

Life Game

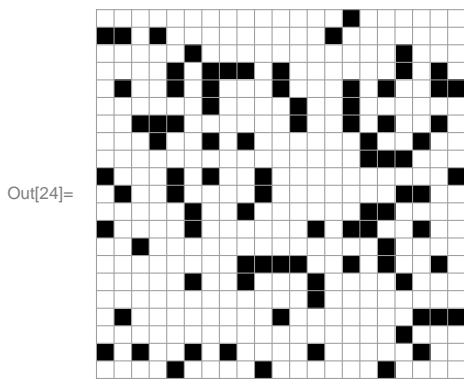
Matsuda@symbolics.jp

serial version

```
In[22]:= brdSize = 21;  
brd = Partition[Table[If[Random[] > 0.8, 1, 0], {brdSize * brdSize}], brdSize];  
(* random generated Life pattern in which 80% in average lives created *)
```

```
In[23]:= showBoard[brd_, opts : OptionsPattern[]] :=  
ArrayPlot[brd, FilterRules[{Mesh -> True, opts}, Options[ArrayPlot]]]
```

```
In[24]:= showBoard[brd, ImageSize -> 180]
```

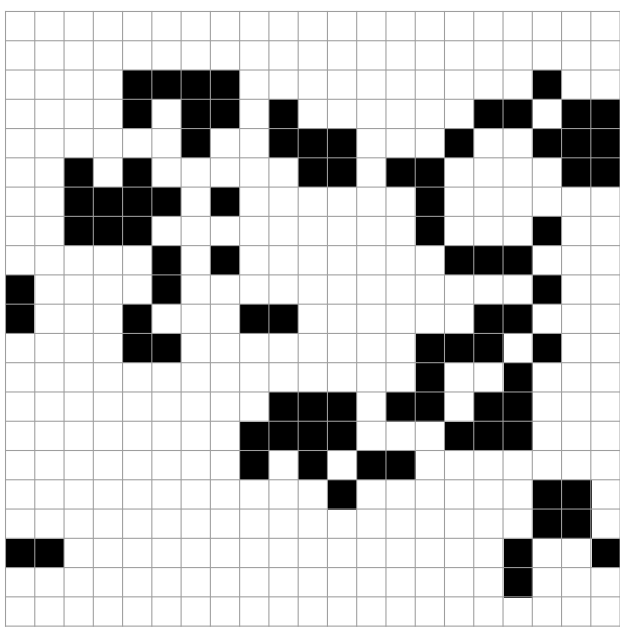
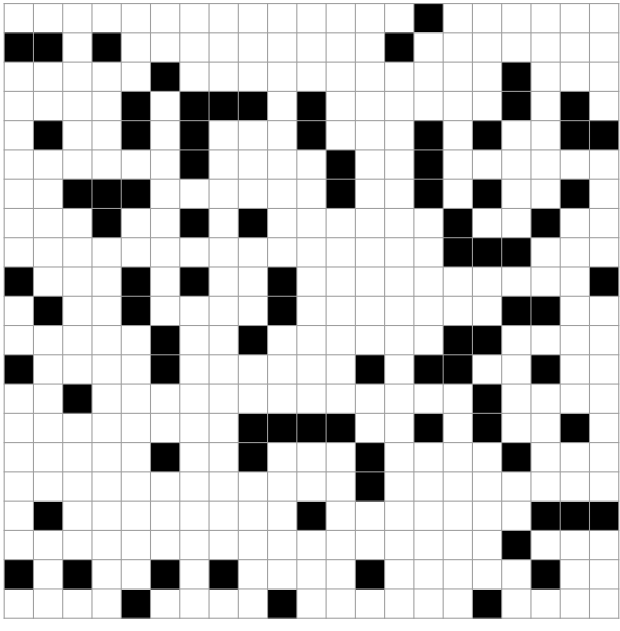


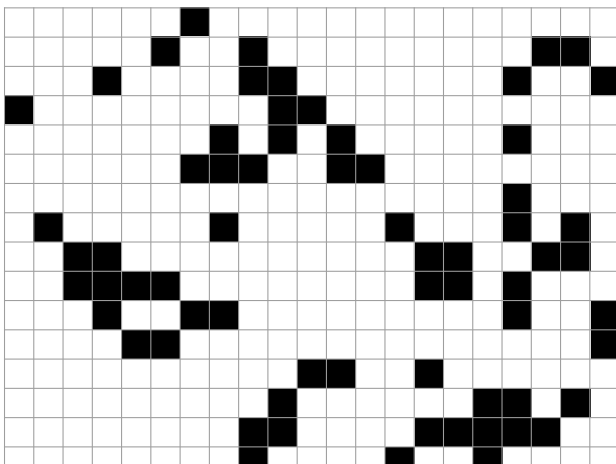
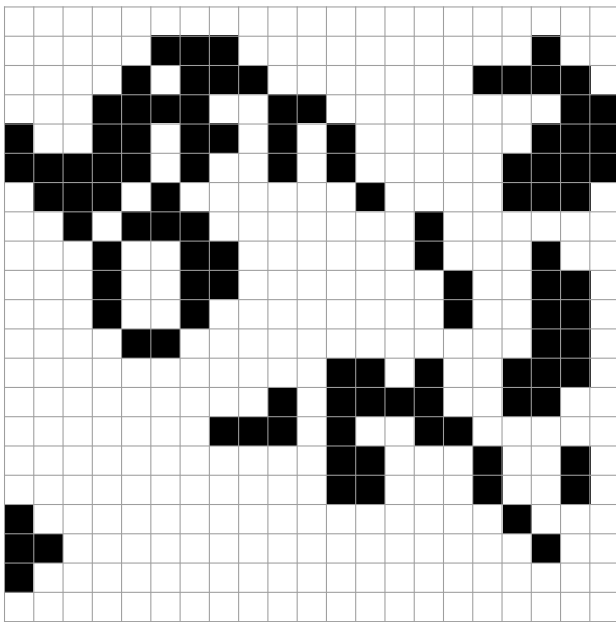
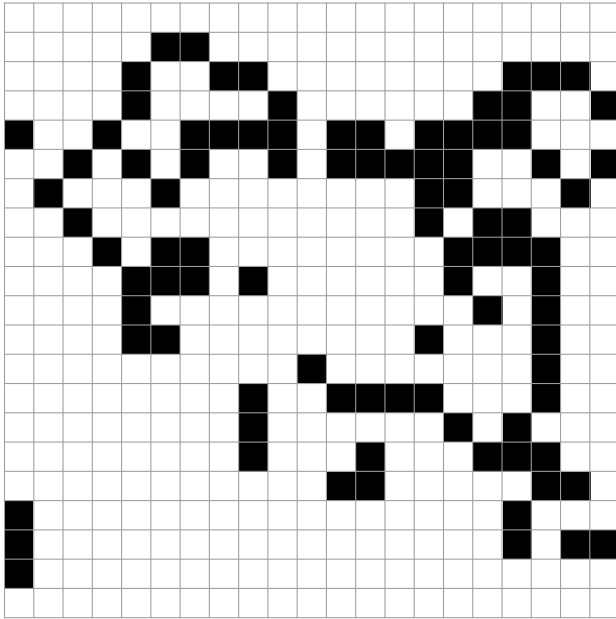
```
In[25]:= LifeGame[config_, t_] := Module[{livingNghbrs, deadAliveCheck, mat, update, gather},  
(* deadAliveCheck[the status of the cell, the number of lives] *)  
deadAliveCheck[1, 2] := 1;  
deadAliveCheck[_ , 3] := 1;  
deadAliveCheck[_ , _] := 0;  
Attributes[deadAliveCheck] = Listable;  
(* calculating the number of lives surrounding each cell *)  
livingNghbrs[mat_] := Plus@@ (RotateRight[mat, #1] &) /@  
{-1, -1}, {-1, 0}, {-1, 1}, {0, -1}, {0, 1}, {1, -1}, {1, 0}, {1, 1}};  
(* repeat to update each status of cells with specified times t *)  
update[mat0_] := deadAliveCheck[mat0, livingNghbrs[mat0]];  
  
NestList[update, config, t]]
```

```
In[26]:= LifeGame2[config_, t_] := Module[{livingNghbrs, deadAliveCheck, mat, update, gather},  
(* Nest version *)  
deadAliveCheck[1, 2] := 1;  
deadAliveCheck[_ , 3] := 1;  
deadAliveCheck[_ , _] := 0;  
Attributes[deadAliveCheck] = Listable;  
(* calculating the number of lives surrounding each cell *)  
livingNghbrs[mat_] := Plus@@ (RotateRight[mat, #1] &) /@  
{-1, -1}, {-1, 0}, {-1, 1}, {0, -1}, {0, 1}, {1, -1}, {1, 0}, {1, 1}};  
(* repeat to update each status of cells with specified times t *)  
update[mat0_] := deadAliveCheck[mat0, livingNghbrs[mat0]];  
  
Nest[update, config, t]]
```

```
In[27]:= showBoard /@ LifeGame[brd, 4]
```

Out[27]= {





parallel version

```
In[28]:= padding[mat_] := Module[{m}, m = Table[0, {Length[mat] + 2}, {Length[mat][[1]]}];
  m[[2 ;; -2]] = mat;
  m];

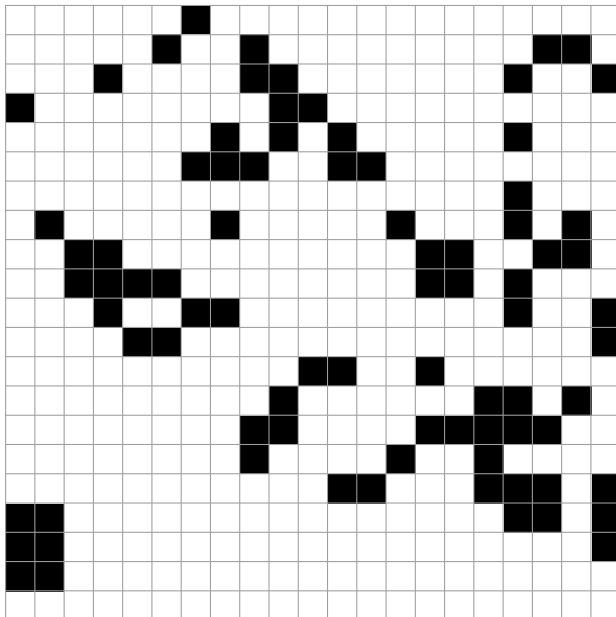
In[29]:= peal[mat_] := Module[{m, n},
  {m, n} = Dimensions[mat];
  mat[[2 ;; m - 1]]];

In[30]:= parallelLifeGame[config_, t_] := Module[{livingNghbrs, deadAliveCheck, mat, update},
  (* deadAliveCheck[the staus of the cell, the number of lives] *)
  deadAliveCheck[1, 2] := 1;
  deadAliveCheck[_, 3] := 1;
  deadAliveCheck[_, _] := 0;
  Attributes[deadAliveCheck] = Listable;
  (* calculating the number of lives surrounding each cell *)
  livingNghbrs[mat_] := Plus@@(RotateRight[mat, #1] &) /@
    {{-1, -1}, {-1, 0}, {-1, 1}, {0, -1}, {0, 1}, {1, -1}, {1, 0}, {1, 1}};
  (* repeat to update each status of cells with specified times t *)
  update[mat0_] := deadAliveCheck[mat0, livingNghbrs[mat0]];

  (* note:
  this program doesn't use NestList but Nest though NestList is easy to implement. *)
  mpiScatter[config, mat, 0, mpiCommWorld];
  mat = padding[mat];
  Nest[Function[{a}, EdgeCell[mat]; mat = update[mat]], mat, t];
  Flatten[peal /@ mpiGather[mat], 1]
]

In[31]:= parallelLifeGame[brd, 4] // showBoard
```

Out[31]=



Timing

In[35]:= \$NProc

Out[35]= 8

```
In[36]:= brdSize = 800;
brd = Partition[Table[If[Random[] > 0.8, 1, 0], {brdSize * brdSize}], brdSize];
```

```
In[41]= Timing[LifeGame2[brd, 10];] (* 10 generation step of the game *)
```

```
Out[41]= {9.73398, Null}
```

```
In[40]= Timing[parallelLifeGame[brd, 10];]
```

```
Out[40]= {1.97058, Null}
```