Programming Challenge 2010
Blobby Land

Gordon J. Pace, Chris Porter, Christian Colombo

February 2010

Abstract

This document outlines the programming challenge task that has to be solved in the 2010 ICTSA Programming Challenge. The competition will open on Friday, 26th February 2010 at 20:00 and closes on Sunday, 28th February 2010 at 20:00 ZST[1]. This document also specifies the criteria that will be used to judge the entries. The full rules of the Programming Challenge can be found on the website http://programming.ictsamalta.org/

1 Judging Panel

The judging panel is made up of the following three members:

- Gordon J. Pace, Department of Computer Science, University of Malta `<gordon.pace@um.edu.mt>`
- Chris Porter, Department of Information Systems, University of Malta `<chris.porter@um.edu.mt>`
- Christian Colombo, Department of Computer Science, University of Malta `<christian.colombo@um.edu.mt>`

Any queries to the panel have to be directed to all three members making use of the designated competition Google group which can be accessed by clicking the 'support' button in the navigation menu of the website. No emails sent directly to the judges will be answered. The role of the panel is to judge all submissions made to this competition and declare the winners. The judging panel's decision is final. The formal rules of the competition can be found at the competition's website. Ensure you read through them before you start.

[1]Zeus Standard Time — the time as set on UNIX server zeus, based at the Departments of Computer Science at the University of Malta. This time can be seen on the Programming Challenge website

2 The Challenge

2.1 Problem Definition

The aim of the challenge is to approximate images using a small number of circular blobs of paint. The images are all rectangular, with each of their constituent pixels ranging over a small number of colours (8). Facing an empty (white) canvas, of the same dimensions as the given image, an artist paints circular blobs of paint (each one possibly covering previously painted ones) in one of the 8 possible colours. The challenge the artist faces, is to end up with a painting similar to the original one, not using more than a given number of paint blobs. Your task is to automate the artist.

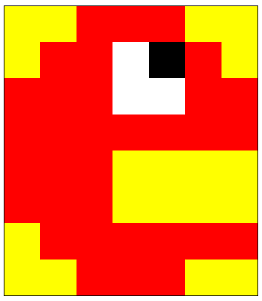
2.2 Input File Syntax

Your program will be given a bitmap saved as a text file consisting of (i) the width w of the bitmap in the first line; (ii) the height h of the bitmap in the second line; and (iii) a separate line for each of the h rows of the image, with each line consisting of w numbers ranging over the characters 0 to 7. Each character gives the colour of the pixel at that location according to the following table:

#	Colour
0	Black
1	Blue
2	Red
3	Magenta
4	Green
5	Cyan
6	Yellow
7	White

An input file will look like the following, corresponding to the image shown alongside:

```
7
8
6622266
6227026
2227722
2222222
2226666
2226666
6222222
6622266
```



You may assume that the input files are syntactically correct (they contain the right number of lines and characters per line as specified in the first two lines, and that the characters all fall within the range 0 to 7), and contain no empty lines or extraneous white space. You can also assume that all bitmaps are at least 1 by 1 pixels.

2.3 Output File Syntax

Your program will be given a valid bitmap and the maximum number of paint blobs permissible n. It must output a single text file describing a blob-map consisting of up to $n+2$ lines. The first two lines give the width and height of the blob-map (and should match the width and height of the bitmap, while the remaining lines describe the blobs of paint to be applied. The lines describing the blobs of paint contain four space delimited values giving respectively (i) the x-coordinate of the centre of the blob of paint; (ii) the y-coordinate of the blob of paint; (iii) the radius of the blob of paint; and (iv) the colour of the paint to be used. The file must satisfy the following constraints:

- There are no empty lines;
- No extra white space in the lines;
- The colours must range over 0 to 7;
- The radii must be non-negative.

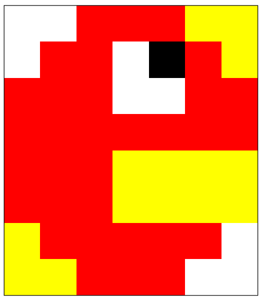
Also note that:

- Each blob of paint $\langle x\,y\,r\,c\rangle$ changes the colour of each pixel inside the blob (a pixel with coordinates (α, β) will change colour if its centre lies in the circle — $(x-(\alpha+0.5))^2+(y-(\beta+0.5))^2 \leq r^2$) to colour c — coordinates (0,0) correspond to the bottom left hand corner of the bottom left pixel of the bitmap;

- It does not matter if parts (or the whole) of a circle lie outside the bitmap;
- The blobs are processed top to bottom and the last painted blob covering a pixel sets its final colour. Pixels not covered by any blob remain white (colour 7);
- The coordinates of the centres, and the radii may be non-integers;
- The coordinates of the centres may also be negative.

An example blobmap approximating the example bitmap given earlier using 8 blobs of paint can be seen below:

```
7
8
3.5 4 3.75 2
4 3 1 6
6 3 1 6
4 6 1 7
4.5 6.5 0.5 0
0 0 2 6
-1 -1 2.5 6
7 8 2 6
```



3 Helpful Tools

A number of useful tools can be downloaded from the website to help with understanding the problem and evaluating your solutions:

Sample bitmaps: Three bitmaps of different sizes are available for download, on which you can experiment with your solution.

Bitmap visualisation: The program `bm2ps` generates a postscript image of a bitmap. It takes two parameters: (i) the name of the bitmap file to read; and (ii) the name of a postscript file to generate.

Blobmap visualisation: The program `blbs2ps` generates a postscript image of a blobmap. It takes two parameters: (i) the name of the blobmap file to read; and (ii) the name of a postscript file to generate.

Blobmap to bitmap conversion: The program `blbs2bm` generates a bitmap rendering of a blobmap. It takes two parameters: (i) the name of the blobmap file to read; and (ii) the name of the bitmap file to generate.

Comparison of blobmap with bitmap: The program `cmp` compares the original bitmap to a generated blobmap. It takes three parameters: (i) the name of the original bitmap file; (ii) the maximum number of blobs that should have been used; and (iii) the name of the generated blobmap file. The program outputs the similarity factor and the amount of paint consumed.

4 Judging Criteria

The judging panel will run all submissions on a number of pre-defined test bitmaps and number of permissible blobs. Each program will be given up to 1 minute to generate a blobmap for each problem instance. Programs not terminating on a particular problem within 1 minute, or producing wrong output (too many blobs, invalid syntax or changed bitmap size) will get no points for that problem

For each problem instance, the programs will be ranked according to how similar the blobmap generated is, with respect to the original bitmap.

Given the original bitmap and the output blobmap, we will calculate the following values:

1. The number of exactly matching pixels *correct*;
2. The number of almost correct pixels (a pixel in the output is almost correct, if it is not correct, and it matches the colour of at least three of the eight neighbouring pixels in the original bitmap) *similar*.

The similarity of two such maps (of size *width* × *height*) is calculated using the following definition:

$$similarity = \frac{correct + 0.3 \times similar}{width \times height}$$

For each problem instance, the teams are ordered according to the similarity factor obtained (the larger the better). A score is then given to each team according to their ranking: best 50 points, second 30 points, third 20 points, fourth 10 points and fifth 5 points. The rest will not be awarded any points.

In the case of ties on an individual problem, all submissions solutions with the same similarity factor, the team consuming least paint (where each blob consumes πr^2 paint) is given priority. If they still give the same results, they will be given the same ranking for that problem.

The winning team will be the one which obtains the largest total number of points. In the unlikely case of unresolved ties, the ranking will be done according to the execution time of the program over all the problem set (fastest program wins) after which the judging panel reserves the right to resolve any further tie in as fair a way as possible using a tie breaker.

5 Submission Rules

Any submissions have to be a single Microsoft Windows executable file — a command line program called `blobify.exe` and must take exactly three parameters: the name of the bitmap file it should read, the maximum number of blobs it may use, and the name of the output blobmap file it is to produce. It will be executed as follows:

```
blobify <input file> <n> <output file>
```

No new windows or graphical user interfaces are to be opened by the program. The program will be executed on an Intel-based PC and having 2GB of RAM. The operating system will be Microsoft Windows XP and it will also have the latest version of the .NET Framework installed and the latest Java version installed. Any submissions that fail to execute on the mentioned environment or any submission that take longer than 1 minute to output a result will not be considered by the judging team.

Each registered team is allowed to submit as many solutions as it wishes. The last submission on Sunday, 28th February 2010 at 20:00 will be considered for the prize. After these times no corrections or re-submissions will be allowed. Submissions can only be done through the programming challenge website — note that the program must finish uploading before the aforementioned time.

6 The Judging Suite

To ensure fairness, the problems that will be used for judging the submissions will also be made available when the challenge opens. These will be encrypted and at the end of the competition, the key will be given to ensure that the files were unchanged.

7 Frequently Would-Have-Been-Asked Questions

Question: What happens to paint overflowing off the canvas with circles which are too big?

Answer: It spills to the floor. Your mum may get angry, and you may get spanked. As far as the program is concerned, that paint is still used but will not contribute to the image.

◇

Question: But if I give the centre of a blob outside the bitmap, is it still acceptable input?

Answer: Yes, in fact, you can also have negative x and y coordinates of the blobs painted. The radii must be positive, though.

◇

Question: If the coordinates can be negative, can they be fractional?

Answer: Yes, they can be rational numbers, but not imaginary.

◇

Question: What is the maximum size of the bitmaps?

Answer: The width and height of the bitmaps will not exceed a googolplex.

⋄

Question: What is a googolplex?

Answer: A googolplex is 10^{googol}.

⋄

Question: What is a googol?

Answer: A googol is 10^{100}.

8 Clarifications

During the period between Friday, 26th February at 20:00 and Sunday, 28th February 2010 at 20:00, all the teams can send queries for clarification through the use of the designated competition Google group which can be accessed by clicking the 'support' button in the navigation menu of the website. No emails sent directly to the judges will be answered. If the judging panel deems the query to be a valid one, and not answered in this document, the answer will be posted back to the group. The judging panel will do its best to answer these queries in as short a time as possible.