CPSC 424 Assignment 6

Term: September 2022, Instructor: Alla Sheffer, sheffa@cs.ubc.ca
https://www.students.cs.ubc.ca/ cs-424

Due: Nov 25, 2022 in class

Assignment 6.1: Polygon Subdivision (5 Points)

In this part, you will implement a couple of subdivision schemes using either MATLAB or Julia. The assignment template can be downloaded from the course website.

- **MATLAB**:

  The main file for this part of the assignment is `driver_q1.m`. One option to run the program is to open `driver_q1.m` in MATLAB, and click "Run." The goal is to implement the function `subdivide()`, found in the file `subdivide.m`.

- **Julia**:

  Two Jupyter notebooks are provided for running Julia interactively. The file for this part of the assignment is `driver_q1.ipynb`. In the Jupyter notebook interface, navigate to the directory containing the file, then open the file. Note you may need to install certain packages in order to run the notebook.

`subdivide()` takes 5 parameters:

- `X` and `Y`, the x- and y-coordinates of the vertices of the polygon to subdivide.
- `num_iterations`, the number of subdivision iterations to perform.
- `scheme`, a flag indicating the subdivision scheme that should be used.

  Given a closed polygon defined by the sequence of points $P_i = (p_0^i, p_1^i, ..., p_{n-1}^i)$, consider the following subdivision scheme that generates after one step of subdivision the polygon $P_{i+1} = (p_0^{i+1}, p_1^{i+1}, ..., p_{2n-1}^{i+1})$ using the following subdivision rules (note that if $j = 0$, then $j - 1$ becomes $n - 1$, and if $j = n - 1$, then $j + 1$ becomes 0):

  – if `scheme` = 1, Cubic B-spline subdivision scheme should be used:
    * $p_{2j-1}^{i+1} = \frac{1}{8}p_{j-1}^i + \frac{3}{4}p_j^i + \frac{1}{8}p_{j+1}^i$
    * $p_{2j}^{i+1} = \frac{1}{2}p_j^i + \frac{1}{2}p_{j+1}^i$

  – if `scheme` = 2, the following subdivision should be used:
    * $p_{2j-1}^{i+1} = \frac{1}{9}p_{j-1}^i + \frac{7}{9}p_j^i + \frac{1}{9}p_{j+1}^i$
    * $p_{2j}^{i+1} = \frac{1}{2}p_j^i + \frac{1}{2}p_{j+1}^i$

- `to_limit`, a boolean flag indicating whether each vertex should be moved onto the limit curve after subdivision is performed.

Start with the `to_limit` flag set to 0. Your implementation should first subdivide the given polygon `num_iterations` times using the specified subdivision scheme, and then return the subdivided polygon's vertices in the `X` and `Y` variables (MATLAB) or a tuple (Julia).

Only after this part is working, if `to_limit==1`, use eigen-analysis to move each output vertex onto the limit curve. You can use any Matlab built-in functions (*e.g.* `eig()`) or Julia functions (*e.g.* `eigvals(), eigvecs()`) to compute the eigenvalues/eigenvectors of the subdivision matrix. Alternatively, you can precompute and hardcode the required eigenvalue/eigenvector information.

Note that both in MATLAB and Julia, all indices start from 1, so you might need to modify the formulas to take this into account.

Assignment 6.2: Differential Geometry (5 Points)

For this question you will need to write code that computes tangents, curvature, normals, and arc length for the following curves, where $t \in [0, 2]$:

$$
\begin{aligned}
&(1) \quad x(t) = t^2 && y(t) = 2t \\
&(2) \quad x(t) = t - 1 && y(t) = (t+1)^3 \\
&(3) \quad x(t) = cos(\frac{\pi}{2}t) && y(t) = sin(\pi t)
\end{aligned}
$$

We have provided a function, `draw_curve_helper(pos, curvature, arclength, tangent, normal)`, that handles the visualization for you. Therefore, to visualize the values along a curve, compute them at a set of positions on the curve, and then pass the positions and computed values into `draw_curve_helper()`. If your vectors are too big for the plot canvas, multiply them by a constant scale only at the visualization step. Also, make sure that you read the comments after `draw_curve_helper()` in order to use it properly.

Your task is to "fill in the blanks" in the provided MATLAB file `driver_q2.m` or Julia file `driver_q2.ipynb`. Code has been provided that defines the parametric functions that make up each curve, evaluates each curve at set of t-values, and then draws the curve. You should add code to compute tangents, curvature, normals, and arc length at each t-value.

You can use either hard-coded formulas (likely easier) or symbolic differentiation tools to complete this question. If you wish to use symbolic differentiation, the relevant command is `diff()`, which takes a symbolic function as the main argument. We have provided all of the parametric functions as symbolic functions in the script to make it easier to use `diff()`. Note that in MATLAB, in order to evaluate a symbolic function numerically, you first need to convert it using `matlabFunction()`. See the provided code for examples.

You are welcome to write your own functions in order to complete this task.

Submission

Submit your files and a README file containing your name and student number with the `handin` command that you should know from other courses:

```
handin cs-424 a6
```