University of Colorado Boulder

Honors Thesis

Uses of Mathematics in Computer Animation and 3D Rendering Software

Peter Rock

Thesis Advisor:
Jeanne Clelland, Department of Mathematics

Honors Council Representative:
Magdalena Czubak, Department of Mathematics

Thesis Committee:
Graeme Forbes, Department of Philosophy
Jason Potter, Department of Philosophy

9 April 2018

- Abstract -

As the title suggests, this paper discusses the applications of several mathematical concepts to computer animation software generally used in the creation of movies and video games. Topics covered will include differential forms, conformal maps, surface texturing, and lighting techniques. It is not the goal of this paper to present anything particularly novel to the mathematical community, but rather to present something that is entertaining to read that will hopefully engage both mathematicians and sane people alike. This paper has been carefully crafted so that it should be accessible to most people with a Calc. I background. That being said, the final section of this paper, which contains some of the applications of math in animation software, should be readable to most people so long as they don't freak out when they encounter the rendering equation.

Contents

1. Introduction

There is a question that seems to haunt almost every Mathematics classroom from middle school through college. It is a question that math teachers not only know is coming from day 1 but also lament answering every. Single. Time. Yes, the infamous, "When am I going to use this?" is a question that stirs the exasperation of math teachers everywhere. Now, this is not because it is a difficult question to answer (indeed, there are whole libraries filled with the answers to this question), but, rather because this is a question that is difficult to answer in a way that is satisfactory for the student since most of the answers that come to mind are examples from STEM fields.

The first math class.

[1]We might say, "well you need to know how to do this so that you can calculate the concentration of your solution correctly," or "you need to know this so you can determine if your results are statistically significant," or even, "so you can tell if the study you heard about is trustworthy or not." But this does little to appease the budding film student or artist who is only in the math course to fulfill a school requirement.

Fortunately, in this age of technology, there is good reason for even the artsiest of folk to care about math! Why? Because computer graphics are a thing, and are fast becoming the main medium for creative expression in the 21^{st} century. Some may argue that it doesn't really matter if creative people know how their software works so long as they know how to use it. To this I respond: does it really seem like a good idea to use anything that you do not have, at least, a basic understanding for? I think not!

And so, in this paper, we will unravel some of the mysteries that surround computer animation software by examining the mathematics that make them tick. Of course, to keep the math at least somewhat entertaining, there will be plenty of puns and fun references strewn throughout this work. For those that are as daft as I am, there are also some recommendations for further research tucked away in the footnotes[2]. Before we begin, there are two more peculiarities that I should mention. Firstly, for the majority of this paper, we will try to stick to at most 3-dimensional space $\mathbb{R}^3$ (trust me, this is weird for math people). Secondly, whereas you may see many books represent points (x, y, z) in $\mathbb{R}^3$ using subscripts like so (x_1, x_2, x_3), in this paper we will be using superscripts (x^1, x^2, x^3) to represent these points. There are good reasons for this, but they lie a bit beyond the scope of this paper.

[1]Image from https://www.smbc-comics.com/comic/a-new-method

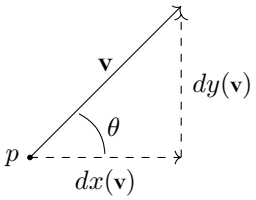
[2]Yes, they will look just like this. This isn't a real recommendation, though; it's merely an example to get you accustomed to the format. I will, however, congratulate you on your superb document navigation skills at this point.

2. A Quick and Dirty Review of Linear Algebra and Calculus

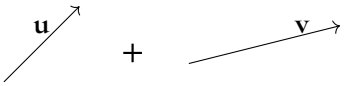
This section is meant to be a hard and fast review of everything that we will need from Calculus and Linear Algebra to approach the rest of this paper. Those that have already taken those courses and feel relatively comfortable with the material can turn to page 394[1].

- Vectors and You -

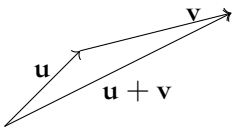
Everything that we will be working on in this paper will require knowledge of vectors. Now, before you go pulling out your can of OFF!™ realize that we are talking about the mathematical object and not the pest (though, I suppose both are usually easily found in fields). Most of the time, people are introduced to the notion of vectors in high school physics where they are told that "vectors have both magnitude and direction." While this is true, it really isn't the most insightful of definitions. Perhaps a better way of thinking of vectors is through the lens of *all* the information that they manage to encode into their structure. Consider the following diagram of a vector $\mathbf{v}$:



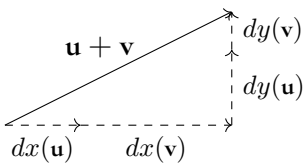
Within this diagram, $\mathbf{v}$ comes with more than just a magnitude and a direction. The vector $\mathbf{v}$ contains a base point p, an angle θ, an x-component which we will call $dx(\mathbf{v})$ (alternatively $dx^1(\mathbf{v})$), a y-component which we will call $dy(\mathbf{v})$ (alternatively $dx^2(\mathbf{v})$), and a length which will be denoted $|\mathbf{v}|$. This is so much better than just "a thing with magnitude and direction." Now we have something that we can actually do math with; more specifically, something that we can add and multiply. Before we do this, however, there is one caveat that we should mention: we can only add and multiply vectors that are based at the same point. For those that have heard of vector addition before, this may seem to fly in direct conflict with the way that you were taught. Many teachers will show the addition $\mathbf{u} + \mathbf{v}$



by



And this is correct in the sense that it produces the right answer, but using this picture can be a bit misleading because it seems to imply that we can just move the base points of vectors around all willy-nilly. A better picture of what $\mathbf{u} + \mathbf{v}$ means is given by the following image:



[1] A cookie to anyone that gets the reference, but seriously, turn to page 11.

It becomes clear from this image that when we talk about adding vectors, it actually means that we are adding the components of those vectors. This is actually a blessing because it means that we can *represent* a 2-dimensional vector $\mathbf{v}$ based at a point p using the following notation:

$$\mathbf{v} = \begin{bmatrix} dx(\mathbf{v}) \\ dy(\mathbf{v}) \end{bmatrix}_p = \begin{bmatrix} dx^1(\mathbf{v}) \\ dx^2(\mathbf{v}) \end{bmatrix}_p = \begin{bmatrix} v^1 \\ v^2 \end{bmatrix}_p.$$

For the sake of simplicity, we tend to drop the p from the subscript since addition is only well-defined between vectors based at the same point. Once we have this representation, though, we can add things really easily! Huzzah!

$$\mathbf{u} + \mathbf{v} = \begin{bmatrix} dx^1(\mathbf{u}) \\ dx^2(\mathbf{u}) \end{bmatrix} + \begin{bmatrix} dx^1(\mathbf{v}) \\ dx^2(\mathbf{v}) \end{bmatrix} = \begin{bmatrix} u^1 \\ u^2 \end{bmatrix} + \begin{bmatrix} v^1 \\ v^2 \end{bmatrix} = \begin{bmatrix} u^1 + v^1 \\ u^2 + v^2 \end{bmatrix}.$$

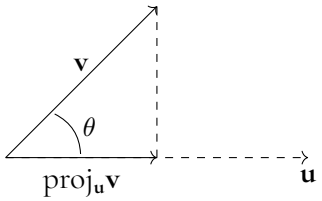
This is wonderful! Now we just need to find a way to multiply vectors and we'll be all set to go! But wait, how on Earth would we want to multiply vectors in the first place? Well, let's go back to what it meant to multiply real numbers. When we talked about multiplying a number a by b, we were actually talking about taking a line segment of length $|a|$ and sticking it end-to-end b times. The resulting line segment was then given a length of $a \cdot b$. There is something that is hidden in this explanation, though[2]: b could have also been represented as a line segment of length $|b|$ that was parallel to the line segment a.

$$\underline{a,\ |a| = 1.5} \qquad \underline{b,\ |b| = 2} \qquad \underline{a \cdot b,\ |a \cdot b| = 3}$$

If we just stick one end of each of these line segments at a point and put arrows at the other end, then these things start to look a lot like vectors. If at all possible, we would like multiplication of parallel vectors to work the same way. So for vectors $\mathbf{u}$ and $\mathbf{v}$ with $\mathbf{u}||\mathbf{v}$, we would like

$$\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}||\mathbf{v}|.$$

But what if $\mathbf{u}$ and $\mathbf{v}$ are not parallel? How could we hope to define multiplication then? Well, we can use a little trick called projection. Pictures become really useful here. The projection of a vector $\mathbf{v}$ onto $\mathbf{u}$ is given geometrically by



At this point, something beautiful happens. If we think back to trigonometry, we realize that we can represent the length of this projection using $\mathbf{v}$, θ, and the definition of cosine for right triangles. That is to say

$$|\text{proj}_{\mathbf{u}}\mathbf{v}| = |\mathbf{v}|\cos(\theta).$$

What's even better is that now we have two parallel vectors. Using what we already know about multiplying parallel vectors, we get that the *dot product* of two vectors $\mathbf{u}$ and $\mathbf{v}$ is

$$\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}||\text{proj}_{\mathbf{u}}\mathbf{v}| = |\mathbf{u}||\mathbf{v}|\cos(\theta).$$

[2] This is starting to become a bit of a theme, no?

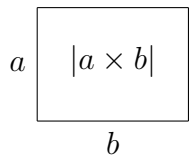
But wait! It gets even better! When we actually want to compute this quantity, it turns out that we only need to multiply the components of each vector and add them together!

$$\mathbf{u} \cdot \mathbf{v} = \begin{bmatrix} dx^1(\mathbf{u}) \\ dx^2(\mathbf{u}) \end{bmatrix} \cdot \begin{bmatrix} dx^1(\mathbf{v}) \\ dx^2(\mathbf{v}) \end{bmatrix} = \begin{bmatrix} u^1 \\ u^2 \end{bmatrix} \cdot \begin{bmatrix} v^1 \\ v^2 \end{bmatrix} = u^1 v^1 + u^2 v^2.$$

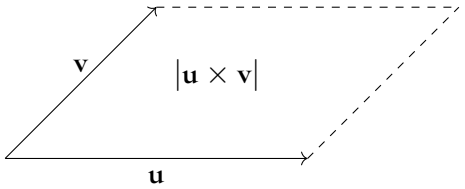
Isn't it beautiful? It almost makes me want to cry. We have managed to multiply two vectors together and in then end the output is a scalar. A wonderful, simple scalar. What's even more wonderful is that we now have a nice way of expressing the magnitude of a vector $\mathbf{v}$ sitting in, say, 3-dimensional space $\mathbb{R}^3$:

$$|\mathbf{v}| = \sqrt{\mathbf{v} \cdot \mathbf{v}} = \sqrt{(v^1)^2 + (v^2)^2 + (v^3)^2}$$

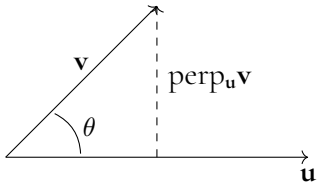
My eyes are starting to water. Before I pull out my handkerchief, however, there is something else that we need to address. When we came up with this definition for the dot product, we motivated it with geometric interpretation of multiplication of parallel line segments a and b. But, there was nothing that said that we *needed* a and b to be parallel. Indeed, if we so chose, we could have placed them perpendicular to each other. In this case, we could define a *different* product $a \times b$ that would give the area of the rectangle with side lengths $|a|$ and $|b|$.



Generalizing this to vectors, we would like to define some sort of product that will give the area of the parallelogram



As we did in the parallel case, we can use a small trick here to deal with arbitrary vectors $\mathbf{u}$ and $\mathbf{v}$. Instead of taking the component of $\mathbf{v}$ that is parallel to $\mathbf{u}$, we can take the perpendicular component



And from this we have the formula

$$|\mathbf{u} \times \mathbf{v}| = |\mathbf{u}||\text{perp}_{\mathbf{u}}\mathbf{v}| = |\mathbf{u}||\mathbf{v}| \sin(\theta).$$

The bad news about this is that giving a nice formula for the computation of this thing for any pair of vectors can be a bit difficult. The good news is that in 3-dimensional space, which is what we will mainly be working with in this paper, the formula isn't too bad. If we were to let

$$\mathbf{u} \begin{bmatrix} u^1 \\ u^2 \\ u^3 \end{bmatrix} \qquad \mathbf{v} = \begin{bmatrix} v^1 \\ v^2 \\ v^3 \end{bmatrix}$$

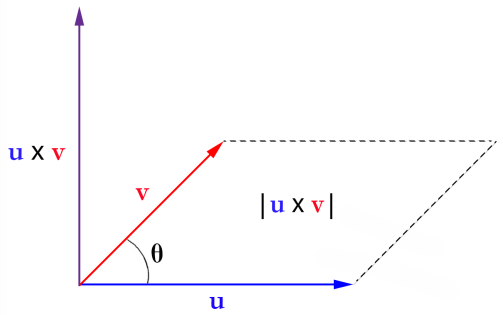
which are three dimensional vectors, then

$$\begin{aligned}|\mathbf{u}\times\mathbf{v}| &= \sqrt{(u^2v^3-u^3v^2)^2+(-(u^1v^3-u^3v^1))^2+(u^1v^1-u^2v^2)^2}\\ &= \sqrt{(u^2v^3-u^3v^2)^2+(u^3v^1-u^1v^3))^2+(u^1v^1-u^2v^2)^2}\end{aligned}$$

As you can imagine, going much higher than dimension 3 for this thing can get really messy (and is better approached using other mathematical tools like the wedge product), so we're going to stick with this definition for now. Now, if we actually look closely at this definition, we can see that there is actually a vector formula hidden in here that we can extract using the definition of "length of a vector" that we derived just a few moments ago. With this we get that the *cross product* of two vectors **u** and **v** is

$$\mathbf{u}\times\mathbf{v} = \begin{bmatrix}u^2v^3-u^3v^2\\ u^3v^1-u^1v^3\\ u^1v^1-u^2v^2\end{bmatrix}$$

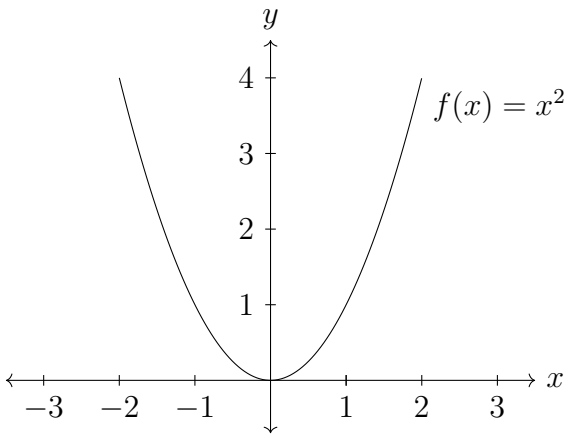
which is another vector sitting in $\mathbb{R}^3$! What's even better, is that this vector sits perpendicular to the plane described by **u** and **v**, so it can be used to *identify* the plane spanned by **u** and **v**.



This method of identification actually becomes really useful when we start talking about tangent planes to surfaces embedded in $\mathbb{R}^3$, which is what we are going to talk about next.

- Parameterizations and Integrals and Derivatives, Oh My! -

Oh dear, where to start? I think that, perhaps, some intuition development is in order here. For this let's think back to what a function, say f, is. Simply put, a function is a set of instructions for how to take some input x and turn it into some output $f(x)$. For example, the function f defined on the real numbers by $f(x)=x^2$ says take the input x, raise it to the second power, and when you're done, that's your output. This has a really nice graphical representation in $\mathbb{R}^2$ (2-dimensional space) as shown below

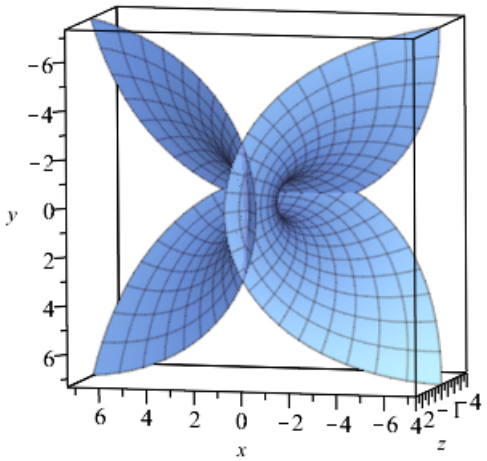


With this representation, there comes a new way of thinking about what functions are. The function $f : \mathbb{R} \to \mathbb{R} : x \mapsto x^2$ (read: function from $\mathbb{R}$ to $\mathbb{R}$ defined by taking x to x^2) shown above actually shows a curve embedded in $\mathbb{R}^2$. So the function f can also be thought of as a set of instructions of how to bend and fold the x-axis (points of the form $(t, 0)$) in $\mathbb{R}^2$ into the curve of points of the form $(x^1, x^2) = (t, t^2)$[3] in $\mathbb{R}^2$. Or, in other words, the function f can be represented by the set of transformations

$$x^1 = t \qquad x^2 = t^2.$$

The equations given above are called *parametric equations*, and since x^1 and x^2 depend on the same single variable, this can be considered a *parameterization of a curve*. Writing out these equations in this way tends to be a bit cumbersome, though, so for the sake of brevity we will use the notation $\widehat{f}(t) = \langle x^1(t), x^2(t)\rangle = \langle t, t^2\rangle$, or, more generally, $\alpha(t) = \langle x^1(t), x^2(t)\rangle$ to denote the parameterization of a curve.

Parameterizations turn out to be very powerful tools in mathematics. In general, they allow us to take k-dimensional objects and represent them in n-dimensional space where $k \leq n$. Even a crazy surface like



can be given a parameterization[4]. This will be really great when we start talking about manifolds embedded in $\mathbb{R}^n$, but for now, let's stick with curves in different dimensions.

We now have an understanding of how to draw curves in arbitrarily many dimensions, but drawing isn't going to be enough. We actually want to do math with these things (surprise!), so we're going to need to develop some tools to work with them. Going back to Calc. I, there were generally two different mathematical objects that we learned to work with: integrals and derivatives.

In a reversal of the usual Calculus curriculum, we are going to talk about integrals first. The wonderful thing about integrals is that the way that we think about them in higher dimensions doesn't really change that much from the way that we thought about them in lower dimensions. Most students are introduced to the idea of a definite integral on an interval $[a, b]$ using the idea of Riemannian sums. In essence, you first draw a bunch of boxes of the same width between a and b and underneath a curve described by $f(x)$ with one of the top corners lying on the curve. Then you add up the the area of these rectangles

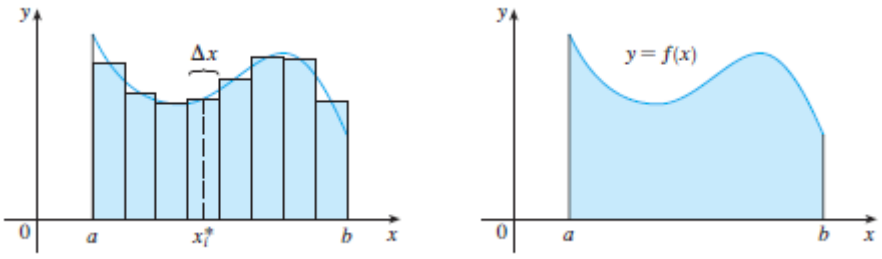
[3] Remember our convention. For spaces of dimension higher than 1, we use superscripts to represent point axes. So a point (x, y, z) in $\mathbb{R}^3$ can also be given the name (x^1, x^2, x^3).

[4] This is actually called Enneper's surface, and it has a parameterization given by $\widehat{f}(u, v) = \langle u - \frac{u^3}{3} + uv^2, v - \frac{v^3}{3} + vu^2, u^2 - v^2\rangle$. I also like to call it the ragoon surface.

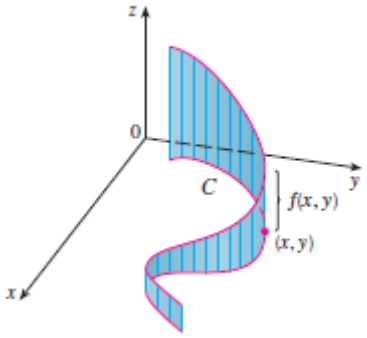
and take the limit of this sum as the number of rectangles goes to infinity. In math speak this looks like

$$\int_a^b f(x)dx = \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i)\Delta x.$$

The images that go with this definition are



Notice that the curve that we are given here is embedded in 2-dimensional space, and that the "area under the curve" is defined to be the area above the x-axis. When we move to 3-dimensional space the idea remains the same except, instead of measuring from the x-axis, we measure from the xy-plane. In graphical terms this looks like



[5]At this point you may say to yourself "Yeah, sure. This looks kinda simple, but how *exactly* am I supposed to integrate this thing?" The answer is that you use the parameterization of the curve. If $C : \mathbb{R} \to \mathbb{R}^2$ is a curve sitting in the xy-plane with parametric equations given by $x(t)$ and $y(t)$ on the interval $[a, b]$, $f : \mathbb{R}^2 \to \mathbb{R}$ is some function sitting above C in $\mathbb{R}^3$, then the line integral of f along C is given by

$$\int_C f(x, y)ds = \int_a^b f(x(t), y(t))\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}\,dt$$

If your eyes are starting to cross while looking at this, then don't worry too much. What's important to remember here is that you *can* take integrals along 1-dimensional objects (curves). If you were wondering if this method could be extended to take integrals along surfaces embedded in $\mathbb{R}^3$, then you would be right; we will actually be covering how to do this in the next section when we start talking about p-forms. For now, though, let's move on to the pièce de résistance: derivatives.

Derivatives are things of beauty in mathematics. They are robust enough that they can give us valuable information about mathematical objects, and yet flexible enough that they are well-defined on a lot of interesting functions. When we originally introduce the derivative of a function at a point in Calc. I, we

[5]Both this and the previous images were from [Ste12].

tend to pitch it as describing the "slope of the tangent line at that point." But now we have a new tool for thinking about curves, and with it, a new way of thinking about derivatives. Let's go back to our old friend $f(x) = x^2$.

When we last saw our friend, we had just given her[6] a parametric formulation $\alpha(t) = \widehat{f}(t) = \langle t, t^2 \rangle$. Now, we already know that the derivative of f is $f'(x) = 2x$. Notice, when we take the derivative of each of the components of $\widehat{f}$ we get $\widehat{f}'(t) = \langle 1, 2t \rangle$. This looks an awful lot like the vector

$$\mathbf{v} = \begin{bmatrix} 1 \\ 2t \end{bmatrix}$$

which we could base at the point (t, t^2). Then when we plug in a value for t we end up with some vector based at a point on the curve described by $\widehat{f}$ that has the same slope as what we expect to see from the tangent line at that same point. This is not a coincidence! Actually, if we have a general curve parameterization $\alpha(t) = \langle x^1(t), x^2(t), \ldots, x^n(t) \rangle$ with vector representation given by

$$\alpha(t) = \begin{bmatrix} x^1(t) \\ \vdots \\ x^n(t) \end{bmatrix},$$

then the derivative of α can be described using the vector formula

$$\alpha'(t) = \begin{bmatrix} \frac{dx^1(t)}{dt} \\ \vdots \\ \frac{dx^n(t)}{dt} \end{bmatrix}.$$

And, if we wanted to be really picky, we could include the point that we are taking the derivative at to get something that looks like

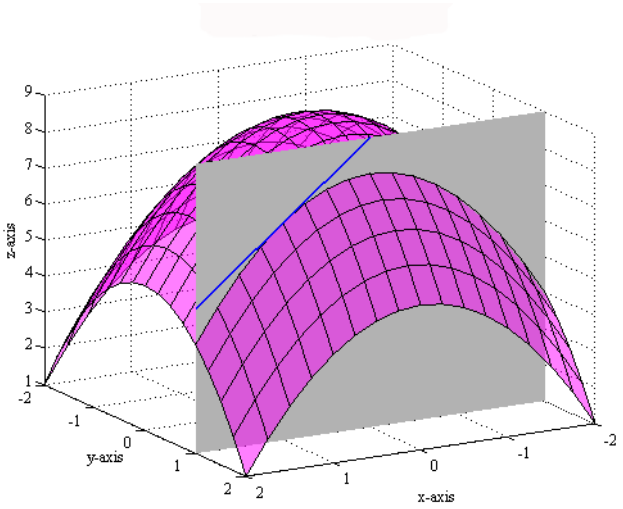
$$\alpha'(t) = \left(\begin{bmatrix} x^1(t) \\ \vdots \\ x^n(t) \end{bmatrix}, \begin{bmatrix} \frac{dx^1(t)}{dt} \\ \vdots \\ \frac{dx^n(t)}{dt} \end{bmatrix} \right).$$

But this notation tends to get a bit cumbersome, so we tend to omit the base point of the tangent vector. However, it's still important to note that it's there! Remember, we only know how to do math with vectors based at the same point.

Brilliant! We can now take derivatives of space curves! All that's left is to figure out how to take derivatives of higher dimensional objects. Well, we run into a bit of a snafu here. While it makes perfectly good sense to talk about the slope of a curve embedded in some n-dimensional space, the idea of taking the derivative of something even as simple as the sphere is abstract enough to leave my head hurting. Fortunately, there is a work around for this. Instead of trying to develop a derivative for the entire surface at once, we can instead pick a point p on our object to take the derivative at and a direction $\mathbf{v}$ to take the derivative in. Then we just need to find some space curve that lies on our object that has a tangent vector that points in the same direction as the one that we've chosen. The slope of that vector then becomes the value of the directional derivative.

Really, like many things that we will encounter in this paper, this concept is something that is best explained with a picture example. Suppose that we want to take the derivative of some function $f(x, y)$ whose graph is a paraboloid at the point $(1, 1)$ and in the x direction. Then our picture may look something like

[6] #feminism



[7] In this image, the gray plane represents the "direction" we are taking the derivative in and the tangent vector for the space curve defined in the x direction will lie on the blue line. There is actually a little bit more nuance that appears in the rigorous definition of the directional derivative, but this sort of pictorial intuition will be more than enough to get us through the rest of this paper.

And that's it for this section! Congratulations on making it through! Now that we have built up a good intuitive foundation for working with Calculus in 3-dimensional space, I'll let you in on a little secret. We have actually been doing Differential Geometry this whole time! Curves like our friend $\alpha(t) = \langle t, t^2 \rangle$ are actually examples of 1-dimensional manifolds, and surfaces like the one in the image above are examples of 2-dimensional manifolds. As we progress into the next section, we are going to delve even deeper into what derivatives are and what we are really doing when we take integrals over manifolds. Isn't it exciting?!

[7] Image from https://math.stackexchange.com/questions/469253/visual-intuition-partial-directional-derivative

3. An Introduction to Differential Geometry

We are finally ready to start discussing Differential Geometry (which I like to call Super Calculus for reasons that will become clear shortly)! Before we get to the exciting topic of Discrete Differential Geometry (DDG), however, we will need to cover some of the basic definitions and theorems that arise in "smooth" Differential Geometry first. Why? Well, over the past thirty years or so, DDG arose as an attempt to apply techniques used in Differential Geometry to computer models. Unfortunately, there are some problems that arise from the attempt to discretize even the simplest notions in Differential Geometry (e.g. the Laplacian, conformal parameterizations, the exterior derivative, etc.). It often turns out that there is no good "canonical" choice for any of these notions, and which version of, say, the Laplacian that is used depends on the properties of the given discrete approximation. This lack of a canonical choice can be attributed to the way that we choose these discrete analogues. There are many ways that a discrete analogue for a smooth operator can be chosen, and, loosely speaking, the discrete analogue is often chosen according to what properties we want to preserve from the smooth case. There are few restrictions on how we choose to go about this, and the only real restriction that we place on these discrete analogues is that they converge to their smooth form under a mesh refinement. There will be plenty more on this later, but first it would be a good idea to get an idea of what some of these smooth objects are.

- Differential Forms -

Differential forms are a wonderful and elegant tool that we use in Differential Geometry to do, well, almost everything. In fact, anyone that has taken Calculus has seen a differential form before; it was just never called that since as soon as we would try to define it rigorously the term "tangent bundle," "cotangent bundle," or "manifold" would pop up and cause undergraduates' heads to explode. But fear not! These objects are not that scary, and, again, we have all seen at least one before. Perhaps the easiest way to think of a differential form is in terms of covectors. Now, mathematically speaking, this notion isn't exactly right for dimensions higher than 1, but it does serve as a great tool.

Let's begin with a review of what a vector is. In the previous section we described a vector as being a point with a direction (or angle) and magnitude (length) attached to it, so a point with a little extra information. There is another way to describe these objects, however. All of the information about the angles of the associated vector is preserved in the relation of this point to the origin.

Now, in mathematics, the prefix "co-" is often used in the nomenclature of the mathematical dual object. Don't worry too much about what this means, precisely, the most important thing to know about covectors is that they behave like functions on vectors. Remember how a normal function f "eats" a scalar x and then "spits out" a different scalar y? Well covectors do almost the same thing except they "eat" vectors and "spit out" scalars.

- 1-forms on a Manifold -

Now, we mentioned that everyone that has taken Calculus has seen a differential form before under a different name. This form normally appears first in the Fundamental Theorem of Calculus

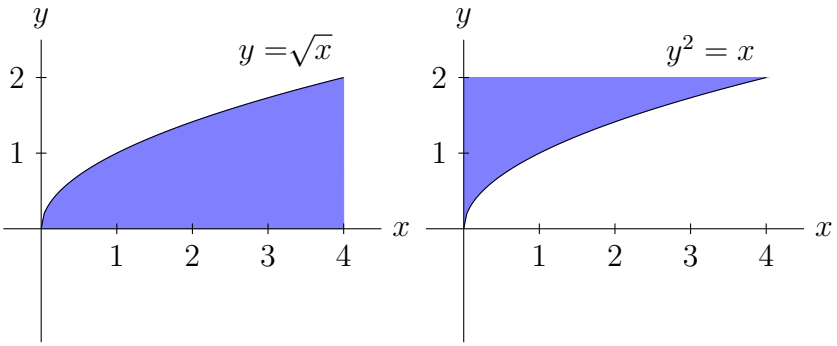
$$\int_a^b f(x)\,dx = f(b) - f(a).$$

In this case the hidden differential form is the dx attached to the end of the integrand. This is an example of a $1\text{-}form$, a differential form that eats a single vector and spits out a scalar (notice: this is the same

as what a covector does). What's the scalar that is spit out? Well if we consider the vector $\mathbf{v} = \langle x, y \rangle$ (which can be thought of as being based at θ and moving x units in the dx direction and y units in the dy direction) and we feed it to the covector dx we see that $dx(\mathbf{v}) = x$ which was the first coordinate of $\mathbf{v}$.

This would seem all well and good but when we take an integral, we don't ever feed the differential form a vector in the first place, so it may almost seem like useless notation that calculus teachers cooked up to torture students. This suspicion is further compounded by the fact that math professors have a propensity to drop the aforementioned 1-form after a couple of steps. The next natural question to ask *why on earth do we include the 1-form in the first place then*? Well, because it's incredibly important, that's why! This 1-form tells us the direction in which we will be evaluating the integral!

When we are first introduced to this differential operator, we are only interested in integrating in one direction: along the positive x-axis. However, there are two ways to integrate the same curve represented in the standard coordinate plane $\mathbb{R}^2$ as seen below.



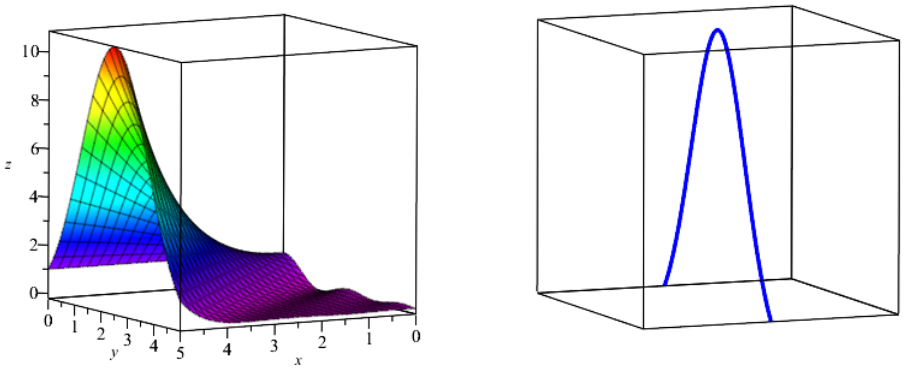
In the first case, we are evaluating the integral of this curve section along the dx direction which ends up looking like

$$\int_0^4 \sqrt{x}\, dx = \frac{16}{3}.$$

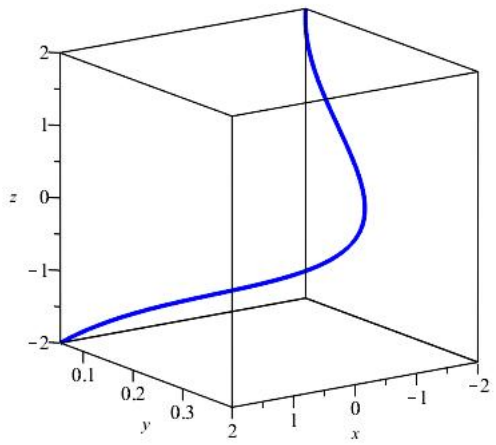
In the second case, we are evaluating

$$\int_0^2 y^2\, dy = \frac{8}{3}.$$

And behold! These two values are not the same! This may seem like an obvious thing now, but it becomes something that is extremely important to keep in mind as we move into higher dimensions. Consider the following two images:



Here, we wanted to integrate in the dy direction over some curve embedded in the surface on the left, so we chose to cut the surface along the plane $x = 0$ which gives the image on the right. This image just looks like a curve in $\mathbb{R}^2$, which is something that we already know how to take the integral over. But what if we want to integrate over something that looks like this:

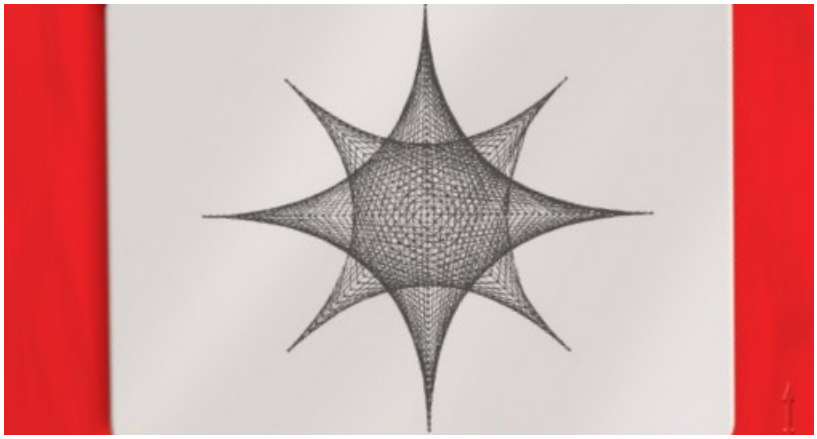


Now, we could try to come up with an equivalent representation of the area under this curve using some sort of transformation to flatten it out, but this is really difficult to do in general, and we actually have a better tool to approach these problems: p-forms.

- p-forms and the Exterior Derivative -

We have arrived at a rather troublesome problem. We want to integrate 3-dimensional shapes, but the only tools that we have so far are limited to integrals where the curve that we are integrating along is a straight line. What we need is a tool that allows us to travel in multiple directions at once and allows us to take integrals over multi-dimensional objects (e.g. surfaces). For inspiration in this we refer to the mighty Etch-a-Sketch.

The principles behind an Etch-a-Sketch are rather simple; the right knob makes lines running from east to west and the left knob makes lines running from north to south. Turning any one knob individually, or sequentially, results in a series of straight lines. But, when the knobs are urned together, it's possible to obtain images like[1]:



[1]Image from http://pixelobby.com/pixeless-thursday-etch-a-sketch/

In our situation, the knobs are 1-forms and we want to find a way to make them work together so that we can take integrals of multi-dimensional objects. We do this using the *wedge product*.

We have actually seen the wedge product before. Back in the linear algebra section, we referred to it as the cross product, but in this case we have something a bit more generalizable. The cross product, wonderful as it is, is really only defined between pairs of vectors. The wedge product, on the other hand, is well defined on both vectors *and* covectors. Unfortunately, the rigorous definition of wedge product lies far beyond the scope of this paper,[2] but we will be able to lay down some basic properties of the wedge product and use some of these properties to derive some useful expressions. We begin with a definition:

Definition 3.1. Let φ and ψ be 1-forms on $\mathbb{R}^n$. Then the *wedge product* of φ and ψ is a skew-symmetric, multilinear product given by

$$\varphi \wedge \psi.$$

Now, there are two new words in this definition that we haven't really talked about yet: *skew-symmetric* and *multi-linear*. Well, an operation like "$\wedge$" is said to be *skew-symmetric* if it has the property that

$$\varphi \wedge \psi = -\psi \wedge \varphi.$$

Now, this property won't be very surprising once we think about it for a moment. If we recall that the sign of cross-product of two vectors depends on the order that the vectors are multiplied in (this is often referred to as the "right-hand rule"), that is if

$$\mathbf{u} \times \mathbf{v} = \mathbf{w}$$

then

$$\mathbf{v} \times \mathbf{u} = -\mathbf{w},$$

then we can see that the cross-product is skew-symmetric. The more difficult property to come to terms with is multi-linearity. Recall (again from linear algebra) that a map $T(\mathbf{u})$ between vector spaces V and W is called linear if

$$T(a\mathbf{u} + b\mathbf{v}) = a\,T(\mathbf{u}) + b\,T(\mathbf{v}).$$

Multi-linearity functions in much the same way except instead of requiring the map to be linear in one variable, we require that it is linear in multiple variables, hence the term. The multi-linearity of a wedge product is pretty easy to express, so we will use this as our pet example:

$$\begin{aligned}(\varphi \wedge \psi)(a\mathbf{u} + b\mathbf{v}, c\mathbf{x} + d\mathbf{y}) = {} & ac\,(\varphi \wedge \psi)(\mathbf{u}, \mathbf{x}) + ad\,(\varphi \wedge \psi)(\mathbf{u}, \mathbf{y}) \\ & + bc\,(\varphi \wedge \psi)(\mathbf{v}, \mathbf{x}) + bd\,(\varphi \wedge \psi)(\mathbf{v}, \mathbf{y}).\end{aligned}$$

As we can see, testing for multi-linearity tends to get rather messy, but we won't have to do that any further here. We just need to know that these wedge products possess this property so that we can make use of it later on. Now, we are finally ready for the definition of a p-form!

Definition 3.2. Let φ and ψ be 1-forms on $\mathbb{R}^n$. Then a *2-form* on $\mathbb{R}^n$ is given by the wedge product

$$\varphi \wedge \psi.$$

Similarly, a *p-form* Φ on $\mathbb{R}^n$ is given by the wedge product of p 1-forms $\{\varphi_i\}_{i=1}^p$ and is denoted by

$$\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_p.$$

[2] If you are curious, there are really good treatments of tensor, symmetric, and wedge products in Chapter 2 of in [Cle17].

We should probably note here that in the same way that 1-forms "ate" singletons (or 1-tuples) of vectors and spat out scalars, p-forms will eat p-tuples of vectors and spit out scalars. Wonderful, we finally have a definition for a p-form, which we mentioned that we can use to calculate integrals, so we can finally start doing useful math, right? Well, wrong, unfortunately. There is still one last thing that we need to work out before we can move on to some of the more applicable stuff. Up until this point we have mainly talked about wanting to be able to integrate things, but in the process of doing this, we have left out a major portion of Calculus: differentiation! If we are to have any notion of what it means to integrate over a p-form, we better have some idea of what it means to take a derivative of one.

This brings us to the notion of the *exterior derivative*, which we will use to turn p-forms to $(p+1)$-forms, but more on that in a minute. This topic can get rather heady and is probably best approached by first defining it on a 0-form. Wait? A 0-form? What is that? Be not afraid, my dear, for 0-forms are some of our best and oldest friends and will be the most familiar of the differential forms that we have met so far. The exact meaning of the term "0-form" is something that we can suss out with only a little more thought. We started this section with 1-forms (a.k.a. covectors) which took 1-dimensional objects (a single vector) and turned it into a 0-dimensional object (a scalar), so if the nomenclature is to be believed, then a 0-form should take a 0-dimensional object (a point) and turn it into another 0-dimensional object (a scalar). Well, this is exactly what a function does! Observe, the function defined by

$$f(x^1, x^2, x^3) = f(x, y, z) = 3xy + y\cos(z)$$

takes some point[3] $(x, y, z) \in \mathbb{R}^3$ and turns it into a scalar $3xy + y\cos(z) \in \mathbb{R}$. Now what happens when we take the derivative of this thing? First we become confused, and we start asking what variable we are supposed to take the derivative with respect to. How about we take all of them? Then we have that

$$\frac{\partial f}{\partial x} = 3y, \qquad \frac{\partial f}{\partial y} = 3x + \cos(z), \quad \text{and} \quad \frac{\partial f}{\partial z} = -y\sin(z).$$

So how do we make a full derivative out of this? Well, we can add them all together, maybe. But to keep track of which direction each part of the derivative was taken in, we better give them a tag. This will give something like

$$df(x, y, z) = 3y\,dx + (3x + \cos(z))\,dy - y\sin(z)\,dz.$$

The astute may recognize this as looking a lot like the definition of the gradient of a function and with good reason. This is exactly the expression for the gradient of the function written in the language of differential forms. Running with this analogy, we know that the gradient of a function gives the direction of maximal increase of the function. This intuitively tells us that if there was to ever be a generalized notion of derivative, this would have to be closely related to it. And it is[4] which means it's definition time!

Definition 3.3. If $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, then the *exterior derivative* of f is a 1-form denoted by df with the property that for any point $\widehat{x} = (x^1, \ldots, x^n) \in \mathbb{R}^n$ and for any tangent vector $\mathbf{u}$ to f based at the point $\hat{x}$

$$df_{\widehat{x}}(\mathbf{u}) = \mathbf{u}[f].$$

In other words, $df_{\widehat{x}}(\mathbf{u})$ is the directional derivative[5] of f at $\widehat{x}$ in the direction of $\mathbf{u}$.

[3] Recall that we are using $(x^1, \ldots, x^n)$ to represent points in $\mathbb{R}^n$

[4] It's almost like I planned this.

[5] For those who have forgotten what exactly this means feel free to look back at the end of Section 2. This stuff is hard and takes some time to get straight in your head.

Looking at this definition, we see that df is, in fact, a 1-form and that by taking the exterior derivative of a 0-form we got a $(0+1)$-form. This is good, but there is another troublesome thing that we have to deal with before we go any further: how do we know that the expression of df that we just derived was the "right" one? Well, we are actually incredibly fortunate in that regard. We happen to know that the 1-forms $(dx^1, \ldots, dx^n)$ form a basis for all the 1-forms on $\mathbb{R}^n$ (in the linear algebra sense) and that any 1-form φ has a unique expression

$$\varphi = \sum_{i=1}^{n} f_i(\widehat{x}) dx^i,$$

where the f_i are 0-forms. Now we move on to p-forms, and again, we will use $\mathbb{R}^3$ as an example.

Working off of the basis of 1-forms (dx^1, dx^2, dx^3) of $\mathbb{R}^3$ can see that in $\mathbb{R}^3$ the basis of 2-forms can be given by $(dx^1 \wedge dx^2,\ dx^2 \wedge dx^3,\ dx^3 \wedge dx^1)$ since we know that $dx^j \wedge dx^j = 0$ (to see why this is, consider our good ol' friend the cross product) and $dx^i \wedge dx^j = -dx^j \wedge dx^i$. So any 2-form $\varphi \wedge \psi$ in $\mathbb{R}^3$ can be expressed as

$$\begin{aligned} \varphi \wedge \psi &= f_1(\widehat{x})\ dx^1 \wedge dx^2 + f_2(\widehat{x})\ dx^2 \wedge dx^3 + f_3(\widehat{x})\ dx^3 \wedge dx^1 \\ &= \sum_{|I|=2} f_I(\widehat{x})\ dx^{i_1} \wedge dx^{i_2} \end{aligned}$$

where I is an indexing set that ranges over all multi-indices (i_1, i_2)[6]. Building off of this, we get a unique expression for a p-form Φ over $\mathbb{R}^n$

$$\Phi = \sum_{|I|=p} f_I(\widehat{x})\ dx^{i_1} \wedge \cdots \wedge dx^{i_p}$$

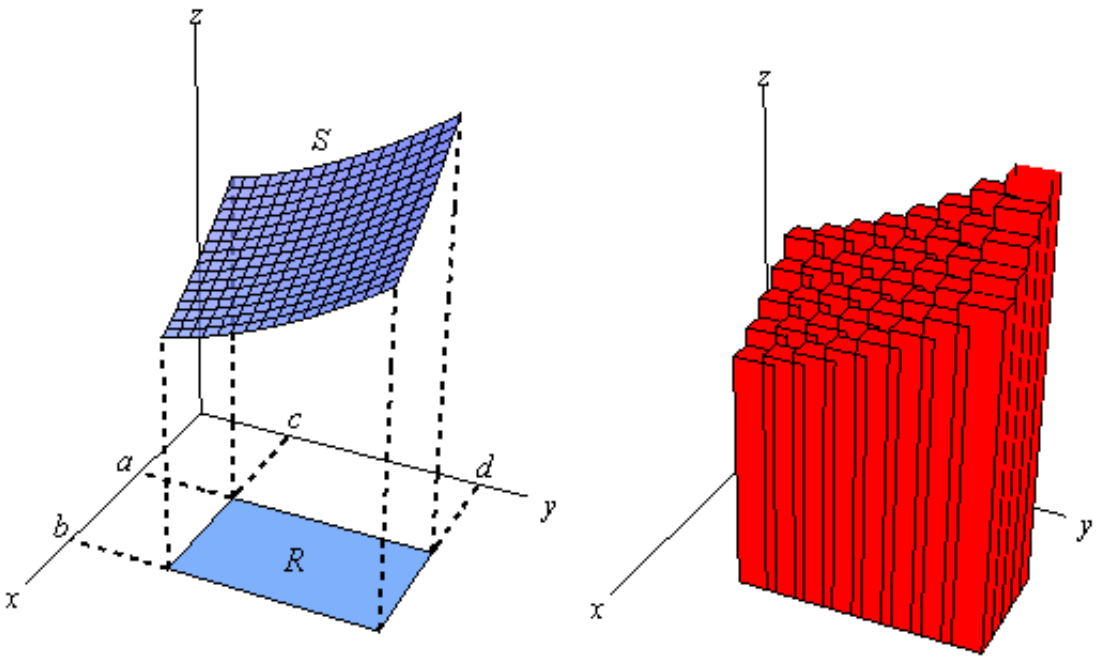
where I is again an indexing set that ranges over all multi-indices $(i_1, \ldots, i_p)$. With this, we are finally able to give a definition for the derivative of a p-form.

Definition 3.4. Let $\Phi = \sum_{|I|=p} f_I(\widehat{x})\ dx^{i_1} \wedge \cdots \wedge dx^{i_p}$ be a p-form on $\mathbb{R}^n$. Then the *exterior derivative* of Φ is given by

$$\sum_{|I|=p} df_I \wedge dx^{i_1} \wedge \cdots \wedge dx^{i_p}.$$

Now my brain hurts. Why did we want these things again? They almost look too complicated to be worth the time. Au contraire, my good friend, these things are incredibly useful. We started this whole endeavor on the quest for integrals in n-dimensional space, so perhaps it's time we actually see why p-forms are the thing that we want to integrate. When we first take Calc. I we are taught that integrals are, essentially, sums of infinitely thin rectangles. When we move on to Calc. III, these infinitely thin rectangles turn into infinitely skinny rectangular prisms as shown below:

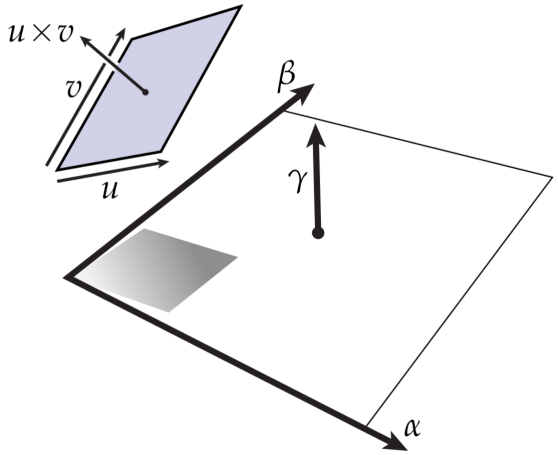
[6] If the second line with the big sigma doesn't make sense, don't worry too much. The important thing is to understand why the first line is true.



[7] How does this all relate to p-forms? Well, it's because when p-forms "eat" vectors, they give us the volume of a p-dimensional parallelogram. This actually translates really well to the cross-product that we have been using throughout this section. When we originally defined the cross product $\mathbf{u} \times \mathbf{v}$ of two vectors $\mathbf{u}$ and $\mathbf{v}$, we said that the vector that the product returned was the normal vector to the plane that they spanned and that the length of this normal vector was the area of the parallelogram that $\mathbf{u}$ and $\mathbf{v}$ created. Now, if we consider the 2-forms that we already defined on $\mathbb{R}^3$, then we get that

$$\begin{bmatrix} dy \wedge dz \\ dz \wedge dx \\ dx \wedge dy \end{bmatrix} (\mathbf{u}, \mathbf{v}) = \mathbf{u} \times \mathbf{v}$$

More specifically, we get that wedge products like $dy \wedge dz(\mathbf{u}, \mathbf{v})$ *return the area of the parallelogram generated by* $\mathbf{u}$ *and* $\mathbf{v}$ *projected onto the plane spanned by* the y and z-axis. Note, these axes can also be represented by the directions dy and dz. Perhaps this is easier to understand with another picture. Let α and β be two orthogonal 1-forms (e.g. dx and dy) and let $\mathbf{u}$ and $\mathbf{v}$ be two vectors. Then the value of $\alpha \wedge \beta(\mathbf{u}, \mathbf{v})$ is given by the area of the "shadow" shown below[8]:



[7] Images form [Daw18].
[8] Image from [Cra17].

Great! Now that we have all of this wonderful exposition on what p-forms are, can we finally see how on earth we are supposed to use them? Right, I'm sorry for the wait, but I promise that it's worth it. We already talked about how p-forms allow us to integrate in higher dimensions, but that's not what makes them great. If you've ever taken Calc. III, then you have probably seen the following theorem:

Theorem 3.5. (Stokes) Let S be an oriented piecewise-smooth surface that is bounded by a simple, closed, piecewise-smooth boundary curve C with positive orientation. Let $\mathbf{F}$ be a vector field whose components have continuous partial derivatives on an open region in $\mathbb{R}^3$ that contains S. Then

$$\oint_C \mathbf{F} \cdot d\mathbf{r} = \iint_S (\nabla \times \mathbf{F}) \cdot \mathbf{n}\, dA,$$

where $\mathbf{n}$ was the unit normal of the surface S.

Just looking at this thing makes my eyes want to cross! There's a ton of stuff in here; gradient (∇), a scalar triple product, the differential of a line parameterization ($d\mathbf{r}$), and whatever dA turns out to be. And it only really works for three dimensions! Fortunately, there is a better way of defining this thing.

Theorem 3.6. (Grandpa Stokes[9]) Let M be a closed, oriented, $(p+1)$-dimensional manifold, ∂M denote the p-dimensional (possibly empty) boundary of M with orientation corresponding to the orientation of M, and Φ be a p-form on M. Then

$$\int_{\partial M} \Phi = \int_M d\Phi.$$

And now we can take an integral of an n-dimensional object with no problems. "Really?" you say dubiously. Yep! "Why?" you ask, feeling evermore incredulous. Because Stokes was a bloody[10] genius and math is[11] beautiful like that.

Now that we have talked at length about the wonderful world of p-forms, we can finally afford to take a step back and talk about objects that make implicit use of p-forms in their definitions. In our case, we are primarily interested in objects called *conformal maps*, which we will see used in order to create textures for animated models.

- Metrics and Conformal Maps -

When we talk about *conformal maps*, intuitively we are describing a function from one space to another which preserves the angles between (tangent) vectors. Notice: this definition says nothing about preserving the lengths of these vectors and those that do are actually a special instance of a conformal map called an isometry. But I'm getting ahead of myself. We should probably start out with an example of what a conformal map looks like. The easiest place to find an example of a simple conformal map would either be in a third grade geography room or in the study of an action movie villain from the 60's. I'm talking of course about globes. It's pretty easy to see that the angles between two points on a globe are the same as the corresponding points on Earth[12], so the "function" that takes the Earth and turns it into a globe is an example of a conformal map.

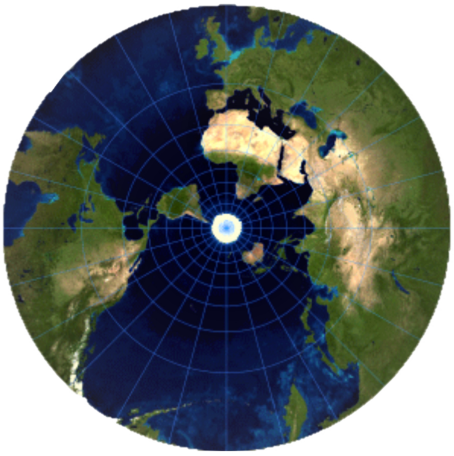
[9] This is actually still called Stokes' Theorem in the literature, but I like to imagine this one wearing a big, gentlemanly mustache.

[10] I feel that there should be some sort of expletive here, but if I did that my proofreader would certainly remove it and I would end up on the receiving end of a lecture I would rather not sit through. Just know that it's there in spirit.

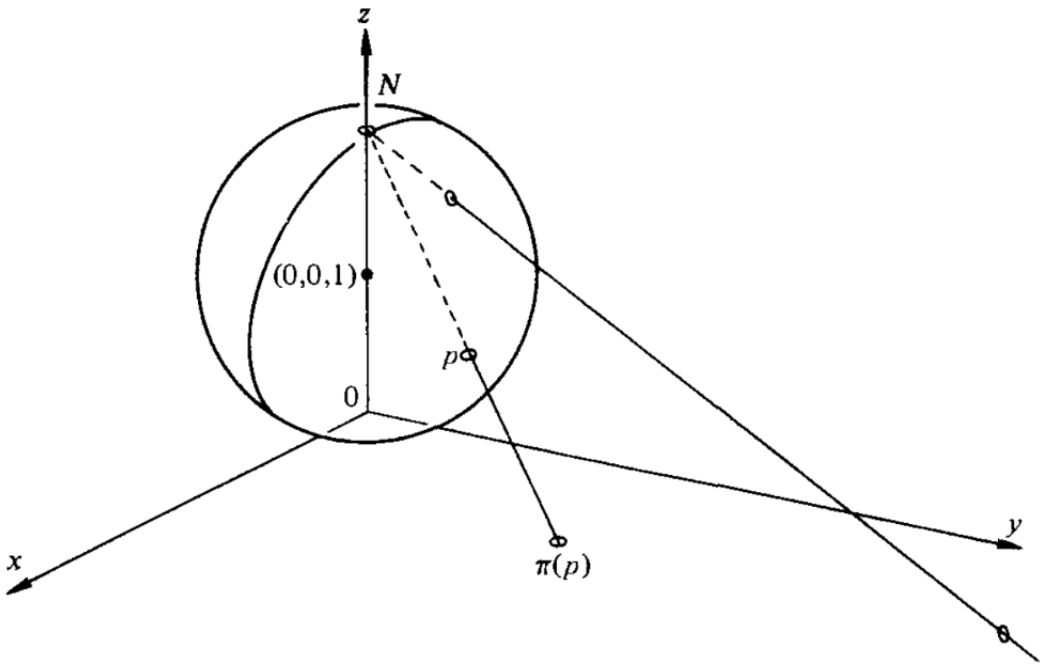
[11] See previous footnote.

[12] Technically, this isn't quite right, but we're going for intuition here so just roll with it.

If we expand on this example, there are other, less obvious examples of conformal maps that we have seen before: World Atlases. In these books you may have seen the following two maps[13]:



If you look closely, you will see that these maps contain complete images of the globe except the one on the left is missing the south pole and the one on the right is missing the north pole. This is not by accident. These charts were constructed using a technique called stereographic projection[14]. Again, a picture becomes useful here:



[15] This image shows the stereographic projection of the globe from the north pole. The projection is constructed by treating the globe like a giant lampshade. First, the north pole of the globe is removed and a "lightbulb" is put in its place. When the light is turned on, the image of the lampshade on the floor forms the stereographic projection of the globe itself.

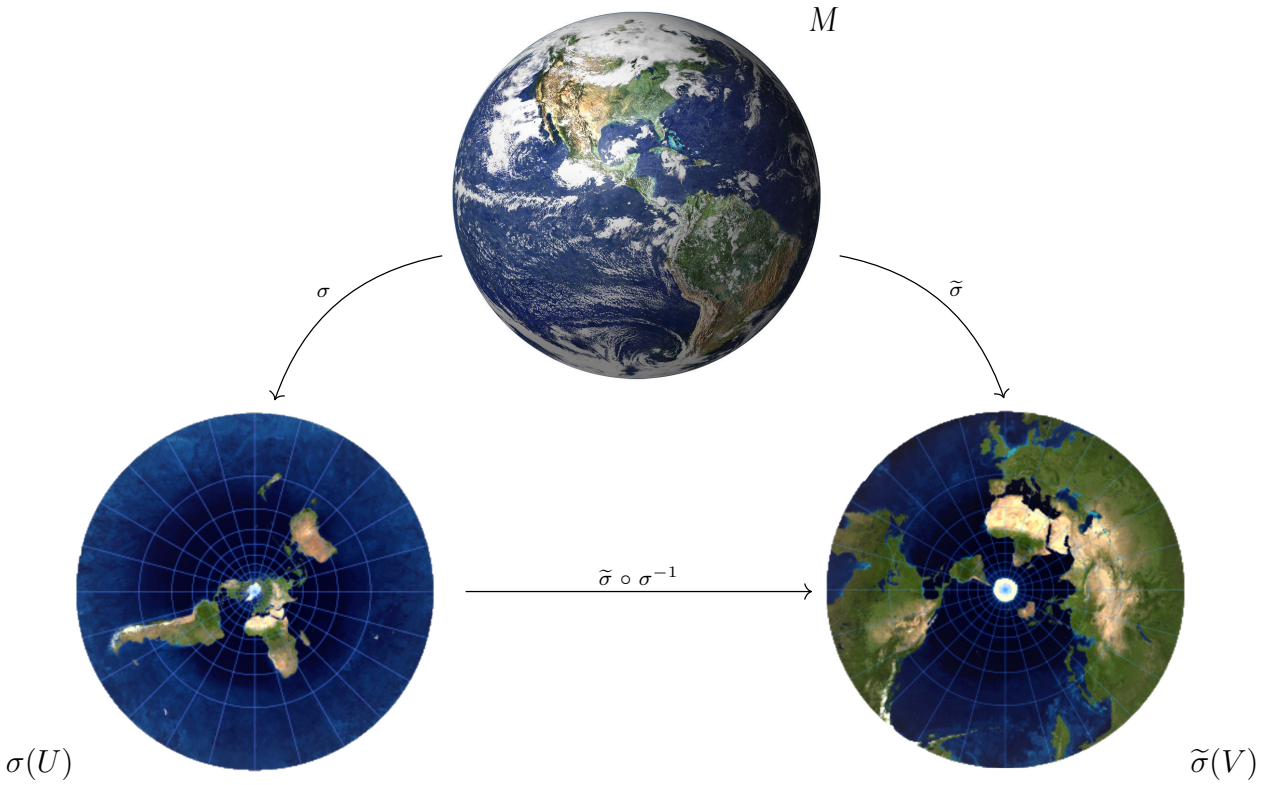
[13] Images from http://www.atractor.pt/mat/loxodromica/projeccao_estereografica-_en.html

[14] If you haven't heard this term before it's worth some looking into. The stereographic projection is one of the main techniques that we use to do math with ∞.

[15] Image from [DC76]

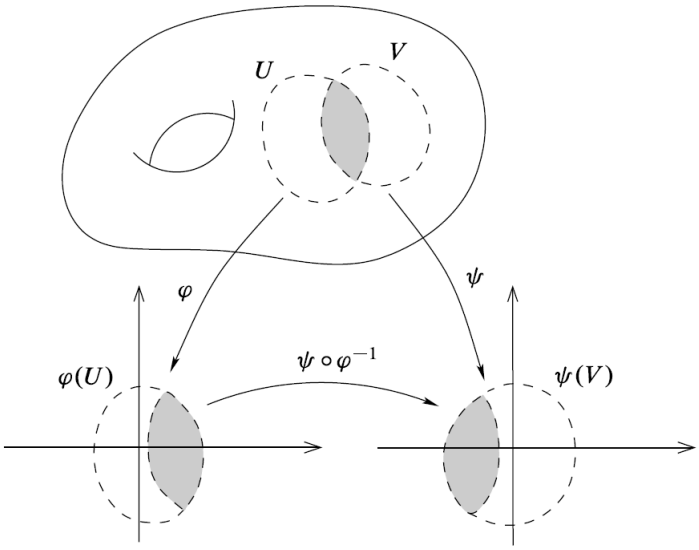
Hopefully, it's believable that the angles between things on the surface of the globe are the same as the angles between their images on the floor. If not, feel free to take a moment to go find a spherical lampshade and test it for yourself.

Back to our two stereographic projections, there is one more thing that we should worry about. Suppose that we define a *chart* on the globe to be the ordered pair (U, σ) such that U is the set that represents the surface of the globe excluding the north pole and σ represents the function that takes U and turns it into its stereographic projection. Similarly, let $(V, \widetilde{\sigma})$ be defined for the globe minus the south pole and consider an angle θ defined by two vectors $\mathbf{u}$ and $\mathbf{v}$ not based at the north or the south pole. We might worry that when we map the image of U on the floor below our lampshade, denoted $\sigma(U)$, to the image $\widetilde{\sigma}(V)$ directly, the angle θ may become distorted somehow. In other words, we may worry that the *transition map* from $\sigma(U)$ to $\widetilde{\sigma}(V)$ is not conformal. This can sometimes be a problem with complex surfaces, but with the way that we have chosen our charts on the globe lampshade this doesn't happen. At this point, we have arrived at something that would be useful to refer to in the future, and, as such, could do with a definition.



Definition 3.7. (Note: There is an image that goes with this on the next page.) Let M be a manifold. If (U, φ) and (V, ψ) are two charts such that the domains U and V have non-empty intersection (i.e. $U \cap V \neq \emptyset$), then the composite map $\psi \circ \varphi^{-1}$ is called the *transition map from* $\varphi(U \cap V)$ *to* $\psi(U \cap V)$ (you can think of this map as a method of transforming the image $\sigma(U)$ from our lampshade into the image $\widetilde{\sigma}(V)$). Two charts (U, φ) and (V, ψ) are said to be *conformally compatible* if either $U \cap V = \emptyset$ or the transition map $\psi \circ \varphi^{-1}$ is a conformal map[16]. We define an *atlas* $\mathcal{A}$ for M to be a collection of charts whose domains (the sets U, V, etc.) cover M. Moreover, we say that $\mathcal{A}$ is a *conformal atlas* if any transition map between two charts is conformal.

[16] It may be worth it to note here that the transition map $\psi \circ \varphi^{-1}$ takes the set $\varphi(U \cap V)$ to the set $\psi(U \cap V)$.



[17] Here's a familiar word. As it turns out, mathematicians aren't the most creative of sorts, and tend to name things with whatever real-life object seems the closest to what we are trying to name. But there may be something a little more concerning here. We have spent the past couple of pages talking about how conformal maps preserve angles, but we haven't really talked about how we can determine the angle between two vectors. We may be tempted to pull out our handy-dandy pocket protractor and just measure the angle using that, but that can become a bit of a hassle especially on a smooth manifold where we would have uncountably infinitely many such angles to measure. Fortunately, we can do this much more efficiently by defining an inner product structure on the given manifold.

Definition 3.8. [18]An *inner product* on the vector space $\mathbb{R}^n$ is a function

$$\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$

with the following properties:

1. *Symmetry:* For any vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$.
2. *Bilinearity:* For any vectors $\mathbf{u}, \mathbf{v}, \mathbf{z} \in \mathbb{R}^n$ and any scalars $a, b \in \mathbb{R}$,

$$\langle a\mathbf{u} + b\mathbf{v}, \mathbf{z} \rangle = a\langle \mathbf{u}, \mathbf{z} \rangle + b\langle \mathbf{v}, \mathbf{z} \rangle$$

   and

$$\langle \mathbf{z}, a\mathbf{u} + b\mathbf{v} \rangle = a\langle \mathbf{z}, v \rangle + b\langle \mathbf{z}, \mathbf{v} \rangle.$$

3. *Positive definiteness:* For any vector $\mathbf{u} \in \mathbb{R}^n$, $\langle \mathbf{u}, \mathbf{u} \rangle \geq 0$ with equality if and only if $\mathbf{u} = \mathbf{0}$.

This actually leads to a definition of something that has been said a couple of times in this paper, but that has not been made explicitly clear.

Definition 3.9. The vector space $\mathbb{R}^n$ endowed with an inner product $\langle \cdot, \cdot \rangle$ is called *Euclidean Space* and is denoted $\mathbb{E}^n$.

Like the wedge product, this inner product is something that we have seen before in disguise. In this case, instead of corresponding to the cross product from linear algebra, the inner product corresponds

[17] Image from [Lee03].
[18] This is exactly the definition given on pg. 70 of [Cle17].

to the dot product. In fact, the way that we calculate the inner product of two vectors in $\mathbb{R}^3$ is exactly the same way that we calculate the dot product of two vectors. What may not be so obvious, however, is that when we impose an inner product structure on a vector space like $\mathbb{R}^n$, we automatically determine a metric (a definition for "length") on that space. In the case of $\mathbb{E}^n$ the length of a vector $\mathbf{u}$ is given by

$$|\mathbf{u}| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}.$$

But for conformal maps we are interested measuring the angle between vectors, not their length. For this it's useful to remember that

$$\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}||\mathbf{v}| \cos(\theta).$$

using this and the fact that the dot product is a special case of the inner product, we get that the angle θ between two vectors $\mathbf{u}$ and $\mathbf{v}$ in Euclidean space is given by

$$\theta = \cos^{-1}\left(\frac{\langle \mathbf{u}, \mathbf{v} \rangle}{|\mathbf{u}||\mathbf{v}|}\right).$$

Notice here that, since we divide by the length of the vectors $\mathbf{u}$ and $\mathbf{v}$, the angle θ between $2\mathbf{u}$ and $\mathbf{v}$ is the same as the angle between $\mathbf{u}$ and $\mathbf{v}$, and, as it turns out, this is exactly the sort of thing that we need to determine if a map is conformal.

Definition 3.10. Let M and N be manifolds with inner product structures given by $\langle \cdot, \cdot \rangle_M$ and $\langle \cdot, \cdot \rangle_N$. A map $F : M \to N$ is called *conformal* if for any pair of vectors $\mathbf{u}$ and $\mathbf{v}$ tangent to M at some point $\widehat{p} \in M$

$$\langle \mathbf{u}, \mathbf{v} \rangle_M = \lambda_{\widehat{p}} \langle dF(\mathbf{u}), dF(\mathbf{v}) \rangle_N,$$

where $\lambda_{\widehat{p}}$ is some scalar dependent on the choice of the point $\widehat{p}$ and dF is the differential of F. (Note: dF behaves like the derivative in the sense that it is the best linear approximation to F at $\widehat{p}$. It is actually a map that operates by taking tangent vectors to M at $\widehat{p}$ to tangent vectors to N at $F(\widehat{p})$.)

Finally! We have arrived at a good definition. At this point, I would like to congratulate you. This stuff is really difficult, especially the last two pages. It may not look like it, but we have secretly been wrestling with objects that most mathematicians don't see until grad school, and even then these things can take a long time to really understand. If you have made it this far, though, I might suggest a five minute break before moving on to the next section; if you have not made it this far then, clearly, you are already ahead of me. Anyway, for the rest of this paper we will look at how exactly these structures that we have defined on smooth manifolds translate to discrete ones and why these sort of things matter for 3D animation and rendering.

4. Moving into the Discrete Case

Before we fully begin this section, a bit of a warning is in order. This will be the section of this paper that is most challenging, mathematically speaking. I will try to provide intuitive examples and explanations for things, but we are fast approaching the limit of where such things are useful or even possible. That being said, with everything that we have covered up until now, the last section of this paper should be accessible. Should you find your eyes glazing over and rolling into the back of your head at any time during this section, feel free to move on to the next section. That is, after all, the upshot of this whole thing.

Alright, disclaimer over. In this section we get into the nitty-gritty of discrete manifolds. What they are, how they are different from smooth manifolds, and, most importantly, how the structures and things that we defined for smooth manifolds translate to the discrete case. As it turns out, not everything that we define on smooth manifolds has a nice, ready-made discrete analogue. In fact, we often find that objects that were relatively intuitively defined on smooth manifolds, like the derivative at a point, become much more difficult to define in a discrete setting. Likewise, things that were a little tricky to define in the smooth setting are readily defined on discrete structures[1]. I think that Dr. Keenan Crane from Carnegie Mellon puts it best when he says research in Discrete Differential Geometry is often organized like a game. This "game" comes with three steps:

1. Write down multiple equivalent definitions of a smooth object or theorem.
2. Apply each of these smooth definitions to a discrete object.
3. Weigh the pros and cons of the discrete analogues under these different definitions.

This last step turns out to be the most important one. We often find that when we choose to apply multiple definitions to the discrete setting, the objects that we get out fail to preserve all of the properties that they used to have in the smooth setting. The definition that we ultimately decide to use is often determined by the properties that we want to preserve and how "good" an estimation the discrete analogue provides.

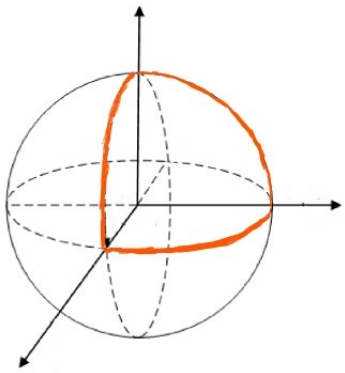
- Dealing With Discrete Manifolds -

At the beginning of the previous section, we alluded to some of the major differences between the smooth and discrete versions of Differential Geometry. These differences arose principally from the way that we chose to define what it meant to be a "good" discrete version of a smooth concept. Specifically, we said that that a given estimation was "good" if it *converged under mesh refinement*. What exactly does this mean? Before we can start into this, we need to talk about triangles.

In mathematics, and especially in geometry, triangles are wonderful things. In fact, they can be used to encode many things about the geometry of the manifold that they lie on. For example, the sum of the angles of a triangle can determine if a manifold is locally concave, flat, or convex. Consider the the normal unit sphere embedded in $\mathbb{R}^3$ and the triangle along the surface of the sphere described by the points $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$. This triangle has angles that sum to 270^o, which is greater than the standard 180^o that we see in Euclidean space. This small fact tells us that the sphere is an example of a locally elliptic manifold. Similarly, if the angles of a triangle add up to less than 180^o, the manifold

[1] A good example of this is the Euler characteristic, which is given a good discussion in §6.6 of [Opr07].

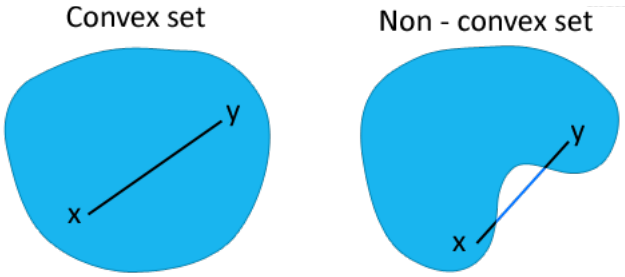
is locally hyperbolic[2].



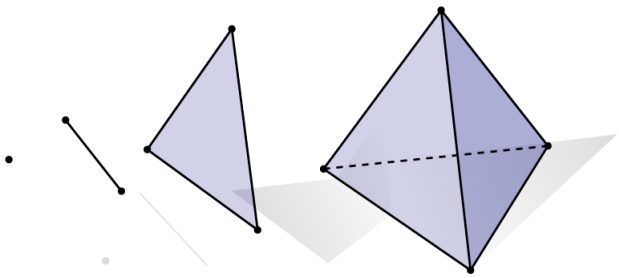
The triangles in the image above (one highlighted in orange) are made using curved lines which don't do us any good when we are modeling using computers. The point still stands, however, that triangles are important to our understanding of the geometry of the manifold itself. But manifolds are not always nice and have this annoying tendency to live in n-dimensional space, so, we have to construct an n-dimensional analogue of a triangle called a *simplex*.

Definition 4.1. Let $V = \{v_0, v_1, \ldots, v_k\}$ be a collection of $k+1$ points. A k-simplex is the minimal convex set that contains the points $\{v_0, v_1, \ldots, v_k\}$.

And how, exactly, is this thing supposed to be a triangle? Let's start with what it means to be a convex set. Simply put, a convex set is a connected set such that any pair of points in that set can be connected with a straight line and that line will remain within the set. Here's a picture:



[3] Extending this notion to include vertices of a a polytope[4] we get that k-simplices look something like the following:



[5]In the image above we have, from left-to-to right, a 0-simplex, a 1-simplex, a 2-simplex, and a 3-simplex

[2]Elliptic and Hyperbolic geometries actually pop up everywhere in science. It is actually a really common practice in physics to embed hyperbolic surfaces in Minkowski space which is used to model space-time in the study of Einstein's theory of special relativity.

[3]Image from https://www.easycalculation.com/maths-dictionary/convex_set.html.

[4]Think k-dimensional polygon.

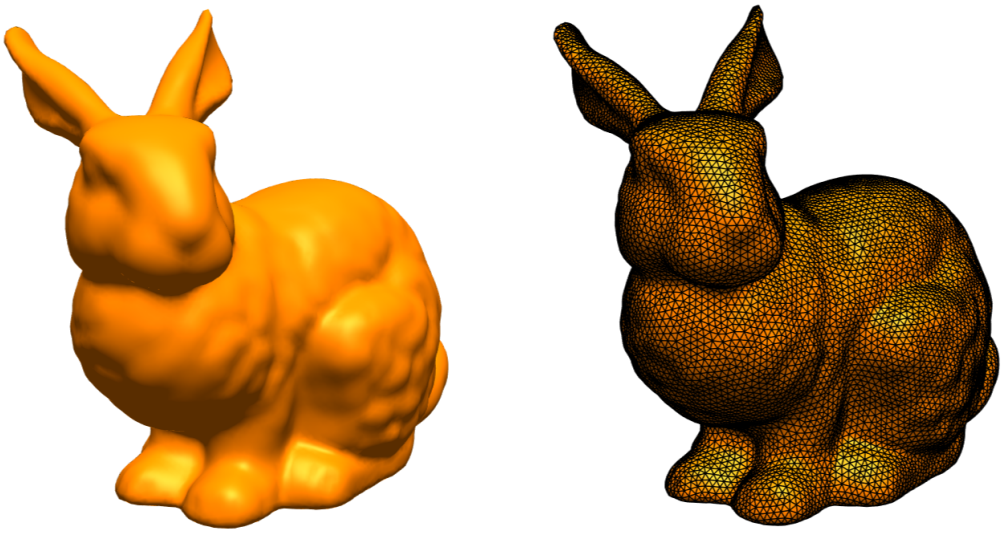
[5]Image from [Cra17].

all of which correspond to what we would think a "triangle" would look like in 0, 1, 2,and 3-dimensional space. However, in our work, we will not just be working with a single k-simplex because that would be boring. Instead, we want to be able to talk about groups of simplices and the relation of each of those simplices to the overall group. First of all, if we have a simplex named σ and another simplex named τ where τ is contained in σ ($\tau \subseteq \sigma$), then τ is called a *facet* of σ. For example, if σ is a tetrahedron and τ is one of the faces of σ, then τ is a facet of σ. This brings us to the notion of a simplicial complex.

Definition 4.2. A *simplicial complex* Σ is a union of simplices, such that

1. If a simplex σ belongs to Σ, then all its facets also belong to Σ.
2. If two simplices in Σ intersect, then their intersection is a common facet.

How does this all relate to manifolds? As it turns out, there is an old theorem from topology that says any 2-manifold (think surfaces) sitting in Euclidean space can be triangulated. That is to say, any smooth surface can be approximated using a triangular mesh. What, exactly, does this look like? Fortunately, there are pictures for these things:



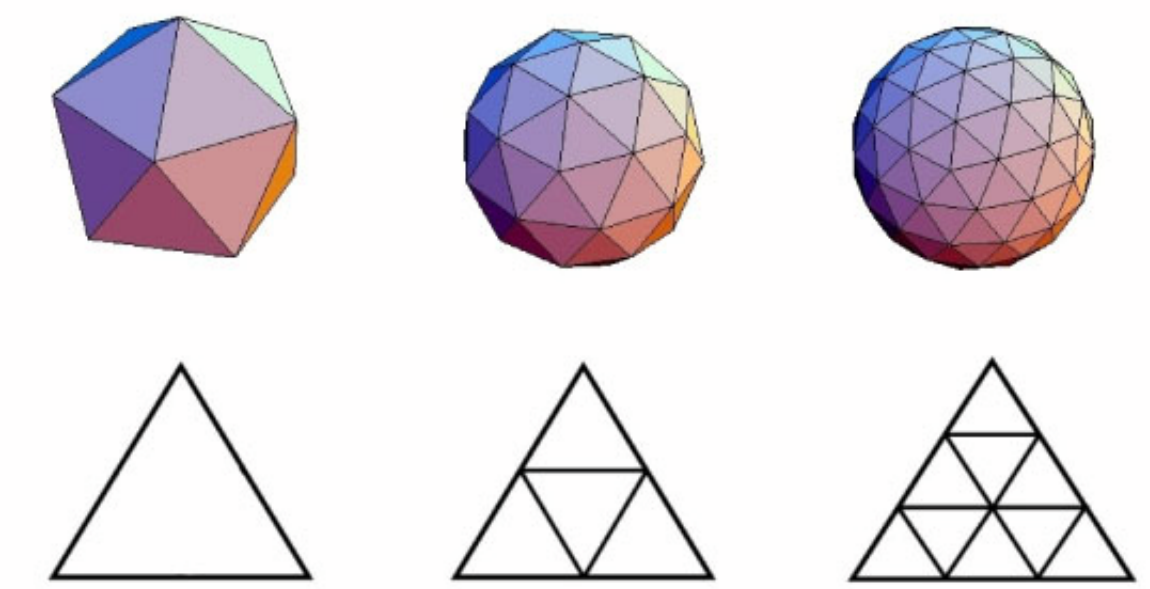
In the image above on the left we have a smooth version of the animator's favorite lagomorph: the Stanford Bunny. On the right, we have a triangulation (a.k.a. a triangular mesh or just a mesh) of the smooth bunny. The astute reader will notice at this point that the image on the right looks suspiciously like a simplicial complex, and that's because it is. A triangulation of a 2-manifold actually turns out to be 2-dimensional simplicial complex (i.e. all the facets are at most 2-simplices) that is also a manifold in its own right. And just like that we have stumbled upon our first example of a discrete manifold[6].

This is wonderful but, now that we know what a discrete manifold is, we have to start doing math on them. This can get a bit tricky when we start defining mathematical objects that we hope to be similar to those that we defined on smooth manifolds. How are we to do this? Well, we have now mentioned several times that we want our discrete objects to converge to their smooth analogues under refinement of a mesh, so, perhaps, it would be a good idea to talk about what exactly it means to refine a mesh.

As with most things in DDG, there are several good ways that we can choose to refine a mesh. Since we are working with triangular meshes (and, yes, you could choose to work with a mesh made of, say, quadrilaterals), we will usually refer to a $\sqrt{3}$ mesh refinement. But, I'm getting ahead of myself. First we

[6]We are actually making a bit of an assumption here. Not all discrete manifolds turn out to be simplicial complexes, but they can be turned into simplicial complexes fairly easily.

need to discuss what a mesh refinement actually does. Well, the best way that I could describe it is that a mesh refinement takes a mesh and makes it *look* smoother while preserving the underlying geometry of the object. It does this by breaking up the triangles in such a way that all of the triangles in the resulting mesh are significantly smaller by comparison. As we do this ad infinitum, we want these triangles to converge into points and our mesh to converge to the smooth object that we were approximating.



So what does it mean for an object in DDG to *converge under refinement* to its smooth analogue? It simply means that if we were to approximate a smooth 2-manifold with a triangular mesh then any object that we define on that triangulation needs to behave like its smooth analogue as the number of faces in the mesh go to infinity.

- Calculus on Discrete Manifolds -

Now that we have the necessary slew of definitions out of the way, it's time to do some math![7] Right, so the main reason that we have had manifolds up to this point is so that we can do calculus on them, and discrete manifolds are no different. Now we have to be more careful, however, as we have mentioned multiple times that discrete surfaces can wreak havoc on even the most carefully constructed of definitions. We started the last section with differential forms, so why don't we revisit those again in this new setting.

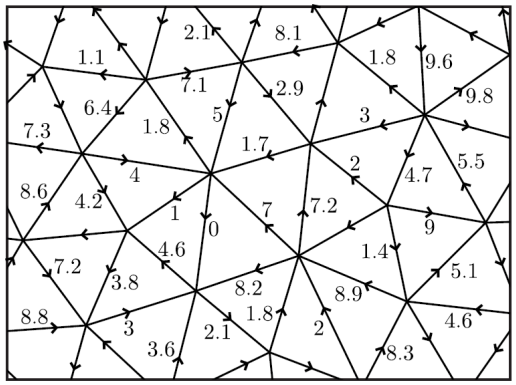
First let's remember what 1-forms were supposed to do. We had an example of the 1-form dx which took any vector $\mathbf{v}$ and pulled off the x component. So, the 1-form gave us the degree to which the vector $\mathbf{v}$ went in the same direction as dx. When we started integrating 1-forms over a curve C we actually took the tangent vector at each point along the edge of C, fed it to the 1-form, and summed up the resulting values. We can use this same principle to describe a discrete 1-form. If we let φ be a 1-form, Σ a simplicial complex, and $e \subseteq \Sigma$ a 1-simplex, then the discrete 1-form of φ along e is given by

$$\widehat{\varphi}_e := \int_e \varphi.$$

By describing the discrete 1-form in this way, we are then able to encode the information gathered by

[7] I can only imagine enthusiasm radiating off you at this exact moment.

this 1-form into the simplicial complex itself by storing the values of the these 1-forms along the edges of the mesh as can be seen in the following image:



[8] If we actually extend our notion of a discrete 1-form to p-forms, we get that discrete differential p-forms are just the values of p-forms evaluated over a p-dimensional cell.

Now, this seems all well and good, but when we start to define 1-forms in this way we can run into a bit of trouble. Since our discrete 1-form is just a set of values that is stored along the edges of our mesh, there is nothing that really stops us from futzing with these values. In fact, we could just assign numbers to the edges of the mesh randomly and it would make a perfectly good discrete 1-form. Likewise, there is nothing that says that the values that we store on these edges need to be real-valued; choosing instead to assign complex numbers, vectors, or counts of pineapples to each edge works out completely well so long as we remain consistent across all edges in a mesh. But if these things are so easy to exploit, then why on earth would we want to define our 1-forms this way? Well, because it makes taking derivatives and integrals almost stupidly easy.

Remember Grandpa Stokes' Theorem? In case you don't, here's a refresher:

$$\int_{\partial M} \Phi = \int_M d\Phi.$$

Basically, this says that we can integrate any differential form so long as we know what the integral is along the boundary. But wait just a tick; we already calculated the integral of a 1-form over all the edges in our mesh, which means that for any triangle σ and any smooth 1-form φ,

$$\int_\sigma d\varphi = \int_{\partial\sigma} \varphi = \sum_{i=1}^{3} \int_{e_i} \varphi = \sum_{i=1}^{3} \widehat{\varphi}_i.$$

Furthermore, we get that the thing on the left is actually a discrete differential 2-form, which we will now call $\hat{d}\widehat{\varphi}$. This new operation $\hat{d}$ is exactly what we expect it to be: it's the *discrete exterior derivative*. Well now, this is quite lovely. By defining our 1-forms as values associated to edges of our simplicial mesh, we have gotten that the discrete exterior derivative simply applies Grandpa Stokes' to a p-form that has already been integrated over each p-simplex and returns the integral of the derivative over each $(p+1)$-simplex. Now, there needs to be a small word of caution when it comes to applying this thing

[8] Image from [Cra17]. For those that are interested, this image is also an example of a weighted, directed graph. It turns out that many of algorithms for actually computing the values of discrete objects described in DDG actually employ results graph theory.

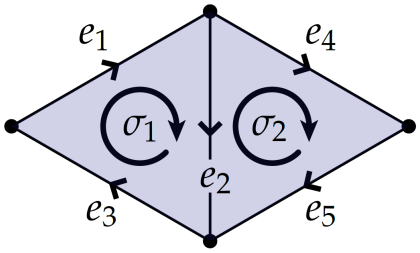
to a real mesh. In the statement of Grandpa Stokes' Theorem, we assumed that the manifold that we were working with had an orientation. We want this to still be the case when we are integrating over simplicial meshes! This tells us that we need to define d by adding the piecewise integrals over the simplicial complex, and that we need to be especially conscientious of the sign of the 1-form that we are adding. For example, we may have

$$(\hat{d}\widehat{\varphi})_1 = \widehat{\varphi}_1 + \widehat{\varphi}_2 + \widehat{\varphi}_3$$

and

$$(\hat{d}\widehat{\varphi})_2 = \widehat{\varphi}_4 + \widehat{\varphi}_5 - \widehat{\varphi}_2$$

when we consider the following pair of 2-simplices:



[9]With all of this groundwork finally in place, we can move on to the upshot of this whole section: discrete conformal mappings.

- Discrete Conformal Mappings -

It's finally time for the last topic of this section. It's been long and arduous process getting here, but, out of everything that we have seen, this is actually the thing that will be most directly applicable to the world of 3D rendering. If we remember way back to the end of the last section of this paper, we will recall that a map $F : M \to N$ is called conformal if for any pair of vectors $\mathbf{v}$ and $\mathbf{w}$ tangent to M at some point $\widehat{p} \in M$,

$$\langle \mathbf{v}, \mathbf{w} \rangle_M = \lambda_{\widehat{p}} \langle dF(\mathbf{v}), dF(\mathbf{w}) \rangle_N,$$

where $\lambda_{\widehat{p}}$ is some scalar dependent on the choice of the point $\widehat{p}$. This notion of conformality is dependent on there being a defined metric (sense of length) on the manifold that we are interested in, so we should first give a definition for what it means for a discrete manifold M. First we will note that the types of discrete manifolds that we are working with come equipped with a set of Vertices (V), a set of Edges (E), and a set of Triangles (T) and can be described using the triple $M = (V, E, T)$.

Definition 4.3. A *discrete metric* on M is a function ℓ on the edges of M that assigns to each edge $e_{ij} \in E$ a positive number ℓ_{ij} so that the triangle inequality holds for all triangles $t_{ijk} \in T$.

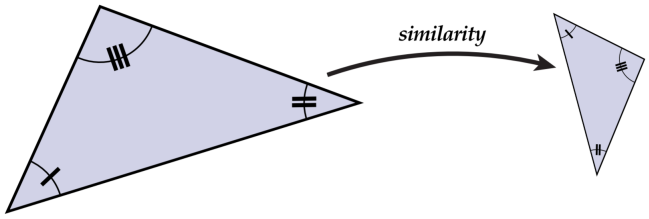
Great, now we have a way to measure lengths and, since the manifolds that we are interested in are already embedded in Euclidean space, we also have a way to measure angles between vectors, so we are completely ready to start trying to construct conformal maps.

We saw in the last section that if we considered the Riemannian sphere with one of the poles removed then we could use the stereographic projection to map the globe down into a small disk. This is actually a special case of the Riemann mapping theorem which says that any simply-connected (no holes), bounded (has an end that you can fall off) Riemannian manifold can be conformally mapped to the unit disk. If we were to design some sort of conformal map on discrete 2-manifolds for use in computer animation, it

[9]Image from [Cra17].

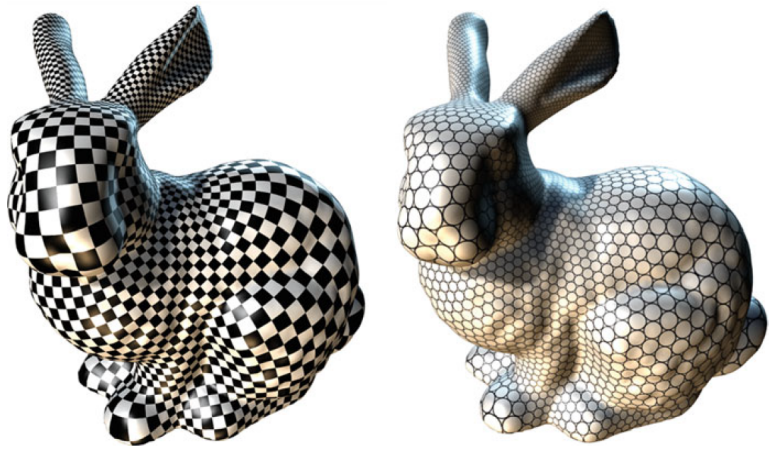
would be really nice if the map would have this property since this would allow us to fold plane textures (like wood) into complicated shapes without distorting that texture.

Well, the last time that we wanted to make a conformal map, we required that that the angles between vectors on our manifold be preserved, so it would be nice if discrete manifolds would allow us to use this same definition. Unfortunately, this definition turns out to be a little more rigid than we would like it to be. By requiring that the angles between vectors be preserved completely, we end up imposing an unintended condition on the set of triangles T of our discrete manifold. If all of the angles between vectors are preserved under the conformal mapping of M, then so too must the interior angles of any triangle t_{ijk} be preserved. This means that the only thing that a given triangle t_{ijk} would be able to map to would be a similar triangle



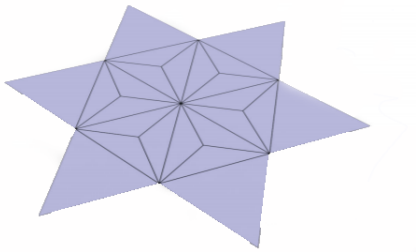
[10]Why is this a problem? All of the triangles in our mesh are connected; more importantly, they are connected along their edges. This means that once we make a choice on how to scale a single triangle, the entire rest of the map is completely determined. What's even worse is that when we do this, we can't perfectly flatten the manifold, much less turn it into a disk. To even get close requires solving a non-linear optimization problem with non-convex, non-linear constraints (translation: it's a massive headache).

This is just peachy. We've hit a brick wall; what on Earth are we supposed to do now? It's time to consult our favorite long-eared friend.

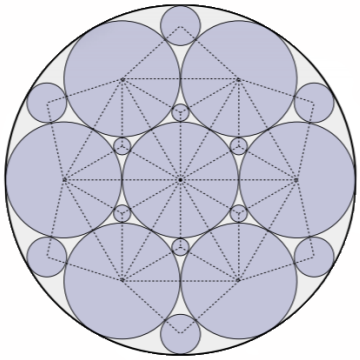


[11]Our next idea comes to us from a man by the name of William Thurston. Back in the 1980s, Thurston observed that conformal mappings between manifolds preserved infinitesimal circles. To see why this could be exciting, we first need to note that any planar graph $G = (V, E)$ can be turned into a circle packing. How? Well if we consider a planar graph like the following:

[10]Image from [CWW+18]
[11]Image from [ZG13].



Then if we were to project the vertices of this graph onto a plane (easy to do in this case), a circle packing for this graph could them be constructed by first placing each of the vertices at the center of some small circle. After that, each of the circles can be inflated in such a way so that they touch tangentially and fill some circumscribing circle like so:



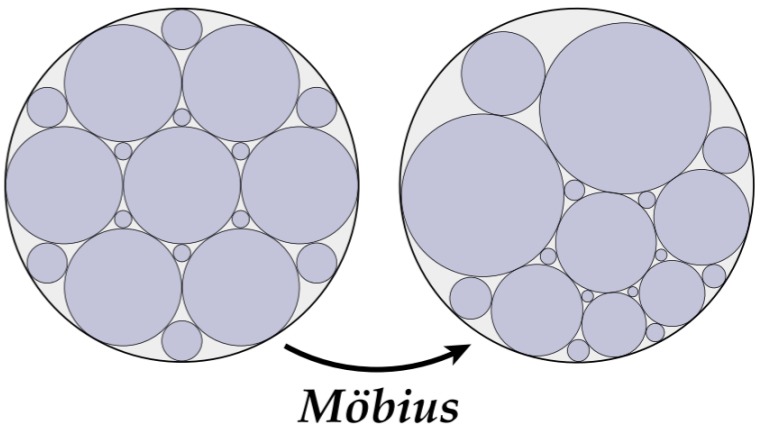
If we combine this with the fact that we want conformal maps to behave well under Möbius transformations, then this idea from Thurston becomes more promising as we consider a theorem from Koebe in 1936:

Theorem 4.4. *(Koebe)* If G is a finite maximal planar graph, then a circle packing of G is *unique* up to Möbius transformations and reflections where a Möbius transformation is a function $f : \mathbb{C} \to \mathbb{C}$ of the form:

$$f(z) = \frac{az + b}{cz + d}$$

where $a, b, c, d \in \mathbb{C}$.

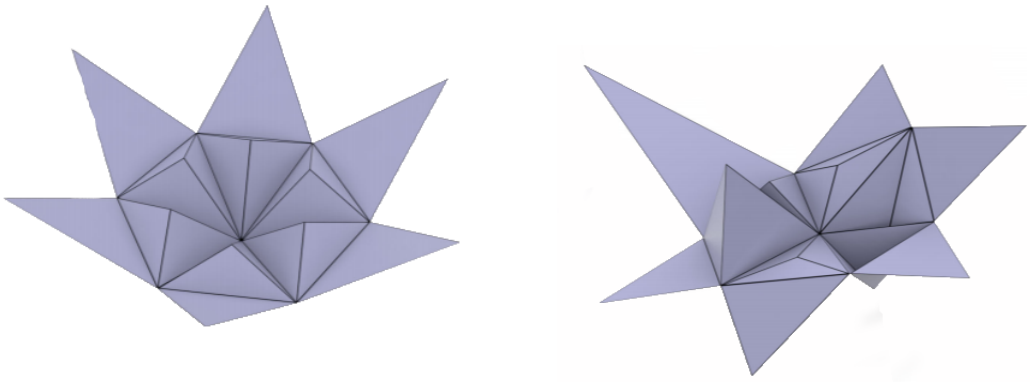
This means that if we take some Möbius transformation on the circle packing that we just provided for our planar graph



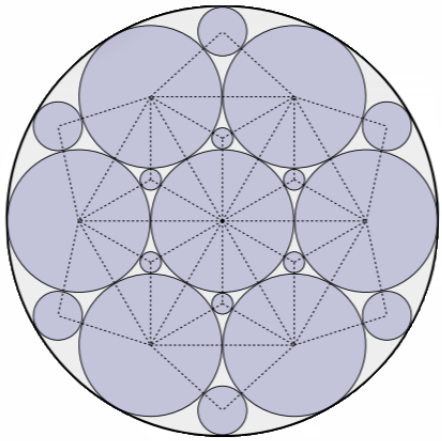
then the resulting packing will also work as a circle packing for the same planar graph. This may seem wrong at first glance, but if you take the time to look at the provided example, you will find that everything turns out fine. Okay, now this is getting good. We now have assurance that any circle packing that we use to describe a given graph will be *unique* up to some equivalence relation. Now we just need a way of formalizing the idea that a circle packing can behave like conformal map. Here, our friend Thurston comes to our aide once more with the following theorem:

Theorem 4.5. A circle packing of a regular hexagonal tiling of a region in the plane approximates a smooth conformal map by preserving the angles between centers of circles in a given packing.

Don't worry about what a hexagonal tiling is at this point. This sounds almost perfect, but it is at this point that we should take a step back and examine what it is that we are doing, exactly. We claim that by turning a piece of our simplicial complex into a circle packing we have constructed a conformal map. But we have yet to actually use anything about the *geometry* of the simplicial complex itself. All that we have used are the combinatorics. This means that that these two simplicial complexes



will both map to the packing that we saw earlier



[12]This can't be right! The metrics on our simplicial complexes have been almost completely ignored! This tells us that, in general, the notion of circle packing is too flexible to be used to determine conformal maps.

So, what now? Well, we use what we have learned to narrow down our choice of definition that we start with and we try again. We saw in our first attempt that it is too much that to ask that the angles

[12]Images from [CWW+18].

inside of the triangles be preserved completely, but we also saw from the second example that we can't just forget about the geometry of the mesh entirely. We need a definition that depends on the metric of our manifold, but that won't be determined by the way that we deform a single triangle. Consider the following definition:

Definition 4.6. Two Riemannian metrics g and $\widetilde{g}$ on a differentiable 2-manifold M are said to be *conformally equivalent* if

$$\widetilde{g} = e^{2u} g$$

for some smooth function $u : M \to \mathbb{R}$.

This definition turns out to be equivalent to the definition that we gave for the smooth case earlier, but it is convenient because it allows us to deal with logarithms of lengths instead of straight scalar multiples. In this sense, this definition is more forgiving on the ways that we distort the metric and it provides us with a simple discrete analogue.

Definition 4.7. Two discrete metrics ℓ and $\widetilde{\ell}$ on M are *(discretely) conformally equivalent* if, for some assignment of numbers u_i to the vertices v_i, the metrics are related by

$$\widetilde{\ell}_{ij} = e^{(u_i+u_j)/2} \ell_{ij}.$$

This definition is particularly nice because it behaves how we would want it to under transformations of space[13]. It's rather easy to notice, however, that with this definition neither the angles of triangles in our mesh nor the lengths of the edges of these triangles are well-preserved, and so we might ask what *is* preserved by these transformations. The answer is cross ratios.

Definition 4.8. Given a discrete metric ℓ and an interior edge e_{ij} between t_{lij} and t_{ijk}, we can associate with e_{ij} the *length cross ratio* which is defined by

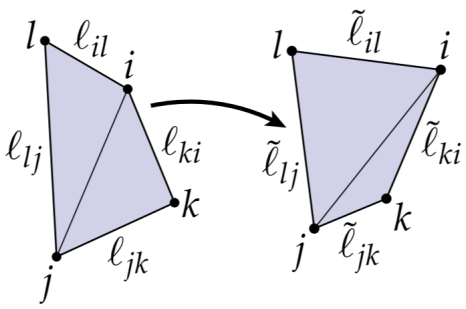
$$\mathfrak{c}_{ij} := \frac{\ell_{ik}}{\ell_{kj}} \cdot \frac{\ell_{jl}}{\ell_{li}}.$$

With this wonderful observation, we are now ready to state the final theorem of this section.

Theorem 4.9. Two discrete metrics ℓ and $\widetilde{\ell}$ are conformally equivalent if and only if they induce the same cross ratios. That is to say that for all i, j,

$$\mathfrak{c}_{ij} = \widetilde{\mathfrak{c}}_{ij}.$$

What exactly this theorem means is captured really well in the following picture[14]:



[13] Most notably, this definition behaves well under Möbius transformations.
[14] Image from [CWW+18]

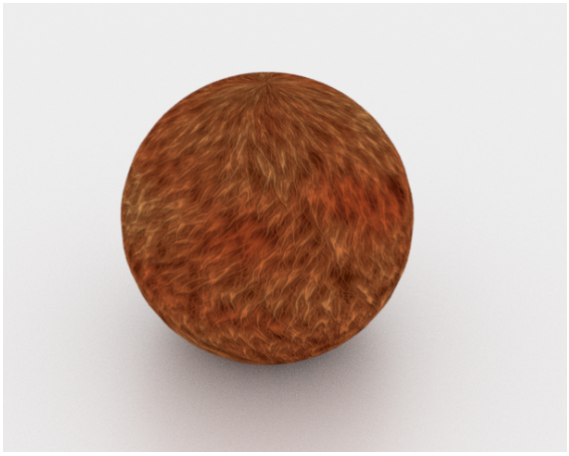
At last! We have reached a definition that allows us to take sections of our simplicial complex and conformally map it into a disk! This means that the useful theorem that we had for smooth manifolds that said we could conformally map surfaces to disks will still hold in the discrete case. But what are these things good for anyway? Do not worry, we wouldn't have made such a big fuss over these things if they were completely useless. Actually, the fact that we can even do this in general has been one of the more important discoveries in the past several years for the world of DDG and geometry processing. As we will see in the next section, these conformal maps and their inverses are needed to apply textures and shaders to different 3D models. We will also get to see how p-forms play a role in determining how the eye perceives 3D models after they have been rendered.

5. Applications of DDG to Surface Textures and Lighting

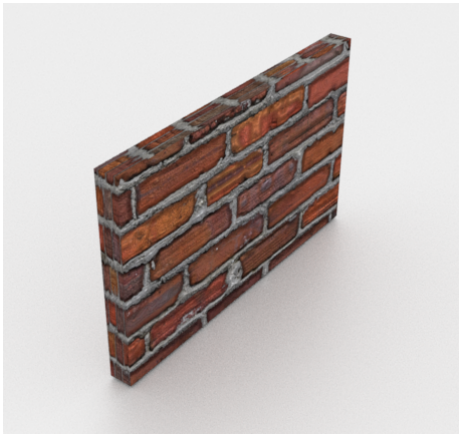
It's time for all of our hard work to pay off! At this point we can finally start talking about all of the fun things that we've been teasing until now, and we will finally see how, exactly, all of this math makes our favorite movies and games look fantastic. For this section, we are actually going to go in the reverse order of what we have been doing so far and talk about the ways that conformal maps are used in 3D textures before we move on to see how our fundamental forms can be used to help us bring light to these textures and objects.

- Surface Textures in Movies and Video Games -

I suppose it's best to start off with talking about what a surface texture is. I like to think of textures like wall papers or paints that can be put on the surface of objects in order to make them look a certain way. For example, we could make flat plane look like a hard-wood floor, a pyramid look like crumbling stone, or, my personal favorite,



a hairy ball[1]. We need to draw a bit of an important distinction here. While surface textures may *look* like they add depth to a surface by giving it creases and ridges, they actually don't do any of this. Remember: textures are just like paint. This may not be very convincing from looking at just one image, so let's look at one more example. consider the following brick wall:



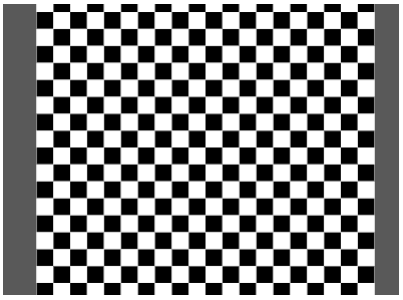
[1]There is actually a really fun theorem in algebraic topology called the *Hairy Ball theorem*. It's worth a look-up if you have some time.

This thing looks pretty normal as far as brick walls go, and you would be forgiven for thinking that every bit of this was was modeled (think sculpted out of clay, but on a computer) and colored to be made to look real. You would be wrong, though. In actuality what has happened is that a basic box shape has had the image of a brick wall painted onto it and then some clever shading has been applied to make the wall appear to have depth. This is much easier to see from a different angle.

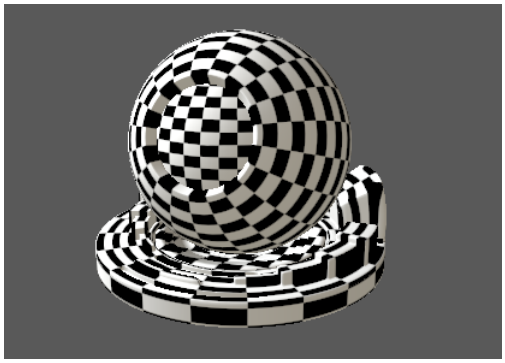
This is an image of the very same wall we saw before, but the angle has been adjusted to look along the wall. In this way we can see that the wall is, in fact, still completely flat.

The fact that we can manipulate images like this probably isn't too surprising. In fact, anyone that is even remotely familiar with art has probably seen some variation of this trick before. However, I bring it up here because this trick is essential for both 3D artists and render engine programmers to understand because it allows for a substantial amount of simplification when it comes time to generate 3D images.

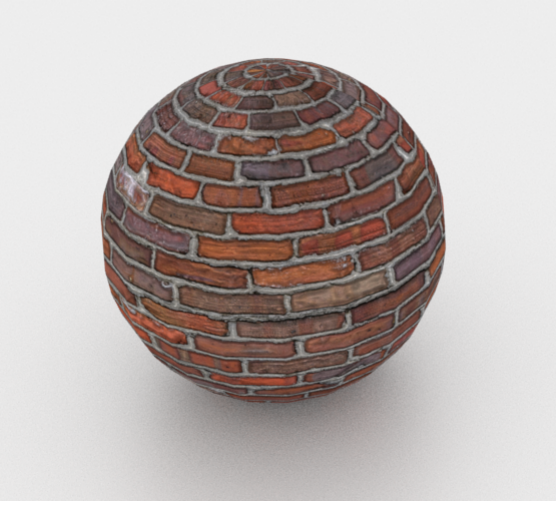
So how does this all relate to our conformal maps? Well, as we just discussed, all the information for a texture can be stored in a 2-dimensional image. And what do conformal maps do? They take 2D images and map them onto the surfaces of 3-dimensional objects! So, if we consider a 2D texture like the following checker pattern



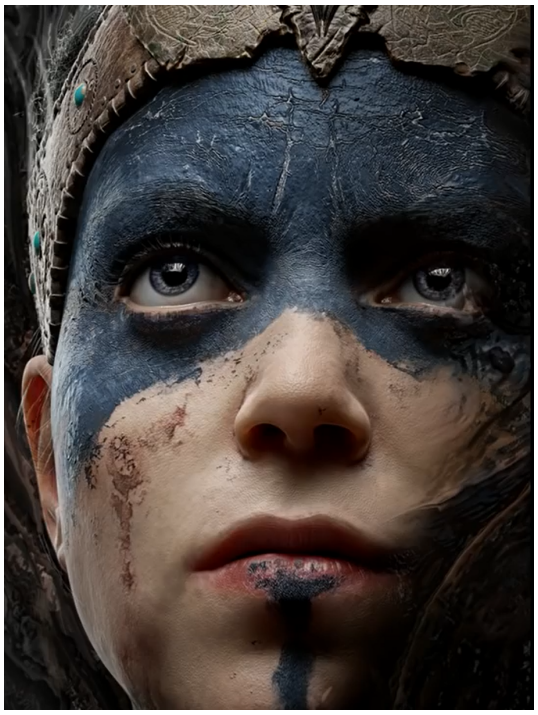
then this pattern is applied to a surface using a conformal parameterization which produces something like



Or, if we want to use our brick texture again, we can make



And that's pretty much it for how textures work. It seems kinda lack-luster to have taken over 30 pages to build up to this moment only to have the payoff last less than 3 pages. I would argue, however, that this is part of the beauty of the math we have worked through this far. Mathematicians are lazy by nature, so whenever we manage to find some theorem or some formula that minimizes the work that we do later, we call that theorem or formula beautiful. And that is exactly what has happened here, but if you still have a bit of a mathematical itch to scratch, then the next two sections should prove satisfying. Other than that, here are a couple of images that highlight the use of surface textures to replicate skin, metals, and organic materials[2]:

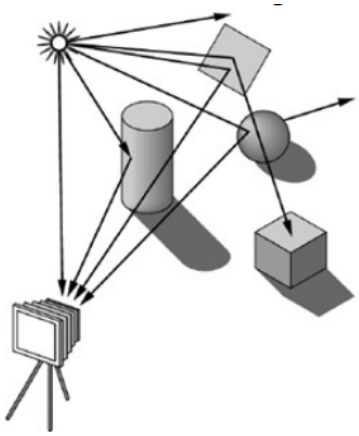


[2]Images from *Hellblade: Senua's Sacrifice*, MAWI United, and *Horizon: Zero Dawn*

- Lights, Camera, p-forms! -

We now know how 3D renderers place materials on meshes to make cool things like brick walls, but we still have no idea how we are even able to see these wonders in the first place. But, before we get into how the computer is able to illuminate environments built in 3-dimensional space, we should first go over how the computer knows what it sees in a given environment.

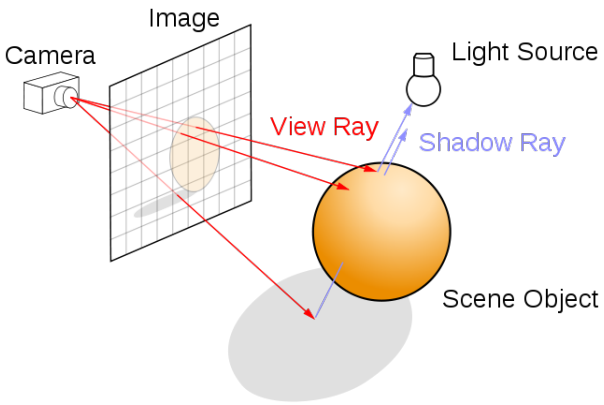
When we study the way that light works in the real world, we tend to think of it in the form of rays. Specifically, we think of it as starting at some source and then emanating out and bouncing off objects. When we look at a given object, some of the rays are then projected onto an image plane in the back of our eye (the retina), which our brain interprets to tell us what we see.[3].



This model does not work well, however, for when we want to calculate what a camera sees in a 3D model. Why? Because there are a lot of wasted rays that don't hit anything that the camera can see. That means that a lot of processing power and RAM would be wasted on things that would ultimately be considered useless. There is an easy fix for this, though; instead of considering the light rays as starting

[3]Image from http://slideplayer.com/slide/7537260/

at the source and making their way to our eye/camera, we can think of the light rays as starting at the camera, passing through an image plane, and making their way back to the given source.



[4]By thinking of light in this way, we ensure that every ray contains valuable information about what the camera can actually see. We are then able to control the definition of the image rather easily by modifying the number of rays that the camera emits.

With a basic understanding of how ray tracing works under our belt, we can now start in on how these things weave together to illuminate a scene[5]. We'll start with how the computer actually deals with the light rays emitted by the camera. When a computer goes to trace a light ray starting at the camera, the ray is parameterized as a line. The line is then shot through the scene and the computer records the first time instance when the light intersects a model in space. This first point of intersection will actually be the only point along the given line which the computer will see, and, therefore, will be the only point along the line that is taken into account when the scene is rendered.

Once the primary points of interaction have been handled, the graphics renderer will then proceed to track the way that the light ray reflected off of the given point interacts with the remainder of the scene and store the resulting information until the intensity of the light reaches below some threshold (usually light is given a quadratic intensity dropoff). The sum of all this data together with information provided by shaders, materials, and bump/displacement maps[6] is put together in the following equation called the *rendering equation*, which is used to determine radiance L at a given point $\widehat{x}$ by taking a surface integral over a hemisphere $\Omega_{\widehat{x}}$ centered at that point

$$L(\widehat{x},\omega) = L_e(\widehat{x},\omega) + \int_{\Omega_{\widehat{x}}} L(h(\widehat{x},\omega'),-\omega')f_r(-\omega',\widehat{x},\omega)\cos(\theta')d\omega'.$$

This equation looks ugly, but I promise that it's actually super pretty as far as spectrograph equations go, and there is an accompanying picture on the next page. In this equation, the point $\widehat{y} = h(\widehat{x},\omega')$ is the first point hit in the scene when shooting a ray from $\widehat{x}$ in the direction ω', and is mainly what determines a point's visibility in a scene; $L_e(\widehat{x},\omega)$ is the emitted spectral radiance (how bright the point is); and f_r is the bidirectional reflectance distribution function (a.k.a. the BRDF which determines how

[4]Image from https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

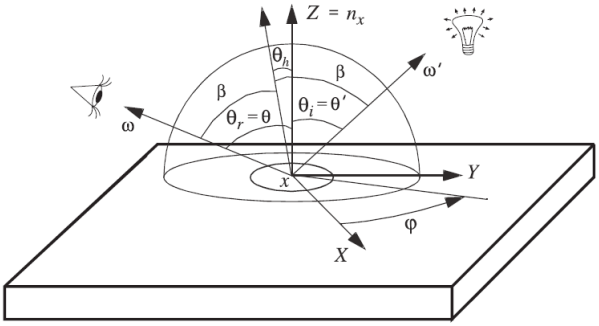
[5]For those of you that are somewhat familiar with the basics of 3D animation already, I will not be discussing what vertex, pixel, geometry, and tessellation shaders are and the role that they play in determining how things are rendered. For those that don't know what these are, feel free to look them up, but I caution that making sense of what's really going on will require some basic background in computer science and will likely involve looking into what the render pipeline is.

[6]Note: Bump maps and displacement maps are actually two different things, but they serve essentially the same purpose.

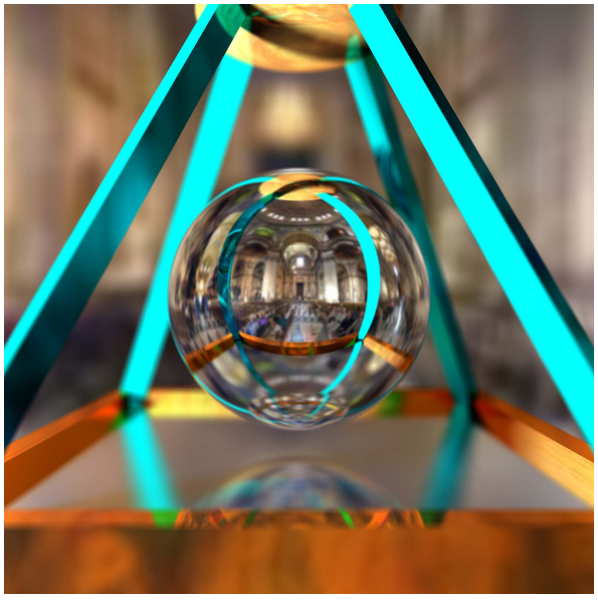
lights bounce off of opaque surfaces). This last equation is, in turn, given by

$$f_r(-\omega', \widehat{x}, \omega) = \frac{1}{\pi}\left(F(\lambda, \theta' = 0) + \frac{F(\lambda, \theta')D(\theta_h)G(\theta_h, \beta, \theta, \theta')}{\cos(\theta)\cos(\theta')}\right)$$

where $F(\lambda, \theta')$ is the Fresnel function (which describes the reflection and refraction of light), $D(\theta_h)$ is the microfacets distribution function (which describes the statistical distribution of surface normals), and $G(\theta_h, \beta, \theta, \theta')$ is the geometrical attenuation factor (which describes how a light dims as it moves through a medium). The rest of the constants in the rendering equation should be discernible from the image[7].



Now, I have mentioned two things so far that I have yet to show. Firstly, I claimed that there were differential forms in here somewhere, and secondly, I claimed that these equations were pretty. Hopefully, after all that we've been through in this paper, it's fairly easy to see that we have just taken an integral which involves a differential form. Actually, the rendering equation is an integral defined over a manifold with boundary, so you can bet that when we are actually computing this thing we use Grandpa Stokes' theorem. That takes care of the p-forms, but why is this fugly[8] rendering equation actually pretty? Well, it's because all the subequations that the exact form of the rendering equation rely on are actually really well-behaved in the sense that it's easy to estimate them. So even if we want to render something that has a lot of reflections in it like[9]



our computers won't crash during the rendering process. Does it still take a lot of time? Yes, but at the very least it's doable.

[7] Image from [DGP05]
[8] Not a typo.
[9] Image from https://courses.cs.washington.edu/courses/cse557/08wi/projects/trace/

6. Conclusion

I suppose that was the last bit of what I wanted to talk about. We've gone through a lot in this paper. From vectors, to derivatives, to differential forms, to conformal maps, we have seen them all and, hopefully, developed some good intuition for them. More importantly, I hope that this paper has provided a satisfactory reason for appreciating math outside of its standard applications within STEM disciplines, and, maybe, convinced a few people that had thought they were done with math to take another look at it. I'm not going to get my hopes up, but it's a nice thought to have. At the very least, I know that if you made it this far that you found this paper mildly interesting, and so I say to you thank you for reading.

References

[Axl97] Sheldon Axler. *Linear algebra done right*, volume 2. Springer, 1997.

[Bai] Joshua Bainbridge. Physically based shading.

[BPS+15] Alexander I Bobenko, Ulrich Pinkall, Boris A Springborn, et al. Discrete conformal maps and ideal hyperbolic polyhedra. *Geometry & Topology*, 19(4):2155–2215, 2015.

[Cle17] Jeanne N Clelland. *From Frenet to Cartan: the method of moving frames*, volume 178. American Mathematical Soc., 2017.

[Cra17] Keenan Crane. Discrete differential geometry: An applied introduction, 2017.

[CW17] Keenan Crane and Max Wardetzky. A glimpse into discrete differential geometry. *Notices of the AMS*, 64(10), 2017.

[CWW+18] Keenan Crane, Max Wardetzky, Johannes Wallner, Yaron Limpan, and Justin Solomon. Ams short course on discrete differential geometry. San Diego, California, Jan. 8-9 2018.

[Daw18] Paul Dawkins. Paul's online math notes, 2018.

[DC76] Manfredo P Do Carmo. *Differential Geometry of Curves and Surfaces*. Courier Dover Publications, 1976.

[dGDT15] Fernando do Goes, Mathieu Desbrun, and Yiying Tong. Vector field processing on triangle meshes. In *SIGGRAPH Asia 2015 Courses*, page 17. ACM, 2015.

[DGP05] Ivan Dimov, Todor Gurov, and Anton Penzov. A monte carlo approach for the cook-torrance model. In *Lecture Notes in Computer Science*, volume 3401, pages 257–265, 02 2005.

[DMA02] Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic parameterizations of surface meshes. In *Computer graphics forum*, volume 21, pages 209–218. Wiley Online Library, 2002.

[DP91] Kees Dullemond and Kasper Peeters. Introduction to tensor calculus. *Kees Dullemond and Kasper Peeters*, 1991.

[GGL+14] Xianfeng Gu, Ren Guo, Feng Luo, Jian Sun, and Tianqi Wu. A discrete uniformization theorem for polyhedral surfaces ii. *arXiv preprint arXiv:1401.4594*, 2014.

[HK] Kenneth Hoffman and Ray Kunze. Linear algebra. 1971. *Englewood Cliffs, New Jersey*.

[ICG86] David S Immel, Michael F Cohen, and Donald P Greenberg. A radiosity method for non-diffuse environments. In *ACM SIGGRAPH Computer Graphics*, volume 20, pages 133–142. ACM, 1986.

[Ize13] Thiago Ize. Robust bvh ray traversal. *Journal of Computer Graphics Techniques (JCGT)*, 2(2):12–27, 2013.

[Kaj86] James T Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.

[Kes18] John Kesig. Personal Communication, Mar. 11 2018.

[Kob00] Leif Kobbelt. $\sqrt{3}$-subdivision. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 103–112. ACM Press/Addison-Wesley Publishing Co., 2000.

[Lee03] John M Lee. *Introduction to Smooth manifolds*. Springer, 2003.

[MDSB03] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, pages 35–57. Springer, 2003.

[Opr07] John Oprea. *Differential Geometry and its Applications*. MAA, 2007.

[SSD08] Adam Smith, James Skorupski, and James Davis. Transient rendering. 2008.

[SSP08] Boris Springborn, Peter Schröder, and Ulrich Pinkall. Conformal equivalence of triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 27, page 77. ACM, 2008.

[Ste12] James Stewart. *Essential calculus: Early transcendentals*. Cengage Learning, 2012.

[ZG13] Wei Zeng and Xianfeng David Gu. *Ricci flow for shape analysis and surface registration: theories, algorithms and applications*. Springer Science & Business Media, 2013.