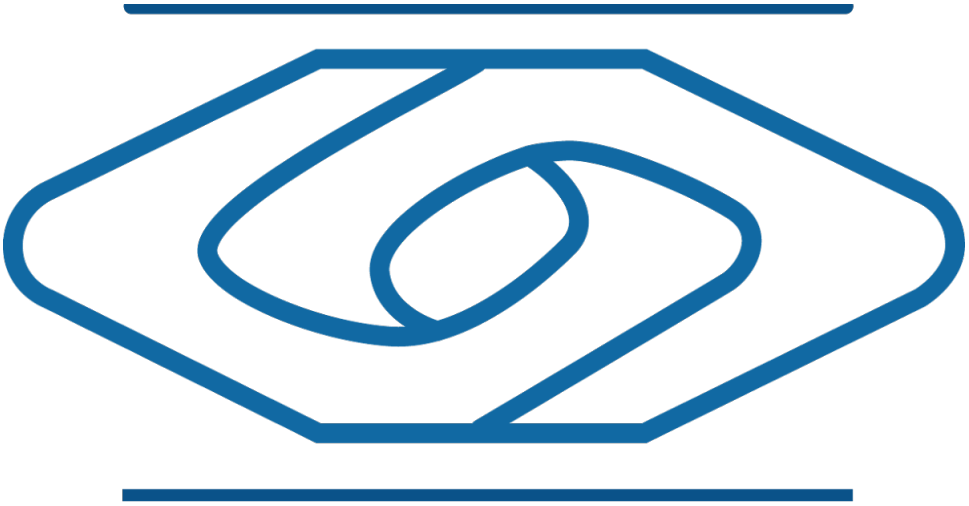

Instituto Politecnico Nacional

Escuela Superior de Computo

Langton's Ant

Submited to:Genaro Juárez Martínez
Submited By: Meza Madrid Raúl Damián
For the class: Computer Selected Topics: Complex Systems

Contents

1 Introduction

Along with the Game of life, Langto's ant is a cellular automaton; more specifically a two dimensional Turing machine. Compared to Game of life, its a simpler version since the rule is only applied to the cells where the ant has been. Since any two dimensional array can be transformed to a one dimensional array, we could say that Langton's Ant is a Turing Machine where the ant is the head of the Turing machine and the space is the tape. The Ant moves to a specific location depending on the input and it can write on the tape.

2 Development

All the source Code for this and other projects for this class can be found on: https://github.com/asdf1234Damian/ComputerSelectedTopics.

2.1 Automata.h

```
 #include <SFML/Graphics.hpp>
#include <vector>
#include <iostream>
#include <ctime>
#include <fstream>
#include <chrono>
#include <random>

class Automata{
public:
  unsigned int size;
  double p;
  Automata(unsigned int,double,short int,short int,short int,short
   ↪  int,short int,short int);
  sf::Texture antUp;
  sf::Texture antRight;
  sf::Texture antLeft;
  sf::Texture antDown;
  sf::Color cleanC= sf::Color(0,0,0);
  sf::Color pheromC= sf::Color(250, 250,250);
  int zoom=1,viewx=0,viewy=0,totalAnts=0, totalPher=0, gen=1;
```

```
21    void run();
22  private:
23    void addAnt(int, int);
24    int getDir(sf::Sprite);
25    void eraseAnt(int,int);
26    void rotateAnt(int, int,int);
27    void rotateAnt(sf::Sprite);
28    sf::Vector2f moveAhead(sf::Sprite);
29    bool standing(sf::Sprite);
30    bool antExist(int, int);
31    void flipCell(int x, int y);
32    void updateView();
33    void randomStart();
34    void update();
35    sf::RenderWindow grid;
36    sf::View view;
37    bool running=false;
38    bool state=true;//True sets current state to A,false to B
39    std::vector<sf::RectangleShape> cells;
40    std::vector<sf::Sprite> ants;
41    void pollEvent();
42    bool getValue(float, float);
43    bool rule(float, float);
44    size_t getIndex(float , float );
45    void setCell(float , float );
46  };
```

2.2 Automata.cpp

```
1  #include "Automata.h"
2
3  std::default_random_engine
4  generator
    ↪  (std::chrono::system_clock::now().time_since_epoch().count());
5  std::normal_distribution<double> distribution(0.0,1.0);
6
7  Automata::Automata(unsigned int size, double p,short int cr1,short
    ↪  int cr2,short int cg1,short int cg2,short int cb1,short int
    ↪  cb2):grid({1000,1000},"Good old Lang's Ant"),cells(size*size){
```

```
8   grid.setPosition(sf::Vector2i(200,100));
9   view.reset(sf::FloatRect(viewx, viewy, size/zoom, size/zoom ));
10  this-> pheromC= sf::Color(cr1,cg1,cb1);
11  this-> cleanC= sf::Color(cr2,cg2,cb2);
12  this->size=size;
13  this->p=p;
14  this->antUp.loadFromFile("images/03.png");
15  this->antRight.loadFromFile("images/02.png");
16  this->antDown.loadFromFile("images/01.png");
17  this->antLeft.loadFromFile("images/04.png");
18  grid.setFramerateLimit(30);
19  grid.setView(view);
20 }
21 //Ant Functions////////////////////////////////////////////
22 void Automata::addAnt(int x, int y){
23  sf::Sprite ant =sf::Sprite();
24  ant.setTexture(antUp);
25  ant.setScale(.003, .003);
26  ant.setPosition((float) x, (float) y);
27  ants.push_back(ant);
28 }
29
30 void Automata::eraseAnt(int x, int y){
31  for (size_t i = 0; i < ants.size(); i++) {
32    if (ants[i].getPosition()==sf::Vector2f(x,y)) {
33      ants.erase(ants.begin()+i);
34    }
35  }
36 }
37
38 void Automata::rotateAnt(int x, int y, int angle){
39  for (size_t i = 0; i < ants.size(); i++) {
40    if (ants[i].getPosition()==sf::Vector2f(x,y)) {
41      if (angle==1) {
42        switch (getDir(ants[i])) {
43          case 1:
44          ants[i].setTexture(antRight);
45          break;
46
47          case 2:
```

```
48            ants[i].setTexture(antDown);
49            break;
50
51            case 3:
52            ants[i].setTexture(antLeft);
53            break;
54
55            case 4:
56            ants[i].setTexture(antUp);
57            break;
58          }
59        }else{
60          switch (getDir(ants[i])) {
61            case 1:
62            ants[i].setTexture(antLeft);
63            break;
64
65            case 2:
66            ants[i].setTexture(antUp);
67            break;
68
69            case 3:
70            ants[i].setTexture(antRight);
71            break;
72
73            case 4:
74            ants[i].setTexture(antDown);
75            break;
76          }
77        }
78      }
79    }
80  }
81
82  void Automata::rotateAnt(sf::Sprite ant){
83    switch (getDir(ant)) {
84      case 1:
85      ant.setTexture(antRight);
86      break;
87
```

```
88      case 2:
89      ant.setTexture(antDown);
90      break;
91
92      case 3:
93      ant.setTexture(antLeft);
94      break;
95
96      case 4:
97      ant.setTexture(antUp);
98      break;
99    }
100 }
101
102 sf::Vector2f Automata::moveAhead(sf::Sprite ant){
103   float x=ant.getPosition().x;
104   float y=ant.getPosition().y;
105   switch (getDir(ant)) {
106     case 1:
107     y++;
108     break;
109
110     case 2:
111     x++;
112     break;
113
114     case 3:
115     y--;
116     break;
117
118     case 4:
119     x--;
120     break;
121
122   }
123   if (x<0) {x=size-1;}else if (x>=size) {x=0;}
124   if (y<0) {y=size-1;}else if (y>=size) {y=0;}
125   return sf::Vector2f(x,y);
126 }
127
```

```
128 //Aux Functions//////////////////////////////////////////////
129
130 bool Automata::antExist(int x, int y){
131   for (size_t i = 0; i <ants.size(); i++) {
132     if (ants[i].getPosition().x==x && ants[i].getPosition().y==y){
133       return 1;
134     }
135   }
136   return 0;
137 }
138
139 int Automata::getDir(sf::Sprite ant){
140   if (ant.getTexture()==&antUp)
141   return 1;
142   if (ant.getTexture()==&antRight)
143   return 2;
144   if (ant.getTexture()==&antDown)
145   return 3;
146   if (ant.getTexture()==&antLeft)
147   return 4;
148   return 0;
149 }
150
151 bool Automata::standing(sf::Sprite ant){
152   return getValue(ant.getPosition().x,ant.getPosition().y);
153 }
154
155 size_t Automata::getIndex(float x, float y){
156   return (y*size + x);
157 }
158
159 bool Automata::getValue(float x, float y){
160   if (x<0) {x=size-1;}else if (x>=size) {x=0;}
161   if (y<0) {y=size-1;}else if (y>=size) {y=0;}
162   if (cells[getIndex(x,y)].getFillColor()==pheromC) {
163     return true;
164   }else{
165     return false;
166   }
167 }
```

```
168
169 void Automata::flipCell(int x, int y){
170   if (getValue(x,y)!=1) {
171     totalPher++;
172     cells[y*size + x].setFillColor(pheromC);//white
173   }else{
174     totalPher--;
175     cells[y*size + x].setFillColor(cleanC);//black
176   }
177 }
178
179 //Init  Functions////////////////////////////////////////
180
181 void Automata::randomStart(){
182   for (size_t x = 0; x < size; x++){
183     for (size_t y = 0; y < size; y++) {
184       cells[y*size + x].setPosition({(float)x,(float)y});
185       cells[y*size +
          ↪  x].setSize({(float)1000/size,(float)1000/size});
186       cells[y*size + x].setFillColor(cleanC);
187       if (distribution(generator)<p) {
188         totalAnts++;
189         addAnt(x, y);
190       }
191     }
192   }
193   std::cout <<gen<<", "<< totalPher<< '\n';
194 }
195
196 void Automata::run(){
197   randomStart();
198   while (grid.isOpen()) {
199     //grid.clear(cleanC);
200     for (size_t i = 0; i < size*size; i++) {
201       grid.draw(cells[i]);
202     }
203     for (size_t i = 0; i < ants.size(); i++) {
204       grid.draw(ants[i]);
205     }
206     grid.display();
```

```
207     pollEvent();
208   }
209 }
210
211 void Automata::update(){
212   float x,y;
213   for (size_t i = 0; i < ants.size(); i++) {
214     x=ants[i].getPosition().x;
215     y=ants[i].getPosition().y;
216     if(standing(ants[i])){
217       rotateAnt(x, y, 1);
218     }else{
219       rotateAnt(x, y, 3);
220     }
221     flipCell(x,y);
222     if (!antExist(moveAhead(ants[i]).x,moveAhead(ants[i]).y)){
223       ants[i].setPosition(moveAhead(ants[i]));
224     }
225   }
226   gen++;
227   std::cout <<gen<<", "<<totalPher<< '\n';
228   state=!state;
229 }
230
231 void Automata::updateView(){
232   view.reset(sf::FloatRect(viewx, viewy, size/zoom, size/zoom ));
233   grid.setView(view);
234 }
235
236 void Automata::pollEvent(){
237   sf::Event e;
238   if (running) {
239     update();
240   }
241   while (grid.pollEvent(e)) {
242     if (e.type== sf::Event::Closed) {
243       grid.close();
244     }else if (e.type == sf::Event::MouseButtonPressed){
245       int x=(e.mouseButton.x*size) / (zoom*grid.getSize().x);
246       int y=(e.mouseButton.y*size) / (zoom*grid.getSize().y);
```

```
247
248       if (e.mouseButton.button == sf::Mouse::Left){
249         if (antExist(x,y)) {
250           eraseAnt(x,y);
251         }else{
252           addAnt(x,y);
253         }
254       }else if(e.mouseButton.button == sf::Mouse::Right){
255         rotateAnt(x,y,90);
256       }
257
258     }else if(e.type==sf::Event::KeyPressed){
259       switch (e.key.code) {
260         case sf::Keyboard::Right:
261         update();
262         break;
263
264         case sf::Keyboard::Up:
265         zoom++;
266         updateView();
267         break;
268
269         case sf::Keyboard::Down:
270         if (zoom>1) {
271           zoom--;
272           updateView();
273         }
274         break;
275
276         case sf::Keyboard::A:
277         if(viewx>0){
278           viewx-=size/(zoom*10);
279           updateView();
280         }
281         break;
282
283         case sf::Keyboard::D:
284         if((unsigned)viewx<=size-size/zoom){
285           viewx+=size/(zoom*10);
286           updateView();
```

```
287           }
288           break;
289
290           case sf::Keyboard::S:
291           if((unsigned)viewy<size-size/zoom){
292             viewy+=size/(zoom*10);
293             updateView();
294           }
295           break;
296
297           case sf::Keyboard::W:
298           if(viewy>0){
299             viewy-=size/(zoom*10);
300             updateView();
301           }
302           break;
303
304           case sf::Keyboard::Left:
305           running = !running;
306           break;
307
308           default:
309           break;
310         }
311       }
312     }
313   }
314
315   //Main Program////////////////////////////////////////////
316
317   int main(int argc, char const *argv[]) {
318     freopen("gens","w+",stdout);
319     //Size
320     unsigned int size=strtoul(argv[1], NULL,10);
321     //Probability
322     double p=strtod(argv[2], NULL);
323     //Colors
```

```
324   short int cr1=strtoul(argv[3], NULL,10),cg1=strtoul(argv[4],
      ↪  NULL,10),cb1=strtoul(argv[5], NULL,10),cr2=strtoul(argv[6],
      ↪  NULL,10),cg2=strtoul(argv[7], NULL,10),cb2=strtoul(argv[8],
      ↪  NULL,10);
325   //Initialization
326   Automata GOL(size,p,cr1,cr2,cg1,cg2,cb1,cb2);
327   GOL.run();
328   return 0;
329 }
```