Interactive Modeling and Visualization of Scattered Data Interpolation Schemes using Quartic Triangular Bézier Patches

Krassimira Vlachkova[a)] and Krum Radev[b)]

Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", 1164 Sofia, Bulgaria

[a)]*Corresponding author: krassivl@fmi.uni-sofia.bg*
[b)]*Electronic mail: kvradev@uni-sofia.bg*

Abstract. We present two new program packages for modeling and visualization of scattered data interpolation surfaces based on smooth interpolation quartic curve networks that are extended to surfaces comprising of quartic triangular Bézier patches (TBP). Our work and contributions are in the field of experimental algorithmics and algorithm engineering. We have chosen the open-source data visualization libraries `Plotly.js` and `Three.js` as our main implementation and visualization tools. This choice ensures the platform independency of our packages and their direct use without restrictions. The packages can be used for experiments with user's data sets since they work with the host file system. The latter allows wide testing, modeling, and editing of the resulting interpolation surfaces. The packages can be used both for research and educational purposes. We experimented extensively with our packages using data of increasing complexity. The experimental results are presented and analyzed.
Keywords: Scattered data interpolation, Curve network, Minimum norm network, Bézier patch, Plotly.js, Three.js

INTRODUCTION

Interpolation of data points in $\mathbb{R}^3$ by smooth surface is an important problem in applied mathematics, which finds applications in various areas such as automotive, aircraft and ship design, medicine, architecture, archeology, computer graphics, and more. In general the problem can be formulated as follows: Given a set of points $\mathbf{d}_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $i = 1, \ldots, n$, find a bivariate function $F(x,y)$ defined in a certain domain containing points $\mathbf{v}_i = (x_i, y_i)$, such that F possesses continuous partial derivatives up to a given order, and $F(x_i, y_i) = z_i$.

Various methods for solving this problem were proposed and applied, see, e.g., the surveys [1, 2, 3, 4], and also [5, 6, 7, 8, 9]. A standard approach to solving the problem consists of two steps, see [1]:

1. Construct a triangulation $T = T(\mathbf{v}_1, \ldots, \mathbf{v}_n)$;
2. For every triangle in T construct a surface which interpolates the data in the three vertices.

The interpolation surface constructed in Step 2 is usually polynomial or piecewise polynomial. Typically, the patches are computed with a priori prescribed normal vectors at the data points. G^1 or G^2 smoothness of the resulting surface is achieved either by increasing the degree of the patches, or by a procedure called *splitting*. Splitting was originally proposed by Clough and Tocher [10] and further developed by Percell [11] and Farin [12]. In practice using patches of least degree and splitting is preferable since it is computationally simple and efficient.

Shirman and Séquin [13, 14] construct a G^1 smooth surface consisting of quartic TBP. Their method assumes that the normal vectors at points $\mathbf{d}_i$, $i = 1, \ldots, n$, are given as part of the input. Shirman and Séquin first construct a smooth cubic curve network defined on the edges of T, and then degree elevate it to quartic. This increases the degrees of freedom and allows them to smoothly connect the adjacent Bézier patches. Next they apply splitting where for each triangle in T a macro-patch consisting of three quartic Bézier sub-patches is constructed. To compute the inner Bézier control points closest to the boundary of the macro-patch, Shirman and Séquin use a method proposed by Chiyokura and Kimura [15, 16]. The interpolation surfaces constructed by Shirman and Séquin's algorithm often suffer from unwanted bulges, tilts, and shears as pointed out by the authors in [17] and more recently by Hettinga and Kosinka in [18].

Nielson [19] proposes a method which computes a smooth interpolation curve network defined on the edges of T so as to satisfy an extremal property and then extends it to a smooth interpolation surface using an appropriate *blending* method based on convex combination schemes. The interpolation curve network is called *minimum norm network* (MNN) and is cubic. Nielson's interpolant is a rational function on every triangle in T. A significant advantage of Nielson's method is that the normal vectors at the data points are obtained through the computation of the MNN.

Vlachkova and Radev [20] propose an algorithm for interpolation of 3D data which improves on Shirman and Séquin's approach in two ways. First, they use Nielson's MNN and second, they apply different strategy and additional criteria for computation of the control points so that to avoid unwanted distortions and twists which appear in surfaces constructed by Shirman and Séquin's method. As a result, the quality of the resulting surfaces is improved.

In this paper we consider the problem of 3D scattered data interpolation using quartic smooth interpolation curve networks and quartic TBP. The main contributions of our work can be summarized as follows:

- We implemented two new software packages for interactive visualization, manipulation, and comparison of smooth interpolation surfaces consisting of quartic TBP that work with arbitrary quartic G^1 curve networks. The packages are user friendly and have a large number of features that allow easy testing and experimenting.
- We performed extensive experimental work. The results were analysed and compared to reveal main differences between Shirman and Séquin's method and our Algorithm 2 [20] with respect to various criteria including quality and shape of the resulting surfaces. Moreover, the results of the experiments provide valuable information on how to modify the TBP constructed by Algorithm 2 [20] to further improve the shape of the interpolation surface.

RELATED WORK

Let $n \geq 3$ be an integer and $\mathbf{d}_i := (x_i, y_i, z_i),\ i = 1, \ldots, n$ be different points in $\mathbb{R}^3$. We call this set of points *data* and assume that the projections $\mathbf{v}_i := (x_i, y_i)$ onto the plane Oxy are different and non-collinear.

A *triangulation* T of points $\mathbf{v}_i$ is a collection of non-overlapping, non-degenerate closed triangles in Oxy such that the set of the vertices of the triangles coincides with the set of points $\mathbf{v}_i$. Hereafter we assume that a triangulation T of the points $\mathbf{v}_i,\ i = 1, \ldots, n$, is given and fixed.

Furthermore, for the sake of simplicity, we assume that the domain D formed by the union of the triangles in T is connected. In general D is a collection of polygons with holes.

The set of the edges in T is denoted by E. If there is an edge between $\mathbf{v}_i$ and $\mathbf{v}_j$ in E, it will be referred to by e_{ij} or simply by e if no ambiguity arises.

A *curve network* is a collection of real-valued univariate functions $\{f_e\}_{e \in E}$ defined on the edges in E. With any real-valued bivariate function F defined on D we naturally associate the curve network defined as the restriction of F on the edges in E, i.e. for $e = e_{ij} \in E$,

$$f_e(t) := F\Big(\big(1 - \frac{t}{\|e\|}\big)x_i + \frac{t}{\|e\|}\, x_j,\ \big(1 - \frac{t}{\|e\|}\big)y_i + \frac{t}{\|e\|}\, y_j\Big),\ \text{where } 0 \leq t \leq \|e\| \text{ and } \|e\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \quad (1)$$

Furthermore, according to the context F will denote either a real-valued bivariate function or a curve network defined by (1). We introduce the following class of *smooth interpolants* defined on D

$$\mathscr{F} := \big\{F(x,y) \in C(D) \,|\, F(x_i, y_i) = z_i,\ i = 1, \ldots, n,\ \partial F/\partial x, \partial F/\partial y \in C(D),\ f'_e \in AC,\ f''_e \in L^2,\ e \in E\big\}$$

and the corresponding class of so-called *smooth interpolation curve networks* $\mathscr{C}(E) := \big\{F_{|E} = \{f_e\}_{e \in E} \mid F(x,y) \in \mathscr{F}\big\}$, where $C(D)$ is the class of bivariate continuous functions defined in D, AC is the class of univariate absolutely continuous functions defined in $[0, \|e\|]$, and L^2 is the class of univariate functions defined in $[0, \|e\|]$ whose second power is Lebesgue integrable.

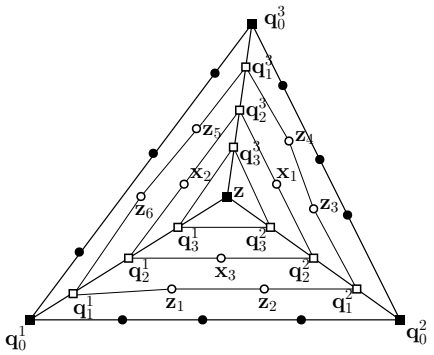
The smoothness of the interpolation curve network $F \in \mathscr{C}(E)$ geometrically means that at each point $\mathbf{d}_i$ there is a *tangent plane* to F, where a plane is *tangent* to the curve network at a point $\mathbf{d}_i$ if it contains the tangent vectors at $\mathbf{d}_i$ of the curves incident to $\mathbf{d}_i$.

The L^2-norm is defined in $\mathscr{C}(E)$ by $\|F\|_{L^2(T)} := \|F\| = \Big(\sum_{e \in E} \int_0^{\|e\|} |f_e(t)|^2 dt\Big)^{1/2}$. We denote the networks of the second derivative of F by $F'' := \{f''_e\}_{e \in E}$ and consider the following extremal problem:

$$(\mathbf{P}) \quad \textit{Find } F^* \in \mathscr{C}(E) \ \textit{ such that } \|F^{*\prime\prime}\| = \inf_{F \in \mathscr{C}(E)} \|F''\|.$$

Nielson [19] showed that $(\mathbf{P})$ possesses a unique solution (MNN). The MNN is obtained by solving a linear system of equations.

Shirman and Séquin's method assumes that the normal vectors at the data points $\mathbf{d}_i$, $i = 1, \ldots, n$, are given a priori. Their Algorithm 1 below takes a triangle in T and the degree-elevated quartic boundary control points of the corresponding macro-patch and computes 19 control points of the three G^1-continuous quartic Bézier sub-patches, see Fig. 1. Algorithm 2 below improves on Shirman and Séquin's Algorithm 1 and was considered in detail in [20].



■ vertices of the sub-patches;
● inner control points on the boundary of the macro-patch;
□ inner control points of the three inner boundary curves of the sub-patches;
○ inner control points of the sub-patches.

FIGURE 1. Construction of a G^1-continuous Bézier macro-patch by splitting to three sub-patches.

Algorithm 1 Shirman and Séquin [13, 14]

Step 1. Compute the control points closest to the boundary of the macro-patch :
1.1 Points $\mathbf{q}_1^i,\ i=1,2,3,$ are centers of the three small triangles with vertices ●■●.
1.2 Then points $\mathbf{z}_i,\ i=1,\ldots,6,$ are computed using Chiyokura and Kimura's method [15, 16].

Step 2. Compute points $\mathbf{q}_2^1=\frac{1}{3}(\mathbf{q}_1^1+\mathbf{z}_1+\mathbf{z}_6),\ \mathbf{q}_2^2=\frac{1}{3}(\mathbf{q}_1^2+\mathbf{z}_2+\mathbf{z}_3),\ \mathbf{q}_2^3=\frac{1}{3}(\mathbf{q}_1^3+\mathbf{z}_4+\mathbf{z}_5)$.

Step 3. Compute points $\mathbf{p}_i=2\mathbf{q}_2^i-\frac{4}{3}\mathbf{q}_1^i+\frac{1}{3}\mathbf{q}_0^i,\ i=1,2,3$.

Step 4. Compute the splitting point $\mathbf{z}=\frac{1}{3}(\mathbf{p}_1+\mathbf{p}_2+\mathbf{p}_3)$.

Step 5. Compute points $\mathbf{q}_3^i=\frac{3}{4}\mathbf{p}_i+\frac{1}{4}\mathbf{z},\ i=1,2,3$.

Step 6. Compute points $\mathbf{x}_1=\frac{3}{2}(-\mathbf{q}_3^1+\mathbf{q}_3^2+\mathbf{q}_3^3)-\frac{1}{2}(-\mathbf{q}_2^1+\mathbf{q}_2^2+\mathbf{q}_2^3),\ \mathbf{x}_2=\frac{3}{2}(\mathbf{q}_3^1-\mathbf{q}_3^2+\mathbf{q}_3^3)-\frac{1}{2}(\mathbf{q}_2^1-\mathbf{q}_2^2+\mathbf{q}_2^3)$,
$\mathbf{x}_3=\frac{3}{2}(\mathbf{q}_3^1+\mathbf{q}_3^2-\mathbf{q}_3^3)-\frac{1}{2}(\mathbf{q}_2^1+\mathbf{q}_2^2-\mathbf{q}_2^3)$.

Algorithm 2

Step 1. Compute the control points in the first layer:
1.1 Points of type □ are centers of the three small triangles with vertices ●■●.
1.2 Then points of type ○ are computed as described in [20].

Step 2. Compute the control points in the second layer:
2.1 Points of type □ are centers of the three small triangles with vertices ○□○ in the first layer.
2.2 Then points of type ○ are mid-points of the segments with vertices of type □ in the second layer.

Step 3. Compute the control points in the third layer: The three points of type □ are centers of the small triangles with vertices ○□○ in the second layer.

Step 4. Compute the splitting point of type ■ as a center of the triangle with vertices □ in the third layer.

OVERVIEW OF OUR WORK

Why Plotly and Three.js?

We implemented two program packages for interactive 3D modeling, visualization, and comparison of smooth interpolation surfaces consisting of quartic TBP. Our main goal was to implement Algorithm 2 proposed in [20] for constructing a G^1-smooth piecewise quartic surface that interpolates a given smooth quartic curve network. We need to visualize the proven mathematical properties of the resulting surface, and to compare them with the surfaces obtained by some known methods such as Shirman and Séquin's. For this purpose, a computer software for 3D visualization capable of displaying curves and TBP viewed from different angles, coloured in different colours, and illuminated in various ways should be created. Hence, the software needs to be interactive, easily configured as it runs, and allowing dynamic loading of input data. Since Algorithm 2 involves many mathematical calculations, mostly on points in $\mathbb{R}^3$, it would be convenient to display the coordinates of these points on the screen, for example when clicking the mouse,

or to be able to hide and display individual parts of the surface. At the same time the software must be fast enough to work in real time and to visualize more complex surfaces. Taking into account the above requirements, we decided to design our software as a Web application. We used only HTML and JavaScript for the implementation. This decision enabled the resulting surfaces to be easily rendered using various visualization libraries. We have chosen `Plotly` [21] and `Three.js` [22] as our main implementation and visualization tools. Both software are open source and are distributed under MIT license [23]. Using `Plotly` provides conveniences such as displaying the coordinate system, the coordinates of points when hovering over them, and controlling the objects rendered. There are, however, some significant disadvantages: few settings for the light sources and mostly very slow operation which makes Plotly not suitable for rendering more complex surfaces.

On the other hand, `Three.js` offers fast real-time operation but the possibilities to control the image displayed and to obtain information about the coordinates of the objects displayed are limited. `Three.js` has advantages such as better tuning of the light sources and shaders used. It also allows saving the rendered objects in SVG vector format [24].

Description of the packages

In this section, we focus on the options for tuning the surfaces displayed provided by the `Three.js` software implementation since those in the `Plotly` implementation are fewer and largely obvious, see Fig. 2 (left). In both implementations the mouse is used to change the camera position. The surfaces can be edited interactively by rotation and resizing.

The user interface is intuitive and allows changing of the main visualization elements such as lighting, shading, coloring, etc. The main features of our Three.js package are as follows, see Fig. 2 (right):

- The data file can be chosen either from a drop-down menu, or can be loaded from an external file.
- The position of the lights can be changed. Due to the specific characteristics of each surface, it is not possible to make a universal lighting that works equally well for every data set. Two light configurations are defined a priori with eight and two light sources, respectively. It is also possible to load the lights from a text file. The additional option *Light Intensity* sets the brightness of the light sources.
- We can control the number of points by which each side of a triangle is divided when drawing a TBP. Larger values of the parameter *Bezier Grid size* result in higher quality images but also place a greater load on the hardware. Small values lead to poor quality results but fast program performance.
- Options *material* and *surface color* select the shader and the color for the surface displayed.
- Options *wireframe* and *surface* display the mesh of triangles forming the surface in case both options are enabled. Otherwise either the interpolation curve network, or the surface are shown.
- The color of the interpolation curve network and the background color can be specified. An option to color the different TBP in a random color is provided.
- Different TBP are colored in different colors when option *Color gradient* is enabled. However, the choice of the colors is not random. The first color is set by the *Color start* option, the last color is set by the *Color End* option, and the total number of colors is set by the *Gradient Steps* option. The colors between the first and the last colors are obtained by linear interpolation.
- Additional sliders *Scale x*, *Scale y*, and *Scale z* allow to scale the image in any of the three directions x, y, and z, respectively, and the option *Image Resolution* changes the image resolution.
- Options *Take Screenshot* and *Scene to SVG* make a copy of the screen image in JPG and SVG format, respectively, and save it to user's computer.

RESULTS FROM THE EXPERIMENTS

Here we present an example from the experiments performed. The interpolation surfaces are generated using the MNN and Algorithm 2. They are visualized using `Plotly` and `Three.js`. The surface is compared with the corresponding surface generated by Shirman and Séquin's method applied to the MNN.

FIGURE 2. The user interface for the program package created with: (left) `Plotly`; (right) `Three.js`.

TABLE I. The data for Example 1

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
x_i	.21	.46	.83	.97	.67	.53	.28	.07	.06	.25	.48	.67	.77	.90	.66	.50	.32	.25	.46	.57	.75	.94	.46	.18	.14
y_i	.88	.93	.89	.54	.71	.74	.77	.70	.43	.56	.61	.54	.45	.31	.35	.47	44	.31	.33	.20	.25	.05	.07	.19	.06

Example 1 We consider the function $f = \exp\left((x-0.5)^2 + (y-0.5)^2\right)$ which is sampled at 25 points shown in Table I. The triangulation, which is shown in Fig. 3 (left), is the Delaunay triangulation. The corresponding MNN is shown in Fig. 3 (right). The surfaces generated using `Plotly` and `Three.js` are shown in Fig. 4 and Fig. 5, correspondingly, as follows: (left) The surface generated by Shirman and Séquin's method; (right) The surface generated by Algorithm 2.

ACKNOWLEDGMENTS

This work was supported in part by Sofia University Science Fund Grant No. 80-10-109/2022, and European Regional Development Fund and the Operational Program "Science and Education for Smart Growth" under contract № BG05M2OP001-1.001-0004 (2018-2023).

REFERENCES

1. S. Mann, C. Loop, M. Lounsbery, D. Meyers, J. Painter, T. DeRose, and K. Sloan, "A survey of parametric scattered data fitting using triangular interpolants," in *Curve and Surface Design*, edited by H. Hagen (SIAM, Philadelphia, 1992) pp. 145–172.
2. S. Lodha and K. Franke, "Scattered data techniques for surfaces," in *Proceedings of Dagstuhl Conference on Scientific Visualization* (IEEE Computer Society Press, Washington, 1997) pp. 182–222.
3. R. Franke and G. Nielson, "Scattered data interpolation and applications: a tutorial and survey," in *Geometric Modeling*, edited by H. Hagen and D. Roller (Springer, Berlin, 1991) pp. 131–160.
4. M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, G. Guennebaud, J. Levine, A. Sharf, and C. Silva, "A survey of surface reconstruction from point clouds," Comput. Graph. Forum **36**, 301–329 (2017).
5. T. K. Dey, *Curve and surface reconstruction: algorithms with mathematical analysis*, Cambridge Monographs on Applied and Computational Mathematics (Cambridge University Press, 2006).
6. K. Anjyo, J. Lewis, and F. Pighin, "Scattered data interpolation for computer graphics, SIGGRAPH 2014 course notes," (2014), last accessed August 22, 2022.
7. F. Dell'Accio, F. Di Tommaso, O. Nouisser, and N. Siar, "Rational Hermite interpolation on six-tuples and scattered data," Appl. Math. Comput. **386**, 125452 (2020).
8. S. A. A. Karim, A. Saaban, V. Skala, A. Ghaffar, K. S. Nisar, and D. Baleanu, "Construction of new cubic Bézier-like triangular patches with application in scattered data interpolation," Adv. Difference Equ. (2020), 10.1186/s13662-020-02598-w.
9. S. A. A. Karim, A. Saaban, and V. T. Nguyen, "Scattered data interpolation using quartic triangular patch for shape-preserving interpolation and comparison with mesh-free methods," Symmetry **12** (2020), 10.3390/sym12071071.

FIGURE 3. Example 2: (left) The Delaunay triangulation for $n = 25$; (right) The MNN.

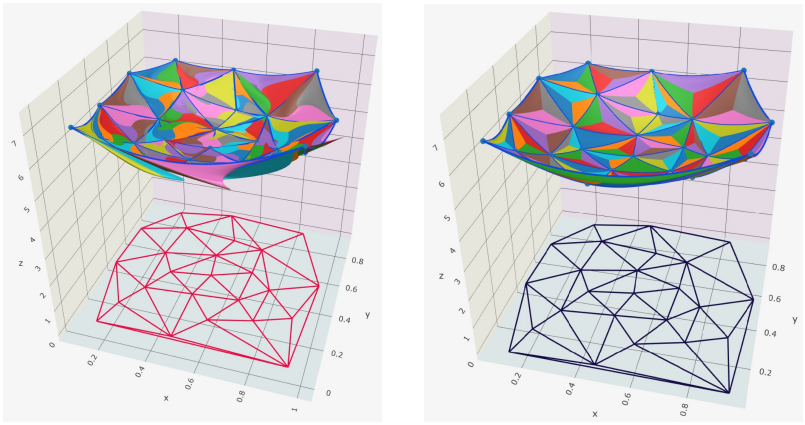


FIGURE 4. Comparison of the two surfaces for Example 1 generated using `Plotly`: (left) The surface generated using Shirman and Séquin's Algorithm 1; (right) The surface generated using our Algorithm 2.

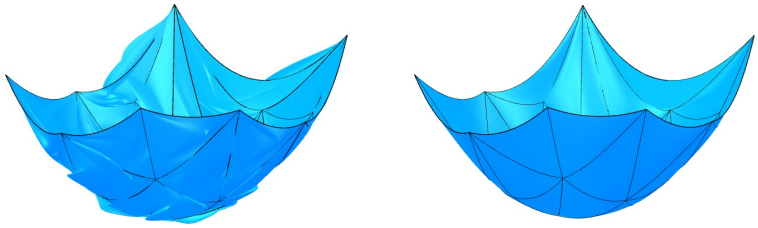


FIGURE 5. Comparison of the two surfaces for the data in Example 1 generated using `Three.js`: (left) The surface generated using Shirman and Séquin's Algorithm 1; (right) The surface generated using our Algorithm 2.

10. R. Clough and J. Tocher, "Finite elements stiffness matrices for analysis of plate bending," in *Proceedings of the 1st Conference on Matrix Methods in Structural Mechanics*, Vol. 66–80 (Wright-Patterson A. F. B., Ohio, 1965) pp. 515–545.
11. P. Percell, "On cubic and quartic Clough-Tocher finite elements," SIAM J. Numer. Anal. **13**, 100–103 (1976).
12. G. Farin, "A modified Clough-Tocher interpolant," Comput. Aided Geom. Des. **2**, 19–27 (1985).
13. L. Shirman and C. Séquin, "Local surface interpolation with Bézier patches," Comput. Aided Geom. Des. **4**, 279–295 (1987).
14. L. Shirman and C. Séquin, "Local surface interpolation with Bézier patches: errata and improvements," Comput. Aided Geom. Des. **8**, 217–221 (1991).
15. H. Chiyokura and F. Kimura, "Design of solids with free-form surfaces," in *SIGGRAPH '83 Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques*, Vol. 17, edited by P. P. Tanner (ACM, New York, 1983) pp. 289–298.
16. H. Chiyokura, "Localized surface interpolation method for irregular meshes," in *Advanced Computer Graphics, Proceedings of Computer Graphics Tokyo'86*, Vol. 66–80, edited by T. Kunii (Springer, Tokyo, 1986) pp. 3–19.
17. L. Shirman and C. Séquin, "Local surface interpolation with shape parameters between adjoining Gregory patches," Comput. Aided Geom. Des. **7**, 375–388 (1990).
18. G. Hettinga and J. Kosinka, "Multisided generalisations of Gregory patches," Comput. Aided Geom. Des. **62**, 166–180 (2018).
19. G. Nielson, "A method for interpolating scattered data based upon a minimum norm network," Math. Comput. **40**, 253–271 (1983).
20. K. Vlachkova and K. Radev, "Interpolation of data in $\mathbb{R}^3$ using quartic triangular Bézier surfaces," in *AIP Conf. Proc.*, Vol. 2325 (2021).
21. "Plotly.js," `https://plot.ly/javascript` (last accessed August 22, 2022).
22. "Three.js," `https://threejs.org` (last accessed August 22, 2022).
23. "MIT License," `https://opensource.org/licenses/MIT` (last accessed August 22, 2022).
24. "SVG," `https://developer.mozilla.org/en-US/docs/Web/SVG` (last accessed August 22, 2022).