情報工学実験Ⅱ

「Tea を用いた複雑系シミュレーション」

担当教員名：赤嶺 有平

提　出　日：２０１０年12月6日
学 籍 番 号：０９５７０７B
氏　　名：大城　佳明

課題 1

以下の仕様を満たす CA シミュレータを作成し，レポートに

1. 近傍則のソースコード（*.tea のソースコードのみ）を記載せよ．

2. 初期状態や遷移規則を変更し結果に与える影響を考察せよ

-状態: 整数値(0 or 1)

-状態遷移規則:

--ムーア近傍の状態の合計が奇数の時，着目セルの状態は 1 に遷移する

--ムーア近傍の状態の合計が偶数の時，着目セルの状態は 0 に遷移する

プロジェクト名「task1」としてプロジェクトを作成する

ムーア近傍とは，着目セルに接する 8 つのセルを指す.

1. 近傍則のソースコード（*.tea のソースコードのみ）を記載せよ．

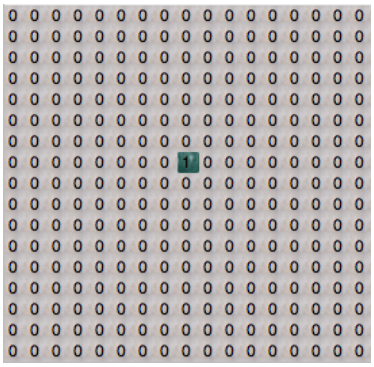
<task1.tea>

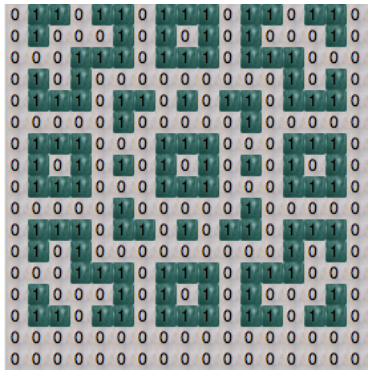
```
cell {
  int state // .状態は、整数型
};

// .リスト変数$neighborに、ムーア近傍のセルの座標を代入する
$neighbor = {  [-1,-1], [0,-1], [1,-1],
               [-1, 0],         [1, 0],
               [-1, 1], [0, 1], [1, 1] };

sync(a_cell of $cell) { // .全セルに{}の操作を適用する
  transform(local(a_cell)) { // .着目セル(a_cell)を原点とする相対座標系を用いる
    local sum=0; // .sync内では，ローカル変数にのみ代入できる
    // 13-19 ムーア近傍のセルの状態stateを合計する
        for_each(nei of $neighbor) {
    // .for_eachは、リスト変数の各要素に対して、与えられた文の実行を繰り返す
    // .$neighborは、 8 つの要素を持つので、 8 回ループする。
    // .neiを通じて、$neighborの各要素にアクセスできる
      sum += @nei.state; // ."@nei"は、位置neiにあるセルをあらわす
                         // ."@nei.state"は、位置neiのセルの状態である
    }
amari = sum %2;

    if(amari == 1) {
      a_cell.state = 1; //sum が奇数の時、次の状態は 1
    }else{
      a_cell.state = 0; //偶数の時、次の状態は 0
    }
  }
}
```

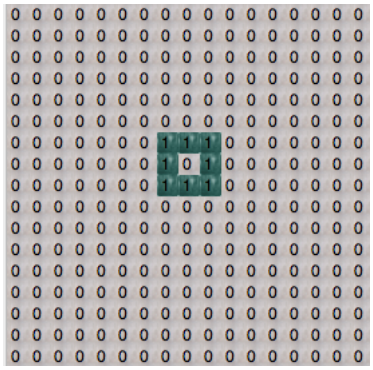
2. 初期状態や遷移規則を変更し結果に与える影響を考察せよ

1.セルを 17×17 の大きさにした場合

<実行結果>







<考察>

A.繰り返しの動作が行われる

B.図では「１」を中央に１つだけ配置し、実行した。

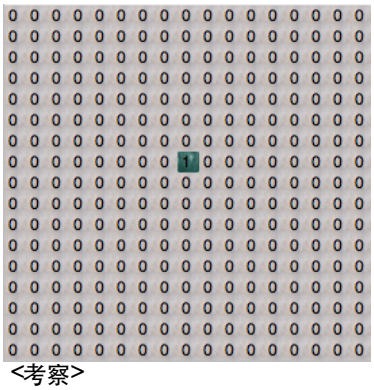
C.このとき time=16 で１周期である。

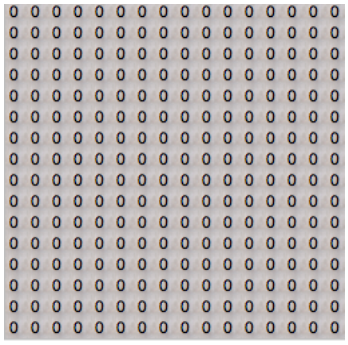
D.初期の配置の数を増やしても繰り返し動作が行われた

E.17 は奇数なので「１」が常にでると考えられる

2.セルを 16×16 の大きさにした場合

<実行結果>





<考察>

A.実行 time=8 の時に「１」がなくなる

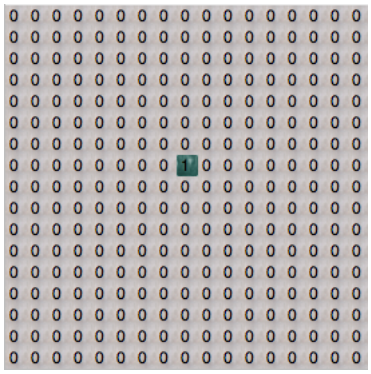
B.実行時に「１」の配置の数は関係なく、time=8 の時にすべて「0」になる

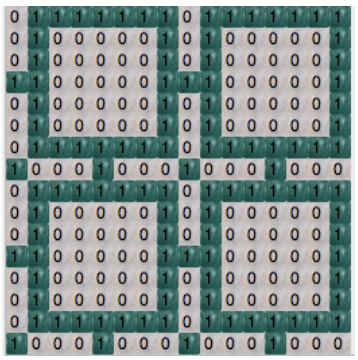
C.縦と横の大きさが偶数の場合はなくなる。

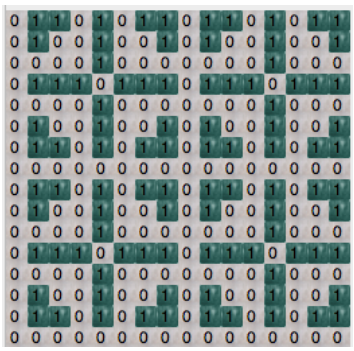
D.どちらかが奇数ならば消えずに形を変え実行され続ける

3.ムーア近傍の状態の合計を３で割った剰余が１の時に「１」を出力した。

<実行結果>







<考察>

A.サイズが 8×8 で一つの図ができる

B.繰り返しいろいろな形の図が表示されるようになった

C.奇数で割るので、サイズは関係なく繰り返し表示が行われると思う

課題２

以下を，レポートとしてまとめよ.

Step 1. ２次元 CA でシンプルに記述可能と考えられるモデル，パターン，アニメーションなどを一つ考えてください

Step 2. 考えたモデルを近傍則として記述し，ASSAM でコード化，TEA で実行してください

Step 3. Step 1 で考えたアイデアと結果は一致しましたか？一致しなかった場合は，なぜ一致しなかったのかを考えてください．

Step 1. ２次元 CA でシンプルに記述可能と考えられるモデル，パターン，アニメーションなどを一つ考えてください

1.「２」が基本は下に動く

2.「１」はは動かない(ブロック)

3.「２」が「１」に衝突すると、「２」は「１」を避けて動く

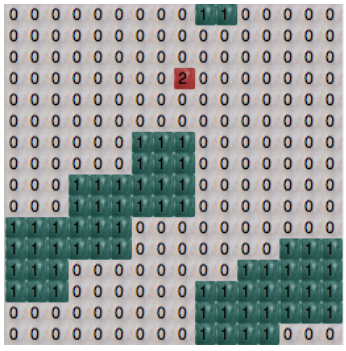
Step 2. 考えたモデルを近傍則として記述し，ASSAM でコード化，TEA で実行してください

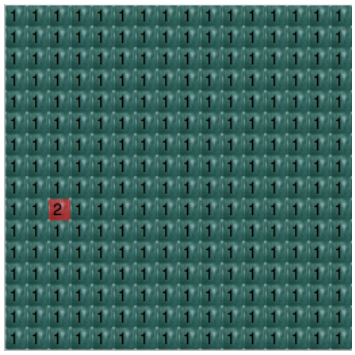
1.「２」は下に動くために、下へのコピーを行う

2.「１」は動かないようにすため、「１」ならば現地の場所を維持するように命令する

3.「２」が「１」に衝突した場合は左にコピーされるようにする。

<実行結果>





Step 3. Step 1 で考えたアイデアと結果は一致しましたか？一致しなかった場合は，なぜ一致しなかったのかを考えてください．

1.考察(一致しなかった理由)

A.今の値が「１」ならば動かないという条件はうまく起動していた

B.移動するために「２」はコピーを行って動いている。

C.その結果「１」にもコピーが行われおり、図のように下に「１」がコピーされている状態になった

２.考察(対処法)

A.「１」の下の値は変化しないようにする

B.上の値が「１」の場合はコピーされないようにする

３.考察(オプション)

A.「２」は降りるだけではなく、登るようなプログラムを作成した

B.「２」と「１」が衝突した場合に「１」の上に移動するようにした

４.ソース

<task2.tea>

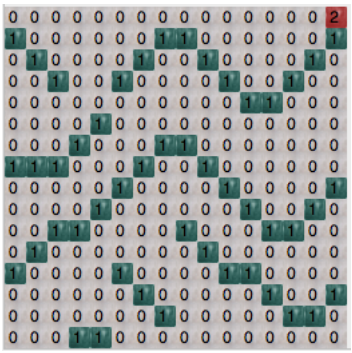
```
//セルの状態変数の宣言
cell {
  //状態変数リスト
  int state
};


sync(a_cell of $cell) {
  transform(local(a_cell)) {
        //「１」の時、変化しない
        if( a_cell.state == 1 ){
        a_cell.state = @[0,0].state;
                }else if(@[0,-1].state == 1){      //上の値が「１」の時、変化しない
        a_cell.state =          a_cell.state;

                }else{
        //「１」に衝突した場合を考える
        //上から「１」に衝突した場合
        if((@[1,0].state == 2) && (@[1,1].state == 1)){
        a_cell.state = 2;
        }else if((@[1,1].state == 2) && (@[0,1].state == 1)){    //左側から衝突した場合
        a_cell.state = 2;
        }else{          //衝突しなかった場合
        a_cell.state = @[0,-1].state;
        }

                }
        }
}
```

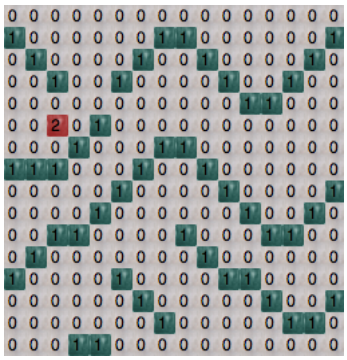
<task2.fld>

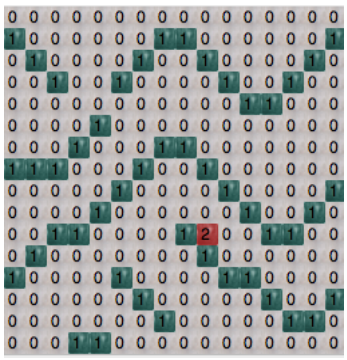
```
16 16
state_name state
0000000000000002
```

```
1000000110000001
0100001001000010
0010010000100100
0000000000011000
0000100000000000
0001000110000000
1110001001000000
0000010000100001
0000100000010010
0011000010001100
0100000001000000
1000010000110000
0000001000001001
0000000100000110
0001100000000000
```

4.実行結果







課題3

sugar scapeモデルのパラメータ（食欲，初期財産，砂糖の再生度など）の変更が結果に与える影響を考察せよ

1.食欲

A.食欲の値が大きいほど、財産を多く消費しやすく、子供を生みにくくなる。

B.しかし、実際にはそれほど変化はない。

2.初期財産

A.初期財産の値が大きいほど、消滅しにくくなる。

B.実際に初期財産の値が大きいほどエージェントの数は多くなる。

C.しかし、どんなにこの値を大きくしてもエージェント数は５００程度である。

3.砂糖の再生度

A.砂糖の再生度の頻度が少ないほど消滅しやすくなる。

B.実際に再生頻度を 43 回に 1 回が生き残れる最小の頻度であった。

課題 4
sugar.tea にコメントを付けよ

<sugar.tea>

```
//セルの状態変数宣言
cell {
  int max_sugar, //最大砂糖量
  int sugar   //現在の砂糖量
};

//エージェントantの内部状態宣言
agent ant {
  int view, //視野範囲
  int eat, //食欲
  int worth,  //財産

  int display
};

//シミュレーション開始時のみ実行される（初期状態設定）
if(time() == 0) {
  for_each (a_cell of $cell) {
    //砂糖の山を生成する
    d1 = 5.0 - length(posof(a_cell) - [30,20])/5.0;
    //座標[30,20]に山の頂点を作る
    d2 = 5.0 - length(posof(a_cell) - [20,30])/5.0;
    //座標[20,30]に山の頂点を作る
    a_cell.max_sugar = (int)max(max(d1, d2), 0.0);
    //d1の山とd2の山の大きい方の値をとる
    a_cell.sugar = a_cell.max_sugar;
    //求めた砂糖の量を現在の砂糖量にする

    //1%の確率でエージェントを生成し、配置する
    if(1%) {
      if(50%) {
        new ant(4, 1, 10, 0) -> posof(a_cell);//セルにエージェントを配置する
        //antの引数は，内部状態の初期値を宣言順に表す
        //視野範囲=4, 食欲=1, 財産=10, display = 0
      }else{
        new ant(8, 2, 10, 1) -> posof(a_cell);
        //視野範囲=8, 食欲=2, 財産=10, display = 1
      }
```

```
   }
  }
}

//4ステップに一回更新される
if(time() % 4 == 0) {
  for_each (a_cell of $cell) {
    a_cell.sugar = min(a_cell.sugar+1, a_cell.max_sugar);
   //sugarの値を増やす。ただし、最大値は越えない
  }
}

$view_dir = { [1,0], [0,-1], [-1,0], [0,1] };
//上下左右の配列定義

for_each (an_ant of $ant) {
  transform(local(an_ant)) {

    an_ant.worth += @[0,0].sugar;
    //現在地のsugarの値を加算している
    @[0,0].sugar = 0;
    //現在地のsugarの値を0にする
    an_ant.worth -= an_ant.eat;
    //antの食欲の分、財産を減らしている

    //antの財産が食欲の10倍より大きい時
    if(an_ant.worth > an_ant.eat*10) {
      new ant(an_ant.view, an_ant.eat, 10, an_ant.display) -> [0,0];
    //新しいantの作成(子の作成)
      an_ant.worth -= 10;
    //子を作成したので財産を10消費する
    }

   //定義している
    max_sugar = -1;
    $move_to = {};

   //視野内で最も砂糖の多い地点を探す
    for_each(dir of $view_dir) {
      //an_antから上下左右にviewの分最大のsugarの値を探す
      for(i=1; i<an_ant.view; i+=1) {
        view_site = dir*i;
        //an_antから距離iを調べる
        sugar = @view_site.sugar;
        //iの位置のsugarの値を入れる

        //大きさの比較を行っている
        if(max_sugar < sugar) {
          max_sugar = sugar;
          $move_to = { view_site };
          //現在の値が大きい場合はmove_toに位置を格納する
```

```
      }else if(max_sugar == sugar) {
        $move_to += view_site;
        //現在の値と同じ場合はmove_toに追加する
      }
     }
    }

    //an_antの位置を最大のsugerの値に移動させる
    //移動させる位置が複数ある場合を考え、ランダムに移動させる
    an_ant -> $move_to[rand() % sizeof($move_to)];

   //財産がなくなると消滅する
    if(an_ant.worth < 0) {
      delete an_ant;
    }
  }
}

//第 2 引数の値をコードウインドウに出力する
log(STDOUT, sizeof($ant));
```

課題 5

公害の導入

エージェントが行動すると公害が発生する様に変更せよ．また，汚染された場所をエージェントが嫌がるようにする．汚染は，時間により減少する．

変更したところだけ記述する

<sugar.tea>(sugarpoll_tsm)

```
//セルの状態変数宣言
cell {
  int max_sugar, //最大砂糖量
  int sugar ,  //現在の砂糖量
    int pollution_disp //汚染度
};

(省略)

//4ステップに一回更新される
if(time() % 4 == 0) {
  for_each (a_cell of $cell) {
    a_cell.pollution_disp = min(10,max(a_cell.pollution_disp-1, 0));
  }
}

(省略)

    an_ant.worth += @[0,0].sugar;
```

```
    //現在地のsugarの値を加算している
    @[0,0].sugar = 0;
    //現在地のsugarの値を0にする
    @[0,0].pollution_disp += 1;
    //汚染度を＋１する
    an_ant.worth -= an_ant.eat;
    //antの食欲の分、財産を減らしている

(省略)

      for(i=1; i<an_ant.view; i+=1) {
        view_site = dir*i;
        //an_antから距離iを調べる
        sugar = @view_site.sugar / (@view_site.pollution_disp+1);
        //砂糖の量を汚染度+1で割っている

         //大きさの比較を行っている
        if(max_sugar < sugar) {
```

<解析>

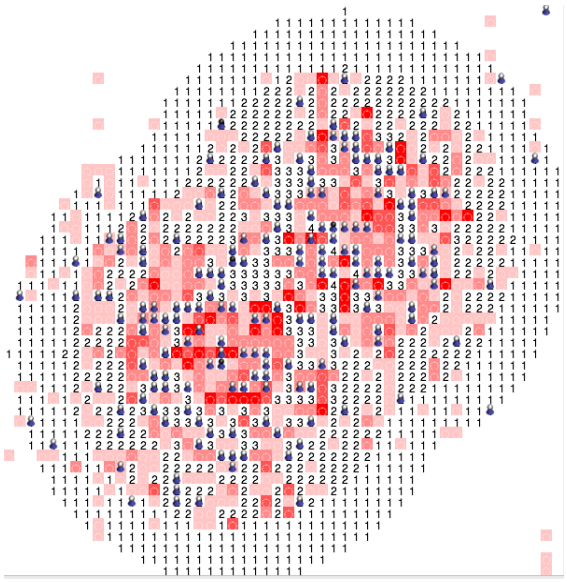
1.int pollution_disp で最初に汚染度の引数の宣言をした

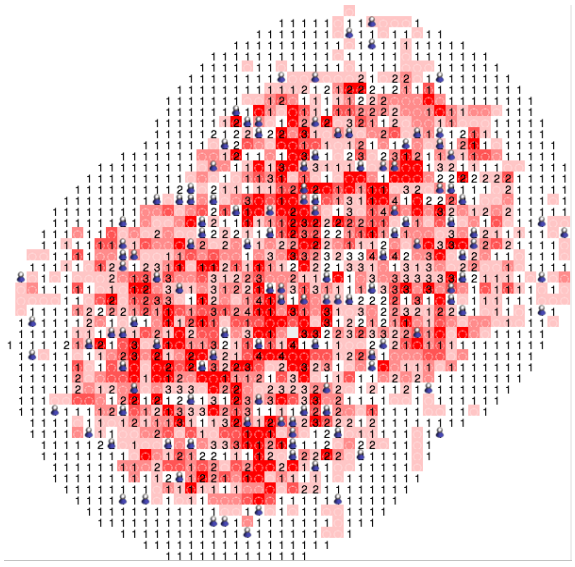
2.4ステップで汚染度を「-1」する。max と min を使って 0 以上 10 以下の汚染度になるようにした。

3. @[0,0].pollution_disp += 1;で汚染度をインクリメントする

4.砂糖の量だけではなく、汚染度を考慮するために「 @view_site.sugar / (@view_site.pollution_disp+1)」をした

<実行結果>





1.汚染度が視覚でわかる

2.砂糖の量が多くても、汚染度が高いと食べられない。