2014 ICTSA Programming Challenge

version 1.4

Christian Colombo, Jean-Paul Ebejer, Karl Fenech,
Kristian Guillaumier, Gordon J. Pace, Chris Porter

November 20, 2014

This document outlines the programming challenge task that has to be solved in the 2014 ICTSA Programming Challenge. The competition will open on Friday, 21st November 2014 at 20:00 and closes on Sunday, 23rd November 2014 at 20:00 ZST[1]. This document also specifies the criteria that will be used to judge the entries.

1 Background

Orson Byar, a world-renowned director, is working on his next movie. Alas, the constraints placed on him by the producers result in a movie that, no matter how well edited, he would rather bin... Except that his contract says that if he does not release the movie by the end of the year, he will be sued for twenty times his worth. Suddenly, he has a brainwave: "what if I edit the frames of the movie and put them in random order?". That way, he will have released a 90 minute movie about a murder on the midnight Gozo Channel ferry thus satisfying the contract he had signed, but without anyone ever being able to watch the movie he is so ashamed of having shot. He locks himself in the editor's room for a week, hands the movie reel to the producers and leaves. The movie is immediately hailed as a postmodern classic. Indeed some critics started referring to "stochastic movies" as a new movie making medium for the new (or at least teenage) millenium, and many other directors released movies using this new 'medium'.

Your task in this year's programming competition is to write a program that restores the correct frame order of a number of movies to be able to make sense of the storyline.

[1]Zeus Standard Time — the time as set on UNIX server zeus, based at the Department of Computer Science at the University of Malta. This time can be seen on the Programming Challenge website `http://www.cs.um.edu.mt/svrg/39fgl1332dgfdds33/Game_of_Codes/`

2 The Task

The task is to write a program which given a command line parameters:

```
unscramble.exe video.vfr prediction.vsq
```

will read a video frame file (`video.vfr`) and output a sequence of predicted frames to the text file specified as the second parameter (`prediction.vsq`). This sequence should render the video correctly. The output should have one frame identifier per line and the same number of frames as the video frame text file (otherwise the prediction will be considered invalid and disqualified).

This section describes the input, outputs, and scoring function used in this year's programming task.

2.1 Input

You will be given a set of **unordered** still images or frames from a video. This set of frames will be described in a video frame text file, with extension `.vfr`. The format of this text file is the following:

- The number of frames, F, in the video followed by a newline.
- The width of each frame, W, followed by a newline. This is constant for all frames in the video.
- The height of each frame, H, followed by a newline. This is constant for all frames in the video.
- A sequence of F frames, where each frame consists of:
  - A sequence of H scanlines, where each scanline consists of:
    - A sequence of W pixels, where each pixel consists of a six character hexadecimal color code (two each for the red, green, blue components, in that order).
    - Each scanline is terminated with a newline character.

A video frame file may be opened with any text editor (e.g. vim or notepad). A simple 3-frame movie (with 3×3 pixel resolution) of a red dot on a white background moving diagonally from top-left to bottom-right is given in Listing 1.

Although you cannot assume anything about the size or number of frames of the video files, you may safely assume that they will fit in the memory of the machine they will be run on.

Listing 1: Anatomy of an video text file.

```
3
3
3
ff0000ffffffffffff
ffffffffffffffffff
ffffffffffffffffff
ffffffffffffffffff
ffffffff0000ffffff
ffffffffffffffffff
ffffffffffffffffff
ffffffffffffffffff
ffffffffffffff0000
```

2.2 Output

The aim of your program is to write out the correct sequence of frames that renders the original video correctly. Your program should write your predicted sequence of frames to a text file (this is specified as the second command line parameter to your program). The output file should have a `.vsq` extension. Each line in the sequence file is separated with a new line character (i.e. one frame number identifier per line). The frame number (starting from zero) refers to a specific frame in the input video frame file (specified as the first command line argument to your program).

We supply an online tool (at `http://www.cs.um.edu.mt/svrg/39fgl1332dgfdds33/Game_of_Codes/resources/WebTools/VideoPlayer.html`) which given an video frame text file and a sequence of frames will show you the resulting video.

2.3 Scoring Function

The scoring function used on a (predicted) frame sequence is described in Equation 1.

$$S = \frac{\sum_{i=1}^{n-1} |idx(F_i) - idx(F_{i-1})|}{n-1} \tag{1}$$

In Equation 1, S is the score of the sequence of frames predicted by your entry, n is the total number of frames in the video, *idx(FrameId)* is the position or index of the specified frame in the predicted sequence and F_i is the ith frame in the original video. S is the average distance between all pairwise subsequent frames in the predicted sequence. The best possible score of S is therefore 1 (smaller S is better). Note that this scoring function gives similar scores to a video frame sequence and its reverse sequence (inverted video sequence file).

Informally, the score gives you the average distance between pairwise frames in your prediction.

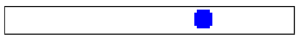
For example, the score for a video with four (n) frames F_0, F_1, F_2, and F_3 and predicted as $\langle F_0, F_2, F_3, F_1 \rangle$ is 2 and is computed as follows:

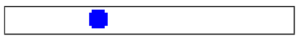
- For $i = 1$, the first pair of frames F_1 and F_0, $idx(F_1) = 3$ and $idx(F_0) = 0$ when subtracted and taken as an absolute value gives 3
- For $i = 2$, the second pair of frames F_2 and F_1, $idx(F_2) = 1$ and $idx(F_1) = 3$ when subtracted and taken as an absolute value gives 2
- For $i = 3$, the third pair of frames F_3 and F_2, $idx(F_3) = 2$ and $idx(F_2) = 1$ when subtracted and taken as an absolute value gives 1
- Summing each pairwise ordering score, $3 + 2 + 1 = 6$, and $6 \div 3 = 2$. Giving a final score of $S = 2$.

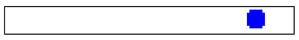
2.4 Sample Problem

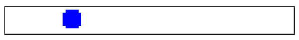
You will be supplied with three examples of video frame text files. You may download these from `http://www.cs.um.edu.mt/svrg/39fgl1332dgfdds33/Game_of_Codes/resources/sample_problems.7z`. The first sample video (reproduced below) is a ball moving from left to right. The second video sample is a ball, moving from left to right but also bouncing and approaching the camera. The third video sample is the opening credits of a popular TV show (no, it's not Star Wars, even if we thought long about that).

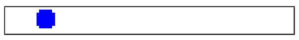
Figure 1 shows a pictorial representation of each frame as defined in the video frame text file (`sample001_ball_moving_right.vfr`).

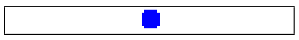

(a) Frame #0


(b) Frame #1

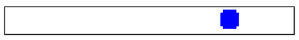

(c) Frame #2

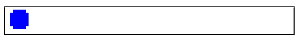

(d) Frame #3


(e) Frame #4


(f) Frame #5

(g) Frame #6


(h) Frame #7


(i) Frame #8

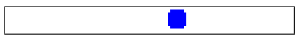

(j) Frame #9

Figure 1: The nine frames of a video (as defined in the video frame text file) of a blue ball moving rightwards, without overlap. A black border is added to each frame to ease visualization. Note that the order of the frames in this video file is what you have to rearrange in order to get an intelligible video.

The first thirteen lines of the video text file of the first sample, `sample001_ball_moving_right.vtf`, are shown in Listing 1. This is truncated due to space constraints.

Listing 2: A few lines from `sample001_ball_moving_right.vtf`.

```
10
110
10
[next 8 frames (80 lines) removed...]
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff[50 more f removed...]
ffffffffffffffffff0000ff0000ff0000ff0000ff0000ffffffffffffff[50 more f removed...]
ffffffffffff0000ff0000ff0000ff0000ff0000ff0000ff0000ffffffff[50 more f removed...]
ffffffffffff0000ff0000ff0000ff0000ff0000ff0000ff0000ffffffff[50 more f removed...]
ffffffffffff0000ff0000ff0000ff0000ff0000ff0000ff0000ffffffff[50 more f removed...]
ffffffffffff0000ff0000ff0000ff0000ff0000ff0000ff0000ffffffff[50 more f removed...]
ffffffffffff0000ff0000ff0000ff0000ff0000ff0000ff0000ffffffff[50 more f removed...]
ffffffffffffffffff0000ff0000ff0000ff0000ff0000ffffffffffffff[50 more f removed...]
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff[50 more f removed...]
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff[50 more f removed...]
[...next frame (10 lines) removed]
```

The correct output sequence file for this video file (this is what you have to predict) is show in Listing 3.

Listing 3: The frame sequence file `sample001_ball_moving_right.vsq`

```
8
4
3
1
6
5
9
0
7
2
```

Note that the identifiers start from zero and the frame sequence file should have as many lines as there are frames in the video file. If the frames in Figure 1 are re-arranged in this order, the correctly-sequenced video will be rendered.

2.5 Supplied Tools

You will be supplied with four tools (developed in .NET and running under Windows and Linux using the Mono project version 3.2.8). These tools may all be downloaded from `http://www.cs.um.edu.mt/svrg/39fgl1332dgfdds33/Game_of_Codes/resources/Tools.zip`.

2.5.1 Scoring Tool

The first tool is the scoring tool, called from the command line using:

```
score.exe original_sequence.vsq predicted_sequence.vsq
```

Given two text files one containing the original video frame sequence and the other containing the predicted sequence this program will calculate the score between the two. It implements Equation 1 and will be used to judge competition entries. The two input files should have the same number of frame identifiers (one per line). The frame identifiers start from zero (0) and have no gaps in the sequence. Also, an online version of this tool may be found at `http://www.cs.um.edu.mt/svrg/39fgl1332dgfdds33/Game_of_Codes/resources/WebTools/VideoPlayer.html`.

2.5.2 Visualization Tool

The second tool is a visualization tool, called from the command line using:

```
visualize.exe frames_definition.vfr sequence.vsq target_dir
```

Given a video frame text file (described in Section 2.1), a text file containing the sequence of frame identifiers (one per line) and a target directory – this program will generate a number of PNG files which can be ordered by the filename to reproduce the defined sequence (contained in the file defined as the second argument). The second and third arguments are optional. If these are not specified, frames (in the form of PNG files) will be generated in the sequence they are found in the video frame file and generated in a directory created in the same location of the video frame file.

2.5.3 VideoEncoding Tool

The third tool supplied is a program which given an mp4, avi or other video file generates a video frame file (`.vfr`). The tool requires the installation of `FFmpeg`[2]. This should be accessible (and in the same directory) of the executable we supply. Note that this tool is not strictly required for the execution and testing of your programming competition entry, however we supply it so you can create video frame files of your own videos. This is the tool we used to encode the judging and sample problems. The tool is run with the following command line:

```
encode.exe video.mp4 video.vfr
```

The first argument is the file path of the source video file (`.mp4`, `.avi`, ...), in any format supported by `FFmpeg`, and the second argument (optional) is the file path of the target video frames text file (`.vfr`). If the second argument is omitted a `.vfr` file is created in the same location as the video file.

2.5.4 VideoRandomizer Tool

This tool gives a new video frame text file with a random ordering and an associated frame sequence file (`.vsq`). This tool was used on all the judging and sample problems to generate a random ordering from the original video frame file. This program is called in the following way:

```
randomize.exe original_video.vfr new_seq_video.vfr new_seq.vsq
```

The first argument is the file path of the original video frames file (`.vfr`). The second argument is the file path to write the randomized video frames file (`.vfr`). The third argument is the file path to write the randomized sequence (`.vsq`).

[2] `https://www.ffmpeg.org/`

3 Judging Criteria

The judging panel will run all submissions on a number of pre-defined video frame text files (each with its associated correct frame sequence file, `.vsq`). Each program will be given up to 1 minute to generate a sequence for each problem instance. Programs not terminating on a particular problem within 1 minute, or producing wrong output (e.g. more frame numbers than actual frames) will get no points for that problem. For each problem instance, the programs will be ranked according to the computed score. For each problem instance, the teams are ordered according to the score in ascending order. 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. In the case of ties, the fastest program runtime execution is used as a tie breaker. The judging panel reserves the right to resolve any further tie in as fair a way as possible using a tie breaker.

4 Submission Rules

All submissions have to be either (i) a single Microsoft Windows or Linux executable file – a command line program called `unscramble.exe`; or (ii) a zip file with one jar file and a batch file called `unscramble.bat`. In both cases, the executable or batch file must take exactly two parameters: the name of the video frame text file it should read and the frame sequence 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 4GB of RAM. The operating systems will be 64-bit Microsoft Windows 8.1 and Ubuntu 14.04. 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 takes longer than 1 minute to output a result **and terminate** will not be considered by the judging panel.

Each registered team is allowed to submit as many solutions as it wishes. The last submission on Sunday, 23rd November 2014 at 20:00 ZST will be considered for the prize. After this deadline no corrections or resubmissions will be allowed. Submissions can only be done through the programming challenge website – note that the program must finish uploading before the aforementioned time.