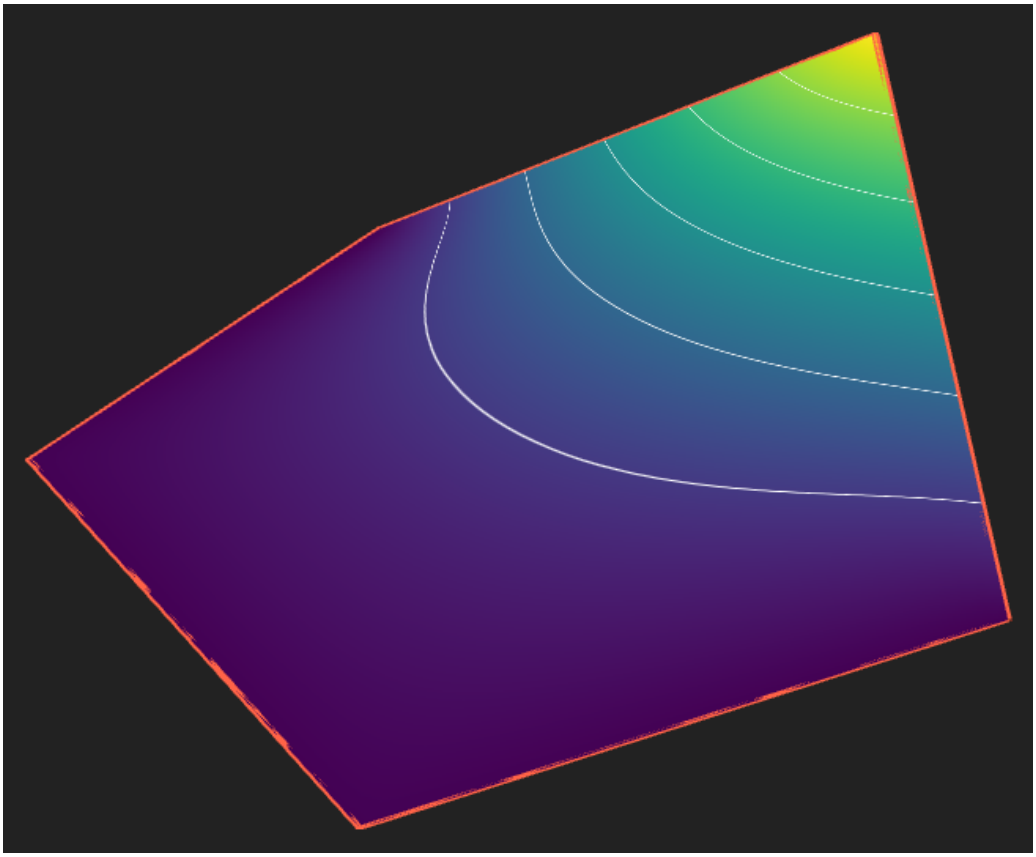
Exploring generalised barycentric coordinates

February 3, 2023



Author: Bob van der Vuurst
Supervisor: Jiří Kosinka

Contents

1 Introduction

Barycentric coordinates are used for many applications, such as shading, interpolation and shape deformation. In classical rendering, barycentric coordinates are used on triangles, where the definition of the coordinates are uniquely defined. However, this is not the case for polygons with more than three vertices. In [1] several types of generalised barycentric coordinate functions are discussed and their behaviour for interpolation is explored. The paper shows several figures where the difference in the interpolation is displayed for each of those functions.

However, the figures in the paper show only a handful of results, and only from one perspective. The goal of this project is to address this limitation by developing a 3D web tool that visualizes the interpolations of the different barycentric functions as well as the functions themselves, similar to Figure 3 in [1].

The tool will have the functionality to choose a barycentric function and the number of vertices in the polygon. The vertex positions should also be configurable, potentially in a separate 2D view.

The resulting interpolation of the barycentric function on the polygon will be displayed in a 3D environment, where the user should have the ability to freely move and rotate the camera around the interpolation. There should also be various colour maps or wireframes to choose from for displaying the shapes. In addition to this, some presets should be included with parameters that show remarkable resulting shapes.

The project targets the problem that the paper only shows a small selection of the many different scenarios, where the barycentric coordinate functions can be used and the differences between them become apparent. The 3D tool should be very flexible and thus be able to show those scenarios more clearly.

2 Functionality

2.1 Overview

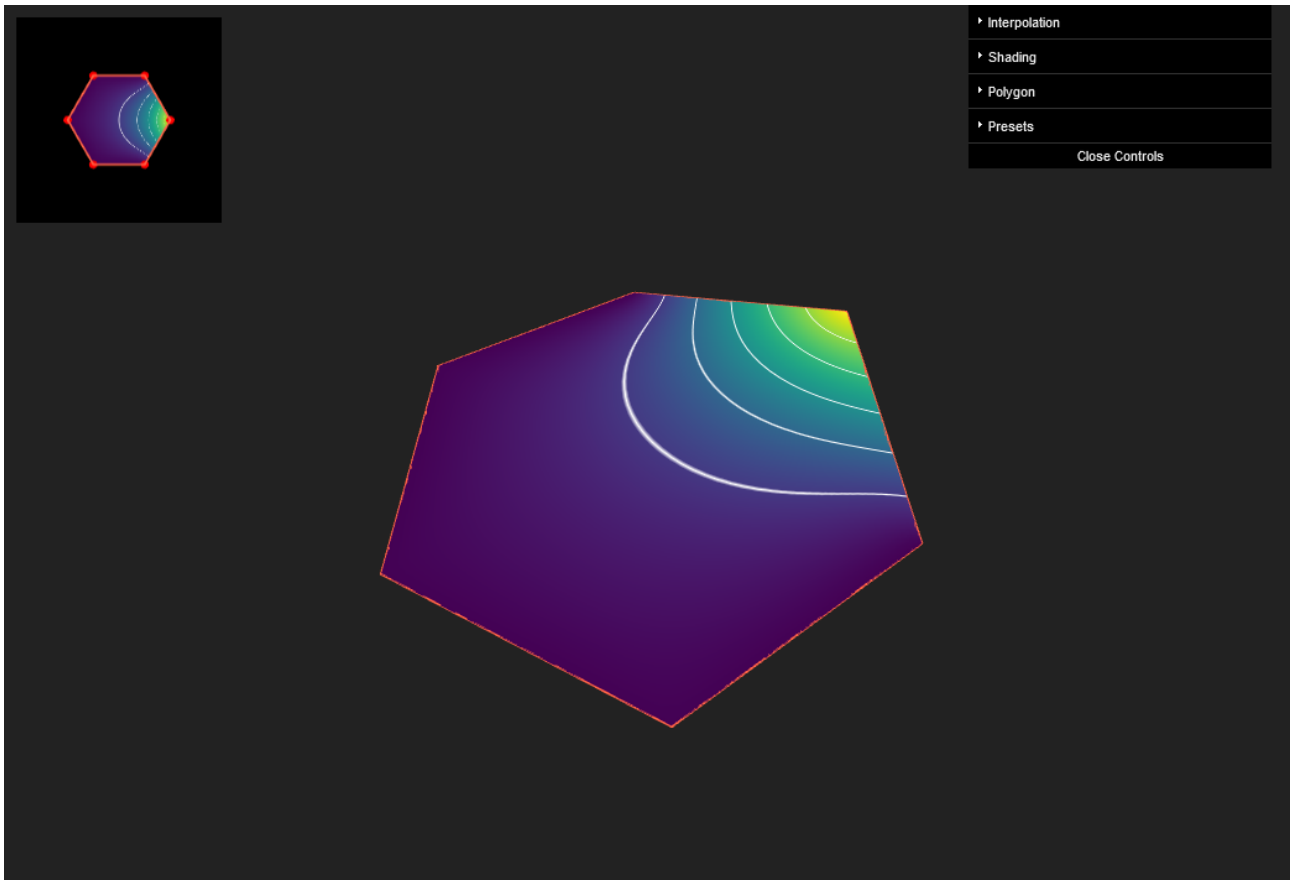


Figure 1: The default view that is shown when visiting the tool's webpage.

The tool can be divided into three parts: The main 3D view, the Graphics User Interface (GUI) and the minimap. The GUI is located on the top right of the screen and the minimap is located on the top left of the screen. See Fig. 1 for an example view. The functionalities of the components will be discussed in detail in their respective subsection.

2.2 3D view

The 3D view allows the camera to be moved, rotated and zoomed. Moving the camera can be done with the left mouse button, rotating with the right mouse button and zooming with the scroll wheel. The controls do not allow the camera to be rolled on its side: The z-axis always points up. This is done on purpose to not confuse the user with complicated camera movement.

2.3 GUI

Figure 2: The GUI with all the folders fully expanded.

The GUI (shown in Fig. 2) allows the polygon and its interpolation to be configured and also provides several visualization options. The GUI is divided into four folders: Interpolation, shading, polygon and presets. By default all folders are closed, but they can be opened by clicking on them. The four folders will be discussed below.

2.3.1 Interpolation

The interpolation folder has the option to configure the interpolation's c_function: Either from the presets or some arbitrary custom function. The preset functions are the same as used in [1]: $r^p, log(1+r), \frac{r}{1+r}, \frac{r^2}{1+r^2}$. Using r^p as the c_function and setting p to 0 gives Wachspress coordinates [2] and setting p to 1 gives mean-value coordinates [3]. The custom function allows any function with variable r to be filled into the custom_function field. There is also support for mathematical functions, such as sin, $log2$, abs and $atan$ to name a few.

2.3.2 Shading

The shading folder provides ways to manipulate the interpolations shading and overall look.

The first setting contains a few colormaps to choose from. The colormaps are all from python's matplotlib library. The colors are based on the relative height to the interpolation: The lowest point in the interpolation will have the first color and the highest point will have the last color, independent of their absolute height.

The wireframe checkbox toggles between a normal and a wireframe view of the interpolation.

The density slider controls the density of the mesh that is generated for the interpolation. This density scales exponentially: For each step increase in density, the number of triangles is multiplied by four. Note that the density of triangles is not uniform, because the main triangles differ in size.

The ContourLines slider configures how many contour lines are displayed on the interpolation. Setting the slider for n contour lines will draw a white line at every $\frac{1}{n}$'th fraction of the interpolation. Setting the slider to 0 will not display any contour lines.

2.3.3 Polygon

The polygon folder contains the functionalities to change the polygon's points.

The pointIndex allows the choice of any point in the polygon, which can then be moved on the z-axis with the z slider below it.

The two buttons below the slider allow for deleting and adding points respectively.

The last button allows the z-coordinates of the points to be cycled counter-clockwise.

2.3.4 Presets

The presets folder contains several presets: a regular polygon, a sine wave mapped to the points' z-axis, a hyperbolic paraboloid and a nonconvex polygon. The number of points the polygon should have can be set with the slider and the preset can be applied with the "Apply preset" button.

2.4 Minimap

The minimap allows the polygon's points to be moved on the xy-plane by dragging them. When the polygon's shape is changed in such a way that it is not strictly convex, a warning is displayed. This is because a nonconvex polygon can cause irregular behavior for most c_functions.

The minimap does not allow the points to be moved such that the polygon is self-intersecting, because that causes irregular behavior by triangulation method that is used for the interpolation.

2.5 Not implemented

2.5.1 Phone support

Currently, the web tool does not work correctly on Android phones. The interpolation is not shown at all, even though the polygon itself is and the rest works fine. Because most Android web browsers do not provide a way to access the developer console directly, this problem is hard to diagnose. It is likely a problem related to the shader, though the tested phone does have WebGL2 support. As most users of this tool will probably access it through a desktop, at this time it is not worth the effort to fix the problem for phones. IPhones have not been tested (yet).

3 Deployment

3.1 Existing deployments

The tool is already deployed at `https://bob-vdv.github.io/BarycentricCoordinates/` using Github Pages. This deployment will be updated each time a commit is made to the master branch.

Another deployment will be added in the future to the tools listed on the University of Groningen's SVCG website, which can be found here.

3.2 Building the web tool

The web tool can be deployed in the following steps:

1. Install NodeJS from the official website.

2. Clone the master branch of the Github page `https://github.com/Bob-vdV/BarycentricCoordinates`
3. Run `npm install`. This will install all of the project's dependencies.
4. Build the deployment with `npm run build`. This will create a directory `dist` which is ready to be deployed.
   The web tool can then be tested locally with `npm run preview`.
   Alternatively `npm run dev` can be used for a local deployment with support for automatic hotswapping if any source file changes.

4 Technology stack

4.1 Typescript

Because the tool should be web-based, the choice of programming languages is limited to variants of JavaScript, the programming language supported by web browsers. TypeScript is a superset of JavaScript that adds support for strong typing to the language. Though this requires writing a bit more code, it enables the text editor to detect wrong types in e.g. functions parameters, which greatly reduces the amount of small mistakes made by the programmer. This results in relatively less time spent debugging those small mistakes and more on writing functional code. TypeScript is fully compatible with existing JavaScript libraries and has the functionality to add externally defined types to JavaScript libraries.

4.2 Npm

Npm is the most popular package manager for JavaScript and is used to install and track the JavaScript packages used for this project.

4.3 Vite

Vite is used to set up the boiler plate code for the server very quickly and also provides fast deployment of both the test server and the build server.

4.4 ThreeJS

ThreeJS is a JavaScript library that provides a high level interface for WebGL and is used to create and render 3D environments. ThreeJS has a vast amount of built-in features for 3D applications, as well as various helper functions. One example is EarCut, which is used by the project to divide the polygon into triangles using ear clipping [4]. There are also numerous demonstrations and visualizations of ThreeJS functionalities, which are great resources start with. These examples can be found on `https://threejs.org/examples/`.

4.5 Dat.GUI

Dat.GUI is a lightweight JavaScript library for providing basic Graphical User Interfaces (GUI). Though dat.GUI does not have many features or customizations, it is sufficient for the project as it is easy to use and provides elements such as sliders, buttons and dropdowns.

4.6 Expr-eval

The expr-eval JavaScript library is used to parse the custom c_i function for the interpolation and convert it to a JavaScript function. It has support for parsing various functions, such as log(), sin(), and floor(). One caveat of expr-eval is that it is roughly 5 times slower than a normal JavaScript function.

4.7 JS-colormaps

JS-colormaps is a javascript file that provides the colormaps from the python matplotlib library. JS-colormaps is not available as a library and thus the file is included in the project manually. It has also been modified heavily for the project to support TypeScript. Also all of the unused colormaps have been removed to reduce the file size.

5 Architecture overview

The overall design of the project tries to follow the Model-View-Controller (MVC) architectural pattern, where the model contains the state of the 3D scene, the view contains the code to present this state and the controller the code to manipulate the state. However, there is one module which does not follow this pattern completely, which is Gui in the view. Because many function calls in the GUI consist of only a few lines, it was decided to leave them inline in the Gui module instead of writing a separate action for it in the Controller.

All the files and their purpose will be briefly discussed below.

5.1 Model

5.1.1 model.ts

The model that initializes the ThreeJS scene with the polygon and the interpolation, and also creates the view and the controller. The program is run with the run() function.

5.1.2 polygon.ts

The polygon that contains all the points. It contains functions for generating the 3D mesh of the lines as well as checks if the polygon is strictly convex or it has self-intersecting edges.

5.1.3 interpolation.ts

Contains the parameters and the polygon to create the interpolation with. It also contains the function to create the 3D mesh of the interpolation using BarycentricGeometry and shading using its colormap.

5.1.4 barycentricGeometry.ts

Divides the polygon into many triangles and then computes the z value at each vertex using the interpolation's c function.

5.1.5 cFunctions.ts

Contains simple functions, such as the function r^p. Also has support for custom expressions using the expr-eval module.

5.1.6 colormaps.ts

Based on JS-colormaps, but every colormap that is not used is removed to reduce the file size.

5.1.7 edge.ts

Defines edges and a function to check if two edges intersect. This is used by the polygon to check if it is self-intersecting.

5.1.8 triangle.ts

Defines triangles and has a function to divide itself into more triangles, which is used by the BarycentricGeometry.

5.1.9 utils.ts

Contains various functions which are used throughout the program such as mathematical modulo (which is different from the % operator with negative numbers).

5.2 View

5.2.1 view.ts

The view contains the renderers and cameras for both the the main 3D view and the minimap, as well as the GUI. The main view uses a perspective camera and the minimap uses a orthographic camera. The minimap and the main view both have their own HTML canvas which the renderers use and the minimap canvas is placed on top of the main canvas using CSS.

5.2.2 gui.ts

Contains the GUI with all the parameters for the polygon and the interpolation.

5.3 Controller

5.3.1 controls.ts

Contains the controls for both the main view and the minimap. The main view controls are ThreeJS MapControls and the minimap controls are ThreeJS DragControls. MapControls allow movement and rotation of the camera in the x and y axes, where the z axis remains upwards. The DragControls does not allow camera movement, but allows the movement of the polygon points instead.

5.3.2 presets.ts

The presets file contains several functions which set the polygon's points to a certain preset. When a preset function is called, all the points in the polygon are discarded and new ones are added, depending on the preset.

5.3.3 updaters

The updaters folder contains two files: interpolationUpdater.ts and polygonUpdater.ts. They are called when the interpolation or the polygon meshes need to be updated respectively. The PolygonUpdater also checks is the polygon is nonconvex. If it is nonconvex a warning is displayed.

References

[1] J. Kosinka, M. Bartoň, Convergence of barycentric coordinates to barycentric kernels, Computer Aided Geometric Design 43 (2016) 200–210, geometric Modeling and Processing 2016. doi:https://doi.org/10.1016/j.cagd.2016.02.003.
URL https://www.sciencedirect.com/science/article/pii/S0167839616300048

[2] E. L. Wachspress, A rational finite element basis, 1975.

[3] M. S. Floater, Mean value coordinates, Computer Aided Geometric Design 20 (1) (2003) 19–27. doi:https://doi.org/10.1016/S0167-8396(03)00002-5.
URL https://www.sciencedirect.com/science/article/pii/S0167839603000025

[4] D. Eberly, Triangulation by ear clipping, 2002.
URL https://www.geometrictools.com/Documentation/TriangulationByEarClipping.pdf