コンウェイのライフゲームの３Ｄ化

卒業研究

明治大学総合数理学部現象数理学科
４年２組６番
上枝和生
2023 年 2 月 28 日

目次

1. はじめに

ライフゲームとは、1968 年にイギリスの数学者ジョン・ホートン・コンウェイ(John Horton Conway)が考案した生命の誕生、進化、淘汰などを簡易的なモデルで再現したゲームである。コンウェイはライフゲームに関する論文を書かなかったため、一般的に知られるようになったのは、Scientific American に Gardner のコラムが掲載されてからである。ライフゲームは単純なルールで複雑な振る舞いをすることが有名で、初期配置を自分で決めたりランダムに選んだりして、その成り行きを眺めるひとり遊び用のゲームである。今でも様々なルールが作成されたり研究が進められていて、最近では 2013 年に自己複製パターンが見つけられている。そんな中で、３D のライフゲームに関して、調べてみてもルールに関する情報が出てこなかったため、本研究ではコンウェイのライフゲームのような面白い動きをする３D のライフゲームのルールを探すことを目標とする。

2. ライフゲームのルール

ライフゲームには碁盤のような格子があり、一つの格子をセルと呼ぶ。セルには二つの状態「生」と「死」がある。各セルの次の世代の状態（生か死）はルールに従い、周囲の８つのセルの状態によって決まる。これから説明するルールは最も有名なコンウェイのライフゲームと呼ばれるルールである。

① 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 つあれば、次の世代が誕生する。

② 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、次の世代でも生存する。

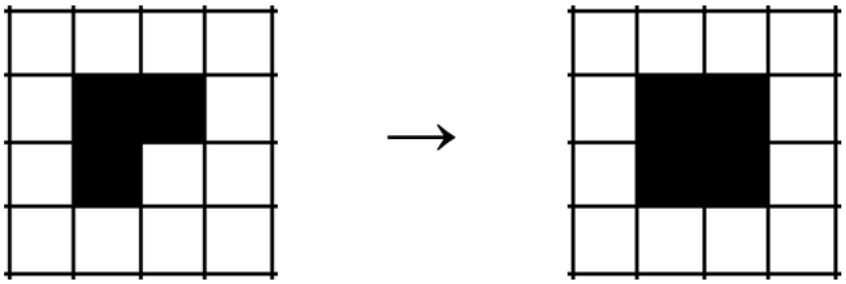

図１　ルール①、②

③ 死亡　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つ以上ならば、次の世代は過疎または過密によって死亡する。

死んでいるセルに隣接する生きたセルが３つ以外の場合、次の世代でも死亡したままである。

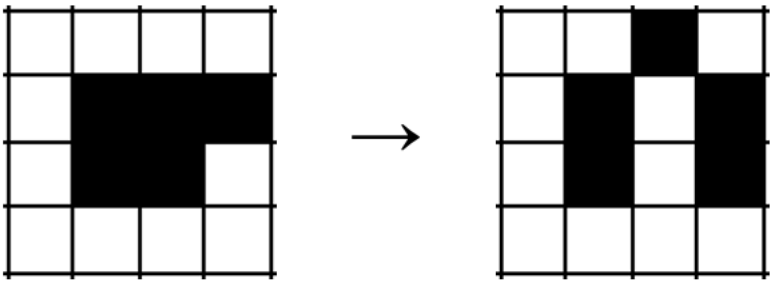

図２　ルール③

このようなルールを生存条件の数字と誕生条件の数字を使って、生存条件の数字/誕生条件の数字として 23/3 のように表記する。

3. コンウェイのライフゲームの代表的な形

コンウェイのライフゲームには代表的な生き続ける形があり、それらは４つに分類できる。

① 固定物体　いつまでも同じ形をしているまとまり

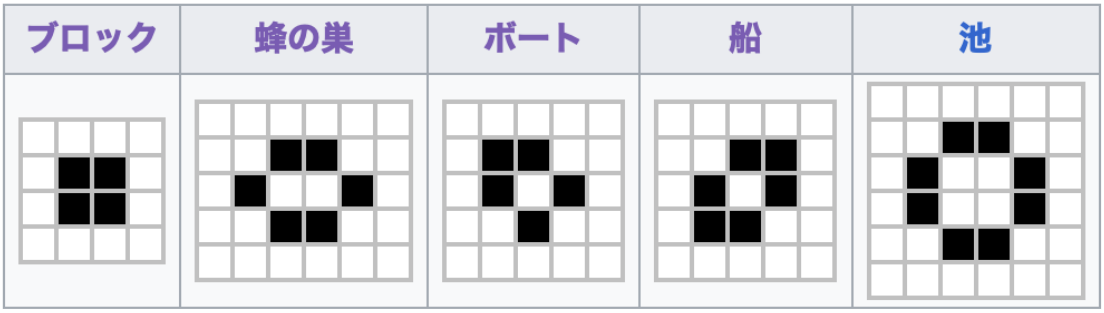

図３　固定物体

② 振動子　ある周期で同じ動きを繰り返すまとまり

下にある図は周期２で同じ動きを繰り返す物体である。

図４　振動子

③　移動物体一定のパターンを繰り返しながら移動するまとまり

下の図の例としてグライダーを説明すると、周期４で同じ動きを繰り返しつつ、物体が右下に移動し続ける物体である。

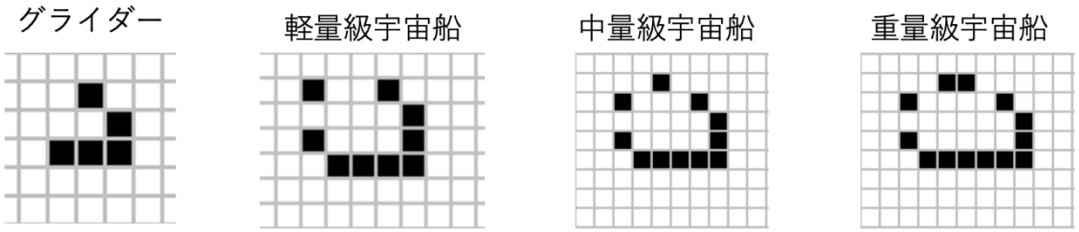

図５移動物体

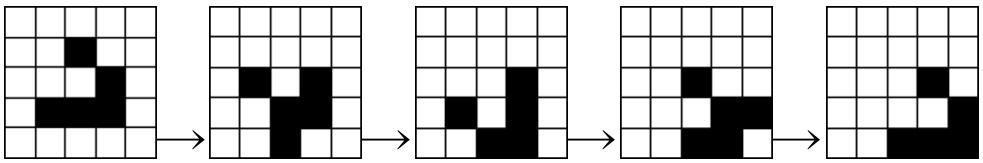

図６　移動物体の例　グライダー

④　繁殖型　マス目が無限にあれば生きているセルが無限に増殖するまとまり

これは繁殖型で有名なグライダーガンと呼ばれる、上で説明したグライダーという物体

を無限に右下に発射し続ける物体である。

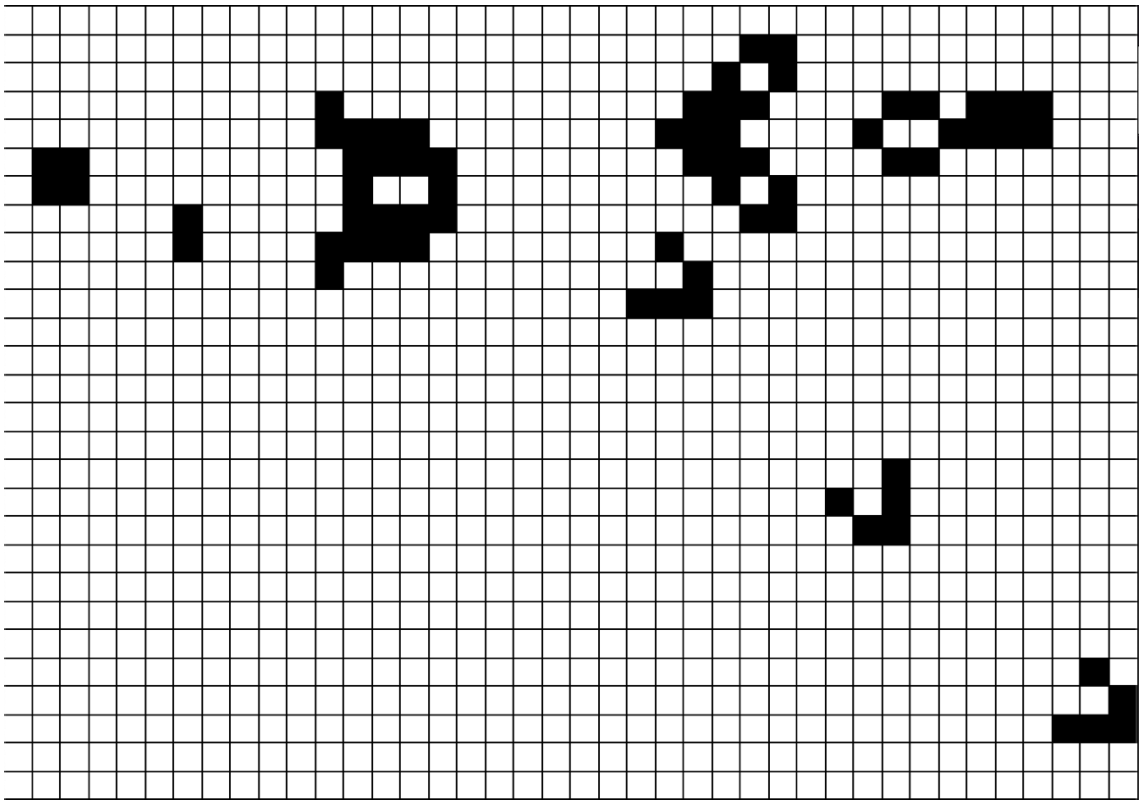

図７　繁殖型の例　グライダーガン

4.ライフゲームの３D 化

コンウェイのライフゲームの繁殖型のような複雑な振る舞いが３D で再現できたら面白いものが出来ると考え、作成した。２次元の時周囲のセルの数が８個であったのに対し、３D になると２６個まで増えるため、ルールを考える際多くのパターンがあることから、最適なものを探すことが難しい。適切なルールを探すことは簡単ではなかったため、二つの方法を用いて探すことにした。

4.1 ルール探し（１）

まず私は、ライフゲームのルールに関して調べた。３D のルールについては見つからなかったため、２次元のルール探しを参考にして３D に応用させようとした。そこで私は高田・坂間[2]を参考にして、ルール探しを進めた。この論文では、様々なルールを考察し、ある法

則で分類している。各ステップで「誕生」、「生存」、「死亡」が適用された数を調べ、その数のステップごとの推移で分類する方法である。下にある図７がそれぞれの推移を表したものになっている。この分類の中で、コンウェイのライフゲームは「死亡」＞＞「生存」＞「誕生」という関係になっているグループである。３D のライフゲームで様々なルールを試し、それぞれ同じグループに含まれるかを確認したが、試した限りでは同じグループに含まれるルールは見つからなかった。

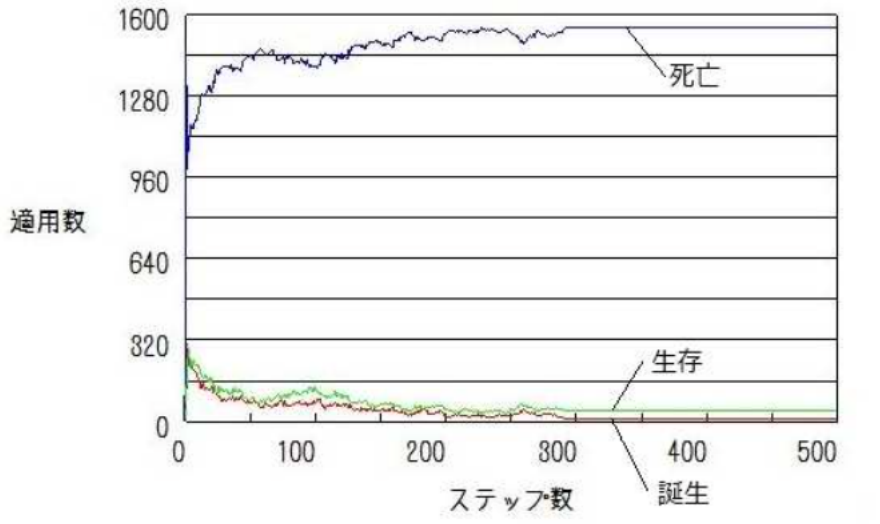

図８　各ステップの「誕生」、「生存」、「死亡」が適用された数の推移(高田・坂間[2])

4.2 ルール探し（２）

次に私は、コンウェイのライフゲームの固定物体をヒントにし、３D で固定物体ができるようなルールを考えた。コンウェイのライフゲームでは２×２の正方形が固定物体になっていたので、３D で２×２×２の立方体が固定物体になるようなルールを作成した。まず固定物体ができるルールーを作る上で、考えなければならないのが生きているセルの生存条件と、生きているセルに隣接している死んでいるセルの誕生条件である。当然、生きているセルが次の世代でも生存する条件でなければならない。そして、隣接している死んでいるセルは次の世代では誕生してはいけない。それを３D でもわかりやすくするために下のような図を使って考えた。この図の見方は、ひとまとまりごとにフロアマップになっていて、数字がセルになっている。数字自体はそのセルに隣接している生きているセルの個数を表していて、赤い数字は生きていて、白い数字は死んでいるセルになっている。図を見ると、生きているセルの隣接している生きているセルの個数は全て７個なので生存条件には７が必ず入る。次に、周囲の死んでいるセルの隣接している生きているセルの個数をみると、１・２・４個なので誕生条件に１・２・４を入れてはいけない。

図９ 3D ライフゲームのフロアマップ

こうしてできたルールをもとに、他にも固定物体や振動子ができるように少しずつルールを改良していった。そうして最終的にできたルールが 4567/7 というルールで、左の図が４つの固定物体で、右の図が周期２の振動子である。

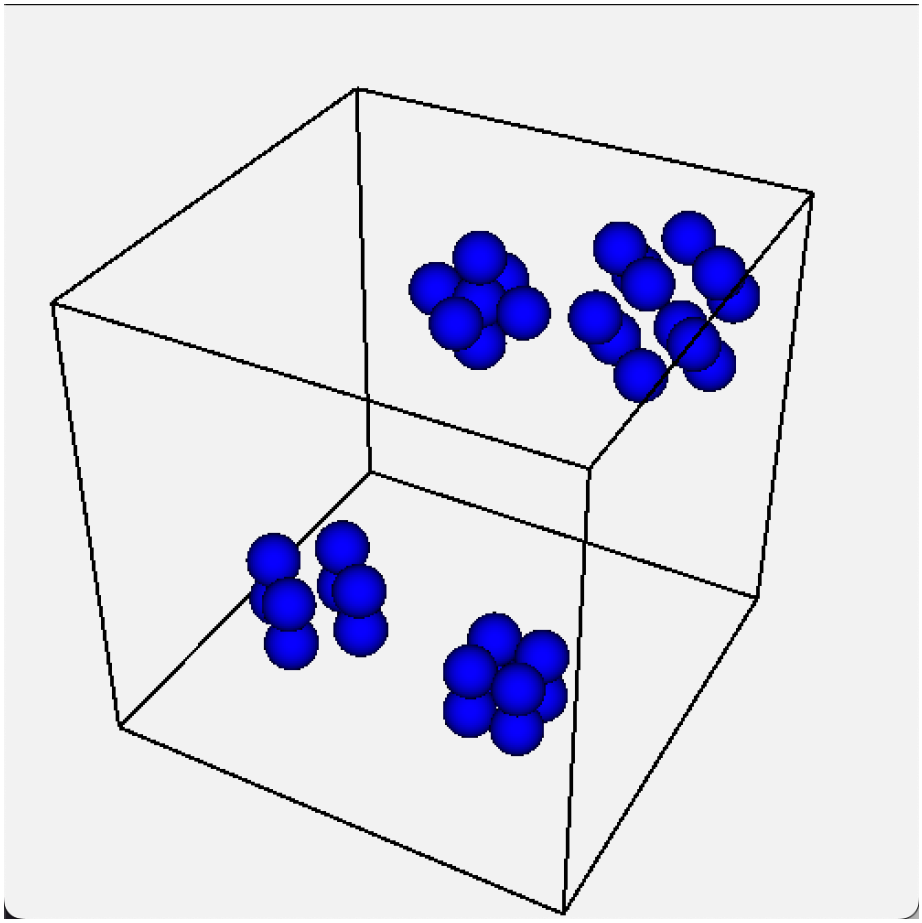

図１０　3D ライフゲームの固定物体の例

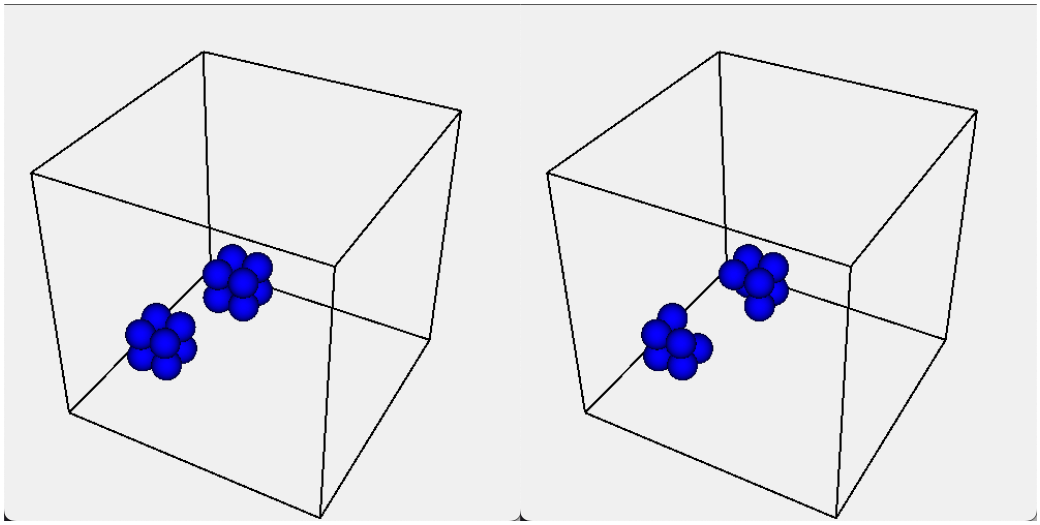

図１１　３D ライフゲームの振動子の例

5. まとめ

３D のライフゲームのルールを考えるにあたり、隣接するセルが２６個もあり、非常に多くのパターンがあるため、ルールを探すことが難しかった。２D のライフゲームの分類を参考にして考えてみたがこの方法では見つからなかったため、かなり古典的な方法でルールを作成した。その結果、固定物体と振動子が再現できるルールは見つけることができた。しかし、まだ移動物体や繁殖型などの複雑な振る舞いは確認できていないので、これらを見つけることが今後の課題である。ルールを探す方法に関しても、より効率的な方法を見つけられればもっと研究が進むだろう。

6. 謝辞

本研修を進めるにあたり、明治大学総合数理学部現象数理学科桂田祐史先生には、指導教員として手厚いサポートをしてくださりました。心から感謝しております。そしてプログラムの参考にさせて頂いた桂田研究室卒業生の里村さんにもお礼申し上げます。

7. 参考文献

[1]ライフゲーム(wikipedia)
https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life#Self-replication
[2]高田祐輔,坂間千秋,擬似ライフゲームの分類に関する一考察,情報処理学会研究報告バイオ情報学,2012(5),pp.1-6
[3]里村孔明,ライフゲームの数理, 2019 年度桂田研卒業研究, 2020 年

8.付録

8.1 コンウェイのライフゲームのプログラムと説明(lifegame23-3.c)

これは2019年度桂田研究室卒業生の里村さんの卒業研究レポートを参考にして作成した。

```
//lifegame23-3.c
#include <stdio.h>
#include <stdlib.h>
#include "glsc3d_3.h"
#define WINDOW_SIZE_X (700)
#define WINDOW_SIZE_Y (700)
#define N 40//一辺のマスの数
#define MAXT 100//ステップ数
#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(){//描画設定
        g_init("main2.c", WINDOW_SIZE_X, WINDOW_SIZE_Y);
        g_def_scale_2D(0, 0.0, N, 0.0, N, 0.0, 0.0, 700.0, 700.0);
        g_def_marker(0, 1, 1, 1, 1, 0, 25);
        g_def_marker(1, 0, 0, 0, 1, 0, 25);
        g_def_line(0, 0, 0, 0, 1, 1, 0);

        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);
        }
        g_sleep(-1);
        return 0 ;
}
double f(int point){//描画設定
        return (float)point * (float)S / (float)N - (float)S / 2;
}
void nextt(int world[][N]) {//次の世代への移行
        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){//各セルに対してルールの適用
        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];//周期境界条
件のためNを足している
                }
        }
        no_of_one-=world[i][j] ;
        if(no_of_one==3){//3は生存条件にも誕生条件にも入っているため必ず1になる
                return 1 ;
        }else if(no_of_one==2){//2は生存条件のみなので次の世代でも同じ状態になる
                return world [i][j];
```

```
        }
        return 0 ;//それ以外は全て死亡する
}
void putworld(int world[][N]) {//線とセルの描画
        int i,j ;
        g_cls();
        g_sel_scale(0);
        g_sel_line(0);
        for(j=N-1;j>=0;--j){
                for(i=0;i<N;++i){
                        if(world[i][j]==0){
                                printf("□") ;
                                g_sel_marker(0);
                                g_marker_2D(i+0.7, j+0.3);
                        }else{
                                printf("■") ;
                                g_sel_marker(1);
                                g_marker_2D(i+0.7, j+0.3);
                        }
                }printf("¥n");
        }
        for(i=0;i<N;i++){
                g_move_2D(i,N);
                g_plot_2D(i,0);
        }
        for(j=N;j>=0;j--){
                g_move_2D(0,j);
                g_plot_2D(N,j);
        }
        g_finish();
        g_sleep(0.1);
}
void initworld(int world[][N]) {//初期配置の読み込み
        /*ターミナルで
        1
        2
```

```
        3
        4
        と入力しもう一度 Enter を押すと(x,y)=(1,2),(3,4)に生きているセルがある初期配置になる*/
        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;
                }
        }
}
```

このプログラムはコンウェイのライフゲームのルール 23/3 を採用している。のプログラムを実行した後に、

1
2
3
4

のように入力し、Enter をもう一度押すと、(x,y)=(1,2),(3,4)のセルが生きているセルになっている初期配置を設定できる。下にグライダーガンの初期配置を例として置く。N を変えると盤面の大きさ、MAXT を変えるとステップ数が変えられる。ライフゲームは本来無限に広がる空間を想定しているが、簡単のためマス目の上下と左右は繋がっているとする周期境界条件を採用している。プログラムには c 言語で書かれている GLSC3D を使用している。GLSC3D は明治大学先端数理科学科の秋山正和教授らによって作られたもので、現象を理解するために数値計算の結果を可視化することを目的に作られたものである。これはフリーで配布されている。

1
31
1
32
2
31
2
32
11
30
11
31
11
32
12
29
12
33
13
28
13
34
14
28
14
34
15
31
16
33
16
29
17
30
17
31

17
32
18
31
21
32
21
33
21
34
22
32
22
33
22
34
23
31
23
35
25
30
25
31
25
35
25
36
35
33
35
34
36
33
36
34

8.2 ３Ｄライフゲームのプログラムと説明(lifegame3D4567-7.c)

これも2019年度桂田研究室卒業生の里村さんの卒業研究レポートを参考にして作成した。

```
//lifegame3D4567-7.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include "glsc3d_3.h"
#define WINDOW_SIZE_X (500)
#define WINDOW_SIZE_Y (500)
#define MAXT (500) //ステップ数
#define N (10) //一辺のマスの数
#define A (256)
#define FN (10.0)
int world[N][N][N];
float resize(int point);
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 s,t;
int main(){
        g_init("main2.c", WINDOW_SIZE_X, WINDOW_SIZE_Y); //描画設定
        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, 30);
        init_wolrd();
        for(t = 0; t < MAXT; t++){
                put_3d_world();
                printf("t=%d¥n",t);
                change_world();
                printf("%d ts rest: %d¥n",t, rest_world());
        }
```

```
	return 0;
}
float resize(int point){//描画設定
	return (float)point - FN / 2;
}
void put_3d_world(){ //glsc3dに描画
	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[px][py][pz] == 1){
					g_marker_3D(resize(px),	resize(py),
resize(pz));
				}
			}
		}
	}
	g_finish();
	g_sleep(1);
}

void change_world(){//次の時刻の配置を設定する
	int ch_y, ch_x, ch_z;
	int next_world[N][N][N];
	int life_count = 0;
	int life = 0;
	for(ch_z = 0; ch_z < N; ch_z++){
		printf("%d段目¥n",ch_z+1);
		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_x][ch_y][ch_z] = life;//次の世代の
セルの状態を入力
                                if(world[ch_x][ch_y][ch_z]==0){//フロアマップ作
成
                                        printf("¥e[0m%d ",life_count);
                                        //ターミナルに死亡セルを白で周りの生存セ
ルの個数を表示
                                }else if(world[ch_x][ch_y][ch_z]==1){
                                        printf("¥e[31m%d ",life_count);
                                        //ターミナルに死亡セルを赤で周りの生存セ
ルの個数を表示
                                }
                        }
                        printf("¥n");
                }

        }
        //世代の移行
        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_x][ch_y][ch_z]                 =
next_world[ch_x][ch_y][ch_z];
                        }
                }
        }
}

int count_arround(int wx, int wy, int wz){
        //その点を中心として 3*3*3 の全ての生死を数える
        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                                       +=
world[(wx+addx+N)%N][(wy+addy+N)%N][(wz+addz+N)%N];
                        }
                }
        }//自分だけ削除
        count_num -= world[wx][wy][wz];
        return count_num;
}
int life_condition(int count, int nowCondition){ //生死条件を考える
        //生きているセルを１、死んでいるセルを０として考える
        if(nowCondition==1){//自分が生存している時
                if(count  ==4||count  ==5||count  ==6||count  ==7){// 生 存 条 件
(4,5,6,7)
                        return 1;
                }else{
                        return 0;
                }
        }if(nowCondition==0){//自分が死亡しているとき
                if (count ==7){//誕生条件(7)
                        return 1;
                }else{
                        return 0;
                }
        }
        return 0;
}
void init_wolrd(){ //初期配置の設定
        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 = "kotei.txt"; //読み込むファイル名をここに入れる
        char readline[A] = {'¥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, A, fp) != NULL ) {
                sscanf(readline, "%d %d %d", &ini_x, &ini_y, &ini_z);
                world[ini_x][ini_y][ini_z] = 1;//ファイルから読み込まれた座標を１
とする
        }
         //ファイルのクローズ
        fclose(fp);
}
int rest_world(){ //世代ｔに生存しているセルの個数を計算する
        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_x][r_y][r_z];
                        }
                }
        }
        return rw_count;
}
```

このプログラムは 4567/7 というルールを採用している。N を変えると盤面の大きさ、MAXT を変えるとステップ数が変えられる。ここでも簡単のためマス目の上下と左右は繋がっているとする周期境界条件を採用している。このプログラムはテキストファイルを読み込んで初期配置を決定するため、あらかじめ下にあるような数字の列のテキストファイルを作っておく必要がある。この例は説明で使用した立方体の固定物体である。１行ずつ１つのセルになっていて、x の値　y の値　z の値となっている。

7 2 2
7 3 2
7 2 3
7 3 3
8 2 2
8 2 3
8 3 2
8 3 3

プログラムには c 言語で書かれている GLSC3D を使用している。