JAPLA 研究会(蓼科夏の合宿)資料 2011/8/3

ライフゲームーその２
ーJ Grid OOP グラフィックスの例としてー

西川　利男

１．ライフゲームを、なぜGrid OOP グラフィックスで行うのか？

まず、この理由を箇条書きであげてみると次のようになるだろう。

・ライフゲームはグラフィックスで動かすのが自然である。
・タテ・ヨコの表示は、Grid のセルが最適である。
・一手ずつの動きは、ボタンのイベント・ドリブンのコントロールで実行する。
・J の Windows Grid グラフィックスは OOP プログラミングで行われる。

２．J の Grid OOP は易しい

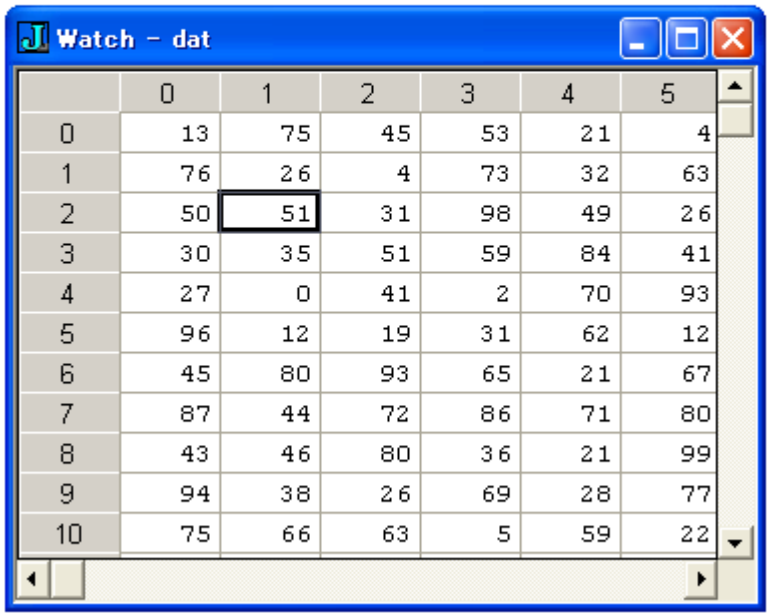
J の Grid とは Excel と同様のスプレッドシートである。これを Excel のように使うだけなら、しごく易しい。以下のように、打ち込むだけで、データが表示され、表計算などが出来るようになる。

Chris Burke による Grid OOP の紹介があるが[1]、最後は次のように結んである。

"OOP is easy in J!"

[1] Chris Burke ,"J4 and OOP", VECTOR, Vol.14, No.4, 103(1998).

```
dat =. ?30 30$100
load 'jwatch'
a =. conew 'jwatch'
create__a 'dat'
```



	0	1	2	3	4	5
0	13	75	45	53	21	4
1	76	26	4	73	32	63
2	50	51	31	98	49	26
3	30	35	51	59	84	41
4	27	0	41	2	70	93
5	96	12	19	31	62	12
6	45	80	93	65	21	67
7	87	44	72	86	71	80
8	43	46	80	36	21	99
9	94	38	26	69	28	77
10	75	66	63	5	59	22

ここでは、やや突っ込んだ使い方をするので、このあと J-OOP の基本を述べよう。

３．Ｊとオブジェクト指向プログラミング(OOP)

いまでは、大きなプログラムの開発は、分割し、モジュール化して行うのが常識である。オブジェクト指向プログラミング(Object Oriented Programming OOP)とは分割モジュール化の一つだが、その最新の手法である。

従来からの手法にサブルーチン・コールがある。ここでは頻繁に使う処理はサブルーチンとして作られ、これを呼ぶのはメイン・プログラムからであり、自分の作業場所は固定しているので単純である。

一方、オブジェクト指向ではこれとは違う考え方をする。Ｊのオブジェクト指向については以前、報告したことがあるので[2][3]、ここではあらすじだけを述べる。

[2] 西川利男、「Ｊのオブジェクト指向プログラミング (OOP) － その１
Ｊの OOP とは－簡単な例でやってみる」JAPLA シンポジウム資料　2005/12/10
[3] 西川利男、「Ｊのオブジェクト指向プログラミング (OOP) － その２
Ｊのスプレッドシート(Grid)と数独パズルへの適用」同上

まず、Ｊではモジュールの論理的な格納場所を示すのに、ロケール(locale)というとらえ方をするが、これに慣れなくてはならない。

Ｊを起動して、最初に自分のいる作業場所は base というロケールである。そして、かつてのサブルーチンに代わって、頻繁に使う処理などのモジュールはクラスとして作られる。

クラスを使用するときには、直接ではなく、一種の写しであるインスタンスを、次のコマンドにより生成し、それを用いて行う。

```
ins =: '' conew 'Class_Name'
```

これらを扱う次のコマンドがある。ここで最初の co は class object の意味である。

```
conl '' …… すべてのロケールを示す
conl 0  …… 名前付きロケールを示す
conl 1  …… 番号付きロケールを示す
coname '' …… 現在のロケール名を示す
```

Ｊの OOP のポイントは、base とインスタンスとの間で、値や関数(OOP ではメソッドと呼ぶ)の参照をどう行うかにかかっている。

すなわち、base の中からインスタンス ins の値を参照するには、次のようにする。

```
value__ins(アンダーバー２個)
```

一方、クラスの中で base の値を参照するには、次のようにする。

```
Value_base_（アンダーバー１個で区切り、さらにアンダーバー１個を付ける）
```

これら２つのＪの特別な記法の違いに注意。

４．ライフゲーム Grid グラフィックス版のプログラム構成

Ｊの OOP プログラムは起動プログラム(base)とクラスプログラムとから成る。

・起動プログラム(Life_Game.ijs)……前回のライフゲーム素朴版を元に拡張
　　ライフゲームの計算プログラム(lif, life など)
　　ライフゲームのデータ
　　ライフゲームの起動……オブジェクト・インスタンス作成
・クラス・プログラム(User¥plifegame.ijs)
　　グラフィックスのフォーム作成
　　ライフゲーム実行のコントロールのボタン、エディットボックス
　　ライフゲームのデータ作成、テスト実行など
　　プログラムは grid 機能の実現のために、jwatch を継承している。

５．起動プログラム

ライフゲームの初期パターンの配列データとライフゲームの計算処理プログラム、は前回のプログラムをそのまま用いる。

これに加えて、つぎのクラスファイルのロードおよびライフゲームの実行のためには、オブジェクト・インスタンス作成のプログラムが必要になる。

```
corequire 'user¥classes¥plifegame.ijs'

NB. Usage: ========================================
NB.  run LFBe
NB.  run Glider

LIFEV =: LValue    NB. running valuable

COL =: 16
ROW =: 20

run =: 3 : 0
if. 0 = #y. do. y.=. LFBe end.
LValue =: (COL, ROW, 1, 2) expand (″. y.)
w =: '' conew 'plifegame'
)
```

実行するには

run （ライフゲームの初期パターンデータ）

でおこなう。

Lvalue は配列データで、LIFEV は計算処理途中の値の格納で、ロケール base とインスタンスでグローバル共用する。Grid 画面のセルのタテ・ヨコの値は COL, ROW でグローバルである。

６．クラス・プログラム

クラス・プログラムの作成といっても、通常のプログラム・スクリプトの場合とそれほど変わることはない。実際には

[File]-[New Class]

とすると Wizard が働き、それに従って行えばよい。次のように最初の設定を行ってくれる。

クラス・プログラムの最初は次のコマンドから始める。他から参照されるべきクラスの名前を示す。ユーザ作成のクラスの名前は'p'で始める。

```
coclass 'plifegame'
```

必要な親クラスなどをロードする。

```
require 'jwatch'
```

ここには、grid で用いるいろいろな機能がすべて備えられている。

このとき、自動的に作られる

```
create
```

は、インスタンス実行の最初に実行される関数で、base からの値などの引渡しを行うことが出来る。

フォームの設計、作成は、[Edit]-[Form Editor]を開くと、やはりWizardにより、コードは自動的に作られる。また、ここでボタン、エディトボックスなどの作成を行い、コードの入力、編集ができる。

このあたりの実際のＪのスクリプトを見てみよう。

```
NB. grid class for life game
coclass 'plifegame'
require 'jwatch'
require 'gl2'
NB. フォームの定義([Edit]-[Form Editor]で自動的に作られる)
PGRID2=: 0 : 0
pc pgrid2;pn "Life Game";
xywh 6 6 209 144;cc grid isigraph;
xywh 224 140 34 11;cc Go button;
xywh 224 71 34 11;cc Set button;
xywh 224 123 34 11;cc fibo button;cn "Fibo";
xywh 225 46 34 11;cc life button;cn "Life-Go";
xywh 6 154 34 11;cc e1 edit ws_border es_autohscroll;
xywh 48 154 32 11;cc e2 edit ws_border es_autohscroll;
xywh 87 154 50 11;cc e3 edit ws_border es_autohscroll;
xywh 225 29 34 11;cc Load button;
xywh 225 89 34 11;cc Test button;cn "Test-Go";
xywh 225 7 33 11;cc e4 edit ws_border es_autohscroll;
pas 6 6;pcenter;ptop;
rem form end;
)
NB. 最初に実行される。フォームとGridの表示をおこなう。
create=: 3 : 0
wd PGRID2
formhwnd=: wd'qhwndp'
Col =: COL_base_
Row =: ROW_base_
I_Life =: 0
wd'pshow'
)
NB. クラスオブジェクトの終了
destroy=: 3 : 0
wd 'pclose'
codestroy''
)

pgrid2_close=:destroy

formselect=: 3 : 'wd''psel '',formhwnd'
```

ライフゲームのデータをロードするには、'Load' ボタンを押して行うが、そのコントロールの内容を示すコードは次のようになる。

```
NB. Push Load Button, then display values of base in grid
pgrid2_Load_button=: 3 : 0
glsel'grid'
glmapraw ''
glmap MM_TEXT
glnoerasebkgnd 1
glgrid ''
glshow''
'XX YY' =: 2{. ". wd'qchildxywhx grid'  NB. current position and size
(XX, YY) make_cell Col, Row
LIFEV_base_ =: LValue_base_             NB. loaded value from base
glgridtext ;(":each LIFEV_base_{' *'),each 0{a.  NB. display values
glpaintx ''
)
```

grid のセル表示を行ったのち、base よりライフ・ゲームのデータをクラス・オブジェクトのデータに取り込む。

ここで、サブルーチン make_cell は、配列からセル位置に貼り付けるものである。

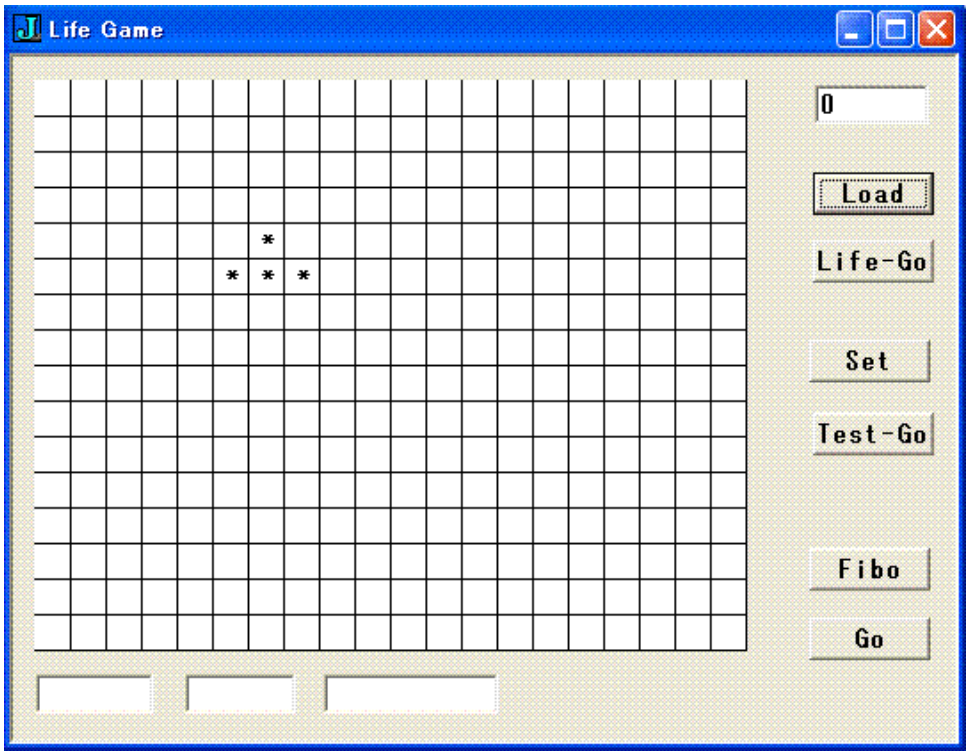
```
make_cell =: 3 : 0
:
'XX YY' =. x.
'R C' =. y.
wd 'setxywhx grid ',": XX, YY, (>:C*WX), (>:R*HX)
glgridrc R, C
glgridw C$WX
glgridh R$HX
)
```

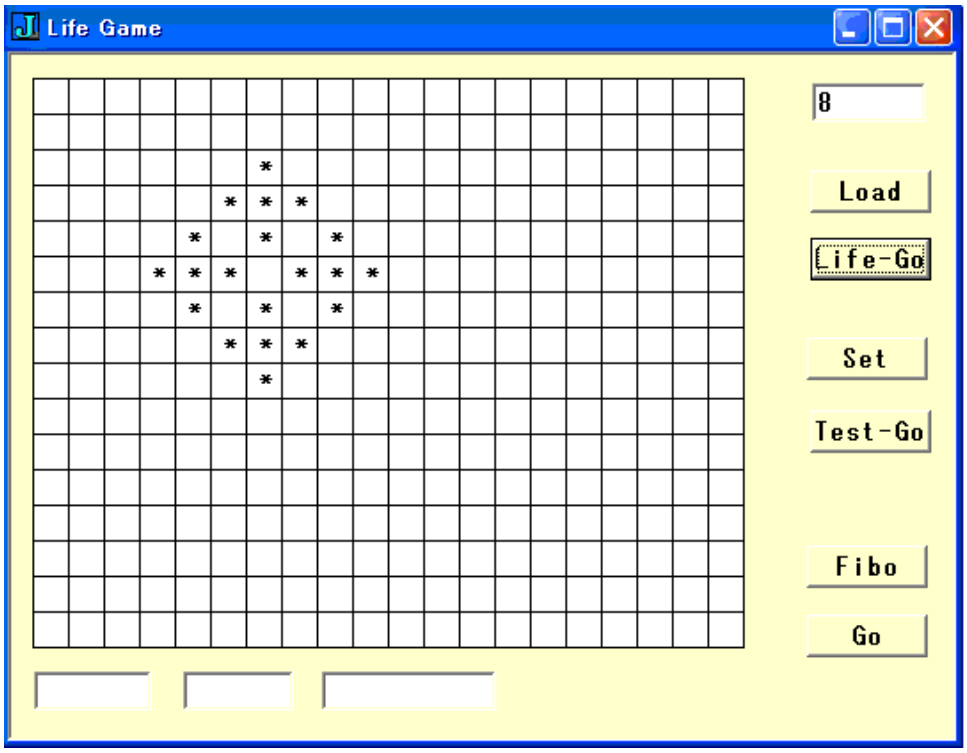
つぎに、ボタン'Life_go'を押すと、ライフゲームは１ステップずつ進行する。

```
pgrid2_life_button=: 3 : 0
LIFEV_base_ =: life_base_ LIFEV_base_
glgridtext ;(":each LIFEV_base_{' *'),each 0{a.  NB. display values
 I_Life =: I_Life + 1
 wd 'set e4 ', ": I_Life
glpaintx ''
)
```

ここでは、いずれも base にあるライフ・ゲームの配列データ LIFEV_base_を、ライフ・ゲームの操作関数 life_base_により１ステップ進め、配列データ LIFEV_base_を更新している。その配列データを文字列にして glgridtext により grid のセルに書き込んで表示するのである。

7．Grid ライフゲームの進行の実際





８．パターンのマニュアル入力によるライフゲーム

　マウスにより位置をきめつつ、ライフゲームのデータをマニュアルで設定したい。まず、マウス指定でセル位置とセルの値を得る左ボタンを作った。そのコントロールのコードを示す。入力したマウスのピクセル位置 sysdata から grid のセルの位置に変換する関数 sys2cel が必要である。

```
NB. mouse left button to display position and value ==========
pgrid2_grid_mbldown=: 3 : 0
 RC =. <:"(0) (1, 1) + sys2cel sysdata
NB. smoutput sysdata   NB. output object value on the session manager window
NB. smoutput sys2cel sysdata
 ('R';'C') =. RC
NB. glgridmark RC,1 1
NB. glgriddrawmark''
NB. cell position and data
 wd 'set e1 ', ": R
 wd 'set e2 ', ": C
 cdata =: glgridgettext R, C
NB. smoutput cdata  NB. write character data on ijx window
NB. smoutput val cdata  NB. write asc-val of data on ijx window
 if. 42 = val cdata
   do. wd 'set e3 "*"'
   else. wd 'set e3 ', cdata
 end.
glpaintx ''
)

NB. Subprogram called mbldown and mbrdown
sys2cel =: 3 : 0
 d=. (0{d),-/3 1{d=. ".y.
 gridhs =. Col$HX
 gridws =. Row$WX
 rc=. (+/(1{d)>:+/¥gridhs) , +/(0{d)>:+/¥gridws
 rc
)
```

マウスでセルを指定して、セルにデータを入力、編集するには思いがけない手法が必要になる。そのボタンコントロール(マウスの左ボタン)のコードを見てみよう。

```
editflag=: 0
pgrid2_grid_mbrdown=: 3 : 0
 if. editflag do. doedit'' end.
RC =: sys2cel sysdata
('R';'C') =. RC
NB. smoutput RC
glgridmark RC,1 1
```

```
glgriddrawmark''
d=. glgridgettext RC
glgridedit d
glgridedit 1
NB. glgridedit ''          NB. create edit box (= blank cell)
editflag=: 1
NB. glpaintx ''
)

doedit=: 3 : 0
d=. ". glgridgetedit''     NB. get contents of edit box
glgridrchw RC,1 1          NB. subarray for gridtext command
NB. write data from edit box(d) into cell / '*' for 1, ' ' for 0
select. d
 case. 1 do. glgridtext '*'
 case. 0 do. glgridtext ' '
end.
if. 1 = #d do. LIFDA =: d (< RC) } LIFDA end.  NB. write data to LIFDA array
glgridedit 0               NB. close edit box
editflag=:0
glpaintx ''
)
```

サブプログラム doedit では、マウスクリックのセル位置で、小さなエディットボックスが開き、それにキー入力がなされる、というしくみで、セルにデータが入れられる。
'1' でセルには'*'が、'0'で' 'が表示される。'1','0'の値はライフ・ゲームのパターン配列として以後使われる。

ボタン'Set'を押した後、上のようにマウスの左ボタンで、ライフ・ゲームのデータを作りセットする。

```
NB. Push Set Button, then display values of base in grid
pgrid2_Set_button=: 3 : 0
glsel'grid'
glmapraw ''
glmap MM_TEXT
glnoerasebkgnd 1
glgrid ''
glshow''
'XX YY' =: 2{. ". wd'qchildxywhx grid'  NB. current position and size
(XX, YY) make_cell Col, Row
LIFDA =: (Col, Row)$0, 0
glgridtext ;(":each LIFDA{' *'),each 0{a.
 I_Life =: 0
 wd 'set e4 ', ": I_Life
```

```
glpaintx ''
)
```

実行にはボタン'Test_go'により１ステップずつ、ライフ・ゲームを進めることは同様である。

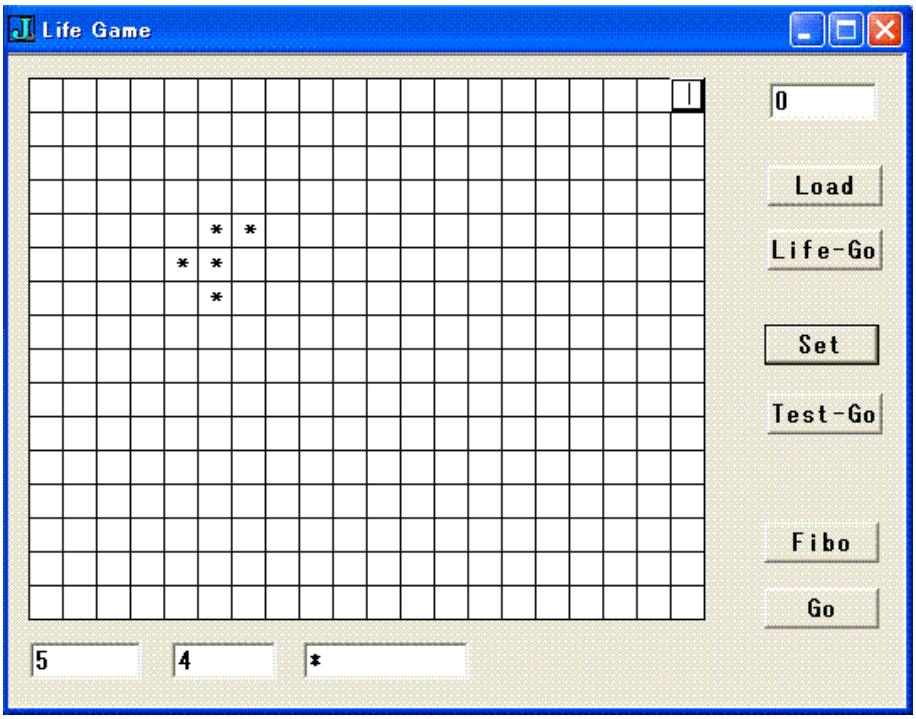
```
pgrid2_Test_button=: 3 : 0
LIFDA =: life_base_ LIFDA
'XX YY' =: 2{. ". wd'qchildxywhx grid'  NB. current position and size
(XX, YY) make_cell Col, Row
glgridtext ;(":each LIFDA{' *'),each 0{a.  NB. display values
 I_Life =: I_Life + 1
 wd 'set e4 ', ": I_Life
glpaintx ''
)
```

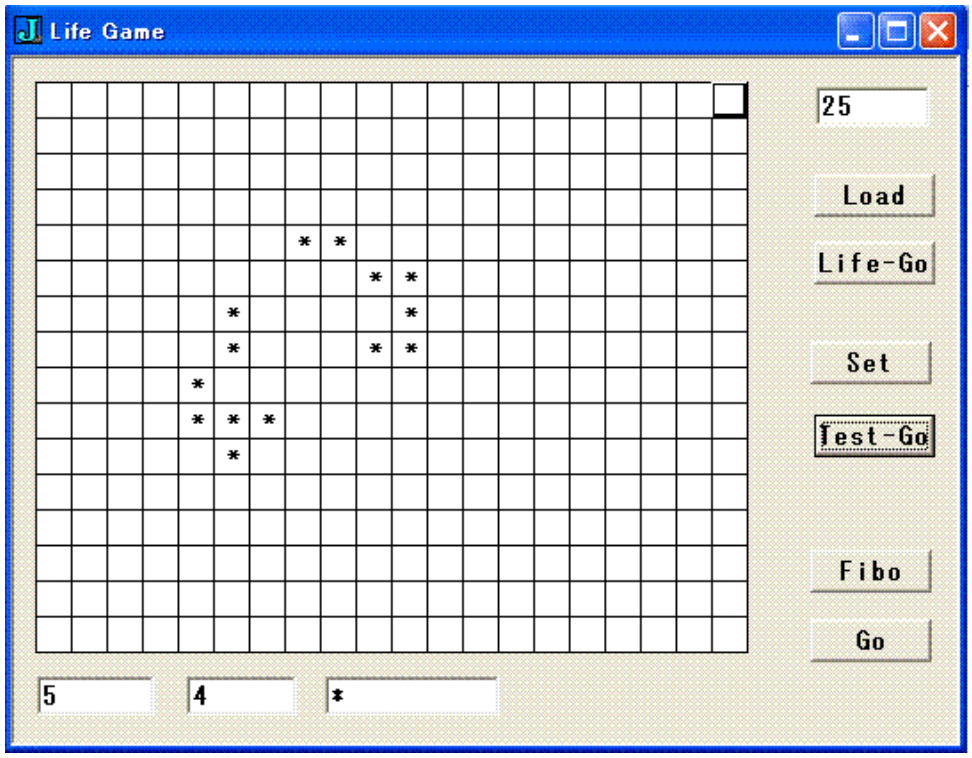
９．マニュアル設定によるライフゲームの進行の実際

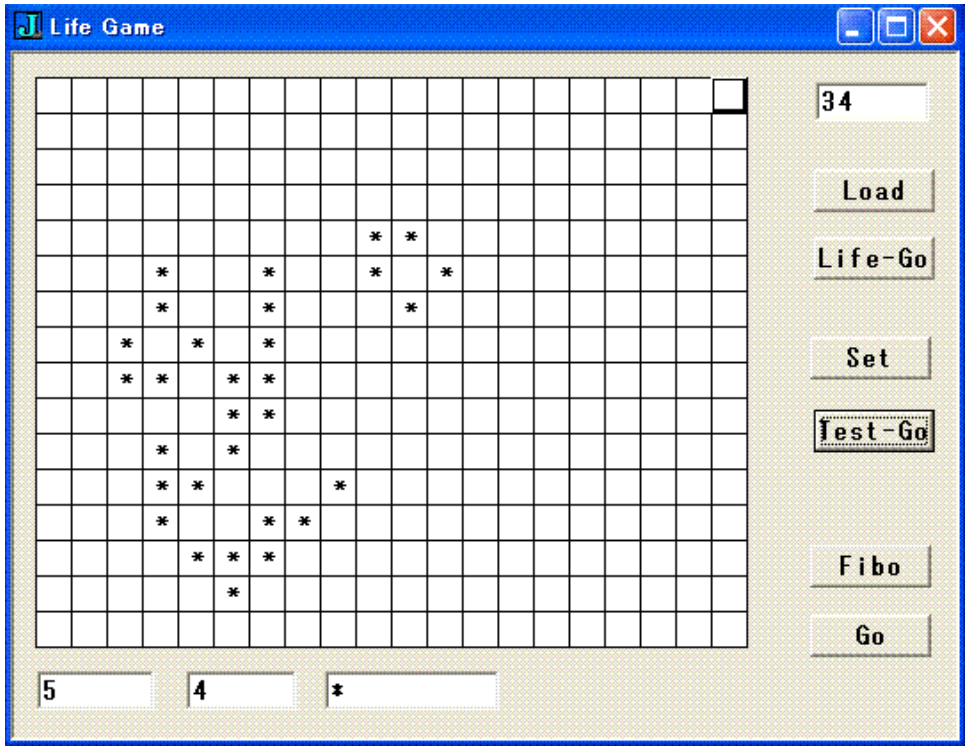
初期パターンとしてはごく簡単なのに、さまざまに変化するので有名なR-ペントミノと呼ばれるライフ・ゲームを追ってみよう。

ボタン'Set'を押した上で、マウスの右ボタンを押すとセルに入力が可能となり、'1'を入力すると、セルには'*'が表示され、'0'では空白になる。

また、マウスの左ボタンでは、セルの位置とその値が表示される。




Life Game
25
Load
Life-Go
Set
Test-Go
Fibo
Go
5
4


Life Game
34
Load
Life-Go
Set
Test-Go
Fibo
Go
5
4

ライフ・ゲーム　Jのソース・コードーbase プログラム・スクリプト

```
NB. Life Game
NB.  programmed by Toshio Nishikawa
NB.  2011/6/1

indx =: 3 : 0
Y =. <"(0) y.
Y (,L:0) /"(0 1) Y
)
NB.    indx <: i. 3
NB. +-----+----+----+
NB. |_1 _1|_1 0|_1 1|
NB. +-----+----+----+
NB. |0 _1 |0 0 |0 1 |
NB. +-----+----+----+
NB. |1 _1 |1 0 |1 1 |
NB. +-----+----+----+



env =: 3 : 0
:
BX =: indx <: i.3
ind =. (,(<y.) +L:0 BX) -. (<y.)
ind { x.
)

NB.     i. 4 4
NB.  0  1  2  3
NB.  4  5  6  7
NB.  8  9 10 11
NB. 12 13 14 15
NB. Usage:
NB.     (i. 4 4) env (1, 1)
NB. 0 1 2 4 6 8 9 10
NB.     (i. 4 4) env (1, 2)
NB. 1 2 3 5 7 9 10 11

lif =: 3 : 0
:
ST =. +/ x. env y.
select. ST
  case. 0, 1 do. 0
  case. 2    do. (<y.) { x.
  case. 3    do. 1
  fcase.     do. 0
end.
)
```

```
NB. number matrix data
DA0 =: 5 5$0 0 0 0 0, 0 1 0 0 0, 0 1 0 0 0, 0 0 1 0 0, 0 0 0 0 0

NB. DA1 is character matrix (5 x 9)
NB. if you want number matrix, then convert as ". DA1
DA1   =: ] ;._2 (0 : 0)
0 0 0 0 0
0 1 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 0 0
)

LFAa =: ] ;._2 (0 : 0)
0 0 0 0 0
0 1 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 0 0
)

LFAd =: ] ;._2 (0 : 0)
0 0 0 0 0
0 1 1 0 0
0 1 0 0 0
0 0 0 0 0
0 0 0 0 0
)


LFBb =: ] ;._2 (0 : 0)
0 0 0 0 0 0
0 0 0 0 0 0
0 1 1 1 1 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
)

LFBe =: ] ;._2 (0 : 0)
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 1 1 1 0 0 0
```

```
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
)

DAX =: ] ;._2 (0 : 0)
0 0 0 0 0
0 0 1 0 0
0 1 1 1 0
0 0 0 0 0
)

LF_glider =: ] ;._2 (0 : 0)
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0
0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
)

Glider =: ] ;._2 (0 : 0)
0 0 0 0 0
0 0 1 0 0
0 0 0 1 0
0 1 1 1 0
0 0 0 0 0
)

R_pentomino =: ] ;._2 (0 : 0)
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 1 1 0
0 0 1 1 0 0
0 0 0 1 0 0
0 0 0 0 0 0
)
```

```
life =: 3 : 0
DA =. y.
'M N' =. $ DA
DB =. }."(1) }. (<:N) {."(1) (<:M) {. M index N
DC =. > DA lif L:0 DB
(0,"(1) 0, DC, 0) ,"(1) 0
)

NB. Examples
NB.     (life^:(5) ". LFBe) {'_*'
NB.     (life^:(7) ". LFBe) {'_*'

expand =: 3 : 0
:
'm n a b' =. x.
MA =. y.
MB =. m {. 0 ,^:(a) MA
MC =. 0 ,"(1)^:(b) MB
n {."(1) MC
)

NB. Uasage:
NB.    (10 16 3 4) expand (". DA1)
NB. 8 8 8 8 9 9 9 9 9 0 0 0 0 0 0 0
NB. 8 8 8 8 9 9 9 9 9 0 0 0 0 0 0 0
NB. 8 8 8 8 9 9 9 9 9 0 0 0 0 0 0 0
NB. 8 8 8 8 0 0 0 0 0 0 0 0 0 0 0 0
NB. 8 8 8 8 0 1 0 0 0 0 0 0 0 0 0 0
NB. 8 8 8 8 0 1 0 0 0 0 0 0 0 0 0 0
NB. 8 8 8 8 0 0 1 0 0 0 0 0 0 0 0 0
NB. 8 8 8 8 0 0 0 0 0 0 0 0 0 0 0 0
NB. 8 8 8 8 0 0 0 0 0 0 0 0 0 0 0 0
NB. 8 8 8 8 0 0 0 0 0 0 0 0 0 0 0 0

NB.    (life^:(0) (11 16 1 2) expand (". LFBe))
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NB.   (life^:(8) (11 16 1 2) expand (". LFBe))
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
NB. 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0
NB. 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0
NB. 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0
NB. 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NB. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


NB.   (life^:(8) (11 16 1 2) expand (". LFBe)) {'_*'
NB. ________________
NB. ________________
NB. ______*_________
NB. _____***________
NB. ____*_*_*_______
NB. ___***_***______
NB. ____*_*_*_______
NB. _____***________
NB. ______*_________
NB. ________________
NB. ________________

NB. Life Game grid display ===================================
NB.   2011/7/8
NB.   2011/7/20
corequire 'user¥classes¥plifegame.ijs'

NB. Usage: ===========================================
NB.  run LFBe
NB.  run LF_glider
LIFEV =: LValue    NB. running valuable
COL =: 16
ROW =: 20

run =: 3 : 0
if. 0 = #y. do. y.=. LFBe end.
NB. LValue =: (11 16 1 2) expand (". y.)
```

```
LValue =: (COL, ROW, 1, 2) expand (". y.)
FValue =: 1
w =: '' conew 'plifegame'
)
```

ライフ・ゲーム　Jのソース・コードークラス・スクリプト

```
NB. grid class for life game
coclass 'plifegame'

NB. modified from 'pgrid2'

require 'jwatch'
require 'gl2'

PGRID2=: 0 : 0
pc pgrid2;pn "Life Game";
xywh 6 6 209 144;cc grid isigraph;
xywh 224 140 34 11;cc Go button;
xywh 224 71 34 11;cc Set button;
xywh 224 123 34 11;cc fibo button;cn "Fibo";
xywh 225 46 34 11;cc life button;cn "Life-Go";
xywh 6 154 34 11;cc e1 edit ws_border es_autohscroll;
xywh 48 154 32 11;cc e2 edit ws_border es_autohscroll;
xywh 87 154 50 11;cc e3 edit ws_border es_autohscroll;
xywh 225 29 34 11;cc Load button;
xywh 225 89 34 11;cc Test button;cn "Test-Go";
xywh 225 7 33 11;cc e4 edit ws_border es_autohscroll;
pas 6 6;pcenter;ptop;
rem form end;
)

create=: 3 : 0
wd PGRID2
formhwnd=: wd'qhwndp'
Col =: COL_base_
Row =: ROW_base_
I_Life =: 0
wd'pshow'
)

destroy=: 3 : 0
wd 'pclose'
codestroy''
)

pgrid2_close=:destroy

formselect=: 3 : 'wd''psel '',formhwnd'
```

```
WX =: 20  NB. Width  30 pixels
HX =: 20  NB. Height 20 pixels


NB. Col =: 11
NB. Row =: 16

NB. Push Load Button, then display values of base in grid
pgrid2_Load_button=: 3 : 0
glsel'grid'
glmapraw ''
glmap MM_TEXT
glnoerasebkgnd 1
glgrid ''
glshow''
'XX YY' =: 2{. ". wd'qchildxywhx grid'  NB. current position and size
(XX, YY) make_cell Col, Row
LIFEV_base_ =: LValue_base_             NB. loaded value from base
glgridtext ;(":each LIFEV_base_{' *'),each 0{a.  NB. display values
glpaintx ''
)



NB. make grid sub ================================================
make_cell =: 3 : 0
:
'XX YY' =. x.
'R C' =. y.
wd 'setxywhx grid ',": XX, YY, (>:C*WX), (>:R*HX)
glgridrc R, C
glgridw C$WX
glgridh R$HX
)

pgrid2_fibo_button=: 3 : 0
NB. display data and renew
Value_base_ =: fib_base_ Value_base_        NB. renew values
glgridtext ;(":each Value_base_),each 0{a.  NB. display values
glpaintx ''
)

pgrid2_life_button=: 3 : 0
LIFEV_base_ =: life_base_ LIFEV_base_
```

```
glgridtext ;(":each LIFEV_base_{' *'),each 0{a.  NB. display values
 I_Life =: I_Life + 1
 wd 'set e4 ', ": I_Life
glpaintx ''
)

pgrid2_ectrl_fkey=: 3 : 0
editenable__grid=: -.editenable__grid
)


pgrid2_grid_char=: 3 : 0
smoutput sysdata
char__grid sysdata
glpaintx ''
)



NB. Convert sysdata to cell_data ========================
NB. Subprogram called mbldown and mbrdown
sys2cel =: 3 : 0
 d=. (0{d),-/3 1{d=. ".y.
 gridhs =. Col$HX
 gridws =. Row$WX
 rc=. (+/(1{d)>:+/¥gridhs) , +/(0{d)>:+/¥gridws
 rc
)

val=: a. & i.
chr=: val ^:_1
NB. mouse left button to display position and value ==========
pgrid2_grid_mbldown=: 3 : 0
 RC =. <:"(0) (1, 1) + sys2cel sysdata
NB. smoutput sysdata
NB. smoutput sys2cel sysdata
 ('R';'C') =. RC
NB. glgridmark RC,1 1
NB. glgriddrawmark''
NB. cell position and data
 wd 'set e1 ', ": R
 wd 'set e2 ', ": C
 cdata =: glgridgettext R, C
NB. smoutput cdata  NB. write character data on ijx window
NB. smoutput val cdata  NB. write asc-val of data on ijx window
```

```
 if. 42 = val cdata
   do. wd 'set e3 "*"'
   else. wd 'set e3 ', cdata
 end.
glpaintx ''
)

NB. Set New Life Data on the grid cell ========================
NB. Select Cell-position by mouse right button down
NB.  and enter new value,
NB.  afterward to another cell, renewed new value
NB.  ============================
editflag=: 0
pgrid2_grid_mbrdown=: 3 : 0
 if. editflag do. doedit'' end.
RC =: sys2cel sysdata
('R';'C') =. RC
NB. smoutput RC
glgridmark RC,1 1
glgriddrawmark''
d=. glgridgettext RC
glgridedit d
glgridedit 1
NB. glgridedit ''          NB. create edit box (= blank cell)
editflag=: 1
NB. glpaintx ''
)

doedit=: 3 : 0
d=. ". glgridgetedit''    NB. get contents of edit box
glgridrchw RC,1 1         NB. subarray for gridtext command
NB. write data from edit box(d) into cell / '*' for 1, ' ' for 0
select. d
 case. 1 do. glgridtext '*'
 case. 0 do. glgridtext ' '
end.
if. 1 = #d do. LIFDA =: d (< RC) } LIFDA end.  NB. write data to LIFDA array
glgridedit 0              NB. close edit box
editflag=:0
glpaintx ''
)


NB. Push Set Button, then display values of base in grid
pgrid2_Set_button=: 3 : 0
```

```
glsel'grid'
glmapraw ''
glmap MM_TEXT
glnoerasebkgnd 1
glgrid ''
glshow''
'XX YY' =: 2{. ". wd'qchildxywhx grid'  NB. current position and size
(XX, YY) make_cell Col, Row
LIFDA =: (Col, Row)$0, 0
glgridtext ;(":each LIFDA{' *'),each 0{a.
 I_Life =: 0
 wd 'set e4 ', ": I_Life
glpaintx ''
)



pgrid2_Test_button=: 3 : 0
LIFDA =: life_base_ LIFDA
'XX YY' =: 2{. ". wd'qchildxywhx grid'  NB. current position and size
(XX, YY) make_cell Col, Row
glgridtext ;(":each LIFDA{' *'),each 0{a.  NB. display values
 I_Life =: I_Life + 1
 wd 'set e4 ', ": I_Life
glpaintx ''
)
```