死则显示白色。每一个格子旁边都有邻居格子存在，如果我们把3\*3的9个格子构成的正方形看成一个基本单位的话，那么这个正方形中心的格子的邻居就是它旁边的8个格子。

每个格子的生死遵循下面的原则：

1． 如果一个细胞周围有3个细胞为生（一个细胞周围共有8个细胞），则该细胞为生（即该细胞若原先为死，则转为生，若原先为生，则保持不变） 。

2． 如果一个细胞周围有2个细胞为生，则该细胞的生死状态保持不变；

3． 在其它情况下，该细胞为死（即该细胞若原先为生，则转为死，若原先为死，则保持不变）

设定图像中每个像素的初始状态后依据上述的游戏规则演绎生命的变化，由于初始状态和迭代次数不同，将会得到令人叹服的优美图案。

这样就把这些若干个格子（生命体）构成了一个复杂的动态世界。运用简单的3条作用规则构成的群体会涌现出很多意想不到的复杂性为，这就是复杂性科学的研究焦点。

细胞自动机有一个通用的形式化的模型，每个格子（或细胞）的状态可以在一个有限的状态集合S中取值，格子的邻居范围是一个半径r，也就是以这个格子为中心，在距离它r远的所有格子构成了这个格子的邻居集合，还要有一套演化规则，可以看成是一个与该格子当前状态以及邻居状态相关的一个函数，可以写成f:S\*S^((2r)^N-1)->S。这就是细胞自动机的一般数学模型。

## 二、结构与功能介绍

1.LifeInDiv类

LifeIndiv用于创建生命对象，每一个对象代表着一个微观世界（方格），对象储存了世界的生命信息和演化过程信息，提供了读取世界数据的一些方法，和清除世界内容的方法。

class LifeIndiv

{

private:

int LifeArray[22][22];

//int status[102][102];

int life\_num, dead\_num,life\_sum,dead\_sum;

//life\_num equals real-time living block

//dead\_num equals dead blocks in last evolve-process

//life\_sum equals the addition of living blocks in all turns

//dead\_sum equals the addtion of dead\_num in all turns

int step,increase[5000];

public:

LifeIndiv();

void createWorldRandom();

//int\* GetWorldInfo();

void evolve();

int getStatus(int i, int j);

int getStatistics(int i);

void clearWorld();

void createWorldCrowd();

int getStep();

void create10Cell();

void create3Boats();

};

其中

createWorldRandom()方法用于创建普通模式的稀疏数组；

evolve()方法用于单次的系统演化，改变数组元素值；

getStatus(int i, int j)函数用于获得确定位置的生命状态；

clearWorld()方法用于清空现有世界，为创建新世界做准备；

createWorldCrowd()方法用于创建普通模式的富饶数组（初始生命较多）；

getStep()方法用于获取系统当前的演化步数；

create10Cell()方法用于展示中心直线的系统演化特例；

create3Boats()方法用于展示三船模型的系统演化特例；

2.LifeGameFrameMain类

LifeGameFrameMain类用于实现交互行为，程序窗体中含有两个Panel用于放置所有的窗体和控件。主要使用的控件是wxStaticText，用于显示面板数据，并拒绝用户修改面板数据。提供的数据有，当前存活的生命数Life RT（Real-Time）,上轮死亡的生命数Dead LS（Last Turn）、世界诞生生命总数、世界死亡生命总数。

其中，

prepareStep()方法用于更新系统状态，强制刷新世界窗口；

setTexts()用于实时更新面板数据；

class LifeGameFrame: public wxFrame

{

public:

LifeGameFrame(wxWindow\* parent,wxWindowID id = -1);

virtual ~LifeGameFrame();

friend wxGrid\* GetGrid();

friend class LifeIndiv;

void prepareStep();

void setTexts();

private:

//(\*Handlers(LifeGameFrame)

void OnQuit(wxCommandEvent& event);

void OnAbout(wxCommandEvent& event);

void OnPanel2Paint(wxPaintEvent& event);

void OnStartNormalMode(wxCommandEvent& event);

void OnSwitchStepMode(wxCommandEvent& event);

void OnNextStep(wxCommandEvent& event);

void OnSystemPause(wxCommandEvent& event);

void OnSystemContinue(wxCommandEvent& event);

void OnButtonStartClick(wxCommandEvent& event);

void OnButtonPauseClick(wxCommandEvent& event);

void OnChoice1Select(wxCommandEvent& event);

void OnCreateCrowdWorld(wxCommandEvent& event);

void OnChoice2Select(wxCommandEvent& event);

//\*)

//(\*Identifiers(LifeGameFrame)

static const long ID\_GRID1;

static const long ID\_STATICTEXT2;

static const long ID\_STATICTEXT3;

static const long ID\_STATICTEXT5;

static const long ID\_STATICTEXT6;

static const long ID\_STATICTEXT7;

static const long ID\_STATICTEXT8;

static const long ID\_STATICTEXT9;

static const long ID\_STATICTEXT4;

static const long ID\_PANEL2;

static const long ID\_STATICTEXT10;

static const long ID\_CHOICE1;

static const long ID\_BUTTON1;

static const long ID\_BUTTON2;

static const long ID\_STATICTEXT1;

static const long ID\_STATICTEXT12;

static const long ID\_STATICTEXT13;

static const long ID\_STATICTEXT14;

static const long ID\_CHOICE2;

static const long ID\_STATICTEXT15;

static const long ID\_PANEL1;

static const long StartEvolveNormal;

static const long StartCrowdedWorld;

static const long idMenuQuit;

static const long StepButton;

static const long NextStep;

static const long PauseButton;

static const long ContinueButton;

static const long idMenuAbout;

static const long ID\_STATUSBAR1;

//\*)

//(\*Declarations(LifeGameFrame)

wxStaticText\* StaticText10;

wxStaticText\* StaticText9;

wxMenuItem\* MenuItem8;

wxMenuItem\* MenuItem7;

wxStaticText\* StaticTextDead;

wxStaticText\* StaticTextLife;

wxMenuItem\* MenuItem5;

wxStaticText\* StaticText2;

wxMenu\* Menu3;

wxStaticText\* StaticText6;

wxMenuItem\* MenuItem4;

wxStaticText\* StaticText8;

wxPanel\* Panel1;

wxStaticText\* StaticText3;

wxGrid\* Grid1;

wxMenuItem\* MenuItem3;

wxStaticText\* StaticText5;

wxStaticText\* StaticText7;

wxStaticText\* StaticTextSystemStatus;

wxStaticText\* StaticTextLifeSum;

wxStatusBar\* StatusBar1;

wxMenuItem\* MenuItem6;

wxStaticText\* StaticTextStep;

wxButton\* ButtonStart;

wxStaticText\* StaticTextDeadSum;

wxPanel\* Panel2;

wxStaticText\* StaticText4;

wxButton\* ButtonPause;

wxChoice\* Choice1;

wxChoice\* Choice2;

//\*)

wxTimer m\_timer;

void OnTimer(wxTimerEvent &event);

//void evolve(LifeIndiv &l);

//void evolveProceed(wxGrid \*grid);

DECLARE\_EVENT\_TABLE()

};

static const long ID\_BUTTON1;

static const long ID\_BUTTON2;

static const long ID\_STATICTEXT1;

static const long ID\_STATICTEXT12;

static const long ID\_STATICTEXT13;

static const long ID\_STATICTEXT14;

static const long ID\_CHOICE2;

static const long ID\_STATICTEXT15;

static const long ID\_PANEL1;

static const long StartEvolveNormal;

static const long StartCrowdedWorld;

static const long idMenuQuit;

static const long StepButton;

static const long NextStep;

static const long PauseButton;

static const long ContinueButton;

static const long idMenuAbout;

static const long ID\_STATUSBAR1;

//\*)

//(\*Declarations(LifeGameFrame)

wxStaticText\* StaticText10;

wxStaticText\* StaticText9;

wxMenuItem\* MenuItem8;

wxMenuItem\* MenuItem7;

wxStaticText\* StaticTextDead;

wxStaticText\* StaticTextLife;

wxMenuItem\* MenuItem5;

wxStaticText\* StaticText2;

wxMenu\* Menu3;

wxStaticText\* StaticText6;

wxMenuItem\* MenuItem4;

wxStaticText\* StaticText8;

wxPanel\* Panel1;

wxStaticText\* StaticText3;

wxGrid\* Grid1;

wxMenuItem\* MenuItem3;

wxStaticText\* StaticText5;

wxStaticText\* StaticText7;

wxStaticText\* StaticTextSystemStatus;

wxStaticText\* StaticTextLifeSum;

wxStatusBar\* StatusBar1;

wxMenuItem\* MenuItem6;

wxStaticText\* StaticTextStep;

wxButton\* ButtonStart;

wxStaticText\* StaticTextDeadSum;

wxPanel\* Panel2;

wxStaticText\* StaticText4;

wxButton\* ButtonPause;

wxChoice\* Choice1;

wxChoice\* Choice2;

//\*)

wxTimer m\_timer;

void OnTimer(wxTimerEvent &event);

//void evolve(LifeIndiv &l);

//void evolveProceed(wxGrid \*grid);

DECLARE\_EVENT\_TABLE()

};

wxButton\* ButtonStart;

wxStaticText\* StaticTextDeadSum;

wxPanel\* Panel2;

wxStaticText\* StaticText4;

wxButton\* ButtonPause;

wxChoice\* Choice1;

wxChoice\* Choice2;

//\*)

wxTimer m\_timer;

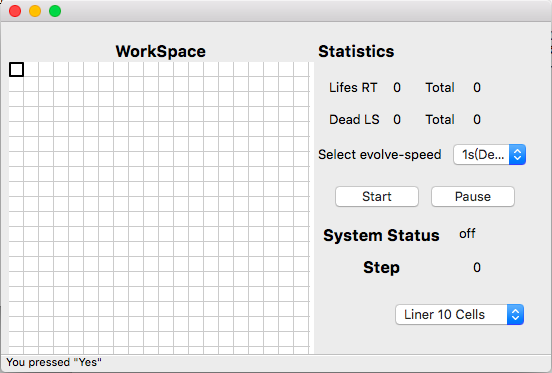
void OnTimer(wxTimerEvent &event);

//void evolve(LifeIndiv &l);

//void evolveProceed(wxGrid \*grid);

DECLARE\_EVENT\_TABLE()

};

 3.界面

界面中左侧的网格世界对应了一个wxGrid对象，用于显示网格世界演化过程，右侧的面板数据采集了网格世界的必要信息。使用了两个wxChoice，用户可选择调整时间流速，选择特定实例（收集的比较有特点的世界系统）。System Status显示了系统的运行状态，还使用了两个wxButton来随时控制系统状态。

在菜单栏中，还可将系统切换至Debug模式，步进来观察系统演化过程，能够更细致地观察到一类精细结构。

## 三、缺陷和不足之处

在程序设计之初想实现实时绘制折线图的形式来表现生命系统的过程，可通过前后的变化进一步地分析演化阶段。但是在wxWidgets中并未找到会用的图表模块，因此这一部分缺失。

不足之处是在设计时就没有找到足够的功能点，初衷是实现生命游戏，但对生命游戏的信息采集和统计还不到位，功能上存在先天不足。

本来还设计了系统信息导入导出模块，但是还没有完全掌握这部分内容和对信息格式的转化没有考虑到（即暂时无法实现由用户来操纵世界大小）。

## 四、代码原创性

程序所有代码，除了Timer定时器部分是参考了网上实例编写，其他部分均为原创，不存在抄袭借鉴情况，这一点可以从比较个性化的变量命名和程序结构看出。