ライフゲームの数理

総合数理学部現象数理学科

2610150027 里村孔明

目次

1 はじめに

2次元のセルオートマトンにおいて、ライフゲームはとても単純なルールであるにも関わらず、様々な複雑な形が出現する、とても良い研究材料である。しかしその扱いやすさゆえに様々な研究が進んでおり、新たな発見を得るのは難しいと思われる。本研究では今まで発見された事柄や代表的な形をまとめつつ、新たなシミュレーションの方法について検討していきたいと思う。

2 ライフゲームとは

ライフゲームとは、2 次元のセルオートマトンであり、各セルには生と死の 2 つの状態があり、時間とともにその状態が変化していく。各セルの次の世代の状態はそのセルおよびそのセルの周囲に生存している個体の数によって決定する。以下の 3 つのルールを持つものを一般的にライフゲームと呼び、本研究で扱う。

① 死んでいるセルに隣接する生きたセルがちょうど 3 つあれば次の世代が誕生する。
② 生きているセルに隣接する生きたセルが 2 つか 3 つであれば次の世代も生存する。
③ 生きているセルに隣接する生きたセルが 1 つ以下または 4 つ以上であれば、過疎または過密により次の世代のセルは死滅する。

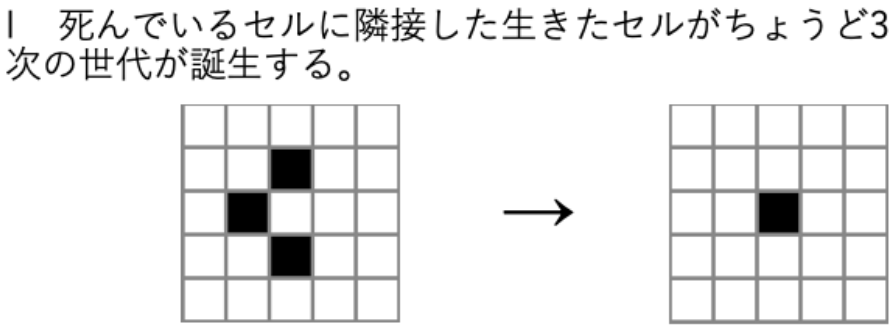
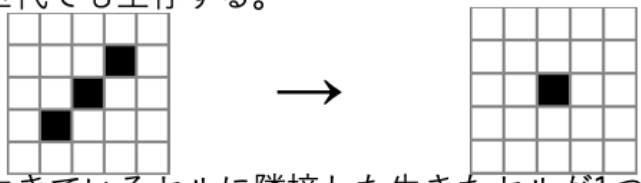


図1 ライフゲームのルール①

II　生きているセルに隣接する生きたセルが2つか3つであれば、次の世代でも生存する。



III　生きているセルに隣接した生きたセルが1つ以下または4つ以上であれば、過疎または過密により次の世代のセルは死滅する。

図2　ライフゲームのルール②、③

3 ライフゲームの既存の形など

ライフゲームでは、世代を進めていくうちに全ての個体が消滅してしまう場合と、一定の形を保って生存し続ける場合がある。生存し続ける場合については以下の 4 パターンに分けられている。

① 固定物体
　世代が進んでも変化しないものを指す

② 振動子
　ある周期で同じ図形に戻るものを指す

③ 移動物体
　一定のパターンを繰り返しながら移動するものを指す

④ 繁殖型
　マス目が無限であれば無限に増え続けるものを指す

それぞれの代表的な形にはそれぞれ固有の名称がある。固定物体の代表的なものには以下のようなものがある。

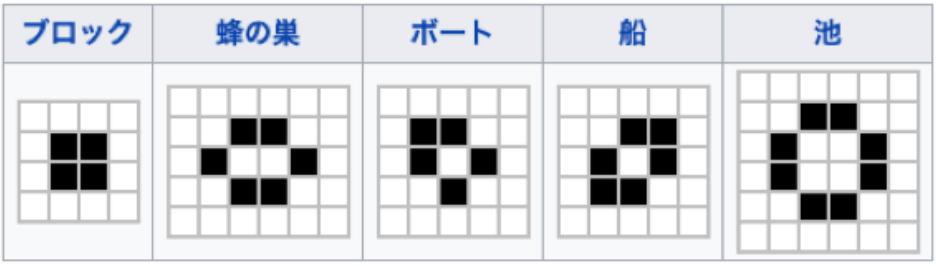


図 3 固定物体の例

次に、振動子については以下のようなものがある。

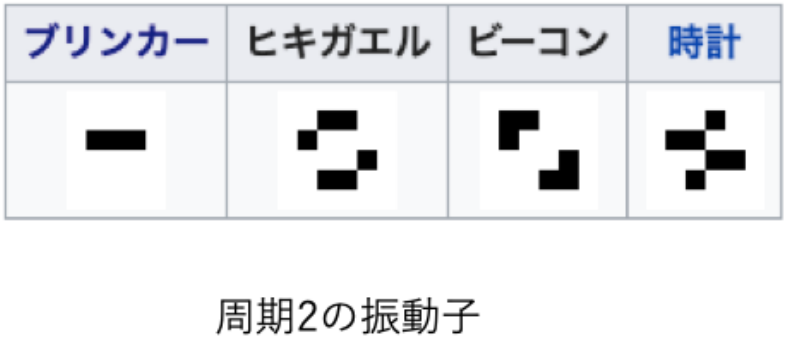


図 4 振動子の例

次に、移動物体については以下のようなものがある。

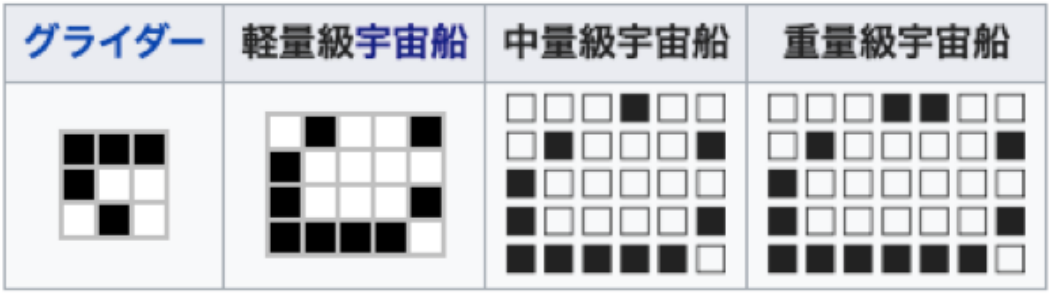


図 5 移動物体の例

繁殖型については作成したプログラムで動かして確かめてみることにする。

4 ライフゲームのプログラム

C による数値計算とシミュレーション(小高知宏　2009)に掲載されているライフゲームのシミュレーションプログラムを参考としてプログラムを作成した。以下がその C 言語によるソースコードである。

```
#include <stdio.h>
#include <stdlib.h>

#include "glsc3d_3.h"

#define WINDOW_SIZE_X        (1)
#define WINDOW_SIZE_Y        (1)

#define N 42
#define MAXT 100
#define BUFSIZE 256
#define S 10

void initworld(int world[][N]) ;
void putworld(int world[][N]) ;
void nextt(int world[][N]) ;
int calcnext(int world[][N],int i,int j) ;
double f(int point);



//int main(int argc, char *argv[])
int main(){
g_init("main2.c", WINDOW_SIZE_X, WINDOW_SIZE_Y);
//g_def_scale_2D(0, -S/2, -S/2, -S/2, -S/2, -S/2, -S/2, WINDOW_SIZE_X,
WINDOW_SIZE_Y);
//g_def_marker(0, 1, 1, 0, 1, 1, 25);
```

```
  int t ;
  int world[N][N]={0} ;

    initworld(world) ;
    printf("t=0¥n") ;
    putworld(world) ;

    for(t=1;t<MAXT;++t){
          nextt(world) ;
          printf("t=%d¥n",t) ;
          putworld(world) ;
    }
     return 0 ;
}

double f(int point){
   return (float)point * (float)S / (float)N - (float)S / 2;
}

void nextt(int world[][N])
{
       int nextworld[N][N]={0} ;
       int i,j ;

       for(i=0;i<N;++i)
         for(j=0;j<N;++j)
           nextworld[i][j]=calcnext(world,i,j) ;
       for(i=0;i<N;++i)
         for(j=0;j<N;++j)
           world[i][j]=nextworld[i][j] ;
}

int calcnext(int world[][N],int i,int j)
{
  int no_of_one=0 ;
```

```
  int x,y ;
  for(x = i-1;x <= i+1;++x){
    for(y = j-1; y <= j+1;++y){
    no_of_one += world[(x + N) % N][(y + N) % N];
    }
  }
      no_of_one-=world[i][j] ;
      if(no_of_one==3) return 1 ;
      else if(no_of_one==2) return world [i][j] ;

  return 0 ;
}

void putworld(int world[][N])
{
  int i,j ;

  g_cls();
  g_sel_scale(0);
  g_sel_marker(0);


  for(j=N-1;j>=0;--j){
    for(i=0;i<N;++i)
        if(world[i][j]==0){
        printf("　") ;
        }else{
        printf("■") ;
        //g_marker_2D(f(i), f(j));
        }
      printf("¥n") ;
  }

  g_finish();
  g_sleep(0.5);
}
```

```
void initworld(int world[][N])
{
  char linebuf[BUFSIZE] ;
  int x,y ;
       while(fgets(linebuf,BUFSIZE,stdin)!=NULL){
         if(sscanf(linebuf,"%d",&x)<1) break ;
         if(fgets(linebuf,BUFSIZE,stdin)==NULL) break ;
         if(sscanf(linebuf,"%d",&y)<1) break ;
         if((x>=0)&&(x<N)&&(y>=0)&&(y<N)) world[x][y]=1 ;
       }
}
```

最初に
1
12
2
13
のように初期値を決め、入力する。上のように入力すると、1 行目 12 列目と 2 行目 13 列目に生きているセルが存在することとなり、時間ごとに変化が始まる。全部で 100 ステップとなるようにした。また、ライフゲームは無限に広がる空間を想定してはいるが、実際のプログラムではどうしても端が存在してしまうので、上下と左右はつながっているという想定で、周期境界条件というものを採用した。よって、例えばグライダーが下に進み続けると上から出てくる、というプログラムになっている。

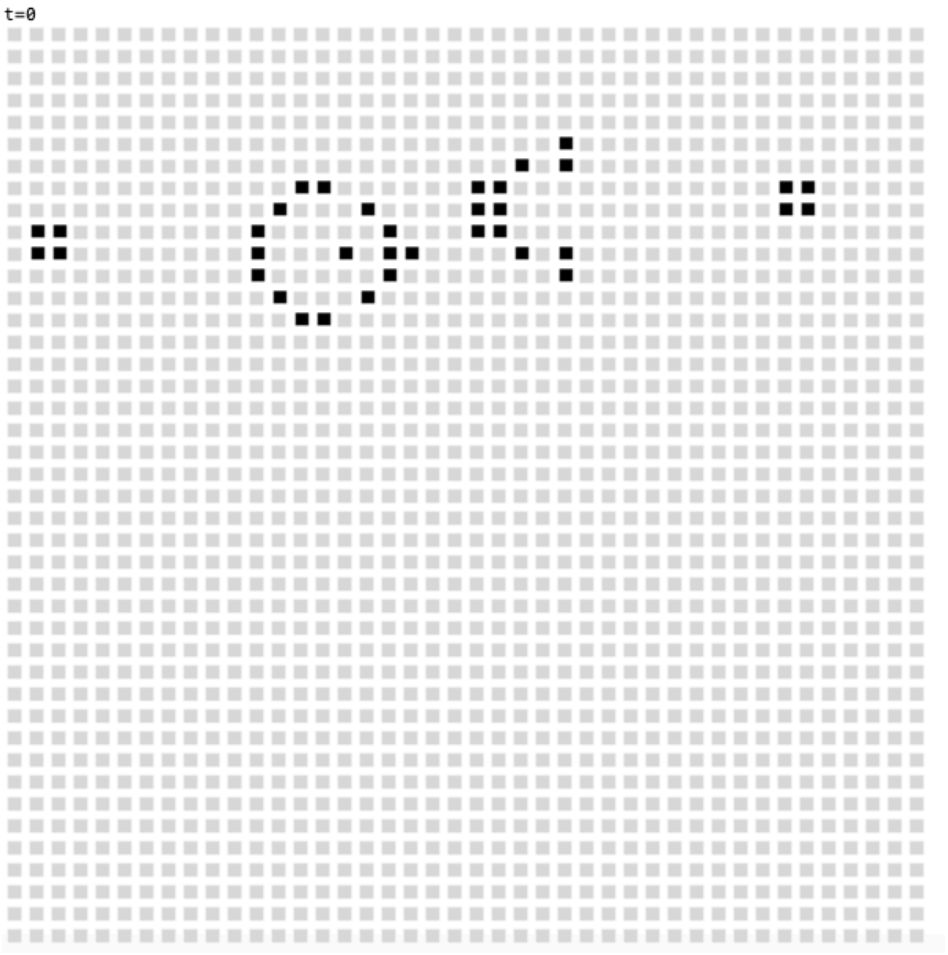


図 6 グライダーガン

初期配置を上図のようにしたものをグライダーガンと言い、次々にグライダーを打ち出す繁殖型の 1 つである。数ステップ後ににはグライダーを打ち出している様子が確認できる。

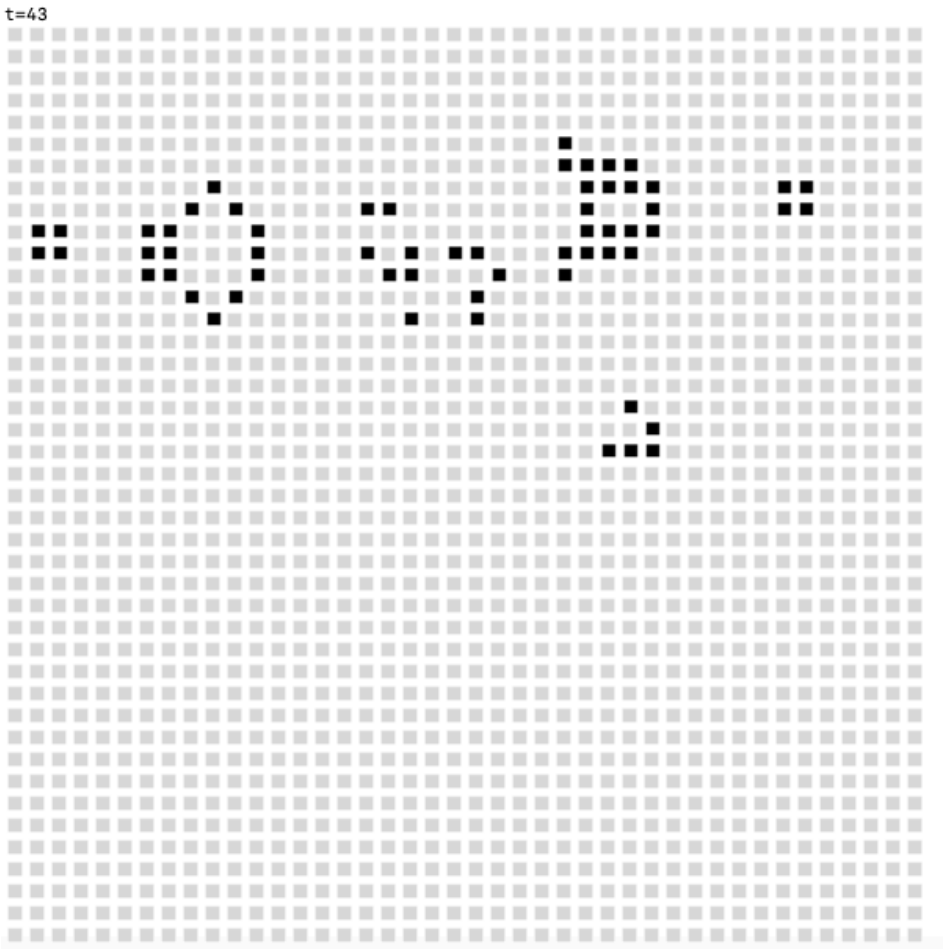


図 7 グライダーガン 2

上図は図 6 から 43 ステップ後のグライダーガンである。グライダーが打ち出されている様子が確認できる。

他にもシュシュポッポ汽車やマックスと言った様々な繁殖型が存在し、このプログラムでも同様に動いている様子が確認できる。

6 ライフゲームの 3D 化

先ほどの 2D のライフゲームを 3D にしたら面白い結果が得られるのではないかと期待し、作成をした。2D のプログラムを少し変え、3D の表示に関しては glsc のサンプルプログラムを参考にした。以下がそのソースコードである。

```
#include <stdio.h>
#include <string.h>

#include "glsc3d_3.h"

#define WINDOW_SIZE_X		(500)
#define WINDOW_SIZE_Y		(500)

#define MAXT (5)
#define N (6)
#define FN (6.0)

#define SIZE (4)

int world[N][N][N];

float resize(int point);
void put_world();
void change_world();
int count_arround(int wx, int wy, int wz);
int life_condition(int count, int nowCondition);
void init_wolrd();
int rest_world();
void put_3d_world();

int main(){
	g_init("main2.c", WINDOW_SIZE_X, WINDOW_SIZE_Y);
	g_def_scale_3D(0,
		-N/2, N/2,
		-N/2, N/2,
		-N/2, N/2,
		-N/2, N/2,
		-N/2, N/2,
		-N/2, N/2,
		-N/2, N/2,
		WINDOW_SIZE_X, WINDOW_SIZE_Y);
```

```
	g_def_marker(9, 0, 0, 1, 1, 2, 20);

	int i, j, k;
	int time;
	init_wolrd();
	for(time = 0; time < MAXT; time++){
		put_world();
		printf("%d times rest: %d¥n",time, rest_world());
		put_3d_world();
		change_world();
	}

	return 0;
}

float resize(int point){
	return (float)point - FN / 2;
}

void put_3d_world(){
	g_cls();
	g_sel_scale(0);
	g_sel_marker(0);
	g_box_center_3D(
		0, 0, 0,
		N, N, N,
		G_YES, G_NO
	);
	g_sel_marker(9);
	int px, py, pz;
	for(pz = 0; pz < N; pz++){
		for(py = 0; py < N; py++){
			for(px = 0; px < N; px++){
				if(world[py][px][pz] == 1){
					g_marker_3D(resize(px), resize(py),
resize(pz));
```

```
                                        }
                                }
                        }
                }
                g_finish();
                g_sleep(3);
}

void put_world(){
        int px, py, pz;
        for(pz = 0; pz < N; pz++){
                printf("[%d]¥n", pz);
                for(py = 0; py < N; py++){
                        for(px = 0; px < N; px++){
                                printf("%d", world[py][px][pz]);
                        }
                        printf("¥n");
                }
        }
}

void change_world(){
        int ch_y, ch_x, ch_z;
        //生死判定
        int next_world[N][N][N];
        int life_count = 0;
        int life = 0;
        for(ch_z = 0; ch_z < N; ch_z++){
                for(ch_y = 0; ch_y < N; ch_y++){
                        for(ch_x = 0; ch_x < N; ch_x++){
                                life_count = count_arround(ch_x, ch_y, ch_z);
                                life = life_condition(life_count,
world[ch_x][ch_y][ch_z]);
                                next_world[ch_y][ch_x][ch_z] = life;
                        }
                }
```

```
		}

		//世界の移行
		for(ch_z = 0; ch_z < N; ch_z++){
			for(ch_y = 0; ch_y < N; ch_y++){
				for(ch_x = 0; ch_x < N; ch_x++){
					world[ch_y][ch_x][ch_z] =
next_world[ch_y][ch_x][ch_z];
				}
			}
		}
}

int count_arround(int wx, int wy, int wz){
		int ca_x, ca_y, ca_z;
		int extension_world[N+2][N+2][N+2];
		//周りに一マス壁を作る
		for(ca_z = 0; ca_z < N+2; ca_z++){
			for(ca_x = 0; ca_x < N+2; ca_x++){
				for(ca_y = 0; ca_y < N+2; ca_y++){
					extension_world[ca_x][ca_y][ca_z] = 0;
				}
			}
		}

		for(ca_z = 0; ca_z < N; ca_z++){
			for(ca_x = 0; ca_x < N; ca_x++){
				for(ca_y = 0; ca_y < N; ca_y++){
					extension_world[ca_x+1][ca_y+1][ca_z+1] =
world[ca_x][ca_y][ca_z];
				}
			}
		}

		//その点を中心として 3*3 の全ての生命を数える
		int addx, addy, addz;
```

```
	int count_num = 0;
	for(addx = -1; addx <= 1; addx++){
		for(addy = -1; addy <= 1; addy++){
			for(addz = -1; addz <= 1; addz++){
				count_num +=
extension_world[wx+1+addx][wy+1+addy][wz+1+addz];
			}
		}
	}
	//自分だけ削除
	count_num -= extension_world[wx+1][wy+1][wz+1];

	return count_num;
}

int life_condition(int count, int nowCondition){
	//生死条件を考える
	if(count ==3){
		//3 以上は生存
		return 1;
	}else if (count > 1){
		//現状維持
	  return nowCondition;
      }else{
		//1 以下は死滅
		return 0;
	}
}

void init_wolrd(){
	int ini_x, ini_y, ini_z;
	//初期化
	for(ini_z = 0; ini_z < N; ini_z++){
		for(ini_y = 0; ini_y < N; ini_y++){
			for(ini_x = 0; ini_x < N; ini_x++){
				world[ini_x][ini_y][ini_z] = 0;
```

```
                        }
                    }
                }

                FILE *fp;
            char *filename = "life_birth.txt";
            char readline[N] = {'¥0'};

            /* ファイルのオープン */
            if ((fp = fopen(filename, "r")) == NULL) {
                fprintf(stderr, "%s のオープンに失敗しました.¥n", filename);
            }

            /* ファイルの終端まで文字を読み取り表示する */
            while ( fgets(readline, N, fp) != NULL ) {
                sscanf(readline, "%d %d %d", &ini_x, &ini_y, &ini_z);
                world[ini_x][ini_y][ini_z] = 1;
            }

            /* ファイルのクローズ */
            fclose(fp);
}

int rest_world(){
            int r_y, r_x, r_z;
            int rw_count = 0;
            for(r_z = 0; r_z < N; r_z++){
                    for(r_y = 0; r_y < N; r_y++){
                            for(r_x = 0; r_x < N; r_x++){
                                    rw_count += world[r_y][r_x][r_z];
                            }
                    }
            }
            return rw_count;
}
```

このプログラムも先ほどと同様に初期値を与え、時間によって表示させるプログラムである。ただし、3D なので、

2 2 2

2 3 4

4 5 5

のように縦、横、高さの座標を打つと、そこに生きているセルが存在するようになっている。このプログラムでは同じフォルダ内に life_birth.txt というテキストファイルを用意し、そこに上のような座標を打っておけば、それが初期値となるように設定してある。例えば

2 2 2

2 2 3

2 2 4

と入力すると、

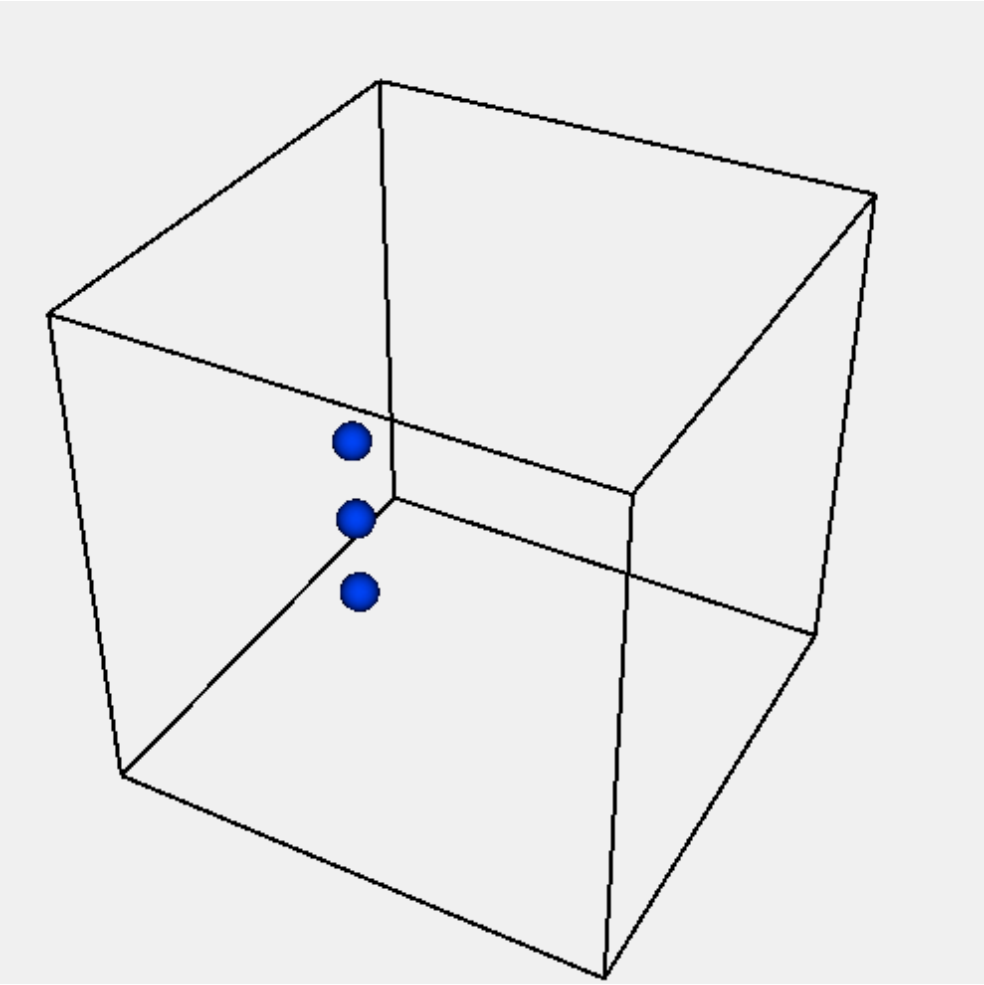


図 8　3D ライフゲーム

と表示され、ここから変化が始まる。

ちなみに、このプログラムも 3 方向に対して周期境界条件を採用している。

6 ライフゲームの 2D と 3D の違い

2D と 3D の最大の違いはルールの設定である。3D の場合は隣接するマスが 26 マス存在するので、隣接する生きているセルが何個以上で生存し、何個以上で現状維持する、というルール設定が非常に多くのパターンがあり、面白い性質が見つかるパターンを探すのが難しい。上記のプログラムでは 2D の場合と同じく 3 個で生存、2 個で現状維持、それ以外は死滅、としてみたが、2D のように上手くいかず、すぐに増殖してしまうケースが多くなってしまった。先行研究なども調べてみたが、3D に関して取り扱っているものは少なく、明確なルールも決まっていないようであった。今回の研究ではプログラムを作成するだけとなってしまったので、今後はこのプログラムを用いて固定物体や振動子などを作り出すことができるルールを発見することが課題である。

7 おわりに

今回の研究ではプログラムを作るだけで終わってしまいましたが、これまで数々の提案やサポートをしていただいた桂田先生と、プログラムの作成に協力してくれた卒業生の山岸君に改めて感謝させていただきます。今後このプログラムが何かの研究などの役に立てば嬉しい限りです。

参考文献

C による数値計算とシミュレーション(小高知宏　2009)

https://ja.wikipedia.org/wiki/%E3%83%A9%E3%82%A4%E3%83%95%E3%82%B2%E3%83%BC%E3%83%A0 （Wikipedia ライフゲーム）図3、4、5を引用