Project Proposal: Interactive Explainers for Geometry Processing Algorithms

Jerry Yin and Jeffrey Goh

1 Introduction

Every two years, the University of British Columbia offers an undergraduate course in geometric modelling (CPSC 424). The course begins with a detailed foray into different ways of modelling curves, such as Bézier curves, Hermite curves, and B-splines. The course then moves from 2D to 3D by discussing methods for modelling surfaces and solids. The course differs from related courses in the mathematics department in that it explicitly discusses *meshes*: in particular triangle meshes, mesh data structures, and mesh algorithms. Meshes are ubiquitous in the field of computer graphics, and have a variety of practical applications.

Because the course is offered infrequently and comparatively few students take the course (due to it being a fourth year course), the notes for the course can be somewhat sparse. In particular, the course lacks detailed notes for many topics, and often uses third-party interactive demos for the first half of the course (see Previous Work section for examples). However, there are a lack of interactive demos on the internet for the second half of the course (meshes), and students must download 3D models and install (and learn) software such as MeshLab or a 3D modelling application in order to interactively explore some of these topics.

Our project aims to create a set of detailed, interactive, and accessible online notes for current and future students of CPSC 424. We will focus on content covered in the second half of the course (meshes), since the first half of the course is already over, and there are already many sufficiently good interactive demos online for curves. Since this project is for a visualization course, the notes will emphasize *visual explanations*, choosing to teach via interactivity and visuals wherever possible, rather than relying mainly on text as in a traditional textbook.

2 Personal expertise

2.1 Jerry Yin

Jerry is currently a teaching assistant for the undergraduate geometric modelling course (CPSC 424) at UBC, and is also researching computational geometry algorithms as part of his Masters degree. He has experience with writing and teaching tutorials on topics in geometric modelling, and has been creating interactive demos with the Desmos graphing application for the current set of tutorial notes. He also has experience with JavaScript, HTML, CSS, and building static websites.

2.2 Jeffrey Goh

Jeffrey is interested in web development and machine learning. During his postgraduate studies, he worked part time as a full stack web developer for a startup in Brisbane, Australia. Besides having experience programming in the frontend (AngularJS) and backend (Python), Jeffrey has used D3.js to visualize information for a university project. Other university projects include a fully functional recipe website, social media analysis and music genre classification using machine learning and signals.

- *Jerry Yin is with The University of British Columbia. E-mail: jerryyin@cs.ubc.ca.*
- *Jeffrey Goh is with The University of British Columbia. E-mail: gohzenhao@gmail.com.*

3 Data and Task

As part of this project, we are planning to create a series of interactive articles on topics in computational geometry, for the benefit of students in the undergraduate geometric modelling course at UBC. These articles are intended to be in the style of interactive articles such as those in the online Distill journal [4]. These "interactive explainers" differ from traditional articles in that they include *reactive diagrams* and animations to assist with the text explanations. Reactive diagrams are diagrams that can be interacted with by the reader, providing a controlled environment in which the reader can experiment with different parameters of the visualization as part of deepening their understanding of the topic being explained.

For our reactive diagrams, we plan on using hand-crafted datasets (geometric models, for the most part) that are designed to be instructive and simple enough to be interactive. Models should be small enough that any changes the user make should be displayed immediately. For example, the simplest 3D model, a tetrahedron, only has four vertices and six edges, but we also plan on providing more interesting models where applicable, such as the Stanford bunny. The more interesting models will be as large as possible (likely around a thousand vertices) while still allowing fast responses to user interactivity. Geometric models are networks, where vertices contain at least position information, and edges connect vertices. Models also have a collection of faces, where a face is defined as a cycle of edges.

4 Proposed Solution

We plan on writing and implementing articles on a small number of topics in geometry processing. Our first article will be on the *half-edge data structure*, a popular in-memory data structure for manipulating meshes. The second article will be on the *mesh subdivision algorithm*.

The design of the articles will follow those of the existing tutorial notes written by Jerry [12], which are in turn based on the design of Edward Tufte handouts. Fig. 3 shows what the notes currently look like.

The following subsections describe some potential visualizations we could implement.

4.1 Half-edge visualizations

For visualizing half-edge data structures, we plan to implement the visualization mocked up in Fig. 1. The OBJ editor, inspired by the live-reloading JPEG editor in "Unravelling the JPEG" [10], displays the contents of a mesh in the simple and popular OBJ text format. The user can edit the contents of the OBJ file, and the half-edge diagram (top right) and memory layout tables (bottom) will update in real-time.

The half-edge diagram shows the mesh laid out in 2D. Its marks are coloured based on type: boundary half-edges are coloured blue, interior half-edges are coloured red, faces are coloured orange, and vertices are coloured black. In addition, vertices, half-edges, and faces are labelled based on their names in the memory layout table. The positions of the vertices are determined by the positions specified in the data (the OBJ file).

If the user hovers their cursor over any vertex, edge, or face in either the half-edge diagram or the memory layout table, the same item will be highlighted in all other places it appears.

If we have time, interactivity will be bidirectional. The user will be able to move vertices in the half-edge diagram and have the positions update in the OBJ editor and memory layout tables in real-time. More

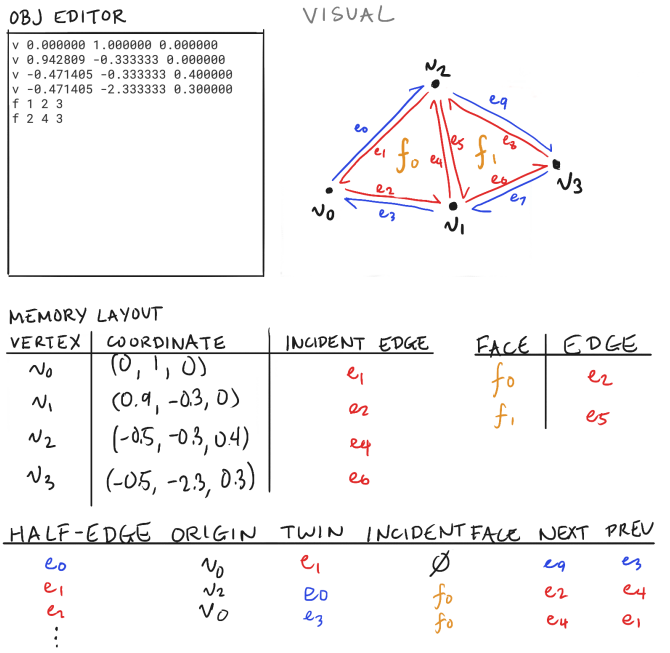

Fig. 1. Half-edge data structure visualization.

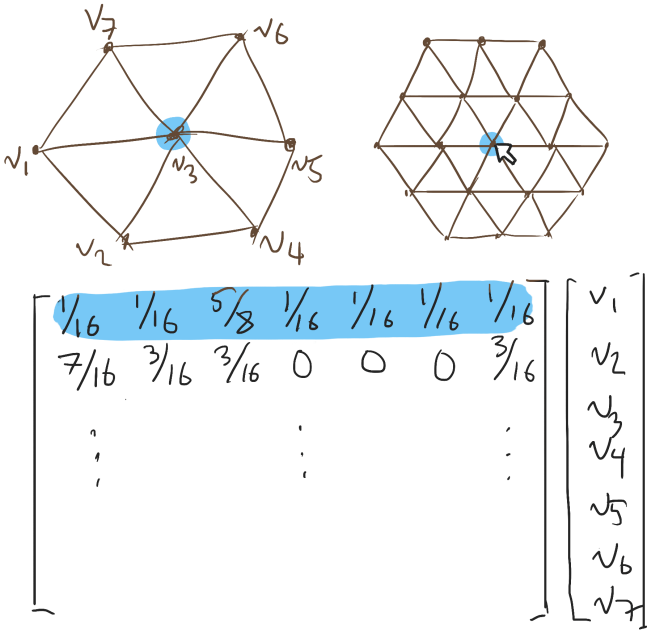

Fig. 2. Mesh subdivision visualization.

interestingly, if we allow users to edit the memory layout tables, then the user will be able to corrupt the half-edge data structure. After corrupting the half-edge data structure, the user can try to repair it by making further changes to the memory layout tables.

4.2 Mesh subdivision visualizations

To visualize the subdivision algorithm, we plan to implement the visualization mocked up in Fig. 2. At the top, we display a mesh laid out in 2D before and after subdivision. At the bottom, we display the subdivision matrix for this mesh. If a user hovers over a vertex in the subdivided mesh, its corresponding row in the subdivision matrix is highlighted. The vertex in the original mesh corresponding to the hovered vertex is also highlighted, if it exists (not all subdivided vertices correspond to vertices in the original mesh).

The user can also drag the vertices in the original mesh to change their positions and see the subdivided mesh change in real-time. In addition, the user can also edit the weights in the subdivision matrix to see how different weights affect the subdivision scheme.

5 Proposed Implementation Approach

Once finished, our project will consist of a small number of single-page articles, one per topic. These articles will be generated with the Idyll toolkit [2], which provides a React container and implementations of common interactive web components written in JavaScript. We will likely need to create some custom components of our own, in addition to leveraging some of theirs. Any 3D visualization will likely be done with regl. There have been a couple of examples of getting regl working with Idyll [1, 9].

6 Milestones and Schedule

In total, we intend to spend around 104 hours on the project across our entire team. Table 1 describes the breakdown of this number, along with estimated dates by which we will complete each task.

Most of the writing tasks regarding computer graphics/meshes will be done by Jerry. Implementation-related tasks will be done together, although Jeffrey will perform slightly more work in this area.

7 Previous Work

A substantial amount of work has been done on interactive explainers. For example, an interactive system has been implemented to view complex 3D models [7]. The diagram of the complex 3D model changes based on the interaction of users. User interactions included clicking, dragging and rotating. The method used in the paper was creating exploded views, which was different from traditional conventions. An iterative algorithm was used in determining how to remove unblocked parts of the 3D model. The paper serves as a good guideline on creating and viewing interactive interactive exploded views with 3D models. Although our project will most likely not use exploded views, it will be similar to our project as our goal is to create an interactive system to allow easier understanding of 3D models for students. Other related work includes [6] and [11]. One of the papers used exploded views as well and the papers provided explanations on subdivision/meshes which are relevant to our project. What can be added on top of this is to visualize the explanation in an interactive manner to give students a better experience. A wide variety of examples on interactive explainers can be found on the Idyll website [5], where past works that were done using Idyll are displayed.

A few interactive articles and demos are already being used in CPSC 424. For instance, an article on Lagrange curves demonstrating Runge's phenomenon [3] is shown in lecture. The site allows the user to modify the control points of a Lagrange curve and also vary its knots. A plot of the curves and a plot of the Lagrange basis functions for the user-specified knots are juxtaposed. The same site has many other interactive articles on splines. Other interactive demos include one on manipulating Hermite curves [8] and one visualizing the de Casteljau algorithm (made by Jerry) [13], although these are not articles because they do not contain any explanation.

References

[1] M. Conlen. Using regl with Idyll. https://web.archive.org/web/20191031190033/https://idyll-lang.github.io/idyll-regl-component/, 2017.

[2] M. Conlen and J. Heer. Idyll: A markup language for authoring and publishing interactive articles on the web. In *ACM User Interface Software & Technology (UIST)*, pp. 977–989, 2018.

[3] E. Demidov. Interpolating Lagrange curve. https://web.archive.org/web/20191104080854/https://www.ibiblio.org/e-notes/Splines/lagrange.html, 2001.

[4] Distill: Latest articles about machine learning. https://distill.pub. Accessed: 2019-11-01.

[5] Idyll: A toolkit for creating data-driven stories and explorable explanations. https://idyll-lang.org. Accessed: 2019-11-01.

[6] O. Karpenko, W. Li, N. Mitra, and M. Agrawala. Exploded view diagrams of mathematical surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1311–1318, 2010.

[7] W. Li, M. Agrawala, B. Curless, and D. Salesin. Automated generation of interactive 3D exploded view diagrams. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pp. 101:1–101:7, 2008. doi: 10.1145/1399504.1360700

[8] Lior. General cubic hermite spline demo. https://codepen.io/liorda/full/KrvBwr. Accessed: 2019-11-03.

[9] R. Reusser. From nothing to something in WebGL using regl. https://web.archive.org/web/20191031190014/https://rreusser.github.io/from-nothing-to-something-in-webgl-with-regl/, 2016.

[10] O. Shehata and M. Conlen. Unravelling the JPEG. *Parametric Press*, 2019.

[11] J. J. van Wijk. Unfolding the earth: Myriahedral projections. *The Cartographic Journal*, 45(1):32–42, 2008. doi: 10.1179/000870408X276594

[12] J. Yin. CPSC 424 tutorial notes. https://www.students.cs.ubc.ca/~cs-424/tutorials/. Accessed: 2019-11-04.

[13] J. Yin. De Casteljau's algorithm. https://www.desmos.com/calculator/s78usaowv9. Accessed: 2019-11-03.

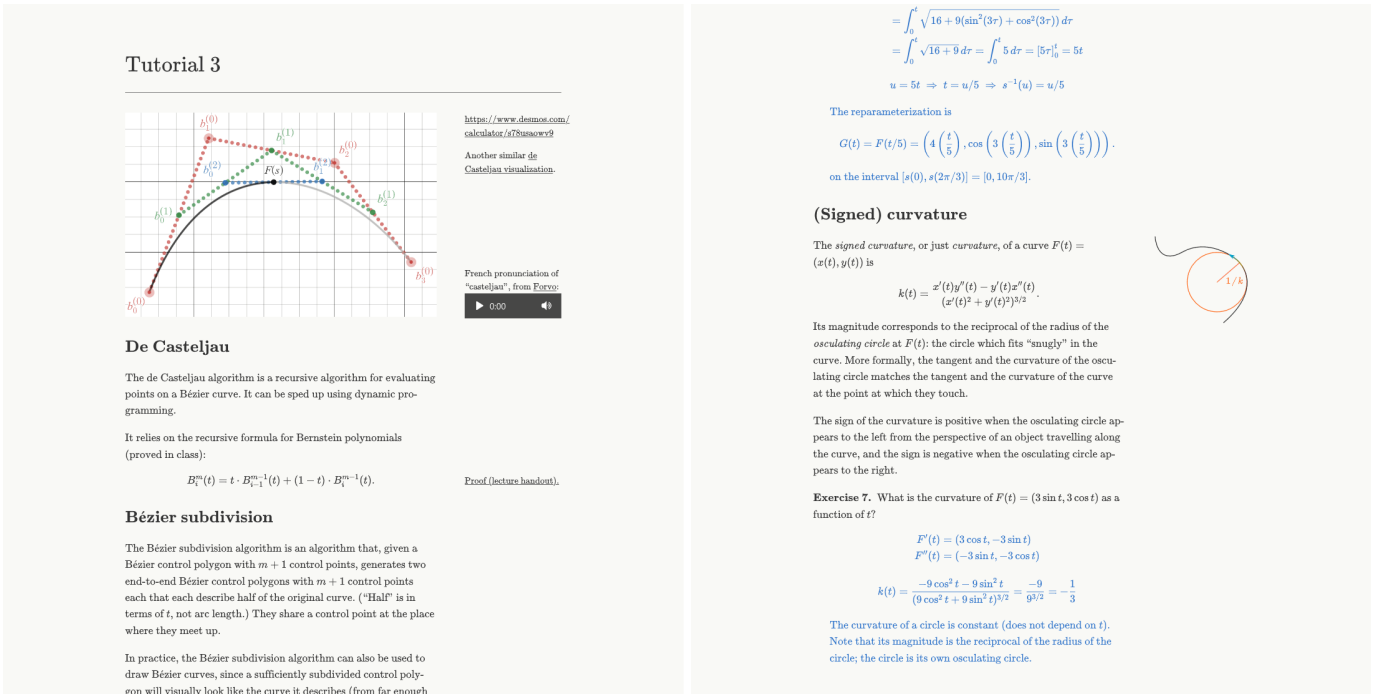

Fig. 3. Some of the current tutorial notes for CPSC 424. In the notes on the left, the image links to an interactive demo of the de Casteljau algorithm for drawing Bézier curves. The notes on the right show a version of a tutorial with solutions to exercises in blue.

Table 1. Proposed schedule for deliverables. Hard (course) deadlines are in italics.

Task	Est. time (hours)	Deadline	Description
Project proposal	10	*Nov. 4*	*Write the project proposal (this document).*
Set up Idyll	2	Nov. 6	Set up Idyll project skeleton and development environment (1 hour/person).
Create first article (implementation)	21	Nov. 15	Create half-edge data structure visualizations
Create first article (writing)	5	Nov. 15	Write text explanations for half-edge data structures
Peer project review 1	2	*Nov. 19*	*Prepare slides and plan a demo.*
Create second article (implementation)	21	Nov. 29	Create visualizations on second topic (either mesh subdivision or simplification)
Create second article (writing)	5	Nov. 29	Write second article text
Peer project review 2	2	*Dec. 3*	*Prepare slides and plan a demo.*
Polish project	6+	Dec. 6	Fix bugs and add finishing touches
Final presentation	10	*Dec. 10*	*Prepare slides and rehearse*
Final paper	20	*Dec. 13*	*See course webpage*
Total	**104**		94 not including proposal