EG 2017 - Education Papers, pp. 17-18. The definitive version is available at http://diglib.eg.org/ (DOI: 10.2312/eged.20171021).

# Generating ASCII-Art: A Nifty Assignment from a Computer Graphics Programming Course

Eike Falk Anderson[1]

[1]The National Centre for Computer Animation, Bournemouth University, UK

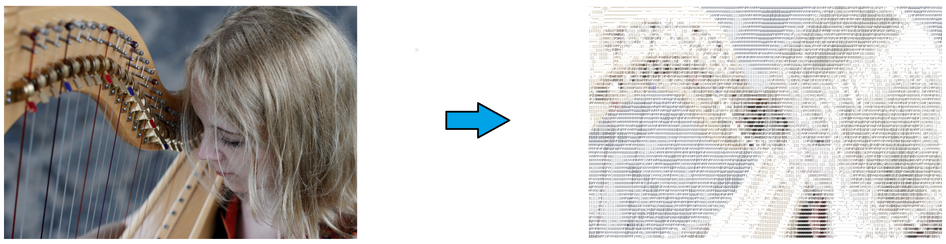


**Figure 1:** *Photo to coloured ASCII Art – image conversion by a successful student submission.*

**Abstract**

*We present a graphics application programming assignment from an introductory programming course with a computer graphics focus. This assignment involves simple image-processing, asking students to write a conversion program that turns images into ASCII Art. Assessment of the assignment is simplified through the use of an interactive grading tool.*

## 1. Educational Context

The assignment is used in the introductory computing and programming course "Computing for Graphics" (worth 20 credits, which translate into 10 ECTS credits in the European Credit Transfer and Accumulation System). The course runs in the first year of one of the programmes of our undergraduate framework for computer animation, games and effects [CMA09] at the National Centre for Computer Animation (NCCA). Students of this BA programme aim for employment as a TD (Technical Director) for visual effects and animation or as a technical artist in the games industry.

## 2. Course Aims and Assessment

This course emphasizes procedural programming (using the C programming language), with a focus on important techniques and concepts that are useful for computer graphics, computer games and computer animation. At the assessment stage, students are expected to demonstrate the ability to implement simple CG algorithms using an appropriate graphics API as well as to design and implement a computer program employing suitable software engineering principles.
The course is assessed through an exam and a coursework assignment (each counting for 50% of the grade), the latter of which typically takes the form of a short (4-week) project at the end of the course. This assignment requires the design and implementation of a computer graphics application supported by a report discussing its design and implementation to assess the students' programming competence at the end of the introductory programming sequence.

## 3. Coursework Assignment

The assignment assesses software development practice, knowledge of procedural programming concepts as well as understanding of basic 2D computer graphics techniques and algorithms. One of the project options – ranging from simple pixel paint programs to image processing tools – is the automatic generation of ASCII Art [Wik17] from 2D images, such as photographs. There exist a number of more or less complex approaches for achieving this [Mik12, XZW10, OR08], but a simple approach is to assign brightness values to character pixel-blocks (automated [Mik12] or manually [Par11]) and mapping these to image areas (blocks) of similar brightness.
Our students use Linux workstations employing *Makefiles* to build projects created with the Geany (`https://www.geany.org`) code editor and the Clang (`http://clang.llvm.org`) C compiler. Graphics context, image and font handling is provided by the SDL2 library (`https://www.libsdl.org`).

## 3.1. Assignment Brief: ASCII Art Image Conversion

*Your task is to write a computer program in C that converts source image(s) to ASCII art, i.e. a representation of the original image using ASCII characters only.*
*The program you create should allow the user to select an image or a sequence of images (e.g. animation frames) and then process them (match pixel regions with ASCII characters of comparable intensity/brightness), saving the results in a format selectable by the user, which should be either as ASCII text files, an image file of ASCII characters (with a selectable text and background colour) or as an image of coloured ASCII characters (using the average colour of the pixels they replace) on a neutral background. Your submission needs to include the following:*

1. *One or more well commented source code files and a Makefile that will build the program without errors on any machine in the labs.*
2. *A PDF user Manual for your application. The User Manual should explain how to initialise and run your program.*
3. *A PDF report documenting and explaining your application. The report should be approximately 6–8 A4 pages of text and should not exceed 10 A4 pages. The report should contain a section called "Background" or "Introduction" explaining the algorithms, techniques, and ideas used in your project. It should also have a section called "Implementation" explaining the structure of your program in terms of the implementation of the algorithms and techniques used, describing flow of control, and explain the implementation of the most important functions or procedures. This section should serve to illuminate but not replicate your code. Finally the report should contain a section called "Results" which includes images, or references to images (and video) demonstrating and explaining the results produced by your program.*
4. *Appropriate sample results (and source data, such as original images where this applies), such as animations or images generated by your program.*

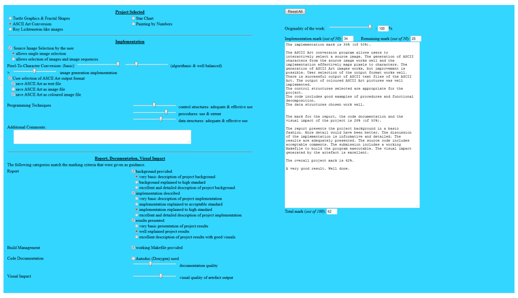
## 3.2. Additional Guidance

In the assignment's introduction lecture a basic strategy that could be employed to complete the assignment is outlined:

- First a suitable monospace (fixed-width) font (e.g. Liberation Mono on Linux) needs to be analysed, e.g. by creating images of white letters on a black background and sorting them by their brightness (average pixel colour).
- Then the source image could be divided into pixel groups matching a single character's size (width/height)for which the average colour and (converted to grayscale) average brightness is determined.
- Finally these pixel areas are mapped to the closest matching ASCII characters and printed as ASCII art.

## 3.3. Assignment Assessment

The assessment criteria for the assignment that directly relate to the quality of the source code and usability of the program – including effective use of functional decomposition, relevant control structures and data structures – count for 50% of the assignment grade.



**Project Selected**
Turtle Graphics & Fractal Shapes [✓ ASCII Art Conversion] Roy Lichtenstein-like images Star Chart Painting by Numbers

**Implementation**
[✓ Source Image Selection by the user]
[✓ allows single image selection] allows selection of images and image sequences
Pixel-To-Character Conversion: (basic) (algorithmic & well balanced)
> image generation implementation
[✓ User selection of ASCII Art output format]
[✓ save ASCII Art as text file]
save ASCII Art as image file
[✓ save ASCII Art as coloured image file]

Programming Techniques
control structures: adequate & effective use
procedures: use & extent
data structures: adequate & effective use

Additional Comments:

**Report, Documentation, Visual Impact**
The following categories match the marking criteria that were given as guidance.

Report
[✓ background provided]
[✓ very basic description of project background] background explained to high standard excellent and detailed description of project background
[✓ implementation described]
very basic description of project implementation implementation explained to acceptable standard [✓ implementation explained to high standard] excellent and detailed description of project implementation
[✓ results presented]
very basic presentation of project results [✓ well explained project results] excellent description of project results with good visuals

Build Management
[✓ working Makefile provided]

Code Documentation
Autodoc (Doxygen) used
documentation quality

Visual Impact
visual quality of artefact output

Originality of the work: - 100 %
Implementation mark (*out of 50*): 34 Remaining mark (*out of 50*): 28

```
The implementation mark is 34% (of 50%).

The ASCII Art conversion program allows users to
interactively select a source image. The generation of ASCII
characters from the source image works well and the
implementation effectively maps pixels to characters. The
generation of ASCII Art images works, but improvement is
possible. User selection of the output format works well.
There is successful output of ASCII text files of the ASCII
Art. The output of coloured ASCII Art pictures was well
implemented.
The control structures selected are appropriate for the
project.
The code includes good examples of procedures and functional
decomposition.
The data structures chosen work well.


The mark for the report, the code documentation and the
visual impact of the project is 28% (of 50%).

The report presents the project background in a basic
fashion. More detail would have been better. The discussion
of the implementation is informative and detailed. The
results are adequately presented. The source code includes
acceptable comments. The submission includes a working
Makefile to build the program executable. The visual impact
generated by the artefact is excellent.

The overall project mark is 62%.

A very good result. Well done.
```

Total mark (*out of 100*): 62

**Figure 2:** *Interactive Grading Tool (included with this paper as supplemental material): Sliders and buttons mapped to the assessment criteria generate detailed feedback that can be copied and pasted into the coursework feedback forms used in our faculty.*

The remaining 50% of the grade are determined by the project report, the 'visual impact' of the generated artefacts – based on the 'Results' section of the report and/or submitted artefacts generated by the program – and the source code documentation (provision of relevant and appropriate comments in the source code). To speed up the grading process and to ensure that grading and feedback for large numbers of submissions are consistent, we have created a web-based grading tool aligned to the assessment criteria that suggests a grade and generates appropriate feedback text (Figure 2).

## 4. Discussion

The ASCII Art assignment option is suitable for assessing all of the course aims. It is also very popular, e.g. in the 2014/2015 academic year it was selected by 88 of a total of 113 students. Of these, more than 80% passed, with the students who auto-generated the brightness values generally achieving higher grades.

## 5. Acknowledgements

The teaser image was generated using the assignment submission by Lucy Devlin. Original image courtesy of Yuri Birte Anderson.

## References

[CMA09] COMNINOS P., MCLOUGHLIN L., ANDERSON E. F.: Educating technophile artists: Experiences from a highly successful computer animation undergraduate programme. In *ACM SIGGRAPH ASIA 2009 Educators Program* (2009), pp. 1:1–1:8. 1

[Mik12] MIKOLAY M.: A basic ascii art algorithm, 2012. [accessed 10-January-2017]. URL: http://mattmik.com/articles/ascii/ascii.html. 1

[OR08] O'GRADY P. D., RICKARD S. T.: Automatic ascii art conversion of binary images using non-negative constraints. In *IET Irish Signals and Systems Conference (ISSC 2008)* (2008), pp. 186–191. 1

[Par11] PARBERRY I.: Ascii art on a pixel shader, 2011. [accessed 10-January-2017]. URL: http://larc.unt.edu/ian/art/ascii/shader/. 1

[Wik17] WIKIPEDIA: Ascii art — wikipedia, the free encyclopedia, 2017. [accessed 5-January-2017]. URL: https://en.wikipedia.org/w/index.php?title=ASCII_art. 1

[XZW10] XU X., ZHANG L., WONG T.-T.: Structure-based ascii art. In *ACM SIGGRAPH 2010 Papers* (2010), pp. 52:1–52:10. 1