1　エージェントの考え方

　エージェントは、内部状態を持っていて、外界とやりとりすることのできるプログラムです。ここで外界とは、エージェントの置かれた環境であったり、同じ環境内にいる他のエージェントであったりします。一般にエージェントは、あらかじめ与えられた情報に基づいて、自立的に処理を進めることができます。
エージェントの考え方は、シュミレーションの世界でも有用です。プログラムにより仮想世界を構築し、その中でソフトウェア製のロボットであるエージェントを動かすことで、さまざまなシュミレーションを行うことができます。特に複数のエージェントを同一の環境内で動作させる、マルチエージェントの枠組みを用いると、大勢の人間の挙動や動物の群れの様相など、物理的シュミレートすることが可能です。

C による数値計算とシュミレーション　小高　知宏　オーム社　より

2　シンプルなエージェントシュミレーション sa0.c

```
0001: /*
0002:   sa0.c
0003:   シンプルなエージェントシミュレーション
0004:   ２次元平面内で動作するエージェント
0005: */
0006:
0007: #include <stdio.h>
0008: #include <stdlib.h>
0009: #include <math.h>
0010:
0011: #define BUFSIZE 256;
0012: #define N 1
0013: #define N_OF_A 2
0014: #define TIMELIMIT 100
0015:
0016: struct coordinate{
0017:   double x;
0018:   double y;
0019: };
0020:
0021: struct agent{
0022:   int category;
0023:   struct coordinate coord;
0024:   double dattribite[N_OF_A];
0025:   int iattribute[N_OF_A];
0026: };
0027:
```

```
0028: void calcnext(struct agent a[]);
0029: void putstate(struct agent a[], int t);
0030: void cat0(struct agent *cat0agent);
0031: void reverse(int *i);
0032:
0033: int main()
0034: {
0035:   struct agent a[N] = {0};
0036:   int    t;
0037:
0038:   a[0].iattribute[0] = 1;
0039:   putstate(a, 0);
0040:
0041:   for(t = 0; t <= TIMELIMIT; ++t){
0042:     calcnext(a);
0043:     putstate(a, t);
0044:   }
0045:
0046:   return 0;
0047: }
0048:
0049: void calcnext(struct agent a[])
0050: {
0051:   int i;
0052:   for(i = 0; i < N; ++i){
0053:     if(a[i].category == 0){
0054:       cat0(&a[i]);
0055:     }else{
0056:       fprintf(stderr, "ERROR カテゴリがありません(%d)\n", i);
0057:     }
0058:   }
0059: }
0060:
0061: void cat0(struct agent *cat0agent)
0062: {
0063:   reverse(&((*cat0agent).iattribute[0]));
0064:   reverse(&((*cat0agent).iattribute[1]));
0065:
0066:   (*cat0agent).coord.x += (*cat0agent).iattribute[0];
0067:   (*cat0agent).coord.y += (*cat0agent).iattribute[1];
0068: }
0069:
0070: void reverse(int *i)
0071: {
0072:   if(*i == 0){
0073:     *i = 1;
0074:   }else{
0075:     *i = 0;
0076:   }
```

```
0077: }
0078:
0079: void putstate(struct agent a[], int t)
0080: {
0081:   int i;
0082:   printf("t = %d\n", t);
0083:   for(i = 0; i < N; ++i){
0084:     printf("%d\t%lf\t%lf\n", a[i].category,
0085:                              a[i].coord.x,a[i].coord.y);
0086:   }
0087: }
```

3 「sa0」の操作方法

「sa0」の操作方法

```
./sa0
```

上記の「./」は、Linux でのプログラムを実行する時の例です。
ウィンドウズでは、sa0 「Enter」と入力してください。

4 「sa0」の実行結果

```
t = 0
0 0.000000 0.000000
t = 0
0 0.000000 1.000000
t = 1
0 1.000000 1.000000
t = 2
0 1.000000 2.000000
t = 3
0 2.000000 2.000000
t = 4
0 2.000000 3.000000
t = 5
0 3.000000 3.000000
t = 6
0 3.000000 4.000000
t = 7
0 4.000000 4.000000
```

```
t = 8
0 4.000000 5.000000
t = 9
0 5.000000 5.000000
```

5 シンプルなエージェントシュミレーション gsa0.c

gnuplot へ出力します。

```
0001: /*
0002:   gsa0.c
0003:   シンプルなエージェントシミュレーション
0004:   ２次元平面内で動作するエージェント
0005:   gnuplot で出力します。
0006: */
0007:
0008: #include <stdio.h>
0009: #include <stdlib.h>
0010: #include <math.h>
0011: #include <unistd.h>
0012:
0013: #define BUFSIZE 256;
0014: #define N 1
0015: #define N_OF_A 2
0016: #define TIMELIMIT 100
0017:
0018: #define TMPFILE "tempfile.tmp"
0019: #define GNUPLOT "c:\\gnuplot\\binary\\gnuplot"
0020: #define RANGE 100
0021:
0022: struct coordinate{
0023:   double x;
0024:   double y;
0025: };
0026:
0027: struct agent{
0028:   int category;
0029:   struct coordinate coord;
0030:   double dattribite[N_OF_A];
0031:   int iattribute[N_OF_A];
0032: };
0033:
0034: void calcnext(struct agent a[]);
0035: void fputstate(struct agent a[], int t);
0036: void cat0(struct agent *cat0agent);
0037: void reverse(int *i);
0038:
0039: int main()
```

```
0040: {
0041:   struct agent a[N] = {0};
0042:   int    t;
0043:   FILE *pipe;
0044:
0045:   if((pipe = popen(GNUPLOT " -persist", "w")) == NULL){
0046:     fprintf(stderr, "パイプか開けません。\n");
0047:     exit(1);
0048:   }
0049:
0050:   fprintf(pipe, "set xrange [-%d:%d]\n", RANGE, RANGE);
0051:   fprintf(pipe, "set yrange [-%d:%d]\n", RANGE, RANGE);
0052:
0053:   a[0].iattribute[0] = 1;
0054:
0055:   for(t = 0; t <= TIMELIMIT; ++t){
0056:     printf("t = %d\n", t);
0057:     calcnext(a);
0058:     fputstate(a, t);
0059:     fprintf(pipe,
0060:             "plot \"" TMPFILE "\" with points pt 4\n");
0061:     fflush(pipe);
0062:     usleep(100000);
0063:   }
0064:   getchar();
0065:   return 0;
0066: }
0067:
0068: void calcnext(struct agent a[])
0069: {
0070:   int i;
0071:   for(i = 0; i < N; ++i){
0072:     if(a[i].category == 0){
0073:       cat0(&a[i]);
0074:     }else{
0075:       fprintf(stderr, "ERROR カテゴリがありません(%d)\n", i);
0076:     }
0077:   }
0078: }
0079:
0080: void cat0(struct agent *cat0agent)
0081: {
0082:   reverse(&((*cat0agent).iattribute[0]));
0083:   reverse(&((*cat0agent).iattribute[1]));
0084:
0085:   (*cat0agent).coord.x += (*cat0agent).iattribute[0];
0086:   (*cat0agent).coord.y += (*cat0agent).iattribute[1];
0087: }
0088:
```

```
0089: void reverse(int *i)
0090: {
0091:   if(*i == 0){
0092:     *i = 1;
0093:   }else{
0094:     *i = 0;
0095:   }
0096: }
0097:
0098: void fputstate(struct agent a[], int t)
0099: {
0100:   int i;
0101:   FILE *fp;
0102:
0103:   if((fp = fopen(TMPFILE, "w")) == NULL){
0104:     fprintf(stderr, "一時ファイルが開けません\n");
0105:     exit(1);
0106:   }
0107:
0108:   for(i = 0; i < N; ++i){
0109:     fprintf(fp,"%lf %lf\n", a[i].coord.x,a[i].coord.y);
0110:   }
0111:   fclose(fp);
0112: }
```

6 「gsa0」の操作方法

「gsa0」の操作方法

```
./gsa0
```

上記の「./」は、Linux でのプログラムを実行する時の例です。
ウィンドウズでは、gsa0 「Enter」と入力してください。

7 「gsa0」の実行結果

別紙を参照してください。