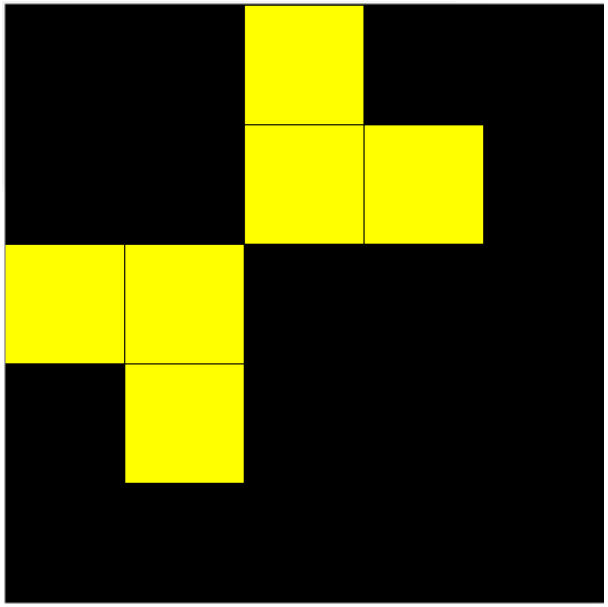
プログラミング演習I （第12回） 課題

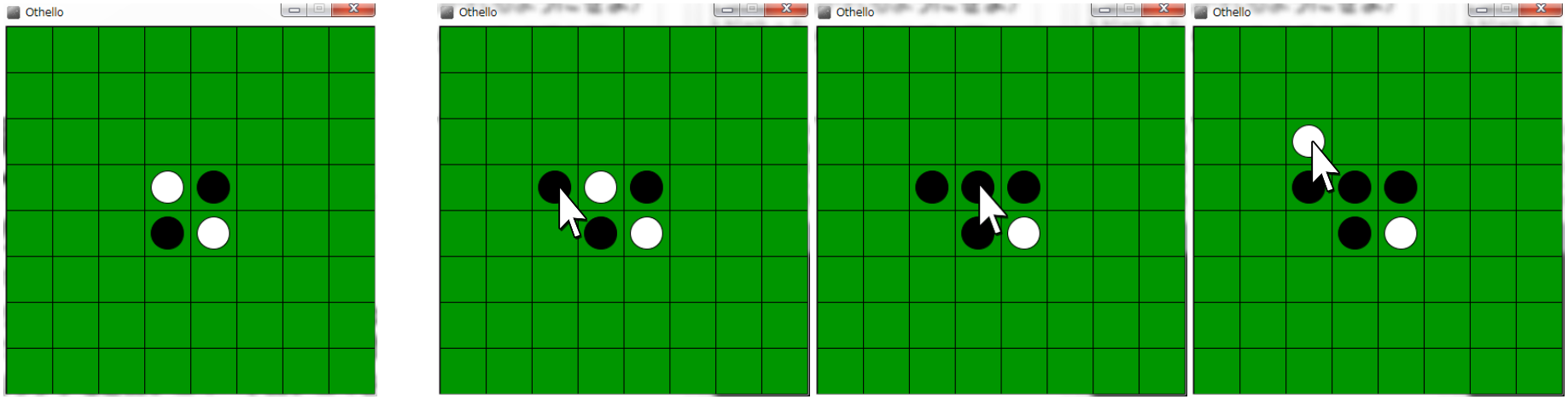
- **基本①　スケッチ名：basic_LightsOut**
  - 横5マス、縦5マスの盤面を作り、そのマス目上をクリックすると、クリックされたマス目の上下左右とそのマス目自体の色を反転させるLights Outを作れ
  - なお、すべてが黒色になったらCLEAR!と表示するようにせよ
  - **ただし、起動したタイミングで下記のような表示になっているようにせよ（予習した人はこれをやるだけ！）**



プログラミング演習I （第12回） 課題

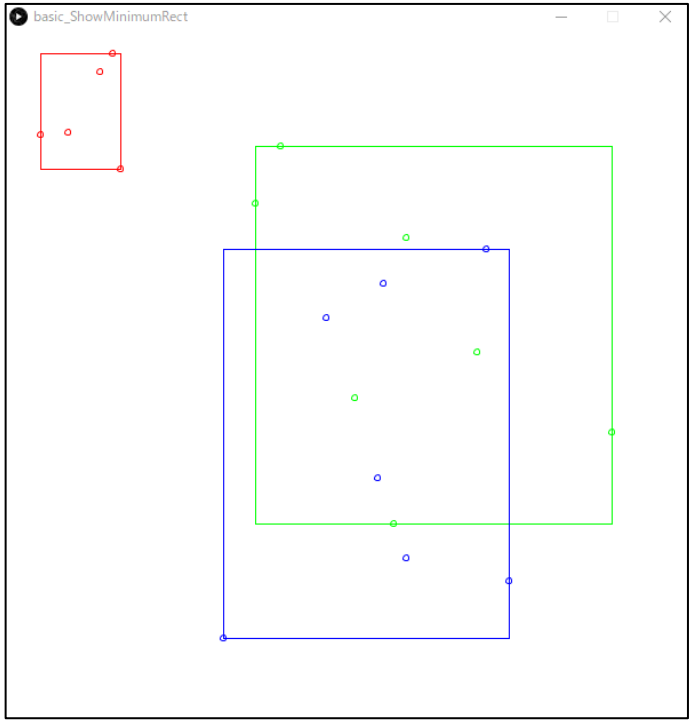
- **基本②　スケッチ名：basic_Othello**
  - 横8マス、縦8マスのオセロの盤面と左下のコマの初期配置を作れ
  - コマがないマスをクリックすると、ターンに応じて白いコマまたは黒いコマが置かれるようにせよ（白いコマ、黒いコマは交互に置かれるようにせよ）
    - ターンを管理する変数を用意して，あきマスに置かれたら値を変更する！
  - また、黒いコマがおかれているマスをクリックすると白いコマへ、白いコマがおかれているマスをクリックすると黒いコマへ変わるようにせよ

初期配置



プログラミング演習I （第12回） 課題

- **基本③スケッチ名：basic_ShowMinimumRect**
  - ある点の座標を(x[i], y[i])と表現する点集合を考える
  - この点集合を引数として与えたときに，その点群を囲う最小の四角形（中を塗りつぶさないもの）を描画する関数drawMinimumRectを作成したい。配布したプログラムの関数を完成させなさい。
  - 下図が出力例となるので確認せよ



プログラミング演習I （第12回） 課題

発展①　スケッチ名：advanced_Matrix

- 3x3の行列を2次元配列の引数としてとり、行列を表示する関数display()、行列式を求める関数det()、対角和を求める関数trace()、ユークリッドノルム（平方和の平方根）を求める関数norm()を作成し、左下図のように表示せよ。なお、displayを除く関数の返り値の型はfloatとせよ。
- 配布するadvanced_Matrixを利用せよ。

```
Matrix:
 2.0  -1.0   0.0
-1.0   2.0  -1.0
 0.0  -1.0  -2.0
行列式（det）=-8.0
対角和（trace）=2.0
ユークリッドノルム（norm）=4.0

Matrix:
 2.0   2.0   3.0
 1.0   4.0   5.0
 0.0  -1.0  -2.0
行列式（det）=-5.0
対角和（trace）=4.0
ユークリッドノルム（norm）=8.0
```

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix}$$

の行列式を求めると次のようになります：

$$\det(A) = a_{00}(a_{11}a_{22} - a_{12}a_{21}) - a_{01}(a_{10}a_{22} - a_{12}a_{20}) + a_{02}(a_{10}a_{21} - a_{11}a_{20})$$

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix}$$

の対角和（主対角線上の要素の和）は、以下のように求めることができます：

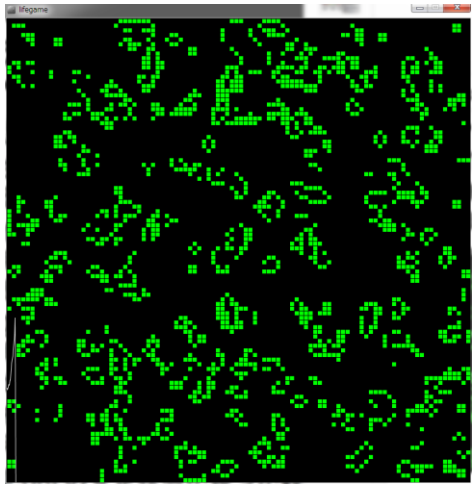
$$\mathrm{trace}(A) = a_{00} + a_{11} + a_{22}$$

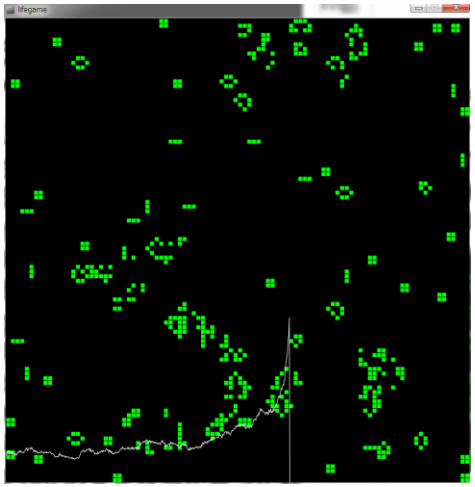
$$\mathrm{norm}(A) = \sqrt{\sum_{i=0}^{2}\sum_{j=0}^{2} a_{ij}^2}$$

協力ChatGPTさん

プログラミング演習I （第12回） 課題

- **発展②　スケッチ名：advanced_LifeGame**
  - 誕生、生存、過疎、過密によってセルが生まれたり死んだりする下図のようなライフゲームを作ろう（下図では生存数をグラフで表示しているが、このグラフはなくても良い）
  - ライフゲームでは、対象とするセルの周囲8マスが生きているか死んでいるかを数え、その結果に応じてセルを生きている状態にするか、死んでいる状態にするかを切り替える。
  - **800x800のウインドウ内に100x100**のマス目を用意し、セルが生きている場合は緑色の四角形を、死んでいる場合は黒色の四角形を描画するようにせよ。ライフゲームのルールは次ページで説明する。





プログラミング演習I （第12回） 課題

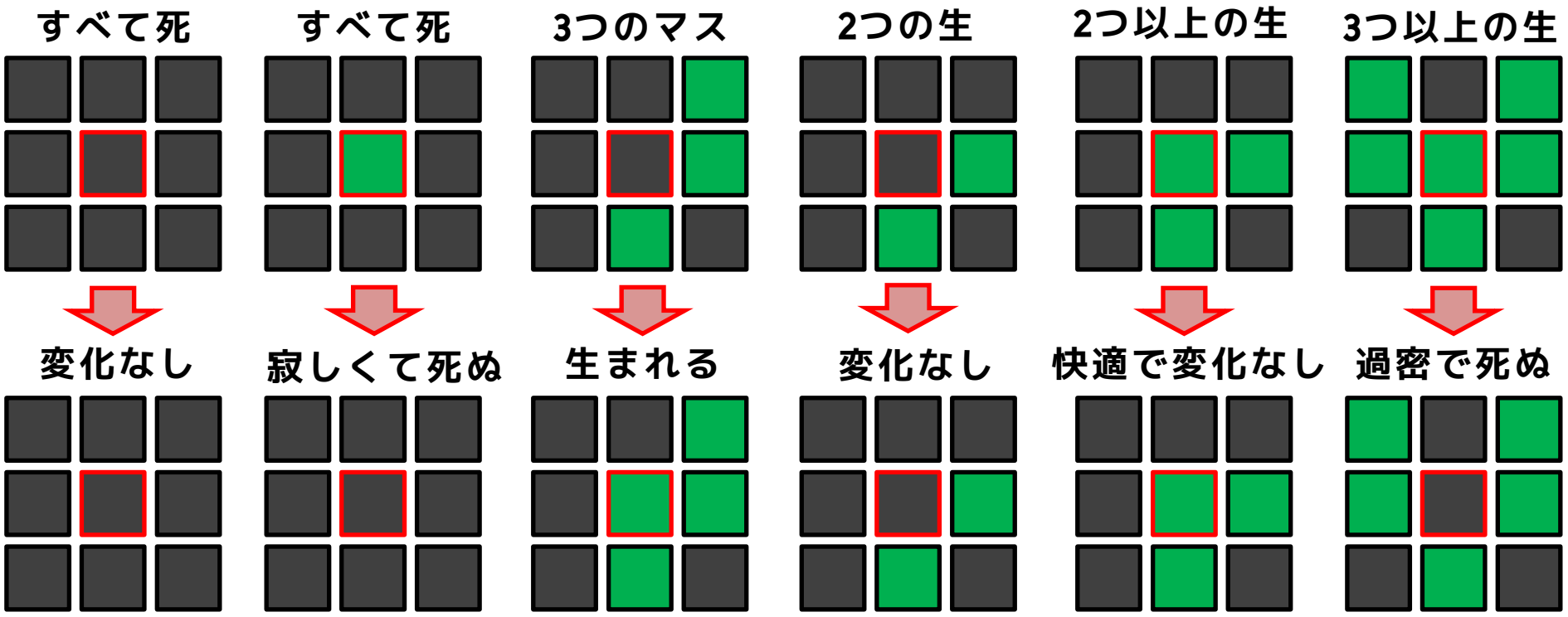
あるマス（赤フレーム）の縦・横・斜めの8マスの生死の状態（生の数）に注目する

【誕生】 死んでいるセルに隣接する生きたセルがちょうど3つならば次世代が誕生

【生存】 生きているセルに隣接する生きたセルが2つか3つならば次世代でも生存

【過疎】 生きているセルに隣接する生きたセルが1つ以下ならば過疎により死滅

【過密】 生きているセルに隣接する生きたセルが4つ以上ならば過密により死滅

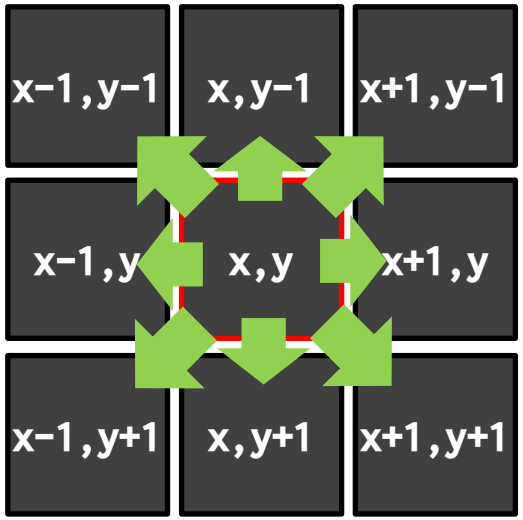


プログラミング演習I （第12回） 課題

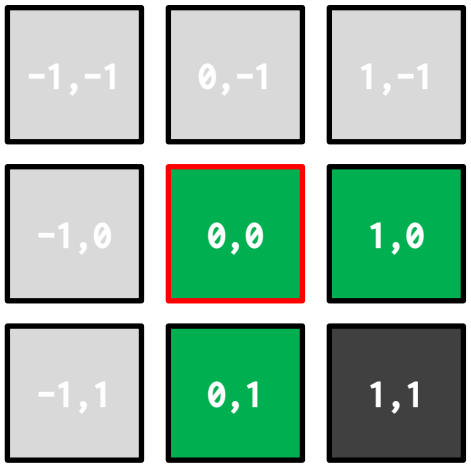
- ルールを整理すると・・・

数が3つなら生 **数が2つなら維持** **それ以外なら死**

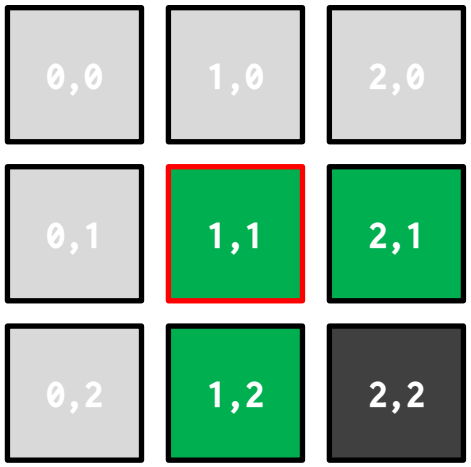
チェックする配列の添字は何になるか？



0から始めると
条件分岐が多くて面倒



1から始めると
条件分岐が少なくなる



表示しない外周を用意して，周囲の「生」の数を数えると楽！

プログラミング演習I（第12回） 課題

ヒント

- 下記URLの安定状態が幾つか観測されたら成功
  - http://ja.wikipedia.org/wiki/ライフゲーム

- 興味が湧いたら、ライフゲームの世界で検索してみよう！