CSCI 3280 Introduction to Multimedia Systems

Spring 2022, Assignment 1 - Inverse ASCII Art

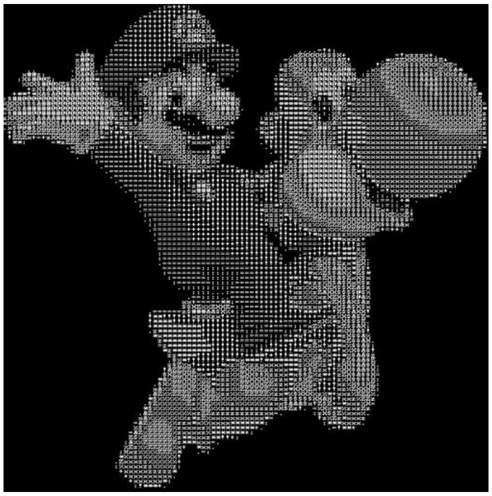
Due Date: Feb. 14, 2022 (11:59pm) Submission via Blackboard

Late submission penalty: 10-point deduction per day (maximum 30%)

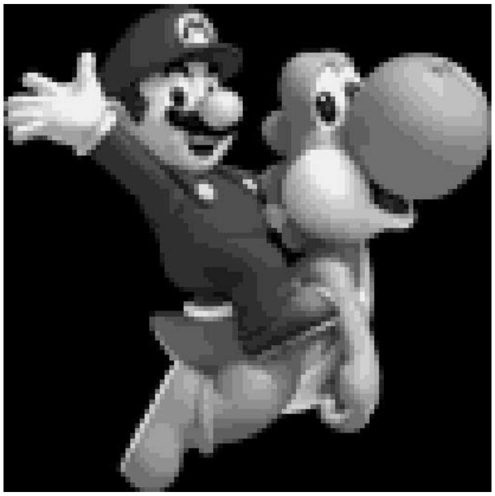
PLAGIARISM Penalty: Whole Course Failed

Introduction

Tone-based ASCII Art is an interesting art form which uses limited ASCII character set as their basic image elements, and it was a popular way to print large graphics posters from dot matrix printers back in the old days but it still remains a popular art form. In this assignment, you are required to complete two small programs, the first one (`inverse.cpp`) converts a tone-based ASCII Art (8 tones) into gray-scale bitmap in .BMP format and the second one (`ascii.cpp`) converts regular RGB bitmap into 8-tone ASCII art.



into



General Requirements

1. Program must be coded in ANSI C/C++, no additional libraries are allowed.
2. The compiled programs must run in **<u>Windows</u>** command prompt as a console program and accepts input with the following syntax.

```
C:\> inverse <art.txt> <art.bmp>
C:\> ascii <input.bmp> <output.txt>
```

   `inverse` is your program executable for the **Inverse ASCII Art** part.
   `<art.txt>` is the full path name to the given ASCII art file.
   `<art.bmp>` is the full path name for the output bitmap.
   `ascii` is your program executable for the **ASCII Art Generation** part.
   `<input.bmp>` is the full path name to the given bitmap.
   `<output.txt>` is the full path name to the output ASCII art file.

3. A simple .bmp file library is included in the package (bmp.h and bmp.cpp).
4. You may assume input ASCII art or bitmap has size not bigger than 256 x 256.
5. You are required to **<u>submit source code only</u>**. We will use Visual Studio 2019 C++ compiler and have your program compiled via visual studio command prompt with the following command line.

```
C:\> cl.exe inverse.cpp bmp.cpp
C:\> cl.exe ascii.cpp bmp.cpp
```

   `inverse.cpp` is the provided skeleton code for part 1.
   `ascii.cpp` is the provided skeleton code for part 2.
   You are required to complete these files to perform the required functionalities.

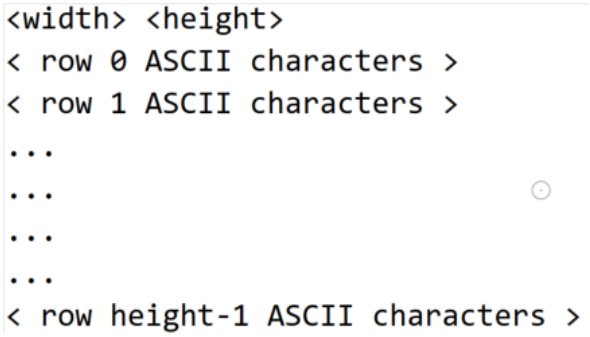
   Please make sure that your source code gets compiled well with it, "failed to compile" receives 10-point deduction for each failed source.

6. ASCII art and bitmap test files are included for testing programs.

Part 1 – Inverse ASCII Art (`inverse.cpp`, 50 points)

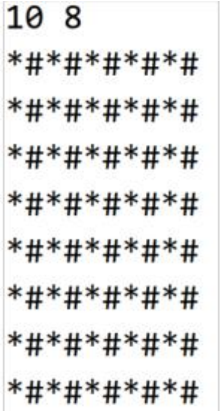
Complete the provided skeleton program source file `inverse.cpp`. Your program should process an ASCII art text file (`art.txt`) and output a grayscale (R, G and B channels have the same value) bitmap in `.bmp` format. A basic 8-level ASCII character mapping is defined in the provided skeleton file `inverse.cpp.` Remember that the R, G and B channels of a bitmap can take values from 0-255 (unsigned char data type).

1. ASCII art file format:



```
<width> <height>
< row 0 ASCII characters >
< row 1 ASCII characters >
...
...
...
...
< row height-1 ASCII characters >
```

   Example:



```
10 8
*#*#*#*#*#
*#*#*#*#*#
*#*#*#*#*#
*#*#*#*#*#
*#*#*#*#*#
*#*#*#*#*#
*#*#*#*#*#
*#*#*#*#*#
```

2. You should derive a quantization to convert the ascii characters to RGB values following the 8-level ASCII character mapping in `inverse.cpp.`

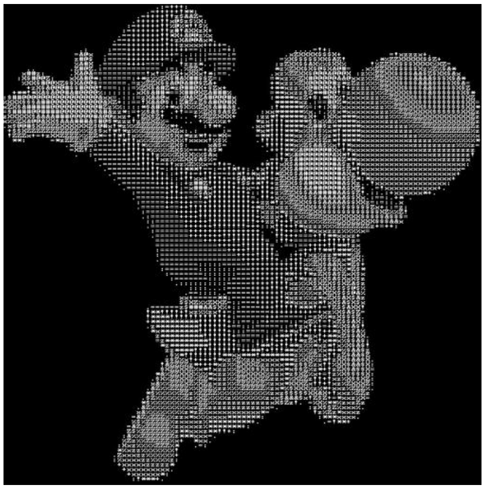
Part 2 – ASCII Art Generation (ascii.cpp, 30 points)

Complete the program source file `ascii.cpp`. Your program should process a given RGB bitmap in `.bmp` format and produce an ASCII art file with the file format given in Part 1. You have to convert RGB into grayscale by using the following formula:

Gray = 0.299 * R + 0.587 * G + 0.114 * B;

Grayscale value should then be quantized into 8 levels properly. The same 8-level ASCII character set used in part 1 should be used to produce your result.

into



Bonus Part (20 points)

You are encouraged to implement some enhancement or features that you find interesting and put this bonus-part program into its own standalone source file named `bonus.cpp`.

Some suggestions:

- Image compression. In part 2, what if the size of the input bitmap is larger than 256 x 256. Can you think about a method to reduce the size of the bitmap first before converting it to an ASCII art?
- Convert the given RGB bitmap into an ASCII Art bitmap (we provide the bitmap image of each ASCII character defined in the program).
- Convert the given RGB bitmap into a Colored ASCII Art bitmap (use your own colored ascii character bitmap set).
- Modify the given ASCII character mapping to achieve better visual qualities.
- And many more…

Please also submit a report that consists of the following items:

- For each additional feature/ enhancement you implemented:
  - The explanation of the feature
  - The source code segment related to the feature
  - Sample runs (including the execution code and the output) (if applicable)
  - The techniques used
  - Anything else we should pay attention to
- References (e.g., the feature you implemented is based on an algorithm found on the Internet or a reference book).

No marks will be given to a feature implemented without mentioning it in the report.

If the features claimed in the report can only be barely used (e.g., with a lot of bugs) or even do not exist in the source code, marks will be deducted.

Submission (Deadline: Feb. 14, 2022 11:59pm)

We expect the following files zipped into a file named by your student ID (e.g. 1155xxxxxx.zip) and have it uploaded to the course's Blackboard system.

- `report.pdf` (tell us what to pay attention to; mandatory if you have done the bonus part).
- `inverse.cpp`
- `ascii.cpp`
- `bonus.cpp` (optional)

*******IMPORTANT*******

All code should be implemented by yourself. Any kind of plagiarism (including copying online source code or/and code of classmates) in all assignment parts (both the general part and the bonus part) will not be tolerated and will be subjected to disciplinary penalties.