Artificial Drawing

Alejandro Lopez-Rincon[1], Evelyne Lutton[2], Pierre Collet[3]

http://easea.unistra.fr/easea/index.php/EASEA_platform

e-mail: alejandro.lopez@iscpif.fr[1], evelyne.lutton@grignon.inra.fr[2], pierre.collet@unistra.fr[3]

Introduction

Art therapy has been proven to improve the quality of life in cases at nursing homes and suffering from dementia. It allows to improve the capacity of the senses and improves the independence of the patients [1]. Better still, in the absence of a cure in the foreseeable future for dementia patients, art therapy has been proven to be a tool to evaluate and improve communication with the patients [2]. For example, in [3,4] the authors made an application to use mandala drawings as a tool to measure the severity of psychological disorder. Art therapy has been proven useful for patients suffering from Alzheimer disease. Improving the mood of both patients and caregivers [5]. There are several associations working along the subject, in France ARTZ[6].

Using the available web resources, such as WebGL and cloud resources we want to create a web-based service capable of giving interactivity and functioning for art therapy. To achieve this goal we want to make an interactive interface where the subject can "play" with a series of video sequences as an interactive movie. The particles are moved by the subject at will. Similar interfaces exist for children with motor and mental disadvantages as the ones sold by AZ toyz[7].

This application was created from the idea of trying to imitate how human beings draw [8]. The basic idea is we take a video sequence, separate the frames (into images) and then, convert each frame into particles that can be manipulated. The particles are actually lines with an initial point, and ending point arranged to give the impression of reproducing the original image. Similar work can be seen in the project ArtiE-Fract[9,10].

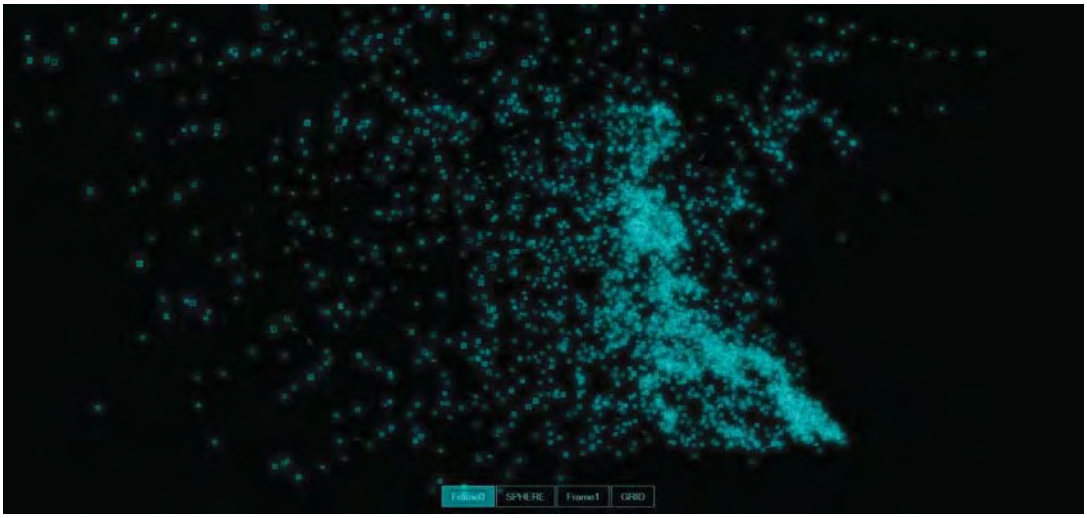


Figure 1 Example of web based application.

Why is it important to try to approximate computer graphics to emulate hand-made drawings? As mentioned in [11]; with great rendered computer graphics we expect perfection. Whereas from simple drawings we communicate easier abstract ideas. In this work we want to mimic how humans draw. For example, if we take into consideration the following hand-made drawing (Fig. 2) we can see this drawing is made from a series of

pencil strokes as seen in the detailed view from Fig. 3.

Figure 2 Hand-made drawing.

Figure 3 Detail of the drawing.

From Fig. 3 we can see that a line is composed by a series of lines. There are similar works trying to imitate real pencil strokes. For example in the work from [12] they made a similar work by trying to approximate the lines made by human using an algorithm to stylize the images. Other works [13] use a given library and they retrieve the pencil strokes to try to emulate the drawing.

Once we have the list of positions for the particles, we can manipulate them, "play" with them and the particles will return to the original position. A similar interactive presentation was presented in the Van Gogh Museum for the painting Van Gogh's Starry Night Interactive by Petros Vrellis [14].

In this research we do not apply a mathematical rule to apply the drawing of a line based on the color of each pixel. We create an algorithm which compares an image line by line to the image we want to reproduce. For example in the Fig 3 we show how particle by particle we approximate to the original drawing.

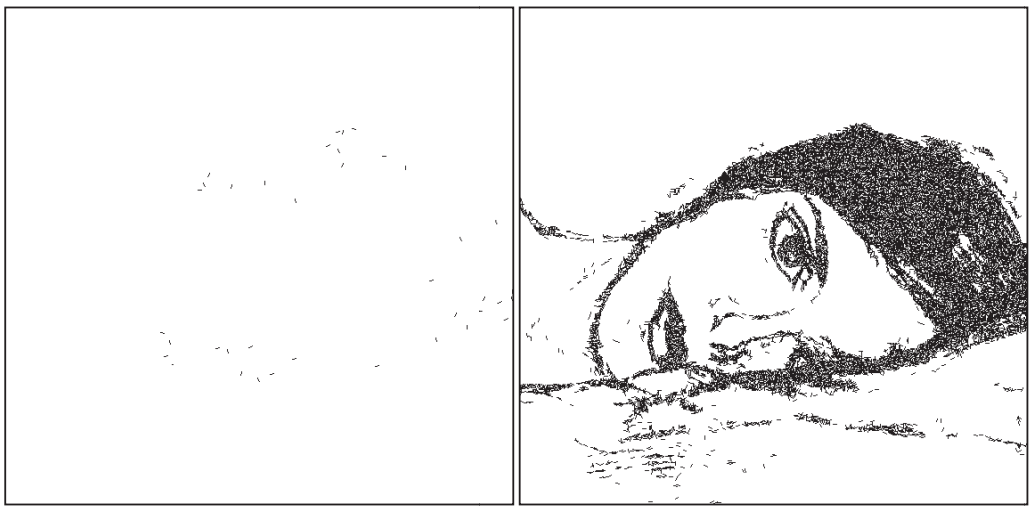


Figure 4 Drawing by 50 Particles (left) and 10,000 particles (right).

The problem we face is how to create the series of particles. For this purpose, we used random search by creating a series of images with a line in a random position, and compare it to the original to see if it was closer to the image we want to reproduce. This approach takes a lot of time, therefore we went to more complicated techniques using genetic algorithms to look for the position of the particles.

To implement the GA techniques we use the library EASEA. EASEA (EAsy Specification of Evolutionary Algorithms) is an Artificial Evolution platform that allows to implement evolutionary algorithms and to exploit the massive parallelism of many-core architectures in order to optimize real-world problems (continous, discrete, combinatorial, mixed and more (with Genetic Programming)) [15-18].

Methodology

For our research we tested 2 different methodologies of treating an image. In both applications we treated the image to only have black and white. In Fig. 5 we have the original image and in Fig. 6 the high contrast in black and white version to apply the algorithms. The first rule is to extract an outline; we do this by making an analysis of the contrast between each colour. If the pixel colour is very different to the subsequent colour, then we consider it a black pixel, in the other case a white pixel. This idea comes from the drawing technique, when we wish to extract the forms of an image or a real life.

Considering the difference in colours for the construction of images by contrast is not something new, and does not requires a complex algorithm. Next, we transform the image into 3 matrixes of the size of the image.

Figure 5 Original Drawing

Figure 6 Black and white image.

The matrixes are for the Red, Green and Blue (RGB) value. Each matrix value is the value of the pixel in that position for the RGB component. The posible values vary from 0 to 255. For the case of black and white we will have that all of the matrixes are the same, and we will only have two possible options, for the [*x,y*] pixel we will have:

Black R[*x,y*],G[*x,y*], B[*x,y*]=0,

White R[*x,y*],G[*x,y*], B[*x,y*]=255.

In the firt approach we create a line with a fixed length (5 pixels) and a random position and random angle. To make the line we use the Bresenham algorithm as implemented in [19]. We use this algorithm because it uses integer matrix for the representation of the image which makes it easier to translate it to several languages and plataforms.

We create 100 images with different lines and compare them to the original image. For the first run the best image becomes the temporal image. To compare it to the original image we use the following cost function:

$$cost = \sum_{i=0}^{m}\sum_{j=0}^{n}(|R_o[i,j] - R_c[i,j]| + |G_o[i,j] - G_c[i,j]| + |B_o[i,j] - B_c[i,j]|) \qquad (1)$$

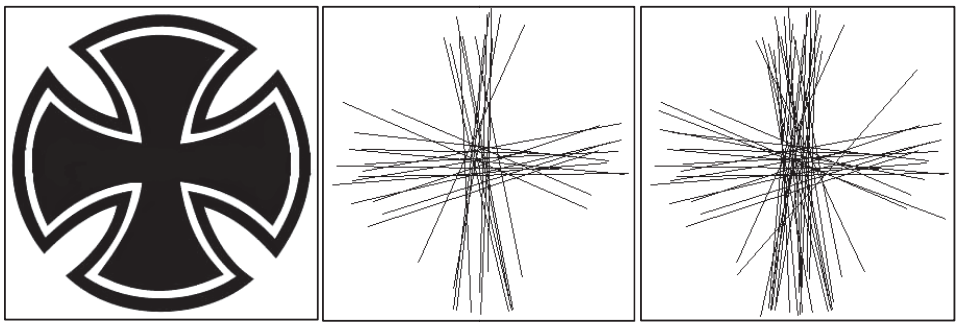
where

$$m = height\ of\ the\ image,$$
$$n = width\ of\ the\ image,$$
$$R_o, G_o, B_o = original\ image\ matrices,$$
$$R_c, G_c, B_c = calculated\ image\ matrices.$$

If any of the 100 images has a lower cost than the temporal image, then we have a new temporal image.We repeat this procedure several times. For example for the Fig. 6 we will have the following creation of particles.

Figure 7 Drawing by 50 particles (top left), 500 particles (top right), 5,000 particles (bottom left), and 10,000 particles (bottom right).

The second approach is to create the particles by choosing two random points and then make a line. For this approach we use the EASEA library and the cost function from Eq. 1. We have 100 individuals in the population, and 30 generations. The parameters to search are the beginning point *x, y* and ending point *x, y* positions.



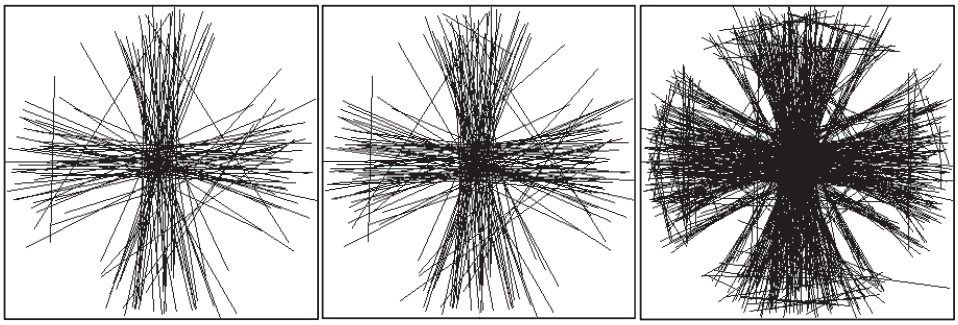


Figure 8 Reproduction of the image. Original picture (top left), 20 particles (top middle), 40 particles (top right), 100 particles (bottom left), 150 particles (bottom middle), and 300 particles (bottom right).

Applying this methodology to the Fig. 6 we get the following result, we added the original image for comparison.

Figure 9 Original Image (left) , and reconstruction 1,000 particles (right).

Finally, as we have the particles we use either WebGL or other visualization technique for example a C# application to move the particles. In

http://www.itsamazigh.com/Resonance/Hope/ArtificialDrawing.html

there is an application for two processed images for 1000 particles using WebGL. The particles change from one position, to the other and also predefined positions as shown in Fig. 10.

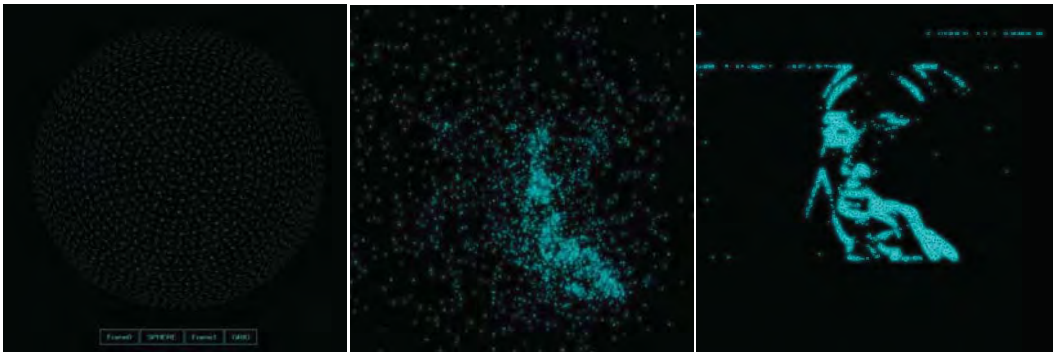


Figure 10 WebGL application for 100 particles. Sphere position (left), transition from sphere to frame (middle), and frame (right).

For these applications and both approaches, the particles are saved with the following format:

var table3 = [0,0,5,0,

822,565,827,564,
817,565,822,566,
976,592,971,594,
991,578,992,583,
986,571,981,571,
1036,612,1037,617,
1018,611,1013,610,
....

giving the beginning and ending point of each line.

Conclusion and Future Work

The tests show that creating the particle system for an image depends on the available resources and for each frame step can take several hours. With a computer with Intel(R) Core(TM) i7-2860QM CPU @ 2.50Hz and 8.00 (RAM) the construction time is around 8 hours. The time varies depending on the image.

To display the particles in a web based interface or an application is not computationally costly. The computational cost is in transforming a full video sequence into a series of particles. For example a 3 minute sequence with 30 fps (frames per second) will take around 43,200 computational hours.

To speed-up our algorithms we are going to add more non-linear constraints to the creation of particles; for example identification of 2d geometric primitves as in [20]. The next step is to start making some tests with individuals to see the application as art therapy which is our main objective.

References

1 Hannemann, Beat Ted. "Creativity with Dementia Patients." *Gerontology* 52.1 (2006): 59-65. Web.

2 Beard, R. L. (2011). Art therapies and dementia care: A systematic review. *Dementia*, 1471301211421090.

3 Kim, S. I., Betts, D. J., Kim, H. M., & Kang, H. S. (2009). Statistical models to estimate level of psychological disorder based on a computer rating system: An application to dementia using structured mandala drawings. *The Arts in Psychotherapy*, *36*(4), 214-221.

4 Kim, S. I., Kang, H. S., & Kim, Y. H. (2009). A computer system for art therapy assessment of elements in structured mandala. *The Arts in Psychotherapy*, *36*(1), 19-28.

5 Chancellor, B., Duncan, A., & Chatterjee, A. (2014). Art Therapy for Alzheimer's Disease and Other Dementias. *Journal of Alzheimer's Disease*, *39*(1), 1-11.

6 "Bonne Rentrée à Toutes Et à Tous!" *ARTZ*. N.p., n.d. Web. 10 Oct. 2014. URL: http://www.associationartz.org/

7 "Hop'Toys - Solutions Pour Enfants Exceptionnels." *Hop'Toys - Solutions Pour Enfants Exceptionnels*. N.p., n.d. Web. 14 Oct. 2014. URL: http://www.hoptoys.fr/

8 Alejandro Lopez Rincon; , " Le hasard à l'oeuvre (the random in the artwork*),"Generative Art Conference 2013*,9-12 Dec. 2013.
URL: http://www.generativeart.com/ga2013xWEB/proceedings1/5.pdf

9 Evelyne Lutton, Emmanuel Cayla, and Jonathan Chapuis. Artie-fract: The artist's viewpoint. In EvoMUSART2003, 1st European Workshop on Evolutionary Music and Art, Essex, April, 14-16 2003. LNCS, Springer Verlag.

10 Evelyne Lutton. Evolution of fractal shapes for artists and designers. IJAIT, International Journal of Artificial Intelligence Tools, 15(4):651-672, 2006. Special Issue on AI in Music and Art.

11 Gooch, B., & Gooch, A. (2001). *Non-photorealistic rendering* (Vol. 2). Wellesley: AK Peters.

12 AlMeraj, Z., Wyvill, B., Isenberg, T., Gooch, A. A., & Guy, R. (2009). Automatically mimicking unique hand-drawn pencil lines. *Computers & Graphics*, *33*(4), 496-508.

13 Hsu, S. C., & Lee, I. H. (1994, July). Drawing and animation using skeletal strokes. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (pp. 109-118). ACM.

14 "Van Gogh's Starry Night Interactive by Petros Vrellis [openFrameworks, IPad]." *CreativeApplicationsNet*. N.p., n.d. Web. 30 Nov. 2014.

15 Evelyne Lutton, Hugo Gilbert, Waldo Cancino, Benjamin Bach, Pierre Parrend, and Pierre Collet. Gridvis: Visualisation of island-based parallel genetic algorithms. In Evopar2014, EvoApplications track of EvoStar, The leading European event on Bio-Inspired Computation, LNCS. Springer, April 2014. Best paper award of Evopar2014, 23-25 April, Granada, Spain.

16 E. Lutton, P. Collet, and J. Louchet. Easea comparisons on test functions: Galib versus eo. In EA01 Conference on Artificial Evolution, Le Creusot, France, October 2001.

17 Enzo Bolis, Christian Zerbi, Pierre Collet, Jean Louchet, and Evelyne Lutton. A gp artificial ant for image processing : preliminary experiments with easea. In LNCS 2038 Springer Verlag, Lecture Notes on Computer Science, editor, EVOIASP2001, pages 246-255, 2001. Lake Como, Italy.

18 Pierre Collet, Evelyne Lutton, Marc Schoenauer, and Jean Louchet. Take it EASEA. In M. Schoenauer, K. Deb, G. Rudolf, X. Yao, E. Lutton, Merelo. J.J., and H.-P. Schwefel, editors, Parallel Problem Solving from Nature - PPSN VI 6th International Conference, Paris, France, September 16-20 2000. Springer Verlag. LNCS 1917.

19 "Draw, Plot 2d Line In C# (csharp) - Bresenham's Line Algorithm." *Dzone.com*. N.p., n.d. Web. 30 Nov. 2014.

20 Lutton, E., & Martinez, P. (1994, October). A genetic algorithm for the detection of 2D geometric primitives in images. In *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on* (Vol. 1, pp. 526-528). IEEE.

Annex EASEA Code

```
/*_____
Template for an EASEA evolutionary algorithm for easea v1.0.3
_____
*/

\User declarations : // This section is copied on top of the output file
#include <fstream>
#include <string>
#include <iostream>
#define M_PI 3.14159265358979323846
#include <math.h>
#include <ctype.h>
#include <cstring>
#include <sstream>
#include <cstdlib>
#include <stdio.h>

int **Rmat;
int **Gmat;
int **Bmat;

int **Rmattemp;
int **Gmattemp;
int **Bmattemp;

int **RmatLine;
int **GmatLine;
int **BmatLine;

int M=0;
int N=0;
```

```
\end

\User functions:

int strtoint(std::string text) {
	int a;
	std::stringstream stream;
	for(int b = 0; b < text.size(); b++) {
		if(!isdigit(text.at(b))) {
			text.erase(b, 1);
		}
	}
	stream<<text<<std::flush;
	stream>>a;
	return a;
}

void readMatrix(char *name, int** &Transfer, int &M, int &N)
{
	std::ifstream file(name);
	std::string str;
	std::getline(file, str);
	M=strtoint(str);
	std::getline(file, str);
	N=strtoint(str);
	Transfer = new int*[M];
for(int i = 0; i < M; ++i)
   Transfer[i] = new int[N];

	for (int i=0; i<M; i++)
	{
		std::getline(file, str);
		std::string word;
		std::stringstream stream(str);
		for (int j=0; j<N; j++)
		{
			std::getline(stream, word, ' ');
			Transfer[i][j]=atoi(word.c_str());
			//std::istringstream os(word.c_str());
			//os >> Transfer[i][j];
		}
	}
	std::cout << "Matrix loaded"<< "\n";
}



void printMatrix(int **matrix, int rows, int cols, char *name)
{
	FILE *f = fopen(name, "w");
	if (f == NULL)
	{
		printf("Error opening file!\n");
		exit(1);
	}
	fprintf(f, "%i\n", rows);
	fprintf(f, "%i\n", cols);

	for (int i=0;i<rows;i++)
	{
		for (int j=0;j<cols;j++)
		{
			fprintf(f, "%i ", matrix[i][j]);
		}
		fprintf(f, "\n");
	}
	fclose(f);
}




 void SwapNum( int &x,  int &y)
     {
	int tempswap = x;
	x = y;
	y = tempswap;

     }

void Line2(int x, int y, int x2, int y2,
	int** &Rmat, int** &Gmat, int** &Bmat,
	int r, int g, int b)
    {
		int x0 = x;
	int y0 = y;
	int x1 = x2;
	int y1 = y2;
	    if (x0<0)
	    {
			x0=0;
	    }
	    if (x0>N)
	    {
			x0=N-1;
	    }
	    if (x1<0)
	    {
			x1=0;
	    }
	    if (x1>N)
	    {
			x1=N-1;
	    }

	    if (y0<0)
	    {
			y0=0;
	    }
	    if (y0>M)
	    {
			y0=M-1;
	    }
	    if (y1<0)
	    {
			y1=0;
	    }
	    if (y1>M)
	    {
			y1=M-1;
	    }

	bool steep = abs(y1 - y0) > abs(x1 - x0);
	if (steep)
	{
	   SwapNum( x0,  y0);
	   SwapNum( x1,  y1);
	}
	if (x0 > x1)
	{
	   SwapNum( x0,  x1);
	   SwapNum( y0,  y1);
	}
	int deltax = x1 - x0;
	int deltay = abs(y1 - y0);
	int error = -deltax / 2;
	int ystep;
	int yt = y0;
	if (y0 < y1) ystep = 1; else ystep = -1;
	for (int xt = x0; xt < x1; xt++)
	{
	   if (steep)
	   {
	      //plot.SetPixel(new Point(y, x));
	      Rmat[yt][ xt] = r;
	      Gmat[yt][ xt] = g;
	      Bmat[yt][ xt] = b;
	   }
	   else
	   {
```

```
            Rmat[xt][ yt] = r;
            Gmat[xt][ yt] = g;
            Bmat[xt][ yt] = b;
            //plot.SetPixel(new Point(x, y));
          }
          error = error + deltay;
          if (error > 0)
          {
            yt = yt + ystep;
            error = error - deltax;
          }
        }


    }

int getCost2(int a,int b, int c, int d)
{


            int temp=0;

            for (int i=0; i<M; i++)
            {

                        for (int j=0; j<N; j++)
                        {
                                    RmatLine[i][j]=Rmattemp[i][j];
                                    GmatLine[i][j]=Gmattemp[i][j];
                                    BmatLine[i][j]=Bmattemp[i][j];

                        }
            }

            Line2( a, b, c, d, RmatLine, GmatLine, BmatLine,0, 0, 0);

            for (int i=0; i<M; i++)
            {

                        for (int j=0; j<N; j++)
                        {
                                    temp=temp+abs(Rmat[i][j]-RmatLine[i][j]);
                                    temp=temp+abs(Gmat[i][j]-GmatLine[i][j]);
                                    temp=temp+abs(Bmat[i][j]-BmatLine[i][j]);

                        }
            }


 return temp;
}

\end

\Before everything else function:
{

            readMatrix(const_cast<char *>("Rmat.matrix"),Rmat,M,N);
            readMatrix(const_cast<char *>("Gmat.matrix"),Gmat,M,N);
            readMatrix(const_cast<char *>("Bmat.matrix"),Bmat,M,N);
            readMatrix(const_cast<char
*>("Rmattemp.matrix"),Rmattemp,M,N);
            readMatrix(const_cast<char
*>("Gmattemp.matrix"),Gmattemp,M,N);
            readMatrix(const_cast<char
*>("Bmattemp.matrix"),Bmattemp,M,N);

            readMatrix(const_cast<char
*>("Rmattemp.matrix"),RmatLine,M,N);
            readMatrix(const_cast<char
*>("Gmattemp.matrix"),GmatLine,M,N);
            readMatrix(const_cast<char
*>("Bmattemp.matrix"),BmatLine,M,N);

            cout<<M<<endl;
            cout<<N<<endl;

            //Line2( 300, 300, 0, 0, Rmat, Gmat, Bmat,0, 0, 0);
            //printMatrix(Rmat, M, N, "Rmat2.matrix");

}
\end

\After everything else function:

//std::cout << (*population) ;

cout <<(population->Best->serialize())<<endl;
string str=population->Best->serialize();

            string word;
            stringstream stream(str);

                        getline(stream, word, ' ');
                        int d=atoi(word.c_str());
                        getline(stream, word, ' ');
                        int c=atoi(word.c_str());
                        getline(stream, word, ' ');
                        int b=atoi(word.c_str());
                        getline(stream, word, ' ');
                        int a=atoi(word.c_str());


cout << a << "\n";
cout << b << "\n";
cout << c << "\n";
cout << d << "\n";

Line2( a, b, c, d, Rmattemp, Gmattemp, Bmattemp,0, 0, 0);
printMatrix(Rmattemp, M, N, "Rmattemp.matrix");
printMatrix(Gmattemp, M, N, "Gmattemp.matrix");
printMatrix(Bmattemp, M, N, "Bmattemp.matrix");
\end

\At the beginning of each generation function:{
}
\end

\At the end of each generation function:

\end

\At each generation before reduce function:
 //cout << "At each generation before replacement function called" << endl;

\end

\User classes :
GenomeClass {
 // need to declare the genome here
 int a, b, c, d;
}
\end

\GenomeClass::display:
            printf("%i %i %i %i\n",Genome.a, Genome.b, Genome.c,
Genome.d);
//printf("%f \n",Genome.a);
\end


\GenomeClass::initialiser : // "initializer" is also accepted
 // the genome to initialise is known as "Genome"
 Genome.a=random(0,N);
 Genome.b=random(0,M);
 Genome.c=random(0,N);
 Genome.d=random(0,M);
\end

\GenomeClass::crossover :

int cost=getCost2(parent1.a,parent1.b,parent1.c,parent1.d);
child.a=parent1.a;
child.b=parent1.b;
```

```
child.c=parent1.c;
child.d=parent1.d;

int costTemp=getCost2(parent1.a,parent1.b,parent2.c,parent1.d);
if (costTemp<=cost)
{
child.a=parent1.a;
child.b=parent1.b;
child.c=parent2.c;
child.d=parent1.d;
cost=costTemp;
}

costTemp=getCost2(parent1.a,parent2.b,parent1.c,parent1.d);
if (costTemp<=cost)
{
child.a=parent1.a;
child.b=parent2.b;
child.c=parent1.c;
child.d=parent1.d;
cost=costTemp;
}

costTemp=getCost2(parent1.a,parent2.b,parent2.c,parent1.d);
if (costTemp<=cost)
{
child.a=parent1.a;
child.b=parent2.b;
child.c=parent2.c;
child.d=parent1.d;
cost=costTemp;
}

costTemp=getCost2(parent2.a,parent1.b,parent1.c,parent1.d);
if (costTemp<=cost)
{
child.a=parent2.a;
child.b=parent1.b;
child.c=parent1.c;
child.d=parent1.d;
cost=costTemp;
}

costTemp=getCost2(parent2.a,parent1.b,parent2.c,parent1.d);
if (costTemp<=cost)
{
child.a=parent2.a;
child.b=parent1.b;
child.c=parent2.c;
child.d=parent1.d;
cost=costTemp;
}

costTemp=getCost2(parent2.a,parent2.b,parent1.c,parent1.d);
if (costTemp<=cost)
{
child.a=parent2.a;
child.b=parent2.b;
child.c=parent1.c;
child.d=parent1.d;
cost=costTemp;
}

costTemp=getCost2(parent2.a,parent2.b,parent2.c,parent1.d);
if (costTemp<=cost)
{
child.a=parent2.a;
child.b=parent2.b;
child.c=parent2.c;
child.d=parent1.d;
cost=costTemp;
}

//second batch
costTemp=getCost2(parent1.a,parent1.b,parent1.c,parent2.d);
child.a=parent1.a;
child.b=parent1.b;
child.c=parent1.c;
child.d=parent2.d;

costTemp=getCost2(parent1.a,parent1.b,parent2.c,parent2.d);
if (costTemp<=cost)
{
child.a=parent1.a;
child.b=parent1.b;
child.c=parent2.c;
child.d=parent2.d;
cost=costTemp;
}

costTemp=getCost2(parent1.a,parent2.b,parent1.c,parent2.d);
if (costTemp<=cost)
{
child.a=parent1.a;
child.b=parent2.b;
child.c=parent1.c;
child.d=parent2.d;
cost=costTemp;
}

costTemp=getCost2(parent1.a,parent2.b,parent2.c,parent2.d);
if (costTemp<=cost)
{
child.a=parent1.a;
child.b=parent2.b;
child.c=parent2.c;
child.d=parent2.d;
cost=costTemp;
}

costTemp=getCost2(parent2.a,parent1.b,parent1.c,parent2.d);
if (costTemp<=cost)
{
child.a=parent2.a;
child.b=parent1.b;
child.c=parent1.c;
child.d=parent2.d;
cost=costTemp;
}

costTemp=getCost2(parent2.a,parent1.b,parent2.c,parent2.d);
if (costTemp<=cost)
{
child.a=parent2.a;
child.b=parent1.b;
child.c=parent2.c;
child.d=parent2.d;
cost=costTemp;
}

costTemp=getCost2(parent2.a,parent2.b,parent1.c,parent2.d);
if (costTemp<=cost)
{
child.a=parent2.a;
child.b=parent2.b;
child.c=parent1.c;
child.d=parent2.d;
cost=costTemp;
}

costTemp=getCost2(parent2.a,parent2.b,parent2.c,parent2.d);
if (costTemp<=cost)
{
child.a=parent2.a;
child.b=parent2.b;
child.c=parent2.c;
child.d=parent2.d;
cost=costTemp;
}


\end

\GenomeClass::mutator : // Must return the number of mutations
```

```
  if (tossCoin(0.4)){
    Genome.a+=random(-10,10);
    return 1;
  }
  if (tossCoin(0.4)){
    Genome.b+=random(-10,10);
    return 1;
  }
  if (tossCoin(0.4)){
    Genome.c+=random(-10,10);
    return 1;
  }
if (tossCoin(0.4)){
    Genome.d+=random(-10,10);
    return 1;
  }
 return 0 ;
\end

\GenomeClass::evaluator : // Returns the score as a real value
              //IndividualImpl::printOn(std::cout);

              double temp=getCost2(Genome.a,Genome.b,Genome.c,
Genome.d);

 return temp;
\end

\User Makefile options:
\end

\Default run parameters :       // Please let the parameters appear in this
order
 Number of generations : 10   // NB_GEN
 Time limit: 0                                          // In seconds, 0 to
deactivate
 Population size : 100
 Offspring size : 50% // or a xx%
 Mutation probability : 1      // MUT_PROB
 Crossover probability : 1     // XOVER_PROB
 Evaluator goal : minimise     // maximise
 Selection operator: Tournament 20
 Surviving parents: 50       // Percentage or absolute
 Surviving offspring: 50     // Percentage or absolute
 Reduce parents operator: Tournament 10
 Reduce offspring operator: Tournament 10
 Final reduce operator: Tournament 7

 Elitism: Strong                                        // Weak or Strong
 Elite: 1
 Print stats: true                                              // Default: 1
 Generate csv stats file:true
 Generate gnuplot script:false
 Generate R script:false
 Plot stats:false                                        // Default: 0

 Remote island model: false
 IP file: ip.txt                                       // List of IP:PORT of islands
to send individuals to
 Server port : 2929
 Migration probability: 0.33   // Probability of sending an individual per
generation

 Save population: true
 Start from file:false
\end
```