Spherical Bézier Patches and Circular Bézier Curves

Cameron Seth

May 1, 2020

1 Introduction

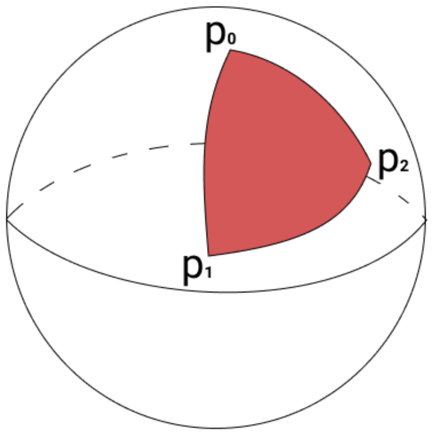
In class we studied Bézier patches/curves that were defined over parameterized domains. A natural variation is Bézier patches/curves defined over spherical/circular domains. This was studied in a series of papers by Alfeld, Neamtu and Schumaker ([1], [2], [3]) to solve the following problem. Let $\{p_i\}$ be a set of points on the sphere, with associated heights $\{r_i\}$. Find a function f that interpolates the points ($f(p_i) = r_i$ for all i). Ideally this f will be continuous (C^0 and C^1) and not too "extreme". Alfeld et al. used piecewise polynomial interpolating surfaces to solve this problem. For my project I implemented the their ideas for both 3D spherical Bézier patches and 2D circular Bézier curves. In the 3D case I applied it to a few interesting examples, including modeling the Earth's elevation. In the 2D case I created a user interface to manipulate the curves.

2 Background

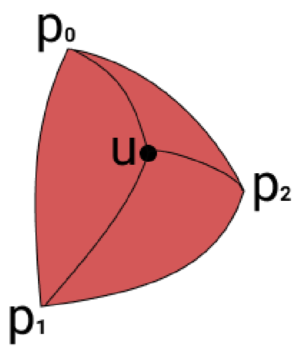
Before solving the problem, Alfeld et al. developed the following definitions [1]. These definitions are analogous to the planar definitions.

Definition 1. *Let S be the unit sphere in $\mathbb{R}^3$ centered at the origin. Let p_0, p_1, p_2 be three linearly independent unit vectors. Then the spherical triangle defined by these vectors is*

$$T = \{u \in S : u = u_0p_0 + u_1p_1 + u_2p_2, u_i \geq 0\}$$



Definition 2. *Let u be a point on the spherical triangle T with $u = u_0p_0 + u_1p_1 + u_2p_2$. Then (u_0, u_1, u_2) are the spherical barycentric coordinates of u.*



Remark 3. *Unlike the planar case, in general $u_0 + u_1 + u_2 \geq 1$. These barycentric coordinates can be calculated using simple geometry, described in [1].*

Definition 4. *Let $u \in T$ with spherical barycentric coordinates (u_0, u_1, u_2). Then the spherical Bernstein basis polynomials on T are defined as*

$$B^n_{\vec{i}}(u) = \binom{n}{\vec{i}} u_0^{i_0} u_1^{i_1} u_2^{i_2}.$$

Further, $B(u) = \sum_{\vec{i},|\vec{i}|=n} P_{\vec{i}} B^n_{\vec{i}}(u)$ is a spherical Bernstein polynomial.

Definition 5. *Let T be a spherical triangle and $B(u) = \sum_{\vec{i},|\vec{i}|=n} P_{\vec{i}} B^n_{\vec{i}}(u)$. Then $\{B(u)u : u \in T\}$ is a spherical Bézier patch.*

Remark 6. *Note that $P_{\vec{i}} \in \mathbb{R}$ and so $B(u) \in \mathbb{R}$. This differes from the planar case where $P_{\vec{i}}, B(u)$ are points. In the spherical case the $P_{\vec{i}}$ will be used as the radial component of the control points.*

Remark 7. *In [1] it was proven that deCastlejau's algorithm can be used to evaluate at a point of a spherical Bézier patch (just like in the planar case).*

The following interesting result is proven in the paper, and I test it in my project.

Theorem 8. *Let T be a spherical triangle, and suppose n is even. Then there exists a unique spherical Bernstein polynomial B of degree n defined on T such that*

$$B(u) = 1, \text{ for all } v \in T.$$

If n is odd then no such B exists [1].

3 Base Project

For my base project I implemented an evaluator and tesselator for spherical Bézier Patches (3D).

- Input Bézier control points in s3d-like format
- Use deCastlejau's algorithm to evaluate Spherical Bernstein Polynomials
- Tesselation will be over spherical barycentric coordinates
- Output in s3d format for cglv

The input format I use is very similar to the s3d-like format used in the datasets for assignment 8. The key difference is how the vectors are inputed. Instead of "v x-coord y-coord z-coord" it will be a normalized vector followed by the radial component ("v normalized-x-coord normalized-y-coord normalized-z-coord r"). Having this extra value is redundant, however I think it makes more sense in solving the spherical problem.

4 Extras

4.1 Examples of using my Base Project

Figure 1b contains a simple example of using my base project implementation. Compared with Figure 1a it illustrates the difference between the spherical case and the planar case. For both images I evaluate a degree 1 Bézier Patch with control points at $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$.

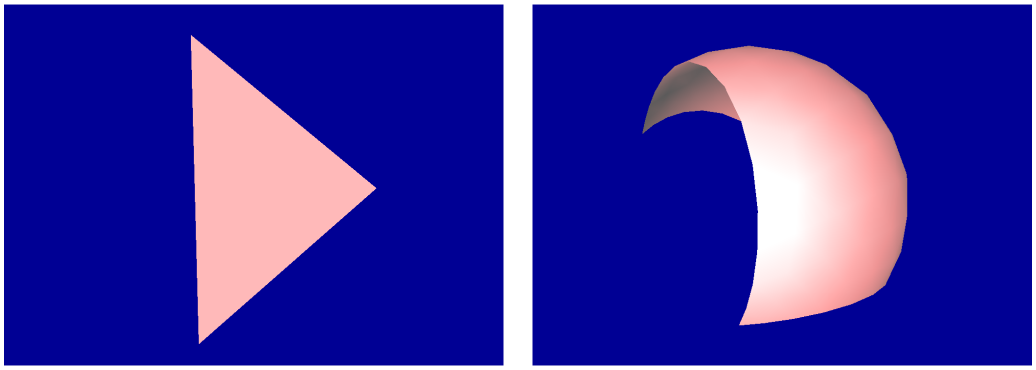

(a) Planar Bézier Patch (b) Spherical Bézier Patch

Figure 1: Degree 1 Bézier patch with control points at $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$

I used my implementation to test Theorem 8. In the degree 1 case there are no internal control points, so the patch in Figure 1b is the only option, and it does not form a surface that is at a constant radius from the center. In the degree 2 case it is possible, see Figure 2. Interestingly, the inner control points go to zero to achieve this constant height.

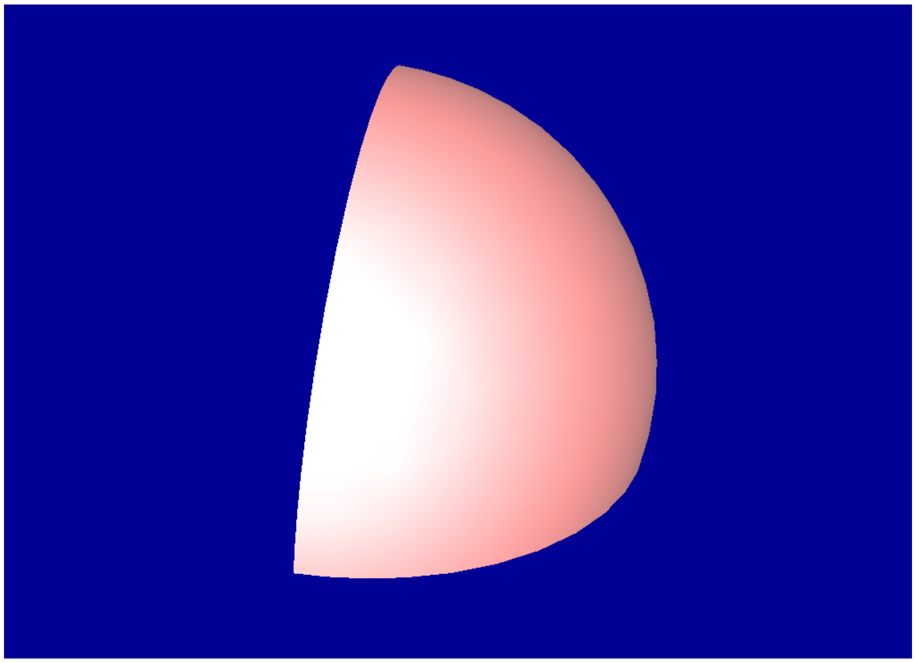


Figure 2: Degree 2 Spherical Bézier Patch with constant height

I also used my implementation to model the surface of the Earth using the Earth's elevation data from [4]. I had to amplify the dataset so that the surface features would be visible. See Figure 3 and 4.

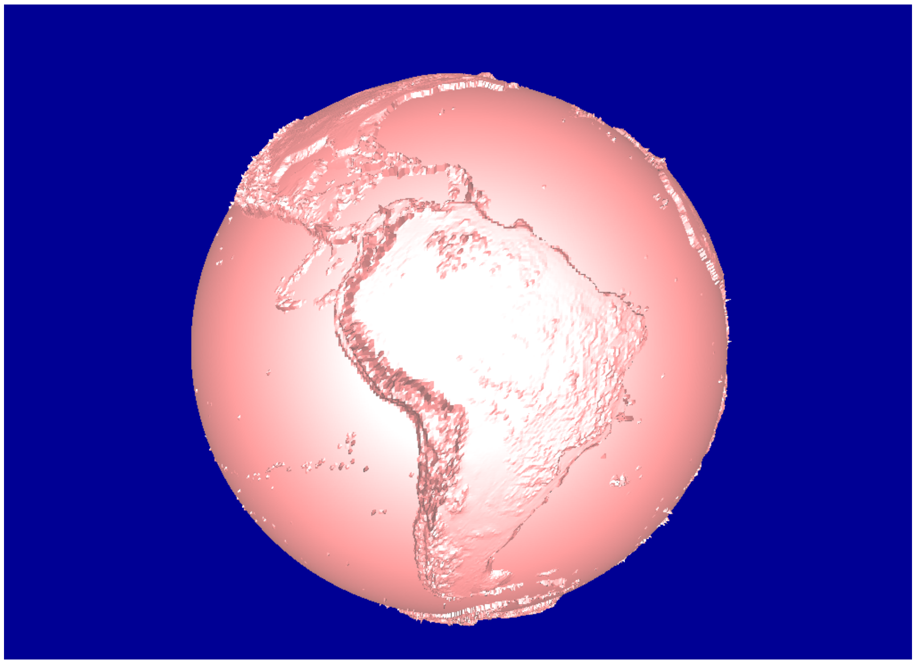


Figure 3: Earth Model, centered on South America

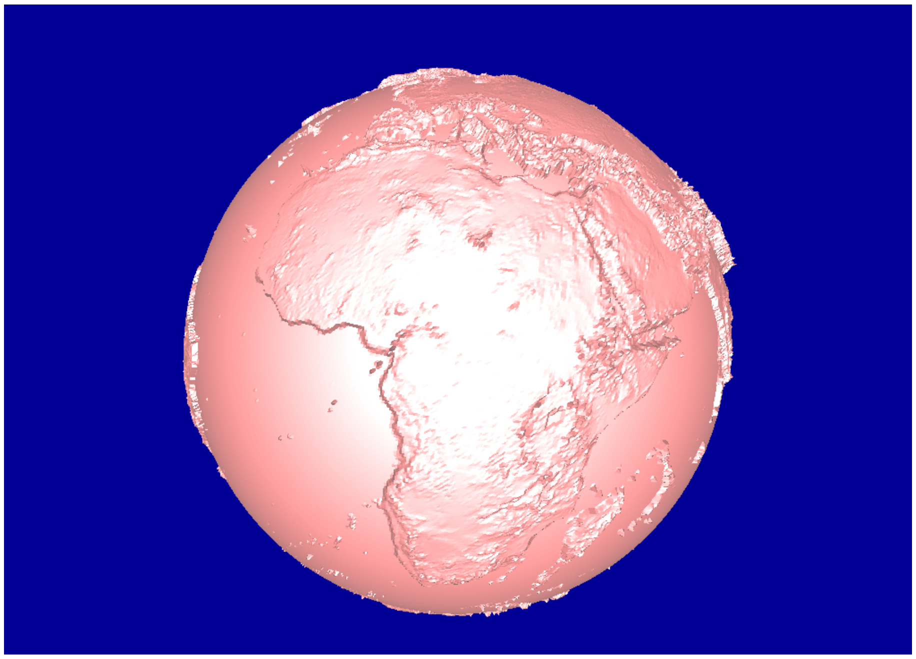
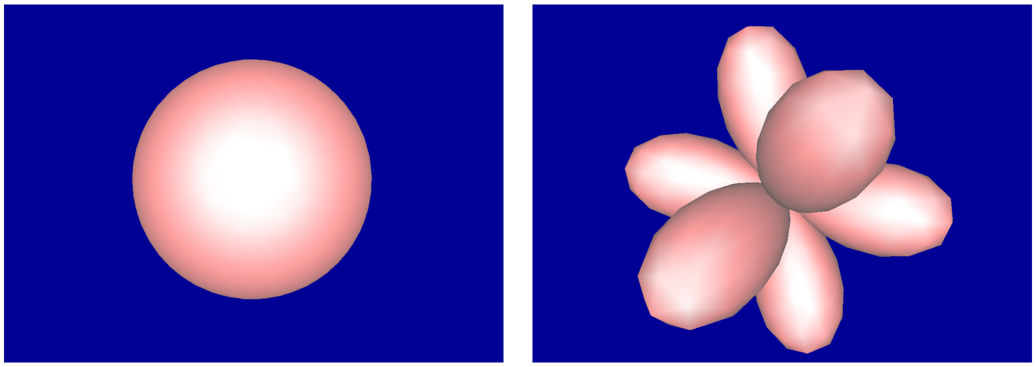


Figure 4: Earth Model, centered on Africa

Finally I used my implementation to model surfaces that are very simple in the spherical case, but are not as simple in the planar case. See Figure 5 for two examples.



(a) Sphere (b) Rose-like surface

Figure 5: Simple modeling with Spherical Bézier Patches

One of the original goals of my project was to implement Clough-Tocher for the spherical case. This was analyzed in [2]. Clough-Tocher is necessary to ensure C^1 continuity across the boundaries of neighbouring patches. Unfortunately I was unable to finish my implementation, but conceptually the algorithm is very similar to the planar case.

4.2 Implement interactive interface for Circular Bézier Curves (2D)

This implementation had the following functionality:

- Same ideas as 3D case: use deCastlejau's to evaluate Circular Bernstein Polynomials
- Control points can be moved with middle mouse button
- Domain circle can be moved and resized
- Menu buttons to give various display options

See Figure 6 for an example using the implementation. One of the nice results is when the circle is resized to be very large, the output approaches the same as the planar case. This is demonstrated in Figure 7. Figure 8 demonstrates the different display settings. One important note: the control points must be spaced out at equal angles from the center of the circle for deCasteljau's algorithm to work. I restrict the control points to stay on the red dotted line.

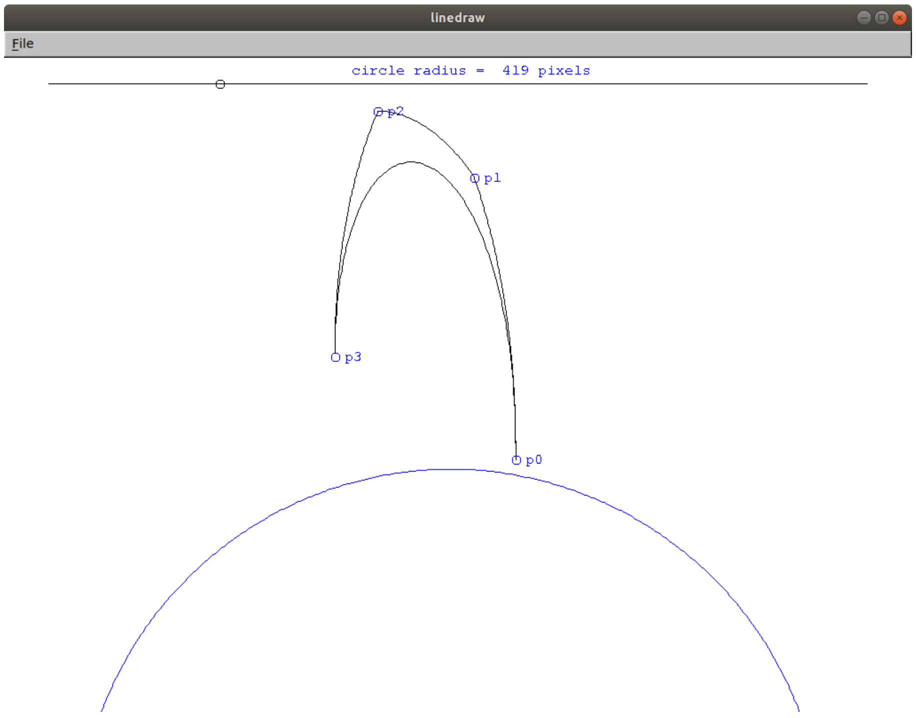


Figure 6: Example Circular Bézier Curve

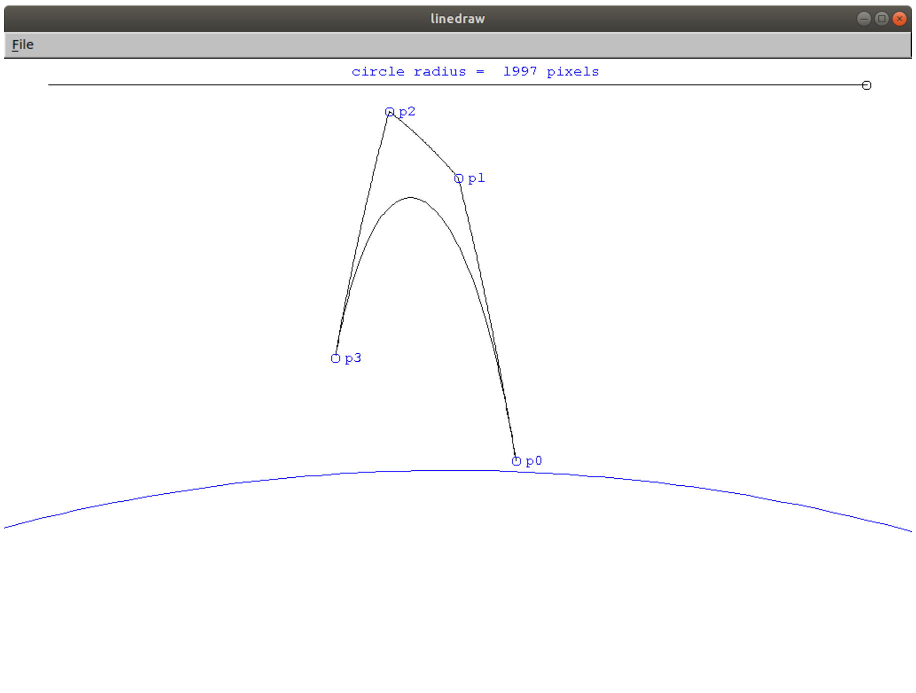


Figure 7: Recover planar results when circle size is large

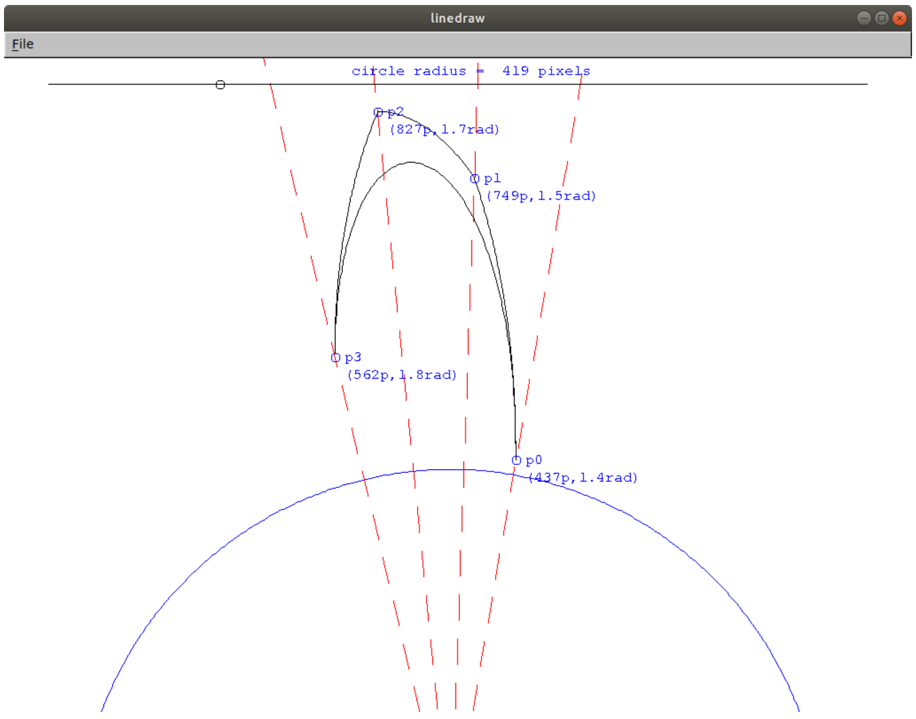


Figure 8: Display Settings

5 What I Learned

- How to solve the problem of interpolating points over the sphere or circle.
- The theory behind spherical Bézier patches and circular Bézier curves (spherical/circular barycentric coordinates, spherical/circular Bernstein polynomials) and how to implement. Compare and contrast with the planar case.
- In the process developed a much better understanding of the standard Bézier patches, Bézier curves, Bernstein polynomials and deCasteljau algorithm.
- The Clough-Tocher algorithm for the spherical case to create C^1 continuity across boundaries (I read about this algorithm in [2], but unfortunately I was unable to complete my implementation of this algorithm).

References

[1] P. Alfeld, M. Neamtu, and L. L. Schumaker, "Bernstein-bézier polynomials on spheres and sphere-like surfaces," *Comput. Aided Geom. Design*, vol. 13, pp. 333–349, 1996.

[2] ——, "Fitting scattered data on sphere-like surfaces using spherical splines," *J. Comput. Appl. Math*, vol. 73, pp. 5–43.

[3] ——, "Circular bernstein-bézier polynomials," in *Vanderbilt University Press (Nashville.* University Press, 1995.

[4] C. Amante and B. Eakins, "ETOPO1 1 arc-minute global relief model: Procedures, data sources and analysis. NOAA technical memorandum NESDIS NGDC-24. national geophysical data center, NOAA," 2009. [Online]. Available: doi:10.7289/V5C8276M