Data Structures and Algorithms

COMP 251, Winter 2013

Assignment 4

Due date: Wednesday, March 27, 2013
8pm

All coding for the assignment must be in Java. All code should be well-commented, in a professional style, with appropriate variables names, indenting, etc. Your code must be clear and readable. **Marks will be very generously deducted for bad style or lack of clarity.**

1. *Bézier curves* allow you to define smooth curves using some number of *control points,* each of which influences the curve shape. Here you will use a *divide-and-conquer* strategy to build both quadratic and cubic Bézier curves in 2D. This formulation is described nicely in Kenneth I. Joy's notes, and you will need to read:

   `http://graphics.cs.ucdavis.edu/~joy/ecs178/Unit-2-Notes/Divide-and-Conquer-Bezier-Curve.pdf`
   `http://graphics.cs.ucdavis.edu/~joy/ecs178/Unit-2-Notes/Quadratic-Bezier-Curves.pdf`
   `http://graphics.cs.ucdavis.edu/~joy/ecs178/Unit-2-Notes/Cubic-Bezier-Curves.pdf`

   (a) Template code for this question is provided, accepting a few command-line arguments. The first parameter is a double value t, and it is followed by one or two control points, each of which is a pair of real numbers. We can summarize the invocation of your program as: **15**

   ```
   java Assig4_1 t x1,y1 [x2,y2]
   ```

   Note that t and all control point coordinates are strictly within $0.0 \ldots 1.0$, and so of real/double type. Also note that the control points are pairs of doubles, where each pair is treated as a single argument (i.e., separated by a comma, and *not* separated by spaces.). The template includes code that parses that format.

   Your program will produce a 1024×1024 image named `outputimage.png` as output.

   You should assume two initial end points for the curve, at $(0, 0.5)$ and $(1.0, 0.5)$. Set an absolute maximum recursive depth of 300 to avoid stack overflows. In all cases the goal is to produce a smooth curve without any visible gaps (within of course the limits of non-aliased, pixel-based output), but without overdrawing pixels too much. Note that to going to depth 300 uniformly will be extremely slow, so you will need to come up with a strategy to ensure you recurse deeply enough, but not overly deep.

   Your program should include two functions, one that produces quadratic Bézier curves, and one that produces cubic Bézier curves. If one control point is given you should compute a quadratic curve; if two control points are given you should compute a cubic curve (and may assume the second control point has an x-coordinate larger than the first).

   Ensure your code works and produces appropriate curves for different control points. What difference does choice of t make in the output image?

   (b) Specify a single control point at $(0.5, 1.0)$ (ie quadratic), and vary t. Plot the maximum recursive depth you need to descend (in any branch) to ensure a complete curve without visual gaps. What do you observe? **5**

   (c) Now, specify two control points, $(0.25, 1.0)$ and $(0.75, 0.0)$ (ie cubic), and vary t. Plot the maximum recursive depth you need to descend (in any branch) to ensure a complete curve without visual gaps. What do you observe? **5**

2. We would like to compute a similarity measure to determine how (un)alike two strings are. For this we can compute a standard "edit distance" measure, calculating the minimum number of additions, insertions, or letter-changes that change a (one-line) string into another. Here we are interested only in word-based changes, so leading, trailing, and multiple spaces between words are not significant. For example,

   distance(" `abc def`", "`abc   def `") = 0

   As an additional complication, we are here interested in a visually-biased solution, considering some pairs of letters more similar than others depending on how they appear. For instance, intuitively, "o" is more similar to "e" than to "w", and so distance("`ooo`","`eee`") < distance("`ooo`","`www`").

   (a) Describe, in pseudo-code, a dynamic programming algorithm you could use to compute similarity of strings given the above constraints. **10**

   (b) In the next question your pseudo-code will become actual code and have to make some clear decisions about which characters are more similar than others. Come up with a reasonable and detailed character-to-character similarity measure you could use within your design. Character similarity depends to some degree on the font, so for uniformity assume a basic courier font, and consider only space, and the letters/symbols: **5**

   `abcdefghijklmnopqrstuvwxyz0123456789.,-`

   The full similarity matrix will be contained within your code. Here you should describe similarity classes/groupings and relations between them.

   (c) Implement your design: write a program that accepts two strings from the above alphabet (and including spaces) on the command-line (as separate arguments) and returns the similarity measure as a real value. **10**

   nb: a template is not provided for this, but it is just a program that expects exactly two command-line arguments.

What to hand in

Submit your assignment to *MyCourses*. Note that clock accuracy varies, and late assignments will not be accepted without a medical note: **do not wait until the last minute**. Assignments must be submitted on the due date **before 8pm**.

Where possible hand in only **source code** files containing code you write. Do not submit compiled binaries or .class files. For the written answer questions submit either an ASCII text document or a .pdf file *with all fonts embedded*. Do not submit .doc or .docx files. Images (plots or scans) are acceptable in all common graphic file formats.

Note that for written answers you must show all intermediate work to receive full marks.

This assignment is worth 6% of your final grade. **50**