Turtle Fractals and Spirolaterals: Effective Assignments for Novice Graphics Programmers

Eike Falk Anderson[1]

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

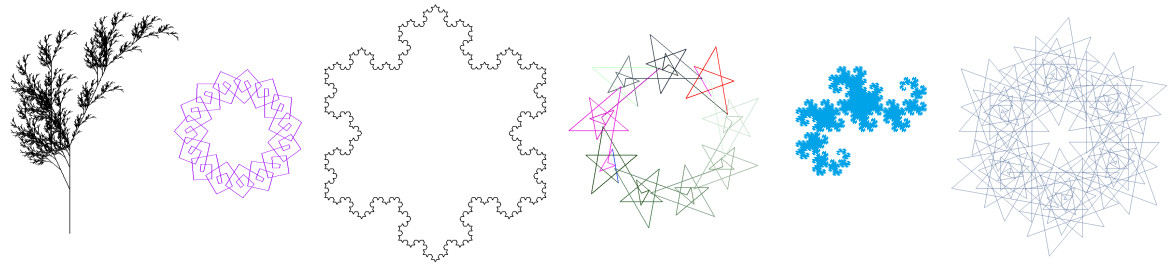

Figure 1: *Example fractals and spirolaterals from student submissions.*

Abstract

This paper presents an effective assignment in the shape of a computer graphics application from an introductory computing course with a graphics programming flavour. The assignment involves basic 2D computer graphics used in combination with fundamental algorithmic elements to create a simple drawing application. Students are first asked to create a data structure and appropriate functions replicating the operations of a Turtle graphics system and then to use this turtle for drawing either fractals or spirolateral curves.

1. Assignment Context

For several decades now, Turtle graphics [GSJ04] has been successfully used to teach programming concepts to all age groups, from small children [SP76] to university students [CC00] (see also `http://nifty.stanford.edu/2013/roberts-turtlegraphics` [PZC*13]). Turtle graphics have also been employed in combination with spirolaterals [Ad81] as well as fractals [PJS92], such as the space-filling Hilbert curve.

The Turtle graphics assignment presented here (see Table 1) has undergone a number of revisions over the past seventeen years and been successfully used for the assessment of several introductory computing and programming courses in the programmes of our undergraduate framework for computer animation, visual effects and computer games [CMA10] at the National Centre for Computer Animation (NCCA).

In its most recent incarnation, the assignment counts for 50% of the grade of the course "Programming Principles" (worth 20 credits, which translate into 10 ECTS credits in the European Credit Transfer and Accumulation System). This course, which covers the introductory programming sequence (as well as basic raster graphics algorithms) and which runs in the first semester of the first year of the "Computer Animation Technical Arts" BA programme, is assessed through a practical coursework assignment in the shape of a short project at the end of the semester (set 4 teaching-weeks before the end of the semester), as well as a written examination (counting for the other 50% of the grade). Graduates of this programme typically aim aim for a career as a Technical Director (TD) in the feature animation or visual effects industries or – if they intend to work in the games industry – as a technical artist.

2. Objectives and Assessment of the Course

The emphasis of the course lies on procedural programming, which is taught in the C programming language, focussing on first principles and fundamental concepts that are applicable to software development for computer graphics, especially for games and animation.

At the conclusion of the course, i.e. at the assessment stage, it is expected of students to be able implement simple algorithms,

DOI: 10.2312/eged.20181005

Summary	Students implement a Turtle graphics system & use this to create fractal/spirolateral drawings.
Learning Outcomes	Understanding of programming principles demonstrating their practical application. Designing a simple application by combining simple programming concepts with basic 2D graphics and implementing this.
Classification(s)	Fundamentals; Graphics & Interfaces.
Audience	Undergraduate students at the end of the introductory programming sequence (CS1).
Dependencies	Requires knowledge of programming basics; familiarity with computer graphics is not needed; knowledge of what Turtle graphics is can be beneficial but is not required.
Prerequisites	None.
Strengths	The simplicitly with which complex graphical patterns can be created.
Weaknesses	A tendency by some to overestimate the complexity of creating a turtle (possibly due to lacking experience with functions and data structures); some students first hard-code the drawing of the fractals/spirolaterals (sometimes within the program's main function), and then try to refactor their code to use Turtle graphics instead, which does not always work smoothly.
Variants	Two assessment versions with option to explore different user interface types & animation; could use existing Turtle graphics system.
Assessment	Source code quality (incl. source documentation), program usability, documentation (report) & visual results (meeting the brief).

Table 1: *Effective Assignment metadata.*

including fundamental CG algorithms. For this they should both apply suitable software engineering principles for the design and successfully employ functions from an appropriate graphics API in the implementation of a computer graphics application, which they are asked to produce as their coursework assignment.

In addition to this software artefact, students are required to support their assignment submission with a report that both discusses and critically evaluates the design and implementation of their program, providing additional insight into the students' understanding of the subject matter, and ultimately their attainment of the learning outcomes, at the conclusion of the introductory programming sequence.

3. Coursework Assignment

The assignment is used to assess a range of different aspects of the introductory programming sequence, including knowledge of fundamental data structures and algorithm design concepts, procedural programming concepts such as functional decomposition and software development practice. As the course has a focus to applied computing for computer graphics, the assignment also assesses the students' grasp of basic 2D computer graphics algorithms and techniques.

When the assignment is set, students are presented with a a number of project options, e.g. the generation of ASCII art from photographs [And17], from which they have to select one. The selection includes one option that asks the students to replicate the operations of a Turtle graphics system, first creating a turtle data structure and appropriate functions, and then to use this system for drawing. The resulting graphics depend on the version of the assignment: one version asks students to draw fractals [IS13], such as the space-filling Hilbert curve or a Dragon curve, e.g. a Heighway Dragon; the second version instead asks students to draw spirolateral curves [Kra99, Kra00]. The two versions of this assignment option are set alternatingly in different iterations of the course: in one year the turtle fractals version may be used in the main assessment (with the other version used for reassessment), whereas in the next year the turtle spirolaterals version would be set.

Our students use Linux workstations that are provided with the SDL2 library (`https://www.libsdl.org`) for the creation of the windowed graphics/rendering context, as well as image handling. The build environment includes the code editor Geany (`https://www.geany.org`) in combination with the C compiler Clang (`http://clang.llvm.org`).

3.1. Assignment Brief: Turtle Graphics & Drawing Spirolaterals/Fractals

Turtle graphics [GSJ04]is a simple and well know method for drawing complex shapes from simple lines. Turtle graphics employs the concept of a "turtle", which is a simple robot or agent in the shape of a turtle that has the following attributes (usually implemented using a record data structure):

- *a **position** (x, y) in 2D space,*
- *a **heading** or **angle** measured in degrees or radians, and*
- *a flag/variable **pen** that could be set either as Down (True/drawing) or as Up (False/not drawing)*

If the turtle's pen is set to have the value Down and the turtle moves then the turtle leaves a trace while it moves. It accepts the following simple instructions (usually implemented as functions):

- ***pen_up** sets the state of the turtle's pen to Up,*
- ***pen_down** sets the state of the turtle's pen to Down,*
- ***move n** makes the turtle move n units along its heading (usually implemented by drawing a line segment between the start point and the end point), and*
- ***turn** α turns the heading of the turtle by α degrees.*

3.1.1. Spirolateral version

The text below is used if the turtle spirolaterals assignment is set:
A spirolateral is a geometric shape/pattern that is constructed from a series of straight lines of growing length that are attached at an angle, which, if repeated after one another, will (eventually) result in a closed curve [Wei18]. Your task is to implement a simple turtle library (data structure & a set of related functions – not implemented as an L-system) in C, employing the SDL library, and use this to create a spirolateral drawing program with a simple user interface (that may be graphical or text-based) that is capable of the following:

1. *allow the user to select/change the drawing parameters such as drawing and background colours, the initial length and number of line segments. The number of repetitions, the drawing angle etc.*
2. *provide a set of (at least) three pre-set spirolaterals that are selectable by the user for drawing*
3. *save the resulting spirolaterals as image files, and optionally save the spirolateral drawing as an image sequence that shows the addition of each line segment as a separate (animation) frame that would show the curve being drawn step by step (segment by segment)*

3.1.2. Fractal version

The text below is used if the turtle fractals assignment is set:
Fractals, i.e. self-similar/self-replicating (recursive) shapes, are often used in modelling and animation to represent organic structures likes trees and plants [man89]. Fractal shapes based on straight lines, such as fractal curves [CG87] are relatively simple to construct. Your task is to implement a simple turtle library (data structure & a set of related functions – not implemented as an L-system) in C, employing the SDL library, and use this to create a fractal drawing program with a simple user interface (that may be graphical or text-based) that is capable of the following:

1. *draw (at least) three different turtle fractals that are interactively selectable by the user (e.g. Koch snowflake, Sierpiński arrowhead curve and Hilbert curve)*
2. *allow the user to interactively select/change the drawing parameters such as drawing and background colours, the recursion depth of the fractal etc.*
3. *save the resulting fractals as image files, and optionally save the fractal drawing as an image sequence that either shows each level of the recursion as a separate (animation) frame or that shows each line segment at the target depth as a separate frame that would show the curve being drawn step by step (segment by segment)*

3.1.3. Submission requirements

The submission requirements are identical for both assignments.
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. Further Guidance given to Students

Various practical exercises from laboratory workshops that have been run in support of the course lectures provide students with starter code to use as the basis for their assignment. This includes code for creating a 2D graphics window using the SDL2 library, line-drawing using Bresenham's line algorithm [Bre65] and library creation. These are highlighted in the assignment's introductory lecture, in which a basic strategy that could be employed to complete the assignment is also outlined:

1. define a turtle (position, angle, drawing-status) – essentially the data that must be stored for a turtle (ideally as a record data structure)
2. define turtle functions (e.g. set-position, set-status, go-forward, turn-by-angle) – the initialisation and relevant algorithms that do the drawing (assuming that each forward step is given a drawing distance) – possibly make this its own turtle library (function parameters should accept the turtle record by reference rather than by value)
3. depending on the assignment type (spirolaterals or fractals)
   - (*for spirolaterals)* design a set of (3) spirtolaterals that can be drawn with a turtle
   - (*for fractals)* design a set of (3) fractals that can be drawn with a turtle
4. then express (implement drawing of) the graphics with the turtle
5. finally, provide some form of user control (interface – this can be text-based or graphical, the latter being considerably more complex than the former)

Students are also made aware of related literature, including most of the references used in this paper and provided with guidance regarding academic (report) writing.

3.3. Assessment Criteria

The assessment criteria for these assignments mainly relate to the quality of the source code and the project report. Only 20% of the grade are derived from the submission's meeting of the functionality criteria stated in the assignment brief (as described in sections 3.1.1 and 3.1.2), which also include the overall usability of the program. Criteria directly relating to source code quality – including the appropriate use of relevant control structures and data structures as well as effective functional decomposition – count for 30% of the assignment grade.

The remaining 50% of the grade are determined by the project report (20%), the visual output, i.e. judging how well the intended results (generated images) are achieved in terms of the brief (15%) – based on the 'Results' section of the report and/or submitted artefacts generated by the submitted program – and the source code documentation (provision of relevant and appropriate comments in the source code).

3.4. Assignment Variations in Previous Course Incarnations

Until about 2010 the Turtle fractals assignment was used in the assessment of the "Computer Programming 1" course [CMA09] – in this course the weighting of the assignment was 40% of the course grade – with almost identical learning outcomes to the current course. The main difference to the current assignment was that students were given an existing Turtle graphics library as starter code. After a course redesign and the integration of the course into an arts degree, the course "Computing for Graphics" [CMA10] (in its first incarnation) used Python with the 'turtle' module (https://docs.python.org/2/library/turtle.html) for the assignment. With the reintroduction of C as the main course language after a further programme redesign, in the 2014/2015 academic year the assignment as presented here was introduced.

4. Discussion

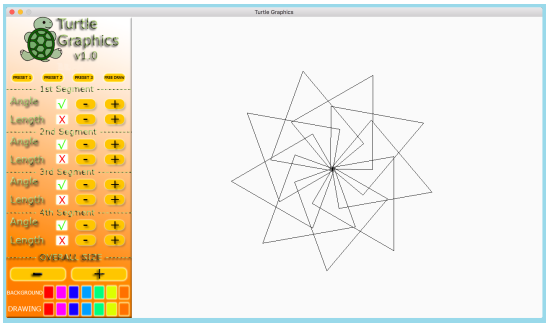

Figure 2: *Student work example – spirolateral curves with GUI.*

Unlike other Turtle graphics assignments (e.g. [PZC*13]), the assignment presented here first expects students to create a simple turtle graphics system, before using it to create a graphics application, although – as previous incarnations have shown – the assignment can just as well be used with an existing Turtle graphics system. The assignment is also platform, language and paradigm agnostic – e.g. if Object Orientation were used, both turtle and graphics (fractal shapes/spirolateral curves) could be implemented as classes.

Over the past three years the pass rate for the assignment has been 95% (average grade 60%). The spirolaterals version seems to be more popular with students than the fractals version, i.e. in years when the spirolaterals version was provided as a project option a three times larger percentage of student cohort took this up than the fractals version in years when this was set instead – reasons for this are unclear, but it is possible that the fractal concept may appear more intimidating to students than spirolaterals. In any case, both versions of the Turtle graphics assignment provide an effective means for assessing all of the course aims and objectives.

5. Acknowledgements

The images showing student work were generated from submissions by (left to right) Quentin Corker-Marin, Karo Nguyen, Jess Cheung, Ruiqui Wang, Carola Gille, Georgia Fearnley (Figure 1), Jake Cross (Figure 2), Ruiqui Wang (Figure 3).

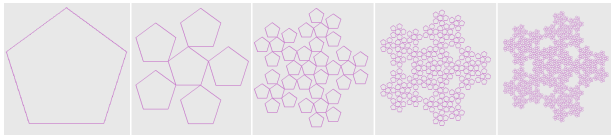

Figure 3: *Student work example – fractal image sequence of increasing fractal depth.*

References

[Ad81] ABELSON H., DISESSA A.: *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*. The MIT Press, 1981. 1

[And17] ANDERSON E. F.: Generating ASCII-Art: A Nifty Assignment from a Computer Graphics Programming Course. In *EG 2017 - Education Papers* (2017), Bourdin J.-J., Shesh A., (Eds.). 2

[Bre65] BRESENHAM J. E.: Algorithm for computer control of a digital plotter. *IBM Systems Journal 4*, 1 (1965), 25–30. 3

[CC00] CASPERSEN M. E., CHRISTENSEN H. B.: Here, there and everywhere - on the recurring use of turtle graphics in cs1. In *Proceedings of the Australasian Conference on Computing Education* (2000), ACSE '00, pp. 34–40. 1

[CG87] CUTTING J. E., GARVIN J. J.: Fractal curves and complexity. *Perception & Psychophysics 42*, 4 (1987), 365–370. 3

[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. 4

[CMA10] COMNINOS P., MCLOUGHLIN L., ANDERSON E. F.: Educating technophile artists and artophile technologists: A successful experiment in higher education. *Computers & Graphics 34*, 6 (2010), 780–790. 1, 4

[GSJ04] GOLDMAN R., SCHAEFER S., JU T.: Turtle geometry in computer graphics and computer-aided design. *Computer-Aided Design 36*, 14 (2004), 1471–1482. 1, 2

[IS13] IRVING G., SEGERMAN H.: Developing fractal curves. *Journal of Mathematics and the Arts 7* (2013), 103–121. 2

[Kra99] KRAWCZYK R. J.: Spirolaterals, complexity from simplicity. In *Proceedings of the 1999 Conference of The International Society of the Arts, Mathematics and Architecture* (1999). 2

[Kra00] KRAWCZYK R. J.: The art of spirolaterals. In *MOSAIC 2000: Millennial Open Symposium on the Arts and Interdisciplinary Computing* (2000). 2

[man89] Fractal geometry: what is it, and what does it do? *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 423*, 1864 (1989), 3–16. 3

[PJS92] PEITGEN H.-O., JÜRGENS H., SAUPE D.: *Fractals for the Classroom – Part Two: Complex Systems and Mandelbrot Set*. Springer-Verlag, 1992. 1

[PZC*13] PARLANTE N., ZELENSKI J., CRAIG M., DENERO J., GUZDIAL M., MALAN D. J., MURALIDHARAN A., ROBERTS E., WAYNE K.: Nifty assignments. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (2013), SIGCSE '13, pp. 539–540. 1, 4

[SP76] SOLOMON C. J., PAPERT S.: A case study of a young child doing turtle graphics in logo. In *Proceedings of the June 7-10, 1976, National Computer Conference and Exposition* (1976), AFIPS '76, pp. 1049–1056. 1

[Wei18] WEISSTEIN E. W.: Spirolateral. From MathWorld – A Wolfram Web Resource, 2018. [accessed 9-January-2018]. URL: http://mathworld.wolfram.com/Spirolateral.html. 2