

AI Assignment

by Ethan Randell

FILE	4154_ETHAN_RANDELL_AI_ASSIGNMENT_177048_479415185.DOCX (130.95K)		
TIME SUBMITTED	22-MAR-2019 07:42PM (UTC+0000)	WORD COUNT	2229
SUBMISSION ID	103043490	CHARACTER COUNT	12030

Artificial Intelligence & Machine Learning Assignment 1

Task 1

This section will be talking about the theory of influence maps and which methods that can be used to implement these influence maps.

The first method that will be analysed first will be Convolution filters here is a brief overview of what convolution filters are, Convolution is a mathematical operation like adding or subtracting. They are used in image blur algorithms is a very popular way to update analyses that spreads values from their source outwards. Influence maps have a particular characteristic and so do other proximity measures. Terrain analyses can benefit from this sometimes but they don't really need the spreading-out characteristic[1].

There are algorithms similar to this that're used outside of video games. The algorithms are also used in physics to try and simulate the behaviours of different kinds of different fields and to form basis models of heat around physical components [1].

The blurring effect inside image editing packages is part of the family of convolution filters, convolution filters have many other names kernel filters, impulse response filters, finite element simulations etc. but the type of name depends on what type of field you're in whether it'd be games development or science [1].

Example of the convolution algorithm

One of these convolution filters that will be analysed will be the Gaussian Filter, blur filters are part of the Gaussian filter family how the work is that they blur up the values that're given to them then spread around and because of this they're brilliant for spreading influence around an influence map.

There can be different size filters but there is only one Gaussian filter, to find the values for the matrix can be done by taking two different vectors which are made up of different numbers from the binomial series [1].

The first few values of the binomial series are

[1, 2, 1]

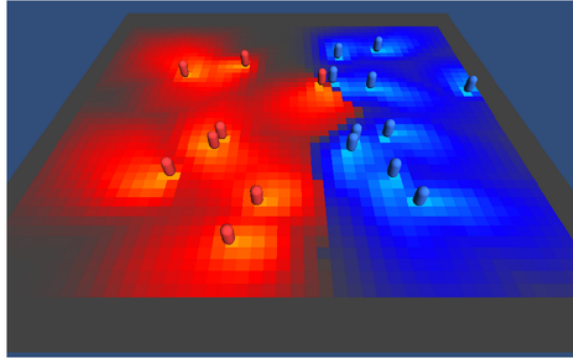
[1, 4, 6, 4, 1]

Then multiply the values to find the filter so for a Gaussian filter of size two we would get

$$\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}.$$

The matrix can be used but the values will increase each time the map is being used so to keep these values at a nice level add up all the values in the matrix then divide that number by the matrix for the matrix above its 256.

$$M = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}.$$



In the image above it shows Gaussian blur being used in an influence map
<https://forum.unity.com/threads/ai-influence-maps.145368/>

The next method will be the boundaries method which mainly focuses more on the edges of the map when using this method, we can no longer use the same matrix like Gaussian and convolution due to the fact that some of the “neighbours for the edge tile do not exist” [1].

There are two ways to rectify this problem one to modify the matrix or two to modify the map itself. One of the ways to modify the matrix is to modify its edges so it can only include the neighbours which exist.

$$M = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

So this is a basic convolution filter so when boundaries are being implemented users would choose either the corner values of the matrix or either the first six values. When using the boundaries method, it can give the best results but the drawback with changing the matrix is that its working with multiple different matrices and then switching between every one of them at the right time.

$$\frac{1}{9} \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix}$$

$$\frac{1}{12} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \end{bmatrix}$$

The next method that may be used is the influence equation first you need to find out where the object in this case a dot for example will be on a grid so to do this you need the dots X and Y coordinates so for this example the X coordinate will be -2 and the Y coordinate will be 2 the next step will be to do a distance check between the two coordinates.

$$Distance = \sqrt{(-2)^2 + 2^2} = 2.82$$

Now that the distance check has been calculated the influence of the dot can now be calculated.

$$Id = \frac{Io}{(e^{(d-r)} + 1)}$$

This is what the equation normally for the example below Io or also known as location of influence source will be 3.5 for example and the r which is radius will be 6. The e in the equation is the exponential function on a calculator.

$$Id = \frac{3.5}{(e^{(2.82-6)} + 1)}$$

The end result of this equation will be 3.36.

Big O Notation

So the big O notation or otherwise known as O(N) is used to show how long it would take for the algorithm to execute and how well it would perform when the algorithm has a massive amount of input data [2].

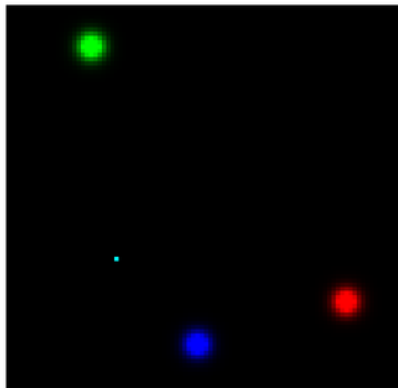
Testing of code

Distance check	<pre>// Distance check for red dot d = sqrt(((check.infx - check.x)*(check.infx - check.x)) + ((check.infy - check.y)*(check.infy - check.y)));</pre> in this screenshot shows the code for the distance check for the red dot so infx and infy are the two coordinates that have been given. <pre> > check {x=1.00000000 y=1.00000000 infx=85.0000000 ...} check.infx 85.0000000 check.infy 20.0000000 </pre> In the screen shot above it shown the distance equation doing the calculations. <pre>d 86.1220093</pre> This is the result of the distance equation
Influence equation	<pre>ID = (check.infmain / (exp(d - check.r) + 1));</pre> In the screenshot above it shows the influence equation but in code form so check.infmain will be the location of influence which in the program is 2 and the check.r is the radius which is set to 2 as well. As a result of this equation the ID will be: <pre>ID 5.85201655e-37</pre>
AI Bot	<pre>if (bot.x == i&& bot.y == j) { glColor3f(0, 1, 1); }</pre> This bit of code is the code that will generate the little ai bot at the x and y coordinates that has been given which is (30,30) and the glColor gives a colour value to the ai bot which in this case it's a turquoise blue.

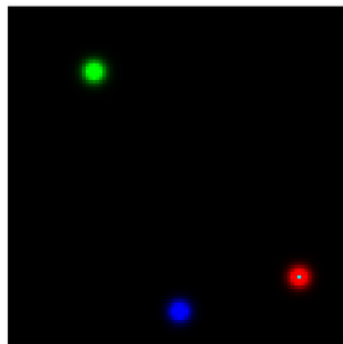
AI bot moving
towards a
certain
influence (red
dot)

```
void whereCanIgo() {  
    rgb lookUp, lookDown, lookLeft, lookRight;  
    lookUp = grid[bot.x][bot.y + 1];  
    lookDown = grid[bot.x][bot.y - 1];  
    lookLeft = grid[bot.x + 1][bot.y];  
    lookRight = grid[bot.x - 1][bot.y];  
    //4 tests  
    if (lookUp.r >= lookDown.r && lookUp.r >= lookLeft.r && lookUp.r >= lookRight.r) // allows the AI bot to move up  
    {  
        bot.y += 1;  
    }  
}
```

In the screenshot above this code helps the AI bot move towards a particular influence which is for this example the red influence so if the bot wanted to look up the bot will go up the y axis by 1, if it wanted to go down it would -1 from the y axis. In the if statements its saying if the lookup. r value is greater than the other three values then the ai bot will move up closer to the red influence.



In the screenshot above shows the red, blue and green influence on the map and also shows the little ai bot as well.



In the screenshot above it shows the ai bot going to the red influence.

References

- [1] J. F. Ian Millington, Artificial Intelligence for Games, Morgan Kaufman Publishers , 2009.
- [2] R. Bell, "A beginners guide to Big O Notation," 23 June 2009. [Online]. Available: <https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/>. [Accessed 22 March 2019].

Appendix

```
//
*****

/* Influence map example using convolution with Gaussian filter to spread out
influence.
//define GRIDSIZE is used to set map size - here its 32 by 32.
//Press space bar to trigger a pass of the filter over the map.
//Some messy techniques here!- be careful.
//*****
*****

///// Includes
#include <stdio.h>
#include <stdlib.h>
#include "glut.h"
#include <iostream>
#include <fstream>
#include <string>
#include <cmath>
#include <math.h>
using namespace std;
#define GRIDSIZE 100

///// Global Variables
typedef struct rgbtag {
    float r;
    float g;
    float b;
}rgb;

struct discheck {
    float x; // position 1
    float y; // position 2

    float infx; // influence
    float infy; // influence
    float r; // radius
    float infmain;
};
discheck check; // distance check structure for red dot
discheck check2; // distance check structure for blue dot
discheck check3; // distance check structure for green dot
struct AI
{
```

```

        int x;
        int y;
};
AI bot;

float d; // distance calculation
float ID; // influence calculation

rgb grid[GRIDSIZE][GRIDSIZE];
rgb gridtemp[GRIDSIZE*GRIDSIZE];
//function prototypes
void init(void);
void display(void);
void drawBox(float w, float h, float l);
void reshape(int w, int h);
void WhereCanIGo();

void init()
// Initialize
//*****
// desc: initializes OpenGL
{
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);          // clears colour buffer
    glEnable(GL_DEPTH_TEST);                       // hidden surface
removal
    glEnable(GL_CULL_FACE);                         // do not
calculate inside of poly's
    glFrontFace(GL_CCW);                           // counter clock-wise
polygons are out

    for (int i = 0; i < GRIDSIZE; i++) {
        for (int j = 0; j < GRIDSIZE; j++) {

            grid[i][j].r = 0;
            grid[i][j].g = 0;
            grid[i][j].b = 0;

        }
    }

    // values for red dot
    check.x = 0;
    check.y = 0;
    check.infx = 85;
    check.infy = 20;
    check.r = 2;
    check.infmain = 2;

    // values for blue dot
    check2.infx = 50;
    check2.infy = 10;
    check2.r = 2;
    check2.infmain = 2;

    // values for green dot
    check3.infx = 25;
    check3.infy = 80;
    check3.r = 2;
    check3.infmain = 2;

    // position values for the AI bot
    bot.x = 30;
    bot.y = 30;

```

```

}

void drawBox(float w, float h, float l)
//drawBox
//*****
//This function first translates a further x,y,z units then draws a box of
//width w, height h, length l
{
    glPushMatrix();          //save modelview
    w = w / 2.0; h = h / 2.0; l = l / 2.0; //adjust values so centre is in middle
of box
                                //draw faces
of box
    glBegin(GL_POLYGON);
    glVertex3f(w, l, h);
    glVertex3f(w, l, -h);
    glVertex3f(-w, l, -h);
    glVertex3f(-w, l, h);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(w, l, h);
    glVertex3f(-w, l, h);
    glVertex3f(-w, -l, h);
    glVertex3f(w, -l, h);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(w, l, h);
    glVertex3f(w, -l, h);
    glVertex3f(w, -l, -h);
    glVertex3f(w, l, -h);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(-w, l, h);
    glVertex3f(-w, l, -h);
    glVertex3f(-w, -l, -h);
    glVertex3f(-w, -l, h);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(-w, -l, h);
    glVertex3f(-w, -l, -h);
    glVertex3f(w, -l, -h);
    glVertex3f(w, -l, h);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(w, -l, -h);
    glVertex3f(-w, -l, -h);
    glVertex3f(-w, l, -h);
    glVertex3f(w, l, -h);
    glEnd();
    glPopMatrix(); //restore previous modelview matrix so leaving things as you
found them
}

// display
//*****
// desc: handles drawing of scene
void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // clear screen
and depth buffer

```

```

glLoadIdentity();
// reset modelview matrix

gluLookAt(0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0); //reset camera position
glColor3f(0, 0, 0); //set colour
glTranslatef(0.5, 0.5, 0.5); //move by x,y and z units
//tests grid by applying
greyscale values to each position
//
for (int i = 0; i < GRIDSIZE; i++) {
    for (int j = 0; j < GRIDSIZE; j++) {
        if (grid[i][j].r > grid[i][j].b && grid[i][j].r > grid[i][j].g)
        {
            glColor3f(grid[i][j].r, 0, 0);
        }
        else if (grid[i][j].b > grid[i][j].r && grid[i][j].b > grid[i][j].g)
        {
            glColor3f(0, 0, grid[i][j].b); //set colour
        }
        else
        {
            glColor3f(0, grid[i][j].g, 0);
        }
        if (bot.x == i && bot.y == j)
        {
            glColor3f(0, 1, 1);
        }
        glPushMatrix(); //save current position/orientation, etc.
        glTranslatef(i, j, 0);
        drawBox(1, 1, 1); //draw
        glPopMatrix(); //restore previous "current" position/orientation
etc.
    }
}

glFlush(); //force drawing

glutSwapBuffers(); // bring backbuffer to foreground
//any errors then display error
codes
GLenum errCode;
const GLubyte *errString;
if ((errCode = glGetError()) != GL_NO_ERROR)
{
    errString = gluErrorString(errCode);
    fprintf(stderr, "OpenGL error : %s\n", errString);
}
}
void keyboardread(unsigned char key, int x, int y)
{
    switch (key) {
        case 'u':
            break;
        case ' ':
            for (int i = 1; i < GRIDSIZE - 1; i++) {
                for (int j = 1; j < GRIDSIZE - 1; j++) {

```

```

        check.x = i;
        check.y = j;

        // Distance check for red dot
        d = sqrt(((check.infx - check.x)*(check.infx - check.x)) +
        ((check.infy - check.y)*(check.infy - check.y)));
        ID = (check.infmain / (exp(d - check.r) + 1));
        grid[i][j].r = ID;

        //Distance check for blue dot
        d = sqrt(((check2.infx - check.x)*(check2.infx - check.x))
        + ((check2.infy - check.y)*(check2.infy - check.y)));
        ID = (check2.infmain / (exp(d - check2.r) + 1));
        grid[i][j].b = ID;

        //Distance check for green dot
        d = sqrt(((check3.infx - check.x)*(check3.infx - check.x))
        + ((check3.infy - check.y)*(check3.infy - check.y)));
        ID = (check3.infmain / (exp(d - check3.r) + 1));
        grid[i][j].g = ID;
    } //convolution calc. very messy tidy up using matrix mult.
}
//for (int i = 0; i < GRIDSIZE*GRIDSIZE; i++) //copy calculated array into
original
    // grid[i] = gridtemp[i];
    ///quicker to swap pointers to arrays - kinda double buffering
    ///2 influence points set here
    //grid[GRIDSIZE / 2 + (GRIDSIZE*GRIDSIZE / 2)].r = 1; //use this if you
want to keep inf at level
    //
    //1 in original unit else unit's influence value is decreased at
each pass.
    //grid[GRIDSIZE / 2 - 5 + (GRIDSIZE*(GRIDSIZE / 2 + 1))].b = 1;

    WhereCanIGo();
    glutPostRedisplay();
    break;
}
}

void reshape(int w, int h)
//reshape
//*****
//OpenGL reshape callback is called when the window is resized
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h); //respecify size of view port to
match new w and h
    glMatrixMode(GL_PROJECTION); //go into projection mode
    glLoadIdentity(); //reset projection matrix
    glOrtho(0.0, GRIDSIZE, 0.0, GRIDSIZE, 0.0, 100.0); //defines orthoganal
projection (parrallel

    // viewing volume

    // window limits of display

    // left =0.0, right =2.0,

    // bottom = 0.0, top = 2.0

    // near =0.0, far = 100.0

```

```

        //gluPerspective(60.0,(GLfloat)w/(GLfloat)h,1.0,200.0); //redefine
projection matrix to match changes in w and h
        glMatrixMode(GL_MODELVIEW); //switch back to modelview mode
        glLoadIdentity(); //reset modelview matrix
        gluLookAt(0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0); //reset camera position
    }

    void WhereCanIGo() {

        rgb lookUp, lookDown, lookLeft, lookRight;
        lookUp = grid[bot.x][bot.y + 1];
        lookDown = grid[bot.x][bot.y - 1];
        lookLeft = grid[bot.x + 1][bot.y];
        lookRight = grid[bot.x - 1][bot.y];
        //4 tests
        if (lookUp.r >= lookDown.r && lookUp.r >= lookLeft.r && lookUp.r >= lookRight.r)
        // allows the AI bot to move up
        {

            bot.y += 1;

        }

        if (lookDown.r >= lookUp.r && lookDown.r >= lookLeft.r && lookDown.r >=
lookRight.r) // allows the AI bot to move down
        {

            bot.y -= 1;

        }

        if (lookLeft.r >= lookUp.r && lookLeft.r >= lookDown.r && lookLeft.r >=
lookRight.r) // allows the AI bot to move left
        {

            bot.x += 1;

        }

        if (lookRight.r >= lookUp.r && lookRight.r >= lookDown.r && lookRight.r >=
lookLeft.r) // allows the AI bot to move right
        {

            bot.x -= 1;

        }
    } // this function is from the AI assignment from last year
http://moodlearchive.uwtsd.ac.uk/moodle1718/mod/assign/view.php?id=110676

    int main(int argc, char** argv)
    {
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(100, 100);
        glutCreateWindow(argv[0]);
        init();
    }

```

```
    glutDisplayFunc(display);  
    glutReshapeFunc(reshape);  
    glutKeyboardFunc(keyboardread);  
    glutMainLoop();  
    return 0;  
}
```

AI Assignment

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

51 /100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

GRADING FORM: SMU GENERIC_GJ

ETHAN RANDELL

50.50

RESEARCH-INFORMED LITERATURE



References a little light! Key areas identified and reviewed.

5.5

KNOWLEDGE AND UNDERSTANDING



Algorithms for calculating influence maps discussed and evidence of an essential understanding of relevant concepts.

11

ANALYSIS



You need to sum both influences in your 2nd (eqn 1) calculation Your convolution calculation is not attempted though you have provided examples from the literature. Test schedule designed and implemented. Complexity and Big O notation referenced but you have not analysed the complexity of the algorithms.

9.5

PRACTICAL APPLICATION AND DEP.



21/40 Evidence of code testing. Algorithms coded and influence calculated. You haven't calculated influence at all points though. No influence map loaded. Sophisticated AI bot behaviours not in evidence.

22.5

SKILLS FOR PROFESSIONAL PRAC.

2/5 Only two refs one of which is electronic.



Structure and presentation of report is however good.

2