Introduction

People playing HyperRogue, or watching our visualizations, naturally want to understand non-Euclidean geometry. While our visualizations are very good for gaining intuitions about this subject, some readers crave more knowledge – what *is* non-Euclidean geometry? Why is it important? How are such games and visualizations created? What are they useful for? Are they even useful for anything? What do all the fancy terms we are using mean? This book is for them.

Non-Euclidean geometry is believed to be a very hard subject. On one hand, it is true, because it changes the very basic intuitions about how space works – various misconceptions about non-Euclidean geometry arise from applying such basic intuitions. On the other hand, our goal is to make this book accessible to high-school students, although they will need to make more effort than someone already familiar with linear algebra, differential geometry, model theory, 3D graphics, automata theory and special relativity. Note that most of them are not things that every mathematician knows – automata theory is usually considered theoretical computer science, special relativity is considered physics, and model theory could be said to be philosophical in nature. In most cases, only the basics are necessary, and we explain these basics. Some chapters are more advanced.

This book has a big focus on computations, that is, a reader with some background in programming should be able to implement their own implementation of non-Euclidean geometry; and applications, counting not only applications in industry, but also creating video games and other kinds of mathematical art. However, the knowledge will be also useful to people willing to use an existing engine (such as RogueViz, the engine of HyperRogue, which is largely based on the ideas presented in this book), and to people who just want to understand how things work. We explain both the classic techniques and our new ideas.

This book is a work in progress. And it is not peer reviewed yet. Contributions are welcome.

Contents

Chapter 1

What *is* non-Euclidean geometry?

1.1 Absolute geometry

Geometry could be described as the study of shapes, angles and distances. Probably the most popular result in geometry is the Pythagorean theorem:

Theorem 1.1.1 (Pythagorean Theorem)**.** *If ABC is a right triangle, c is the length of the hypotenuse, and a and b are the length of the other two sides, then $a^2 + b^2 = c^2$.*

How do we know that the Pythagorean Theorem is true? It is definitely not self-evident. One method is drawing right triangles and noticing that it always appears to be true. That makes sense, but is not great: we would like to know how well could we trust this theorem to always hold. We could just assume it is true, but that leads to memorization, and it is better when our knowledge relies on understanding rather than memorization. The things we assume should be as basic and self-evident as possible.

So this is the approach used in mathematics today: start with as simple assumptions as possible, study the consequences of these assumptions, and hopefully, apply these consequences in our everyday reasoning. One of the works pioneering this approach is Euclid's *Elements* (400 BC?). Euclid starts with basic assumptions (called notions, axioms, and postulates), and attempts to deduce all known facts of geometry from these assumptions. We will not list all Euclid's assumptions, especially that Euclid's work is not formal by modern standards – some things were so self-evident to him that he did not notice he was assuming them and he should list them as assumptions. However, we will list the first fourth postulates:

1. A straight line segment can be drawn joining any two points.
2. Any straight line segment can be extended indefinitely in a straight line.

3. Given any straight line segment, a circle can be drawn having the segment as radius and one endpoint as center.

4. All right angles are congruent.

Euclid was able to deduce a lot from these four postulates and his other assumptions. In the modern language, the following facts are consequences:

- The space is **complete**, that is, straight lines never end. (This is the second postulate.)
- The space is **homogeneous**, that is, if we can drawn a shape at some point X, we can also draw it at some other point Y, and these two shapes would be the same in terms of their angles and distances.
- The space is **isotropic** – which means the shapes can be also rotated by some angle, and again they are the same in terms of their angles and distances.

1.2 Properties of Euclidean geometry

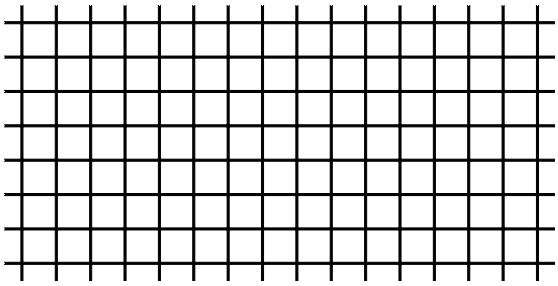

Figure 1.1: The square grid in Euclidean geometry. Are the angles really 90°? Can we extend this forever?

However, not all known results of geometry could be thus obtained. For example, Euclid was unable to prove the following:

1 Pythagorean theorem, listed above.

2 The space is **scale invariant** – which means the shapes can be scaled by any factor f, which keeps their angles, but the distances are multiplied by f. One consequence of scale invariance is that the ratio of circumference of a circle to its diameter is a constant, usually called π.

3 In a plane, given a line and a point not on it, exactly one line parallel to the given line can be drawn through the point. (Here, a line L_2 *parallel* to L_1 is one that does not intersect L_1.)

4 Angles of a triangle sum to 180 degrees.

5 If a line segment intersects two straight lines forming two interior angles on the same side that sum to less than two right angles, then the two lines, if extended indefinitely, meet on that side on which the angles sum to less than two right angles.

6 Rectangles exist. That is, shapes with four straight edges and four right angles.

7 Square grids exist, and can be extended indefinitely (Figure 1.1). The Cartesian coordinates, especially useful e.g. in computer graphics, rely on this assumption.

8 Imagine an observer walking clockwise on the perimeter of some shape, always facing the forward direction. When the observer returns to their starting point and orientation, they must have rotated exactly 360 degrees (assuming that the shape has no self-intersections, and in case of concave shapes, counting counterclockwise rotations as negative). Point [4] is a simple consequence of this.

Interestingly, if Euclid assumed one of the seven things listed above, he could prove the other six. So this is what Euclid did. He wanted his assumption to be as simple as possible, and he picked Euclid's Parallel Postulate [5] as his Fifth Postulate. In more modern expositions, usually Playfair's Axiom [3] is considered simpler (possibly with *exactly one* replaced with *at most one*).

Euclid was unhappy about this, and he believed that including such an overcomplicated postulate was a flaw of his work. Today, we know that it was not his fault: we know that non-Euclidean geometries exist, which satisfy all Euclid's assumptions except Euclid's parallel postulate.

1.3 How space could be weird?

The mainstream meaning of *non-Euclidean* is *any kind of weird space* (W), or *any space that violates some Euclid's assumptions* (A). However, the meaning used in mathematics is more precise than that. To explain why, let's think how a space could be weird, or how could it violate Euclid's assumptions.

Probably the easiest way to violate Euclid's postulates is to violate *completeness*. Most video games take place in a bounded world, and thus, if we used the meaning (A), we could call any such game non-Euclidean. We could also violate *isotropy*, for example by measuring distance between $(0, 0)$ and (x, y) by formula $|x| + |y|$ (Manhattan distance) rather than $\sqrt{x^2 + y^2}$ (which follows from the Pythagorean theorem). Most people, when asked to describe weird space, would probably describe some kind of a looping space (as seen in the video game *Asteroids*), or some kind of building that is "greater on the inside" or otherwise weirdly connected (such spaces are common in works on art, for example, the wardrobe in *the Chronicles of Narnia*, the book *House of Leaves*,

movie *Doctor Who*, or video game *Antichamber*), or impossible structures such as Penrose triangles or staircases. Generally these spaces violate many of Euclid's postulates, but each case violates them in its own way. So, from a practical point of view, such a broad definition of *non-Euclidean* makes no practical sense – we generally want the space to be weird in a specific way. The original meaning of *non-Euclidean geometry* in mathematics, as coined by Gauss, was: a geometry violating Euclid's parallel postulate while satisfying all his other assumptions. Some mathematicians extend this definition, but generally, the term *non-Euclidean geometry* is reserved for spaces that violate the spirit of Euclid's parallel postulate.

1.4 Geometry versus topology

It is time to explain the difference between *geometry* and *topology*. Topology is popularly described as a branch of mathematics studying the properties of shapes which do not change when these shapes are stretched or compressed, but do change when surgery is performed (the shape is cut or glued). Geometry is the opposite: geometry usually changes when shapes are stretched, but do not change when surgery is performed. In other word, geometry is a *local* property, while topology is a *global* property.

Many of the constructions given above (bounded space, looped spaces, portal spaces) are topological constructions, based on surgery. Suppose we live in an infinite universe. Could we ever tell this to be true? No, because we can only study the local geometry; it is always possible that the universe ends or loops somewhere beyond the range of our instruments. So as long as we don't know, it is better to assume that the local geometry simply extends indefinitely. Even if we did know that the world is bounded, such as in a video game, we could still assume the world is infinite – it would be an approximation, but with some care, it would still give correct predictions.

1.5 Models

It is not surprising that lay people would not give actual non-Euclidean geometry (in the meaning coined by Gauss) as an example of weird space – it is far from obvious that constructing such a geometry is possible; as said above, Euclid thought that Euclid's parallel axiom followed from his other assumptions, and so did the other mathematicians researching this problem for 2000 years.

At this point, the readers are welcome to play *HyperRogue*, or possibly watch our video visualizations of hyperbolic space, and see that it is indeed non-Euclidean in the narrow sense of Gauss: all Euclid's assumptions are true, except the Parallel Postulate. Playing HyperRogue is a very good way to get intuitions about how hyperbolic geometry works, but the purpose of this work is to explain how all of this works mathematically. For this, we will need *models*.

The notion of a *model* needs some effort to understand, but it is very important in mathematics, not only in non-Euclidean geometry, but also, for example, in Gödel theorem, and to explain various logical paradoxes; yet, simplified explanations of these concepts tend to gloss over it, which is sometimes misleading. Intuitively speaking, a **model** of X is a structure M such that various elements and relations of X can be *interpreted* in terms of structure M, in such a way that the required properties of X hold in this interpretation. (The interpretation need not be *literal*, for example, points of M need not correspond exactly to the points of X.) What are models of X useful for?

- A model shows that the properties of X are consistent. Suppose we use Euclidean geometry to construct a model of non-Euclidean geometry (in Gauss sense). This shows that non-Euclidean geometry is consistent, and therefore, *Euclid's Parallel Axiom does not follow from other Euclid's assumptions* (as long as Euclidean geometry is consistent, because this is what we constructed our model in).
- Models are useful to explain X, as we will see later.
- Models are used in computer simulations. You use a model of X as a representation of X.
- Euclidean models of non-Euclidean geometry also happen to have artistic uses, and be useful for visualization. As such, they are sometimes confused with *projections*. The difference between *model* and *projection* is significant, but mostly intentional: good models should let us do the math easily, while good projections should let us easily see the important aspects.

1.6 Spherical Geometry

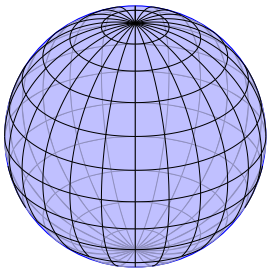

Figure 1.2: We cannot draw the square grid on a sphere.

Since we are interested in computer graphics, we will study the Euclidean property [7] from Section 1.2: *square grids exist, and can be extended indefinitely* (Figure 1.1). We want to construct a Euclidean *model* of non-Euclidean

geometry, that is, a construct that appears to be very similar to what we know from Euclidean geometry, but not having property [7]. In particular, the failure of [7] must be a consequence of the *local* geometry, not *global* topology.

Let us try drawing these squares on a sphere (sitting in the Euclidean space), for example as a grid of parallels of meridians (Figure 1.2). Is this a square grid? No: shapes closer to the poles are narrower than shapes closer to the equator. The angles are all right, however, the parallel are not straight lines – a circle of radius 10 meter around the North Pole is technically a parallel, and it is clear that a person going in such a circle would not consider this straight.

Let us interpret *straight line* as a fragment of a great circle, that is a shape on the sphere that, in the Euclidean three-dimensional space, has the same radius as the sphere itself. (This does make sense, as objects freely moving on the surface of the sphere, without forces other than gravity acting upon them, would indeed move along such lines. Note that, since interpretations need not be literal, they need not make sense – but this one does.) The distances between points on the sphere are measured by measuring the length of the (shortest) great circle arc connecting these two points. Did we obtain a model of non-Euclidean geometry? Let us check some of the properties mentioned in Sections 1.1 and 1.2:

- A sphere is complete, homogeneous, and isotropic.
- Playfair's axiom in formulation [3] is not satisfied, because parallel lines do not exist in our interpretation – great circles always cross.
- Angles of a triangle do not sum to 180°. The easiest way to observe this is to put one vertex at the North Pole, and two at the equator. The sum of angles of such a triangle will be always greater than 180°.

Does our model satisfy Euclid's postulates? This is a bit difficult to tell because of the not very formal nature of Euclid's postulates. Can we extend straight lines indefinitely, or the whole great circle cannot be extended anymore? We can connect the North Pole and the South Pole with a straight line, but is this line *unique*, as Euclid implicitly assumed? In the modern understanding, spherical geometry generally is considered a non-Euclidean geometry, although it would not considered one in the times of Gauss, when Euclid's assumptions were interpreted more strictly. So, spherical geometry does not yet solve Euclid's problem, but it is a good step on the way.

Remark 1.6.1. In some sources, the name *elliptic geometry* is used. This name is a bit confusing – it does not refer to ellipsoids, it still assumes perfect spheres. Two notions are based on the same Greek word. In some sources, *elliptic geometry* and *spherical geometry* are a bit different – the antipodal points are interpreted as the same point in elliptic geometry. This is done to satisfy Euclid's expectations more faithfully, in particular, there is always exactly one straight line going through any two different points. However, this difference is topological, not geometric.

1.7 Hyperbolic geometry

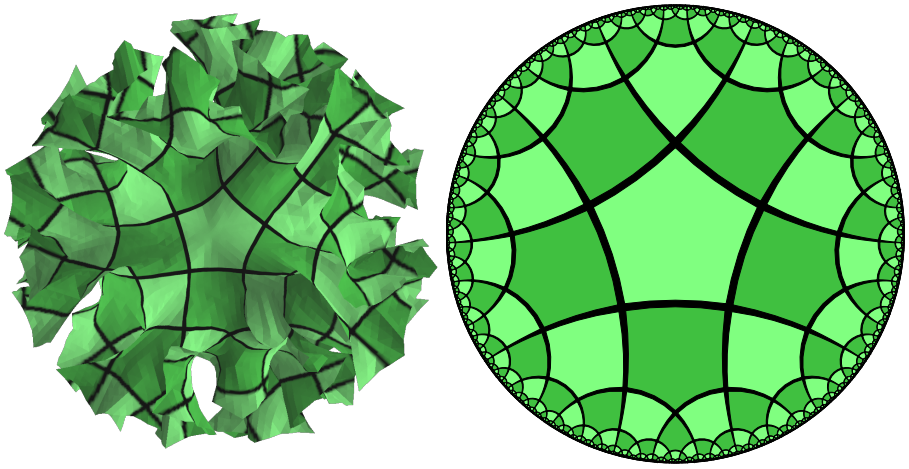

Figure 1.3: (a) A computer simulation of a model of a fragment of hyperbolic plane, created using the Hypersian Rug algorithm. (b) The same model viewed in Poincaré disk model.

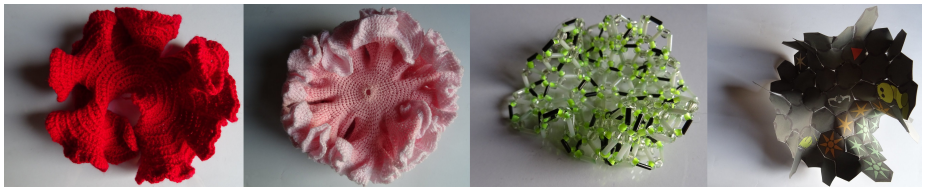

Figure 1.4: Real-world approximations of fragments of the hyperbolic plane.

The reason why spherical geometry is a good step on the way to solve Euclid's problem is that the perfect solution to this problem, *hyperbolic geometry*, is in fact, in some sense, the opposite of *spherical geometry*. We could again model hyperbolic geometry using curved surfaces, but not as nicely as in the case of spherical geometry.

See Figure 1.3a for a computer approximation of such of a surface. Approximations of such surfaces can be obtained using crocheting, beads, paper, computer algorithms (the *Hypersian Rug* in HyperRogue), see Figure1.4, and also sometimes appear in nature (for example, lettuce leaves). The methods of making such approximations are covered in Chapter 7.3. There are two problems though: they cannot be extended indefinitely (they only model an incomplete space), and they are quite messy, having no nice mathematical description in terms of Euclidean geometry. Therefore, they are not very good as models.

To solve these problems, we will use a *projection*. Projections of spheri-

cal geometry are useful in mapmaking: a sphere is projected to the Euclidean plane. Because of the difference between spherical and Euclidean geometry, such a projection will never map distances, angles and areas faithfully; mapmakers use dozens of projections, which differ in the aspect they are faithful in. However, as long as the formula of our projection is known, the actual values can be computed from the map. (As such, projections can be considered models themselves.)

The most popular projection/model used for explaining hyperbolic geometry is called the *Poincaré disk model*. Figure 1.3b shows the hyperbolic plane with some lines in the Poincaré disk model; Figure 1.3b is a projection of 1.3a. The model is restricted to a disk. Circular arcs orthogonal to the disk are interpreted as straight lines. The Poincaré model is *conformal*, which means that angles are mapped faithfully (an angle α between curves is interpreted as angle α). The distances are not mapped faithfully, though: the closer we are to the boundary, the distances are interpreted to be large. All the edges shown in Figure 1.3b are interpreted as lines of the same length.

We will not provide the full formulas for lengths in the Poincaré model. The point of this section is to provide a simple explanation and visualization of the idea of hyperbolic geometry. For easy modelling (especially in the computer), another model, called *the Minkowski hyperboloid model*, tends to be more suitable. This model will be explained in Chapter 2. Instead, we refer to HyperRogue (which uses the Poincar'e model by default) for intuitions, and other books for formal math. This model shows that non-Euclidean geometry is consistent (as long as Euclidean geometry is consistent), thus solving the problem of Euclid we have started this chapter with.

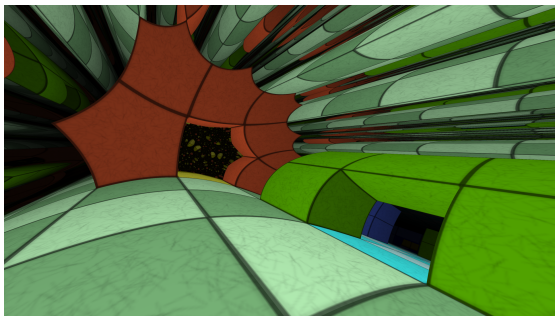

Figure 1.5: A screenshot of our *Portals to Non-Euclidean Geometries* video.

Remark 1.7.1. Some oversimplified sources gloss over the notion of a *model*, by identifying non-Euclidean geometry with its model. For people who understand the concept of a model well, and who can easily convert between X and its model, such a mental shortcut is understandable. However, it is important to note that the properties of the model of X are not *the same* as the properties of X – the interpretation need not be *literal*. There are sources which say things like *non-Euclidean geometry simply means geometry on a curved surface*,

or sources which assume that the Poincaré model is an accurate representation of hyperbolic geometry. This is not true – in both cases, the Euclidean curved lines are *interpreted* as non-Euclidean straight lines; if we actually lived inside a non-Euclidean world, we would consider these lines to be indeed straight, and properties such as *the angles of a triangle do not sum to* 180° to indeed hold. Most three-dimensional visualizations of non-Euclidean geometry accurately represent what we would perceive if we lived in such a world (based on some basic assumptions on physics, like light travelling in straight lines). This means that the perspective and parallax work very differently; it is difficult to convey this in a book, so we recommend our video visualizations (e.g. *Portals to Non-Euclidean geometries*, Figure 1.5), or even better, the interactive visualizations in HyperRogue and RogueViz. Unfortunately, people read *non-Euclidean geometry simply means geometry on a curved surface* and assume that our visualizations, or the literature like *The Call of Cthulhu* by H. P. Lovecraft, are wrong – in fact, their sources did not explain the concept of *models* correctly.

1.8 Comparison

In the previous section, we have mentioned that spherical geometry is the opposite of hyperbolic geometry. However, while we have providen some analogies, this is not apparent from the models shown. Thus, we will finish the chapter with a comparison of spherical, Euclidean, and hyperbolic geometry. We compare the eight properties given in Section 1.2.

1 Spherical Pythagorean theorem is $\cos a \cdot \cos b = \cos c$. The hyperbolic Pythagorean theorem is $\cosh a \cdot \cosh b = \cosh c$. (The functions cosh and sinh will be explained later. The lengths are given in absolute units, which in the case of spherical geometry means that the radius of the sphere is 1.

2 Only Euclidean geometry is scale invariant. Other geometries become closer and closer to Euclidean geometry as the scale gets smaller, so people living in non-Euclidean worlds would still understand Euclidean geometry, just like we understand the geometry of Euclidean plane despite living on a sphere (we are small enough compared to the sphere that we usually do not even notice this). The circumference of a circle of radius r is $2\pi r$ in Euclidean geometry, $2\pi \sin r$ in spherical geometry, and $2\pi \sinh r$ in hyperbolic geometry. There is a common rule for obtain hyperbolic analogs of spherical formulas: change trigonometric functions to their hyperbolic equivalents (if the argument is a distance, not an angle), and possibly change the signs in some places. In Euclidean case, use $\text{sine}(r) = r$ and $\text{cose}(r) = 1$ (which makes the Pythagorean theorem true, but trivial). This will be explored in more detail in the following chapters.

3 In a Euclidean plane, given a line and a point not on it, exactly one line parallel to the given line can be drawn through the point. In hyperbolic plane, there are more; in spherical plane, there are none.

4 Angles of an Euclidean triangle sum to 180 degrees. For hyperbolic triangles, it is always less than 180 degrees. For spherical triangles, this is always more. The bigger the triangle, the bigger the difference.

5 Euclid's Parallel Axiom is not satisfied in spherical or hyperbolic geometry. Intuitively, the lines which should be parallel *converge* in spherical geometry, but *diverge* in hyperbolic geometry.

6 Non-Euclidean rectangles do not exist. This is easy to show from [4] and the fact that quadrilaterals could be split into two triangles.

7 Square grids do not exist, but we can construct grids, for example, of right-angled triangles (in case of spherical geometry) or right-angled pentagons (in case of hyperbolic geometry, like in Figure 1.3). The Cartesian coordinates do not work directly, but we could use the Cartesian coordinates in a model. This will be explained in the following chapters.

8 Imagine an observer walking clockwise on the perimeter of some shape, always facing the forward direction. When the observer returns to their starting point and orientation, they generally will not have rotated exactly 360 degrees. It turns out that their rotation equals 360 degrees minus K times the area *inside* the traversed shape, where K is a parameter called *curvature*, and measures how different the geometry is from Eucldiean. We have $K = 0$ for Euclidean geometry, $K = 1$ for spherical geometry, and $K = -1$ for hyperbolic geometry (assuming absolute units are used).

It is also worth to mention that so far we have been assuming isotropy. In three and more dimensions, there are also *anisotropic geometries*, which are homogeneous, but act differently depending on the direction. The easiest construction of such a geometry is obtained by taking two-dimensional spherical or hyperbolic geometry, and adding an extra dimension in the Euclidean way. The property of being closer and closer to Euclidean geometry as the scale gets smaller is still satisfied (contrary to the Manhattan metric, which is still *quasi-isometric* to Euclidean metric). Such geometries also violate the spirit of Euclid's parallel axiom [5] – this time, the lines neither diverge or converge, but possibly twist in the third dimension. Such geometries will be described in the later chapters (Nil 7.8, Solv 7.9).

Chapter 2

Modelling geometry

This chapter precisely presents the models of Euclidean, spherical and hyperbolic geometry we will be using. While the computations in hyperbolic geometry look mysterious, especially when presented using the Möbius transformations in the Poincaré disk model (as done in many courses), it is in fact very analogous to the obvious model of spherical geometry – the difference is that instead of using a sphere in Euclidean space, we need to use a *pseudosphere* in Minkowski spacetime; and common things are done using linear algebra, in a way similar to the usual method of doing 3D graphics in computers.

The methods used here are not the only possible models. In our experience, many people new to computational hyperbolic geometry use the Poincaré disk model, and then fail to compute (for example, how to compute the midpoint between a and b?). Such computations are more straightforward in our homogeneous models. We will need to understand the basics of Minkowski spacetime, but it will be worth it, especially given that our real-world spacetime (in small scale) is a Minkowski spacetime. Contrary to the Poincaré disk model, the projective version of our model allows us to easily represent ultra-ideal points, which will be relevant for three-dimensional honeycombs (see Chapter 7.7). However, Poincaré disk model has its own advantages: it may be more numerically precise, although in the situations where this matters, it is better to use tessellations anyway (see Chapter 3). Also, other methods may be used to represent rotations and other isometries (quaternions, split-quaternions, etc.). This will be discussed in Chapter 7.5.

2.1 Linear algebra

We denote the set of real numbers by $\mathbb{R}$. The set of tuples of n real numbers is called $\mathbb{R}^n$. We will use such tuples to represents *points* (according to some coordinate system) and *vectors*. because we will use these tuples to denote points, according to some coordinate system. We will denote the individual components of a point $v \in \mathbb{R}^n$ by $v_1,\ v_2, \ldots, v_n$. We will write points as $v =$

$(v_1, v_2, \ldots, v_n)$ or using the vector notation, where the components are written as a column.

The elements of v can be added and subtracted: $(v+w)_i = v_i+w_i$, $(v-w)_i = v_i-w_i$ for every index i, and multiplied and divided by scalars (that is, elements of $\mathbb{R}$): $(k \cdot v)_i = kv_i$, $(v/k)_i = v_i/k$ for every index i. We will generally avoid writing the components directly, and instead treat points as *first class* objects that can be added and multiplied. We denote the point whose every coordinate is 0 also by 0, thus, $0 \cdot v = 0$ for any $v \in \mathbb{R}^d$. By $v[x/i]$ we denote the vector w such that $w_i = x$ and $w_j = v_j$ for $i \neq j$ (this notation is less standard in linear algebra). In particular, $0[1/i]$ is the vector which has 1 on the i-th coordinate and 0 on all the other coordinates. It is easy to verify that, for every $v \in \mathbb{R}^d$, we have $v = v_1 \cdot 0[1/1] + v_2 \cdot 0[1/2] + \ldots + v_n \cdot 0[1/n]$, or, using the sum notation, $v = \sum_{i=1..d} v_i 0[1/i]$.

However, not all basic geometry can be described by the four basic arithmetic operations on points. We will also need linear transformations. A *linear transformation* $T : \mathbb{R}^d \to \mathbb{R}^k$ is a function such that $T(v+w) = T(v)+T(w)$ and $T(kv) = kT(v)$. Note that a linear transformation is described by specifying $f(0[1/i])$ for every index i, because:

$$T(v) = T\left(\sum_{i=1..d} v_i \cdot 0[1/i]\right) = \sum_{i=1..d} T(v_i \cdot 0[1/i]) = \sum_{i=1..d} v_i T(0[1/i]))$$

This can be written using matrix notation, as follows: (we denote $w = T(v)$)

$$\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{pmatrix} = \begin{pmatrix} t_{1,1} & t_{1,2} & \ldots & t_{1,d} \\ t_{2,1} & t_{2,2} & \ldots & t_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ t_{k,1} & t_{k,2} & \ldots & t_{k,d} \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{pmatrix}.$$

The i-th column of this matrix is exactly the vector $T(0[1/i])$. We represent linear transformations computationally using matrices: T will denote both the linear transformation and the matrix representing it. Matrices can be multiplied by vectors ($T \cdot v = T(v)$) and by matrices ($T \cdot U$ is a matrix V such that $V(v) = T(U(v))$ for every v; importantly, it is also a linear transformation). Matrix multiplication is associative, but not commutative, just like function composition.

An important example of linear transformation is $R^\alpha_{i,j}$, which denotes the rotation by angle α in the plane (i, j). This rotation is defined as follows: $R^\alpha_{i,j}(v) = v[v_i \cos\alpha + v_j \sin\alpha/i][v_j \cos\alpha - v_i \sin\alpha/j]$ for $i \neq j$. We measure α in radians, that is, the full turn equals $\tau = 2\pi$, which is the length of a Euclidean circle of radius 1. Radians are very convenient in non-Euclidean geometry, because it will make many of our computations natural. Fractions of τ are usually more convenient to write in degrees; x° is considered equivalent to $\frac{x\tau}{360}$, so for example, we write 90° for $\frac{\tau}{4}$. The power of the matrix representation is that we can denote any arbitrarily long composition of linear transformations as a single matrix.

2.2 Homogeneous coordinates

In our engine, we will use linear transformations to represent *isometries*, that is, functions on the space which do not change distances. In the previous section, we have seen that rotations around axes going through 0 are indeed isometries. So are mirror images $M_i(v) = v[-v_i/i]$. However, we would like our system to also represent translations, which in Euclidean geometry are functions $v \mapsto v+w$ (move in the direction given by vector w). These are not linear transformations, which is easy to see by taking $v = 0$ and $w \neq 0$; for every linear transformation T, we have $T0 = 0$.

We solve this problem by using *normalized homogeneous coordinates*. We represent d-dimensional space S using $d + 1$ coordinates, adding the last coordinate, which equals 1 for every point in the space. It is left as an exercise for the reader to show that not only reflections and mirror images, but also translations are linear transformations. Note that if v and w represent points of S, $(v - w)_{d+1} = 0$, so it does not represent a point (but a *vector* – it can be added to a point in S to obtain another point in S).

Homogeneous coordinates are also useful for linear perspective, and in general, projective transformations. Technically, projective geometry will not be necessary for non-Euclidean geometry, but it is generally relevant and useful for 3D computer graphics, so 2. The d-dimensional projective space, denoted $P\mathbb{R}^d$, is $\mathbb{R}^{d+1}/\{0\}$, where we identify v and $k \cdot v$ for $k \neq 0$ (that is, we consider them alternative, equivalent representations of the same point in $P\mathbb{R}^d$, just like $1/2$ and $3/6$ are equivalent representations of the same rational number). Note that linear transformations and matrices are *well defined*: if v and w are equivalent, so are Tv and Tw. Therefore, our machinery of linear transformations and matrices is still useful in the projective world. (Homogeneous coordinates differ from normalized homogeneous coordinates in that we no longer require the last coordinate v_{d+1} to be 1 – any non-zero value of the last coordinate is a valid representation of a point in S.)

Linear perspective is a simulation of our visual system. Suppose we are looking at a point whose Cartesian coordinates relative to the camera are $v = (x, y, z)$. The camera itself is at point 0. This point is projected on our retina, which is modelled as the plane $F = \{v : v_3 = 1\}$. The intersection of the plane F and the line from eye to v is the point $(x/z, y/z, 1)$. In OpenGL, a point $p \in P\mathbb{R}^3$ is normalized to p such that $p_4 = 1$, and rendered at screen position (p_1, p_2); the value p_3 is used for depth testing (if two points were to be rendered in the same point, only the point with smaller p_3 is rendered). Thus, to obtain the linear perspective projection of v, we use the linear transformation $Pv = v[v_4/v_3][v_4/v_3]$, and let OpenGL do all the work. (The specific matrix P might need to be adjusted based on the field of view.)

Linear transformations are used to convert between various coordinate system. Suppose we need to render a 3D model (described as a set of points in our space S) as seen by an observer at some point. a 3D model is a set of points v, described using coordinates relative to the model; to obtain the coordinates relative to the world (*world coordinates*), we need to apply an isometry, that is,

the global coordinates can be described using Mv for some linear transformation M. Rotating or moving our 3D model corresponds to changing the matrix M. Now, we need to transform the *world coordinates* to *viewer coordinates*, which is again done by multiplying by a linear transformation V. As explained in the last section, we apply the perspective transformation P to finally obtain the coordinates used by OpenGL, which are thus $P(V(M(v)))$.

2.3 Inner product and Euclidean geometry

One more ingredient is the *Euclidean inner product*, which is the function $g_E : \mathbb{R}^{d+1} \times \mathbb{R}^{d+1} \to \mathbb{R}$ defined by $g_E(v, w) = \sum_{i=1}^{d} v_i w_i$. We write $d + 1$ in the dimension to include the extra coordinate introduced in previous section. As seen from the formula, the inner product is easy to compute. In Euclidean geometry, the inner product can be used to compute the following things:

- Distance between two points x and y: $\delta(x, y) = \sqrt{g_E(x - y, x - y)}$.
- The magnitude of a vector v: $|v| = \sqrt{g_E(v, v)}$.
- Whether two vectors v and w go in the same direction. In general, we have $g_E(v, w) = |v| \cdot |w| \cdot \cos\alpha$, where α is the angle between v and w. If $g_E(v, w) = 0$, v and w are called *orthogonal*.

Note that, If I is a linear transformation which is an isometry of the Euclidean space, we have $g_E(v, w) = g_E(Iv, Iw)$. Our model of a *geometry* will consist of the following ingredients:

- The set S of points. In case of the Euclidean space $\mathbb{E}^d$, we have $S = \{x \in \mathbb{R}^{d+1} : x_{d+1} = 1\}$. We will usually use the same symbol for S as the geometry.
- The point $C_0 \in S$ which we consider the origin. We have $C_0 = 0[1/d + 1]$.
- The definition of function g.
- Translation by x units in direction i, which we denote by T_i^x. In case of Euclidean geometry, we have $T_i^x(v) = v[v_i + x/i]$.
- Allowed rotations and mirror images. In $\mathbb{E}^d$, all rotations R_{ij}^{α} and mirror images M_i (for $i, j \leq d$, $i \neq j$)) are allowed.

In this book, we choose our models so that translations, rotations and mirror images are all linear transformations. While g_E being just a function on $\mathbb{R}^{d+1}$ works in this chapter, a more spohisticated approach will be needed for anisotropic geometries.

2.4 Spherical geometry

It is quite natural how to use the system above to represent spherical geometry $\mathbb{S}^d$.

The number d refers to the number of directions we can move in, so spheres in our world (e.g., the surface of Earth) are referred to as two-dimensional ($\mathbb{S}^2$). We will model a sphere $\mathbb{S}^d$ embedded in $\mathbb{E}^{d+1}$. We will be using $d+1$ dimensions again.

- We inherit g from $\mathbb{E}^{d+1}$: $g_S(v,w) = \sum_{i=1}^{d+1} v_i w_i$.
- We have $S = \mathbb{S}^d := \{x \in \mathbb{S}^{d+1} : g_S(x,x) = 1\}$. The sphere is the set of points in distance 1 from 0.
- Just like in $\mathbb{E}^d$, we have $C_0 = 0[1/d+1]$.
- Rotations and mirror images allowed are just like in $\mathbb{E}^d$.
- To translate, we just rotate the sphere: $T_i^x = R_{i,d+1}^x$.

In spherical geometry, the use of g for measuring distances between points is no longer as simple as in $\mathbb{E}^d$. Technically, g is used to measure the magnitudes and angles between *tangent vectors*, while by the distance $\delta(x,y)$ we mean the length of the shortest curve c connecting x and y. It is possible to use g to define the length of curves, using differentiation and integration, although we will not need such a general definition at this point.

It is straightforward to compute the distance $\delta(C_0, x)$. To see this, imagine C_0 to be the north pole. Then, $r = \delta(C_0, x)$ is $\pi/2$ minus the latitude of x. A simple picture shows that $x_{d+1} = \cos(r)$. Therefore, $\cos\delta(C_0, x) = x_{d+1}$, which can be also written as $\cos\delta(C_0, x) = g(C_0, x)$. To compute the distance of two arbitrary points, x and y, we could take an isometry I which moves x to C_0, and thus get $\cos\delta(x,y) = \cos\delta(I(x), I(y)) = \cos\delta(C_0, I(y)) = g(C_0, I(y)) = g(I(x), I(y)) = g(x,y)$ (the last formula follows from isometries of $\mathbb{E}^{d+1}$ preserving g). Another possible method of computing $\delta(x,y)$ is based on the formula $\sin(\delta(x,y)/2) = g(x-y, x-y)/2$.

The proof above of $\cos\delta(x,y) = g(x,y)$ shows the power of our methods. We solve a special case using coordinates, and then by using the properties of g, isometries, and point addition, we show that our formula is in fact general. As a nice exercise in this method, we will prove the spherical analog of the Pythagorean Theorem.

Let ABC be a right triangle with right angle in C, and a, b and c the length of edges opposite vertices A, B and C. First, assume $C = C_0$, $A = T_1^a(C_0)$ and $B = T_2^b(C_0)$. Thus, we have $A = 0[\sin a/1, \cos a/d+1]$ and $B = 0[\sin b/2, \cos b/d+1]$, and $\cos c = \delta(A,B) = \cos a \cos b$. The same formula will be true for any right triangle, because we can always use the isometries to assume C is in C_0 and A and B are on the respective axes.

2.5 Pseudo-Euclidean geometry

Before we proceed to a model of hyperbolic geometry, we will need to explain pseudo-Euclidean geometry (also called Minkowski geometry).

This geometry comes from the physics of spacetime. In spacetime, we have both space and time dimensions, and thus, our events will have both space and time coordinates. We choose the units of space and time so that the speed of light $c = 1$. Thus, if our space unit is meters, our time unit is *light meters* (the amount of time light needs to travel 1 meter).

Space and time dimensions behave in different ways. Before special relativity theory, people assumed that if, according to some observer, an event happens at time t at position x, then, according to another observer moving at speed v, it happens also at time t, but at position $x - tv$. Today, we know this is not true.

This can be explained by the spacetime using a inner product. Let us assume that x_1 to x_d are space coordinates of a spacetime vector x, and x_{d+1} is its time coordinate. The Minkowski inner product $g_H(x, y)$ is defined as $g_H(x, y) = x_1y_1 + \ldots + x_dy_d - x_{d+1}y_{d+1}$. The value of g_H is invariant under the isometries of the Minkowski spacetime. Changing the observer (assuming it is an *inertial frame of reference*, i.e., not accelerating) corresponds to an isometry of the Minkowski spacetime.

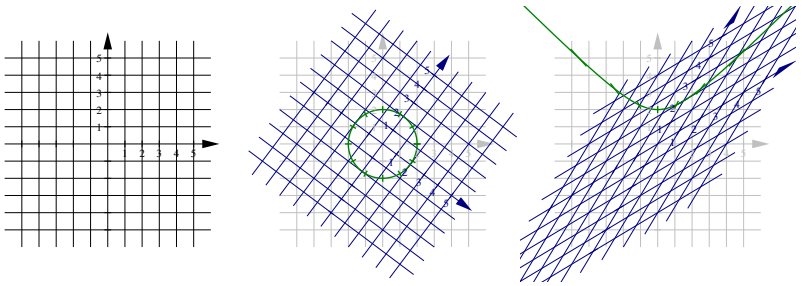

Figure 2.1: (a) A coordinate system. (b) Euclidean rotation. (c) Lorentz boost.

For example, assume $d = 1$ for simplicity. Consider two events in our spacetime, at coordinates $a = (0, 0)$ and $b = (3, 5)$, according to an observer O_1. The event b happens 5 time units in the future, and 4 space units away from us. Consider another observer O_2 starting at a and moving towards b. If x is the spacetime coordinates of an event according to O_1, then its spacetime coordinates according to O_2 will be $x' = f(x)$, where $x'_1 = \frac{5}{4}x_1 - \frac{3}{4}x_2$ and $x'_2 = \frac{5}{4}x_2 - \frac{3}{4}y_2$. In Figure 2.1a, the coordinate system used by O_1 is shown, and in Figure **??**b, the coordinate system used by O_2 is shown. (It is easy to check that $f(x)$ keeps the value of g_H, thus it is indeed an isometry of our Minkowski spacetime.)

Thus, the observer O_2 will see the event b at coordinates $(0, 3)$. In this example, we observe *time dilation*: what takes 4 time units to observer O_1,

takes 5 time units to observer O_1. We did not need to compute the formula for f to get the end result 3 – it was enough to compute $g_H(b-a, b-a) = -9$. If $g_H(b-a, b-a) = x < 0$, it means that there is an inertial observer who considers a and b to be the same location in space, separated by $\sqrt{-x}$ in time. If $x > 0$, then there is an inertial observer who consider a and b to be simultaneous, but separated by $\sqrt{x}$ in space.

This example shows the interaction between space and time coordinates: as we change the observer, the space and time coordinates affect each other. The well-known effects of special relativity (length contraction, time dilation, twin paradox) are caused by this interaction. However, this interaction is in fact analogous to Euclidean rotation – a rotation of our coordinate system, R^α_{12}, also leads to a similar interaction between space coordinates 1 and 2. However, because in the formula for g_H, the sign used for x_i^2 and x_j^2 is different, we will not be using the rotations R^α_{ij}, but instead *Lorentz boosts*, L^α_{ij}. See Figure 2.1b and Figure 2.1c for a comparison of Euclidean rotations and Lorentz boosts.

Exercise 2.5.1. Find the properties that uniquely define the rotations R^α_{ij}, and find the unique L^α_{ij} that satisfy the analogous properties in the pseudo-Euclidean case. (Might require solving differential equations.)

We will not solve this exercise here, we will just provide the result: $L^\alpha_{ij}(v) = v[v_i \cosh\alpha + v_j \sinh\alpha, v_j \cosh\alpha + v_i \sinh\alpha]$. The formula differs from the formula $R^\alpha_{ij}(v) = v[v_i \cos\alpha + v_j \sin\alpha, v_j \cos\alpha - v_i \sin\alpha]$ in that it is using cosh and sinh instead of cos and sin, and + instead of -. Functions sinh and cosh can be computed using $\cosh\alpha = (e^\alpha + e^{-\alpha}/2$, $\sinh\alpha = (e^\alpha - e^{-\alpha}/2$. Just like sin and cos satisfy $(\cos\alpha)^2 + (\sin\alpha)^2 = 1$, their hyperbolic analogs satisfy $(\cosh\alpha)^2 - (\sinh\alpha)^2 = 1$; the graph of $f(\alpha) = (\sinh\alpha, \cosh\alpha)$ is a hyperbola, hence their name. (Figure 2.1c includes a hyperbola which is not affected by Lorentz boosts, just like a circle is not affected by rotations.) The parameter α is called *rapidity*. An observer corresponding to rapidity *alpha*, from our point of view, will move $\sinh\alpha$ units of space in $\cosh\alpha$ units of time, thus their speed is $\tanh\alpha = \sinh\alpha/\cosh\alpha$. Same as for Euclidean rotations, we have $L^\alpha_{ij} L^\beta_{ij} = L^{\alpha+\beta}_{ij}$ (i.e, if observer O_2 is moving relative to O_1 with rapidity α and O_3 is moving relative to O_2 with rapidity β, then O_3 is moving relative to O_1 with rapidity $\alpha + \beta$). Since, for every α, $\tanh\alpha < 1$, we can never reach the speed of light, no matter how much we accelerate.

2.6 Hyperbolic geometry

Hyperbolic geometry turns out to be very similar to spherical geometry, we just embed it into the Minkowski spacetime instead of Euclidean space. We will call this geometry $\mathbb{H}^d$. We will be using $d+1$ dimensions again.

- We inherit g from the Minkowski spacetime:

$$g_H(v, w) = \sum_{i=1}^{d} v_i w_i - v_{d+1} w_{d+1}.$$

- We have $S = \mathbb{H}^d := \{x \in \mathbb{R}^{d+1} : g_H(x,x) = -1, x_{d+1} > 0\}$. The hyperbolic space is the set of points 1 whose spacetime interval from 0 corresponds to 1 unit of time (in the future). It is called a *pseudosphere* or *Minkowski hyperboloid*.
- Just like in $\mathbb{E}^d$ and $\mathbb{S}^d$, we have $C_0 = 0[1/d+1]$.
- Rotations and mirror images allowed are just like in $\mathbb{E}^d$ and $\mathbb{S}^d$.
- To translate, we just rotate the hyperboloid using Lorentz boosts: $T_i^x = L_{i,d+1}^x$.

The rest of Section 2.4 is also analogous.

- Distance between $x, y \in \mathbb{H}^d$ can be computed using $\cosh(\delta(x,y)) = g_H(x,y)$ or $\sinh(2\delta(x,y)) = g_H(x-y, x-y)/2$.
- The hyperbolic analog of the Pythagoras theorem is: $\cosh(c) = \cosh(a) \cdot \cosh(b)$.

2.7 Universal homogeneous model

The models explained for $\mathbb{E}^d$, $\mathbb{H}^d$ and $\mathbb{S}^d$ are very similar, and in fact, we can use one common implementation for all of them.

The three geometries differ by the parameter called *Gaussian curvature*, and denoted with K. Curvature will be discussed in more detail in Chapter 4. Curvature measures how much the geometry is different from Euclidean geometry; in the models discussed so far, we have $K = 0$ for Euclidean geometry, $K = 1$ for spherical geometry, $K = -1$ for hyperbolic geometry. This is because we have been measuring distances in absolute units; in case of spherical geometry, this means that the sphere itself has radius 1. A sphere of radius r has curvature $1/r^2$. Earth is (roughly) a sphere of very large radius, so the effects of curvature are rarely noticeable. To contruct the general model while keeping the rule $C_0 = 0[1/d+1]$, we will divide the $d+1$-th coordinate by $r = \sqrt{1/|K|}$. Let $\sin_K(x)$ and $\cos_K(x)$ be the 1-th and $(d+1)$-th coordinate of the point $T_1^x(C_0)$ obtained by moving C_0 x units in the direction 1. Thus, the functions $\sin_K$ and $\cos_K$ are defined as follows:

$$\sin_K(\alpha) = \begin{cases} \sqrt{1/K}\sin(\alpha\sqrt{K}) & \text{for } K > 0 \\ \alpha & \text{for } K = 0 \\ (1/\sqrt{-K})\sinh(\alpha\sqrt{-K}) & \text{for } K < 0 \end{cases}$$

$$\cos_K(\alpha) = \begin{cases} \cos(\alpha\sqrt{K}) & \text{for } K > 0 \\ 1 & \text{for } K = 0 \\ \cosh(\alpha\sqrt{-K}) & \text{for } K < 0 \end{cases}$$

We have defined the cases $K > 0$ and $K < 0$ separately to avoid computing $\sqrt{K}$ when $K < 0$, although in fact we could just use the formula for $K > 0$ is

every case – we just need to use complex numbvers for the case $K < 0$ since the formula calls for $\sqrt{K}$, and for $K = 0$, compute the limit ($K \to 0$) to avoid $0/0$ in the definition of $\sin_0$.

This leads to the following universal model for $\mathbb{U}_k^d$, a d-dimensional space of curvature K:

- We have $g(v, w) = \sum_{i=1}^{d} v_i w_i + v_{d+1} w_{d+1}/K$. (For vectors in Euclidean geometry, we skip the last component, which is $0 \cdot 0/0$ for Euclidean vectors.)
- We have $S = \{x \in \mathbb{R}^{d+1} : x_{d+1}^2 + (x_1^2 + x_2^2 + x_3^2 + \ldots + x_d^2)K = 1\}$. For $K \leq 0$ we also restruct to $x_{d+1} > 0$. (We do not use g directly because of the Euclidean case.)
- We have $C_0 = 0[1/d + 1]$, and rotations and mirror images allowed are just like in $\mathbb{E}^d$ and $\mathbb{S}^d$.
- Translation works as follows:

$$T_i^x(v) = v[v_i \cos_K x + v_{d+1} \sin_K x/i][v_{d+1} \cos_K x - K v_i \sin_K x].$$

In the RogueViz engine, only the case $K \in \{-1, 0, 1\}$ is modelled – other curvatures are obtained by scaling the models measured in absolute units. This approach leads to simple formulas, although we would recommend using $\mathbb{U}_k^d$ for visualizations and applications which focus on changing the curvature (for example, a smooth transition from spherical geometry through Euclidean geometry to hyperbolic geometry – when the transition is close to curvature 0, the scaled model amounts to division by 0). In this book, we consider $\mathbb{E}^d$, $\mathbb{H}^d$ and $\mathbb{S}^d$ to be the case where $K \in \{-1, 0, 1\}$ (distances measured in absolute units). For the intrinsic mathematical properties, it does not matter which model of hyperbolic or spherical geometry we are using – for example, we could be using the Poincaré disk model instead of the Minkowski hyperboloid model. However, if coordinates are given in this book, that they are coordinates in one of the models above (unless specified otherwise).

In anisotropic geometries, some aspects will change, but a significant part of our work so far will still be recoverable.

Chapter 3

Creating a simple visualization

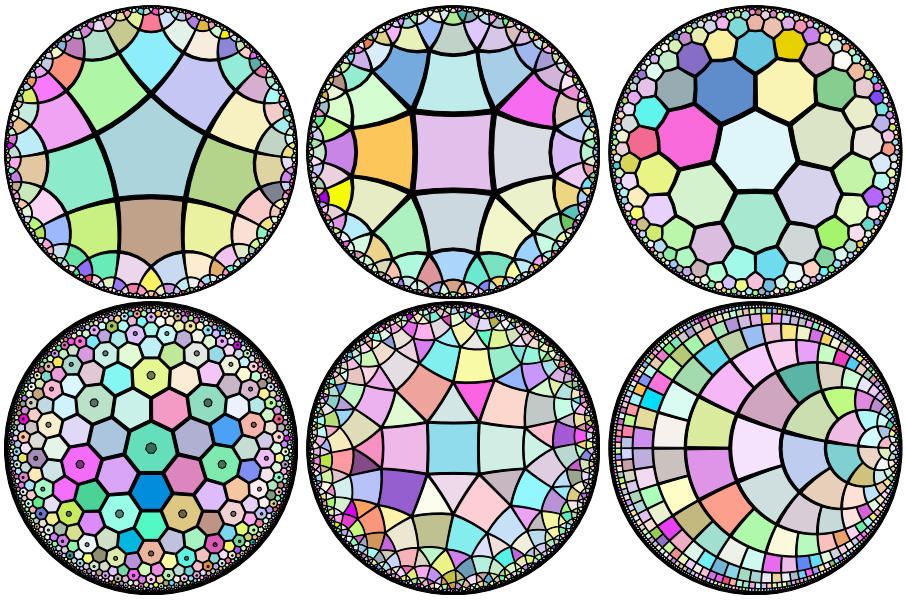

Figure 3.1: Sample tessellations of the hyperbolic plane: (a) {5,4}, (b) {4,5}, (c) {7,3}, (d) bitruncated {7,3} tiling, (e) hybrid tiling, (f) binary tiling.

In this chapter, how the model of $\mathbb{H}^2$ discussed in the previous section can be used to create a simple game, model or visualization on a tessellation of a hyperbolic plane. This will showcase some useful concepts, both mathematical and practical. We also briefly explain what changes for three-dimensional visualizations of non-Euclidean geometry.

Tessellations are not only a great way to organize our game or visualization, but also a great way to avoid numerical precision issues, which are a very serious issue in large-scale computations in hyperbolic geometry. This is of crucial importance in applications, both in game design (many of the most interesting things in the gameplay of HyperRogue are consequences of large-scale behavior of hyperbolic geometry) and applications in data visualization and modelling.

See Figure 3.1 for a sample end result. In the top row, we have three regular tessellations: {5,4}, {4,5}, {7,3}. Tessellation (d) is an Archimedean (1-uniform) tiling: in every vertex, two regular hexagon tile and one heptagon tile meet. Tessellation (e) is a hybrid tiling: every tile is a hyperbolic square or a triangle, but vertices are different. Tessellation (f), the binary tiling, is not based on regular polygons.

Regular tessellations are labelled by a pair of numbers (Schläfli symbol) $\{p, q\}$, where p is the number of sides of the regular shape used as tiles, and q is the *valence*, i.e., the number of tiles which meet in a vertex. The binary tiling is based on horocycles, which will be explained in more detail in Chapter **??** (Exercise 4.2.3).

In the Euclidean plane, each angle of a regular p-gon has the internal angle of $\frac{\tau}{2} - \frac{\tau}{p}$. If we want q of them to meet in a vertex, we need this angle to equal $\frac{\tau}{q}$. By simple algebra, we get $pq = 2p + 2q$, which can be also expressed as the equality of the area and perimeter of a $p \times q$ rectangle. There are three solutions to this: $\{3, 6\}$, $\{4, 4\}$ and $\{6, 3\}$. Spherical tessellations have larger (arbitrarily large) angles, so there are also five spherical tessellations: $\{3, 3\}$, $\{3, 4\}$, $\{3, 5\}$, $\{4, 3\}$ and $\{5, 3\}$, which correspond to five Platonic solids. (If we allow $p = 2$ or $q = 2$, we also get $\{2, p\}$ and $\{p, 2\}$ for any $p \geq 2$; these also correspond to tessellations of the sphere.) All the remaining pairs correspond to hyperbolic tessellations, which have arbitrarily small angles. Generally, the larger the numbers, the larger the tiles are. The theory of tilings will be discussed in more detail in Chapter 7.1.

3.1 Beltrami-Klein and Poincaré disk model

While the Minkowski hyperboloid model is good as the internal model used for computations, it is not very good for visualization. This is because it is three-dimensional, and also because it lives in Minkowski geometry, not in Euclidean geometry – while it maps angles and distance faithfully in Minkowski geometry, the Euclidean hyperboloid might be misleading.

The *general perspective projection* at projection distance d maps the point $\{x, y, z\}$ on the Minkowski hyperboloid or sphere to the point $\{\frac{x}{z+d}, \frac{y}{z+d}\}$ on the plane. This is linear perspective, where the eye is assumed to be placed at point $\{0, 0, -d\}$. Two exceptionally important values of d are $d = 0$ and $d = 1$. These projection are called *models*, because they are often used as alternative methods of modelling the hyperbolic plane. This projection is visualized in Figure 3.2a, for $d = 1$.

The perspective projection at $d = 0$ is called the *Beltrami-Klein disk model*

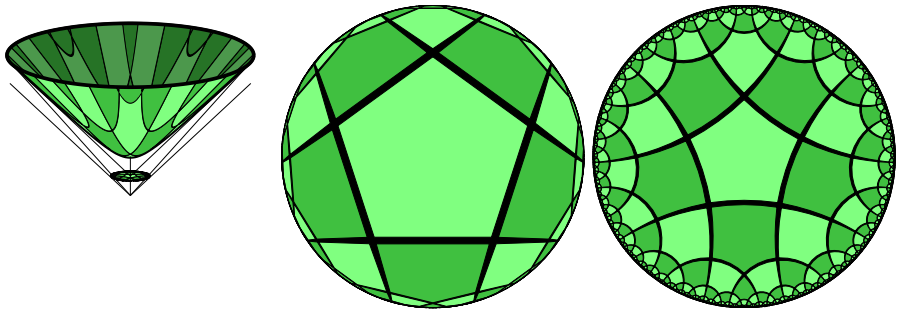

Figure 3.2: The {5,4} tessellation: (a) projection from the hyperboloid model, (b) Beltrami-Klein model, (c) Poincaré disk model.

(Figure 3.2b). The whole hyperbolic plane is mapped into the disk of radius 1. It maps hyperbolic straight lines to Euclidean straight lines. It is the hyperbolic analog of the gnomonic projection, which also maps spherical straight lines to Euclidean straight lines.

The perspective projection at $d = 1$ is called the *Poincaré disk model* (Figure 3.2c). In this case, the whole hyperbolic plane is also mapped into the disk of radius 1. It is a *conformal* projection, which means that angles and small shapes are mapped faithfully. This property makes it good for artistic purposes (starting with the *Circle Limit* series by M. C. Escher), and also the most common visualization of the hyperbolic plane; therefore, in this Chapter we will use this projection. See Chapter 5 for a more detailed study of possible projections of spherical and hyperbolic geometries.

3.2 The structure of the tiling

Most puzzles and simple games take place in the Euclidean square grid. This grid is very simple to implement in a computer: technically, we use a *two-dimensional array* of tiles, indexed by x and y coordinates. A very similar approach can be used for the Euclidean hex grid – the easiest method is to also use x and y coordinates, but make them no longer orthogonal. See Figure **??**.

This approach does not work for hyperbolic tessellations. Instead, we will represent our tiling using a lazily generated *combinatioral map*. This data structure will be appropriate for all kinds of two-dimensional tilings.

Our combinatorial map will consist of tiles. Every tile is represented by an object in computer memory. We will use the $C++$ notation $t \to \text{NAME}$ to denote the attributes of a tile object t. The tile object t has the following attributes:

- $t \to \text{TYPE}$ is the number of neighbors,
- for $i = 0$ to $t \to \text{TYPE} - 1$, $t \to \text{MOVE}[i]$ is the pointer to the i-th neighbor of t, in clockwise order,

- for $i = 0$ to $t \to \text{TYPE} - 1$, $t \to \text{SPIN}[i]$ is j if the i-th edge of t is the j-th edge of $t \to \text{MOVE}[i]$.

Tile objects are created on the fly as they are required (hence *lazy generation*). The pointer can be either a reference to an existing tile, or the *null pointer*, which means that the neighbor is not known yet.

A *walker* is a very useful abstraction. Intuitively, a walker $w = (t, i)$ corresponds to a pointer "looking at" a specific edge i of a specific tile t. By $w + k$ we mean w rotated clockwise by k tiles, that is, a pointer to the $(i + k) \bmod t \to \text{TYPE}$. By $w - k$ we mean w rotated counterclockwise by k tiles. By $w + \text{STEP}$ we mean the walker w looking at the same edge from the other side, that is, if $w = (t, i)$, then $w + \text{STEP} = (t \to \text{MOVE}[i], t \to \text{STEP}[i])$.

The system desscribed above is perfect for working with all kinds of tilings on two-dimensional orientable surfaces. For non-orientable tilings (see Chapter 7.4) we no longer have a global concept of *clockwise ordering*, so we need to extend the tile object with $t \to \text{MIRROR}[i]$, which specifies whether the orientations of t and $t \to \text{MOVE}[i]$ are consistent, and to extend the walker with mirroredness, which specifies whether the orientation of the walker agrees with the orientation of the tile.

3.3 Building a regular hyperbolic tiling

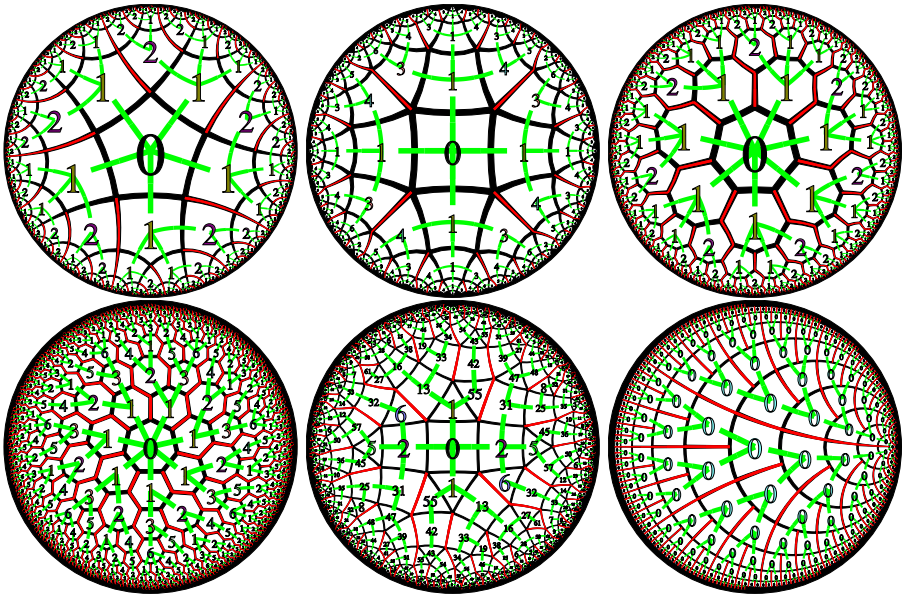

Figure 3.3: Trees of the tessellations from Figure 3.1.

We will use a *geodesic regular tree structure* (GRTS) to generate maps for our

hyperbolic tilings. The idea is visualized in Figure 3.3. The following properties are important:

- Edges are split into *tree edges* and *non-tree edges*. Tree edges give a tree structure to our tiling, that is, for every pair of two tiles t_1, t_2, there is exactly connection between t_1 and t_2 which crosses only tree edges.
- Each non-tree edge connects two vertices. A *wall* is a set of vertices that are connected via non-tree edges, together with the non-tree edges themselves.
- Tree edges are directed: one side is called the *parent*, and the other side is called the *child*. The arrows go from parents to children. In most pictures in Figure 3.3 we have a *root* (labelled with 0), who is not a child of any other tile. In the binary tiling, we have no root – the tree extends infinitely in the parent direction.
- The *subtree* at tile t is the set of all tiles which are descendants of t (t itself, the children of t, their children, and so on). The trees are constructed in such a way that there are finitely many possible subtree shapes. We index these possible subtrees by numbers. In Figure 3.3, two tiles labeled with the same number have subtrees of exactly the same shape. We will call these labels *states*.

To generate a map of one of our hyperbolic tilings, we will need a table describing how subtree shapes connect to each other. We call these tables *geodesic regular tree structures* (GRTS). See Table 3.1 for the GRTS for five of six of our sample tessellations.

We have oriented the tiles in Figure 3.3 so that for every tile t (except the root), $t \to \textsc{move}[i]$ is the parent of t. We will denote the i-th value in row q by $\mathrm{GRTS}[q, i]$. If $t \to \textsc{move}[i]$ is a state index q', this means that a tile t in state q will always have a child in state q' beyond its i-th edge. (Examine Figure 3.3 to see this is correct.) P means *parent*, L means *non-tree edge to the left* and R means *non-tree edge to the right*.

We use a GRTS to construct the whole map as follows. We start our map with a single tile, which will be the root (in state 0). Whenever we want to move to the i-th neighbor of tile t in state q, we check the relevant value x in our GRTS (row q, column i), and proceed as follows:

Child Rule. If x is an index of a state q', we create a new tile t' in state q', and connect the i-th edge of t to the j-th edge of t', where j is the number of index of parent edge of state q' (our example trees are constructed so that j is always 0; in general, j is the position of the unique P in the row q' of the GRTS).

The Child Rule will correctly build the whole tree from the root; however, we will also need to connect non-tree edges.

Left Rule. If x is L, we construct a walker $w := (q, i)$, and compute $w' := w - 1$. Then, we use the following algorithm:

(a)

id	0	1	2	3	4
0	1	1	1	1	1
1	P	2	1	1	R
2	P	L	2	1	R

(b)

id	0	1	2	3
0	1	1	1	1
1	P	3	1	4
2	P	L	3	4
3	P	L	6	4
4	P	3	5	R
5	P	3	1	2
6	P	2	1	4

(c)

id	0	1	2	3	4	5	6
0	1	1	1	1	1	1	1
1	P	L	2	1	1	R	R
2	P	L	L	2	1	R	R

(d)

id	0	1	2	3	4	5	6
0	1	1	1	1	1	1	1
1	P	L	2	3	R	R	
2	P	L	L	4	1	R	R
3	P	L	5	6	R	R	
4	P	L	L	5	R	R	
5	P	L	L	2	R	R	
6	P	L	4	1	1	R	R

(f)

id	0	1	2	3	4
0	P	L	0	0	R

Table 3.1: Tables of GRTS corresponding to the tessellation trees from Figure 3.3.

- Suppose $w' = (t', j)$ where t is in state q'. If $\mathrm{GRTS}[q', j] = R$, connect the j-th edge of t' with the i-th edge of t.
- Otherwise, let $w' := w' + \textsc{Step} - 1$. While computing $w' + \textsc{Step}$, it might be necessary to construct new tiles, possibly even calling our algorithm recursively.

(We probably need an example here.)

Exercise 3.3.1. Assume that all tree edges in Figure 3.3 are already created. Choose one of the red (wall) edges and a red edge to the left of it, and simulate the Left Rule on this edge, to show that it will correctly connect to the tile on the other side on the wall. Argue that the Left Rule correctly connects all non-tree edges. (Use induction on the size of the part of the wall containing the i-th edge of w which is closer to the root.)

Right Rule. If x is R, we use a symmetric algorithm. We start with $w' := w + 1$, stop when $\mathrm{GRTS}[q', j] = L$, and let $w' := w' + \textsc{Step} + 1$ otherwise.

Parent Rule. The case $x = P$ only happens for unrooted trees. In rooted trees, x can never be P, by construction: the root has no P edges, and for other tiles, P edges are immediately connected.

In the unrooted case, when $x = P$ happens, we choose a state q' and index j such that $\mathrm{GRTS}[q', j] = q$, create a tile of state q', and connect its j-th edge to the i-th edge of q. We need to make sure that (q', j) is chosen in a way that does not generate infinitely descending walls, thus breaking the proof by induction in Exercise 3.3.1. In the case of binary tiling this can be done, e.g., by alternating between using the two possible choices.

Exercise 3.3.2. Show that, even though the Left/Right Rule algorithm may call itself recursively, generating n connections will always be performed in time $O(n)$. (Solving this exercise requires some experience in amortized analysis.)

Geodesic regular tree structures can be also found for other hyperbolic tessellations (not necessarily consisting of tiles of a single shape). They can be visualized in HyperRogue using [method]. They are useful not only for generating the combinatorial map, but also for computation of distances in hyperbolic tilings; this subject is discussed in Chapter 7.12. The methods of generating GRTS are discussed in Chapter 7.6.

3.4 From the combinatorial map to the hyperbolic plane

So far, we have constructed our regular tesselations as a structure based purely on combinatorics. This is very important, because such construction is immune to numerical precision issues.

Numerical precision issues are very serious in exponentially expanding geometries. Recall that a circle of radius r in the hyperbolic plane has circumference $\tau \cosh(r)$. Therefore, we can find a set X_r of roughly $\exp(r)$ points on

the circumference of such a circle such that every pair of points is in distance at least 1 from each other. A naive way of representing points of $\mathbb{H}^2$ uses the three coordinates in the Minkowski hyperboloid model, each of them a floating point variable, say, on 64 bits. This gives us 2^{192} representable points, thus, some of the points in $X_1 33$ would have to have the same representation (and thus collapse to a single point). The same will happen for any other representation based on a fixed number of bits. Note that this is only a upper bound estimation; in fact, the problems become visible much earlier. See 7.5 for a deeper discussion of this.

The solution to this is to use a tessellation. Recall (Section 2.2) the approach where the current camera position is modeled as a matrix V (transforming world coordinates to screen coordinates), and every object O in the game has a matrix M_O (transforming object-relative coordinates to world coordinates). We use the tessellation by replacing the matrix V by a pair (V, t_V), where t_V is some tile (probably, the tile the camera is currently centered on), and V is now relative to the tile t rather than any fixed world coordinates. A similar pair representation (M_O, t_O) is also used instead the matrix M_O. When rendering the scene, we find all tiles t' closed to t_V, and for each of them, we compute the matrix $M_{t',t}$ which transforms t'-relative coordinates to t-relative coordinates. An object O is then rendered at position $VM_{t',t}M_O$. We only need to render tiles t' which are close to t; more distant tiles would be too small to be visible anyway.

We still need to explain how to compute $M_{t',t}$. It is sufficient to compute this matrix for adjacent tiles; in general, we can find a path of adjacent tiles $t = t_0, t_1, \ldots, t_k = t'$, and obtain $M_{t',t}$ by multiplying the matrices M_{t_{i+1},t_i} for every step. In this chapter, we will show how to compute this for a regular tiling; binary tiling is discussed in Chapter 4, and other tilings are discussed in Chapter 7.1.

For every tile t, the coordinate system relative to t puts the center of t in C_0, and the center of its 0-th edge on the first axis (in the positive direction). Suppose that $t \to \text{MOVE}[i] = t'$ and $t' \to \text{MOVE}[i'] = t$. Let d be the distance between the center of the two tiles, and n be the number of sides. Then, we have $M_{t',t} = R_{1,2}^{-\tau j/n} T_1^d R_{1,2}^{\tau i/n}$.

We need to compute d. This can be done using the *hyperbolic law of cosines*.

Theorem 3.4.1 (general law of cosines)**.** *Take a triangle with angles α, β, γ in a space of curvature K. Then the length of the edge a opposite α satisfies the formula*

$$\cos\alpha = -\cos\beta\cos\gamma + \sin\beta\sin\gamma\cos_K a$$

Exercise 3.4.2. Compute all the relevant lengths (d) and points (e.g., coordinates of tile vertices) using the hyperbolic law of cosines.

3.5 Small tricks

In this section, we list minor issues related to visualizations of the hyperbolic plane which have not been covered so far.

Nice lines. When using the Poincaré disk model, straight lines are rendered as circle arcs. If we rendered the hyperbolic n-gons polygons as Euclidean n-gons, their edges would be rendered as straight lines, which looks wrong. A simple solution to this is just to subdivide the edges. Just like in spherical geometry, the midpoint of v and w can be computed using the simple formula normalize$(v+w)$, where normalize$(v) = v/|v|$. We can recursively subdivide the edges in halves using the midpoint formula until the line is smooth enough.

Since the Poincaré disk model has variable scale (objects closer to the boundary of the disk are rendered smaller), it is also good to draw the lines finer when they are closer to the boundary.

Matrix tricks. If M is a linear transformation from $\mathbb{R}^d$ to $\mathbb{R}^d$, by M^{-1} we denote its inverse, that is, a linear transformation such that $MM^{-1}(x) = M^{-1}M(x) = x$ for every x. For example, $M_{t,t'}$ and $M_{t',t}$ are inverses.

In general, the inverse matrix is computed by solving a system of equations. However, in most cases, our matrices will represent isometries. If M is an isometry of the sphere ($K = 1$), then the matrix M^{-1} is simply M transposed, that is, $M^{-1}_{ij} = M_{ji}$. If M is an isometry of the hyperbolic plane ($K = -1$), we also need to negate M_{ij} if one (but not both) indexes equals $d + 1$.

Two especially useful linear transformations are R_v, which is a rotation moving v to the first axis (in the positive direction), and T_v is a translation moving C_0 to v. We can write T_v as $R_v^{-1}T^1_{\delta(C_0,v)}R_v$, although nicer formulas exist. Similarly, the operation T_v^x is a translation moving C_0 in direction of v, but by fraction x of the distance, that is, $T_v^x = R_v^{-1}T^1_{x\delta(C_0,v)}R_v$. In some sources, the operations $T_v w$, $T_v^{-1}w$ and $T_v^x w$ are written as $w \oplus v$, $w \ominus v$, and $w \oplus (x \otimes v)$; while this notation looks simple, we consider it bad, because $\oplus$ suggests commutativity, while the beauty of non-Euclidean geometry is that $T_v w$ is *not* commutative.

Exercise 3.5.1. Find nice formulas for R_v, T_v, and T_v^x.

Fragment shaders. So far we have been assuming that the polygon-based rendering method: we represent the scene as a collection of polygons; to render a polygon P, we compute the screen coordinates of every vertex of that polygon, and we use these coordinates to render P.

Another method of rendering is based on rendering every pixel separately. For every pixel p, we compute the part of the scene which contains the pixel p, and color the pixel p accordingly. On modern computer graphics hardware, this method can be done efficiently using a *fragment shader*.

The second method is advantageous in situations where the scene is mathematically simple, for example, when rendering a periodic coloring of $\mathbb{H}^2$. To color the pixel p using this method, we do as follows:

- Convert the coordinates p from the chosen projection to the chosen internal model.
- Transform p from screen coordinates to the coordinates relative to tile $t := t_V$: $p := V^{-1}p$

- Check whether the pixel p is inside the tile t. If not, it must cross the edge in direction of tile t'; set $(p, t) := (M_{t,t'}p, t')$, and repeat this step as needed.
- Now, we know that p is inside the tile t, so we can color it accordingly.

Using this method, we do not need to explicitly create all the tiles drawn, the fragment shader will just compute the coordinates as needed. Furthermore, the issue of curved lines mentioned above disappears. On the other hand, this method is less applicable to visualizations where the contains many objects which are not described using simple periodic math formulas.

Drawing graphics. One minor issue is creating graphics for models to be rendered in our visualization. Most graphics software, whether based on bitmaps, 2D vector graphics or 3D models, assume Euclidean geometry, so graphics designed in them cannot be used directly in hyperbolic geometry. In particular, while most cheaply made but still good looking indie games use pixel graphics, this approach is not appropriate in non-Euclidean geometry, where square grids would have to be weirdly twisted. The following approaches could be used to solve this issue:

- Generate the objects procedurally, using formulas, similar to those from Exercise 3.4.2. This approach is used for simple shapes in HyperRogue.
- Use graphics software which works directly in non-Euclidean geometry. HyperRogue includes such editors.
- Convert graphics from one geometry or size to another. This is also used in HyperRogue, where most of the hand-drawn graphics are designed for the standard HyperRogue geometry, and have to be converted when another geometry or another scale is used.

  The natural approach to convert Euclidean graphics to hyperbolic geometry is to consider the graphics to be viewed in the chosen projection. So, if our visualization uses Poincaré projection, we can convert the Poincaré projection coordinates back to coordinates in the chosen model of hyperbolic geometry; this will look just like as the original picture if the model is centered, and usually good, due to the Poincaré model being conformal. In some cases, we would rather need the shape to fill a regular polygon; then Klein-Beltrami projection is more appropriate, due to mapping regular polygons to regular polygons. Irregular polygons can be triangulated, and again, Klein-Beltrami projection composed with an affine transform applied to every triangle.

3.6 Three-dimensional geometry

Most of the methods outlined in this chapter can be used for rendering three-dimensional hyperbolic or spherical geometry (simply changing the number of

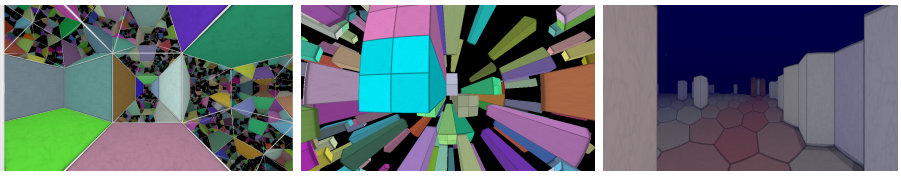

Figure 3.4: Some possible honeycombs for a visualization in $\mathbb{H}^3$.

dimensions). Also, much of it still works in anisotropic geometries, although this will be discussed in more detail in later Chapters. We list the important changes for $\mathbb{H}^3$ and $\mathbb{S}^3$.

The choice of honeycomb. The 3D analogs of tilings are usually called *honeycombs*. We will list three ways of organizing a three-dimensional hyperbolic world:

- Use a regular three-dimensional hyperbolic honeycomb. One such honeycomb is shown in Figure 3.4a; this is the {4,3,5} honeycomb, meaning that its cells are cubes, **five** of them around every edge. 30% of cells are filled with randomly colored cubes; this helps the viewers to avoid the Euclidean intuition that there should be **four** cubes around every edge – the dihedral angles of these cubes are 72°! Working with these is significantly harder than with 2D tessellations. The analogs of walkers and GRTS and significantly more complex. For games, on top of that, level design is significantly harder. Therefore, while this method is the most elegant, we suggest considering another route.

- Use the 3D analog of the binary tiling (Figure 3.4b). This approach does not have most of the issues above.

- Use a 2D tiling, and just allow the object in the game/visualization to move in the third coordinate (probably allowing only bounded movement). Figure 3.4c is a screenshot from HyperRogue using this approach. The eye is on some hyperbolic plane P, making the horizon flat; the floor is 1 absolute unit below P, making it look similar to the Poincaré disk model, and some elements of the scene are above P. Interestingly, for projections based on the general perspective, the pseudo-3D effect can be also obtained without modelling everything in $\mathbb{H}^3$, but also by changing the general perspective parameter depending on the object's altitude, which corresponds to multiplying the Minkowski hyperboloid $\mathbb{H}^2$ representation by a factor. This is explained in more detail in Chapter 4.

The choice of projection. It is possible to render $\mathbb{H}^3$ using the *Poincaré ball model* or *Klein-Beltrami ball model*, which are the direct analogs of their two-dimensional counterparts. In both cases, the whole $\mathbb{H}^3$ is rendered inside

a ball. However, for immersive visualization, most people would prefer first-person view, which simulates what an observer actually living in the modelled non-Euclidean space would see. The screenshots in Figure 3.4 use this approach.

This turns out to be very easy: we can just render the Minkowski hyperboloid coordinates using the standard 3D graphics routines. Since these standard 3D graphics routines are based on projective geometry (Section 2.2), this corresponds to the Klein-Beltrami ball model. It is worth to note that the Klein-Beltrami ball model maps lines to lines and planes to planes, which is very beneficial due to depth testing. In case of $\mathbb{S}^3$, the standard methods render only 90 degrees in front of the camera; the whole scene can be rendered by running them four times.

For even more immersion, e.g. VR, we can render stereographic images, by creating separate images for every eye. One method is *true vision*, which accurately maps rays to each eye.

Exercise 3.6.1. Prove that $\mathbb{H}^3$ rendered using the *true vision* method is perceived as stretched Beltrami-Klein ball projection. By stretched we mean that the point with Beltrami-Klein coordinates (x, y, z) will be perceived at (kx, y, z) for some k.

According to Exercise 3.6.1, the whole infinite hyperbolic plane will be perceived as a ball, which might not be beneficial for our goals. Another method of rendering is *equidistant* rendering, which renders the scene in such a way that objects appear at correct distances and angles to the observer; we need to apply a (non-linear) projection transformation E after applying Model and View and before applying Perspective. Other methods can also be used, e.g., project $\mathbb{S}^3$ or $\mathbb{H}^3$ stereographically to $\mathbb{E}^3$ and render that. In general, though, the observer will still perceive parallax when moving in the simulated space, and this parallax will be perceived as the Beltrami-Klein model again.

Raycasting. The 3D analog of the fragment shader method is called *raycasting*. The advantages and disadvantages of raycasting over polygon-based approach are the same as in the 2D case; raycasters are also generally simpler to implement for anisotropic geometries.

The idea of raycasting is that we simulate the light of ray starting from the camera position p moving in direction v until it hits a wall. For every pixel, we compute the initial p and v in similar way as in the 2D case. Let's assume that some of the faces of the cells of our honeycombs are filled with walls. In our case, v is a unit vector tangent to the (pseudo)sphere at position p, and the point of the ray after moving x units is given by the formula $p' = p\cos_K(x) + v\sin_K(x)$. We can solve an equation to find x for which the ray hits the boundary of the current tile t. If that face has a wall, we color the point accordingly. Otherwise, we compute the new tangent vector v' from the formula $v' = v\cos_K(x) - K\sin_K(p)$, and as in the 2D case, continue with $(p, v, t) := (M_{t,t'}p', M_{t,t'}v', t')$.

Chapter 4

Curvature

In this chapter, we discuss the important concept of curvature. Curvature is a concept used in two distinct, but related, ways: *extrinsic curvature*, which measures how far a curve (or surface, or higher-dimensional manifold) is from a straight line, and *intrinsic curvature*, which measures how far the intrinsic geometry of a surface (or higher-dimensional manifold) is from Euclidean geometry. We also discuss the important curves in hyperbolic geometry: equidistant curves, horocycles, and their higher-dimensional equivalents.

4.1 Euclidean and spherical curves

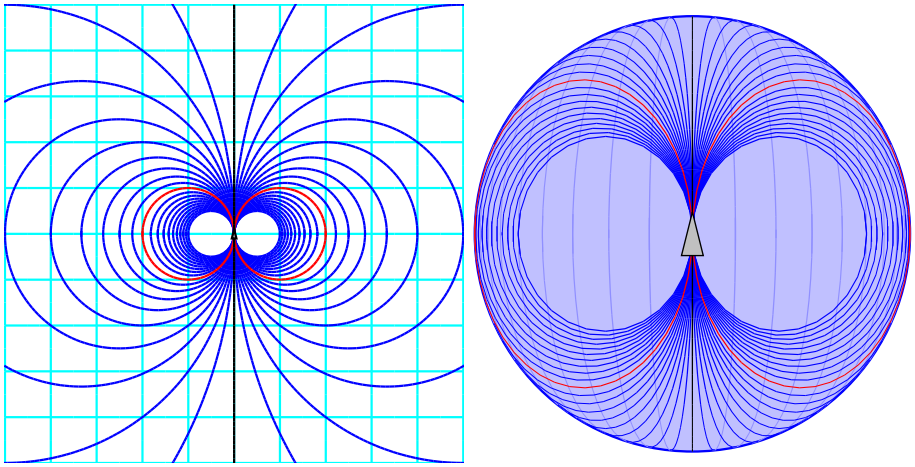

Figure 4.1: Curves of constant curvature in Euclidean (a) and spherical (b) geometry (orthogonal projection). Black is curvature 0, red is curvature $\pm$ 1.

To understand the concept of curvature, imagine that we are moving in the

Euclidean plane. We are currently at some point p, and moving at some speed v. According to Newton's laws of mechanics, if we do not accelerate, we continue moving in straight line and at constant speed. If we accelerate in the direction of movement, we move faster and faster; if we accelerate in the opposite direction, we move slower and slower. In all cases, we move in a straight line.

We can also accelerate in direction which is orthogonal to the direction of movement. Such acceleration a_2 will not change our speed, but our track will no longer be a straight line; instead, it will be a curve. The greater a_2, the greater the curvature is; however, when we move faster, we also need more sideways acceleration for the same curvature. Therefore, we define *curvature* of our track as a_2/v^2, where a_2 is the component of acceleration which is orthogonal to the direction of movement. See Figure 4.1.

Exercise 4.1.1. Show that a Euclidean circle of radius r has curvature $k = 1/r$. (Parameterize this curve, i.e., write position $c(t)$ as a function of time t, differentiate it, and apply the formula.)

Spherical geometry. Now, imagine that we are moving on a sphere embedded in three-dimensional Euclidean space $\mathbb{E}^3$. The acceleration vector will have three components: a component a_1 in the direction of movement; a component a_2 tangent to the sphere, but orthogonal to the direction of movement; and a component a_3 orthogonal (normal) to the sphere itself. The component a_3 is necessary if we do not want to avoid leaving the sphere; if $a_1 = a_2 = 0$, we are moving in a great circle, and a_3 measures the curvature of this great circle in terms of the ambient $\mathbb{E}^3$ space.

Now, imagine that we are a flat curvature living in the two-dimensional world of the sphere. We do not know that our sphere $\mathbb{S}^2$ is in fact embedded in $\mathbb{E}^3$; we can model it as two-dimensional non-Euclidean geometry, and thus, not be aware about a_3. So, from point of view of $\mathbb{S}^2$, the curvature of a line is given by a_2/v^2. In case of $a_1 = a_2 = 0$, our track is not a straight line from the point of view of $\mathbb{E}^3$, but it is a straight line from the point of view of $\mathbb{S}^2$ (and the model of non-Euclidean geometry we are using on $\mathbb{S}^2$) – from the point of view of $\mathbb{E}^3$, it is the straightest line on the surface possible, since $a_2 = 0$ and any curve had a_3. Such lines are sometimes called *geodesics* to clarify that they are not straight in the embedding space $\mathbb{E}^3$.

It is well known that, in Euclidean geometry, straight lines are the shortest paths. Geodesics are *locally shortest* lines on curved surfaces. Locally shortest means that around each point of a geodesic we can find a segment which is the shortest curve connecting its endpoints; on a sphere, if we start at $0°N0°E$ and go 359 degrees east until we reach $0°N1°W$, this path is a geodesic, but clearly not globally the shortest.

Exercise 4.1.2. Compute the curvature of a circle of radius r in spherical geometry $\mathbb{S}^2$. For simplicity, we can imagine the center of our circle is the North Pole, and thus, our circle is a parallel, at latitude $90° - r$. In $\mathbb{E}^3$, this circle would have radius $\sin(r)$ and thus curvature $1/\sin(r)$. In $\mathbb{S}^2$, the curvature is smaller; in particular, for $r = 90°$, we obtain a great circle of curvature 0.

Thus, the curves of constant curvature in $\mathbb{S}^2$ are circles and straight lines. Two properties will be important when studying the curves of constant curvature in hyperbolic goemetry:

- When we consider our sphere as a surface embedded in $\mathbb{E}^3$, curves of constant curvature are intersections of our sphere with a Euclidean plane.
- The stereographic projection maps curves of constant curvature to curves of constant curvature.

4.2 Hyperbolic curves

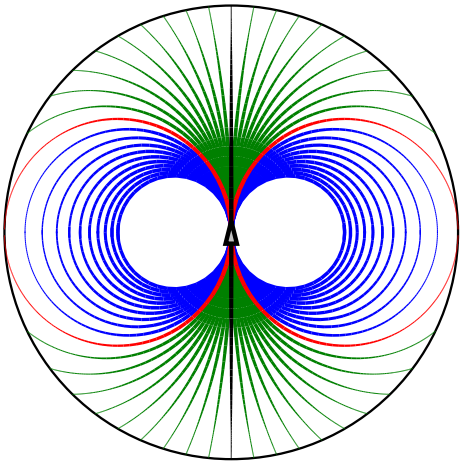

Figure 4.2: Curves of constant curvature in hyperbolic geometry, in the Poincaré disk model. Black is curvature 0, green is curvature below 1 (equidistant curve), red is curvature $\pm$ 1 (horocycle), blue is curvature > 1 (circle).

While both in spherical and Euclidean geometry, curves of constant curvature were circles and straight lines, this is no longer the case in hyperbolic geometry.

To see this, imagine the center of our body is moving in a straight line in $\mathbb{H}^2$, and our right hand is always x ($x \neq 0$) units away from this straight line (orthogonally to the direction of movement). Let c be the straight line path taken by our body, and c_1 be the path taken by our right hand.

Exercise 4.2.1. Determine the equations of these curves in the Minkowski hyperboloid model (Chapter 2). (Should be straightforward.)

The curve $c_1(t)$ is *not* a straight line. This counters our Euclidean intuitions, because a similarly defined curve in Euclidean geometry would be a straight line. However, note that, in spherical geometry, it would not be a straight line either – if c is an equator, then c_1 is a parallel, which is (usually) a non-great circle.

Such curves are called *equidistant curves*. In some sources, they are also called *hypercycles*. indexequidistant curve

By homogeneity of the hyperbolic plane, we see that equidistant curves are curves of constant curvature; also, they are not straight lines (as mentioned above) and not circles (since they do not loop). Intuitively, they arise because if our body and hand both moved in a straight line, they would diverge, due to the nature of hyperbolic geometry; so (at least) one of them must curve to recompensate this. The greater x, the more curvature is required.

Lines of constant curvature k will be equidistant curves when k is small; if k is sufficiently large, they will be circles. There is also the boundary case, when k is not small enough to produce an equidistant curve, but also not large enough to produce a circle. Such a curve is called a *horocycle*.

This completes the classification of curves of constant curvature in hyperbolic geometry – in the order of increasing curvature, we have *straight lines*, *equidistant curves*, *horocycles* and *curves*. To analyze these cases, we will note that the properties mentioned in the end of Section 4.1 still hold:

- Curves of constant curvature are intersections of the Minkowski hyperboloid with a plane. However, due to the unbounded nature of the Minkowski hyperboloid, they are no longer necessarily circles – they can be hyperbolas (for straight lines and equidistant curves) or parabolas (for horocycles).
- The Poincaré disk model maps all curves of constant curvature to curves of constant curvature. The relation between such a curve and the boundary of the disk depends on the type of the curve. Straight lines cross the boundary at right angle. Equidistant curves cross the boundary at any angle. Horocycles are tangent to the boundary. Circles do not intersect the boundary.

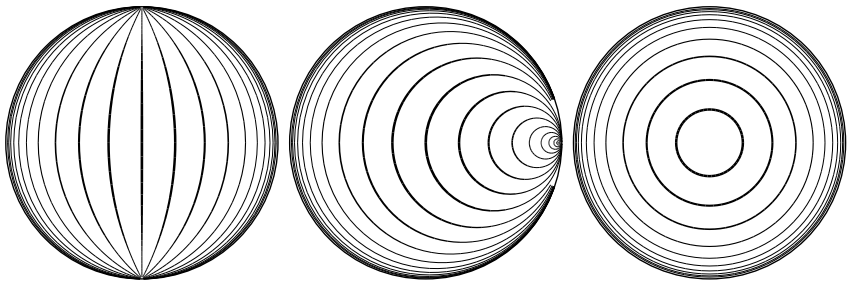

Figure 4.3: Concentric curves of constant curvature in hyperbolic geometry, in the Poincaré disk model. (a) equidistant curves, (b) horocycles, (c) circles.

Equidistant curves, horocycles, and circles in the Poincar'e disk model are shown in Figure 4.2. Another visualization is in Figure 4.3, where *concentric* curves, horocycles and circles are shown. Two curves (of constant curvature) are *concentric* is they are at a constant distance from each other; in Figure 4.3, this

distance is $\frac{1}{2}$ for two consecutive curves. As expected, concentric circles have a common center (C_0 in Figure 4.3c). Concentric horocycles also have a common center, but it is not a *material point* in the hyperbolic plane, but rather an *ideal point*, i.e., a point in infinity, represented in Poincaré disk model or Klein-Beltrami disk model as a point on the boundary of the disk. A center can also be defined for concentric equidistant curves, but this time, it is an *ultra-ideal point*; such points have no representation in the Poincaré disk model, but in Klein-Beltrami disk model, they are represented as points in $P\mathbb{R}^2$ *outside* of the disk.

Exercise 4.2.2. Show that horocycles have curvature $k = 1$. This can be done easily in Poincaré disk model: consider a horocycle passing through C_0 (as in Figure 4.2); the Poincaré disk model maps it to an Euclidean circle; compute the curvature of this Euclidean circle; and rescale this curvature to the original $\mathbb{H}^2$, by taking into account the scale factor for the Poincaré disk model close to C_0.

Exercise 4.2.3. Consider the horocycle h which passes through C_0, and is tangent to the straight line $v_1 = 0$. There is an isometry P_t of the hyperbolic plane which maps the horocycle h to itself, and moves the horocycle by t units (that is, the point p_x upwards on the horocycle such that the length of the arc from C_0 to p_x is x, is mapped to C_0). Find the matrix of this isometry. This matrix can be used to implement the *binary tiling* discussed in Chapter 3 – can you see the horocycles in Picture 3.1f? One possible solution of this Exercise is to consider the horocycle as a limit of larger and larger circles.

4.3 Surfaces and hypersurfaces

Having established the intuitions about curves in two-dimensional geometries, we can now consider surfaces in three-dimensional geometries.

For curves, if a curve c in $\mathbb{E}^2$ has curvature k, it means that c is well approximated by a circle of radius $1/k$. Instead of a circle, we could also use a parabola $y = kx^2/2$, which has the same curvature. Of course, the Euclidean space has to be translated and rotated, so that the point $(0, 0)$ is the point where we are measuring the curvature.

A similar approach can be used in $\mathbb{E}^3$, but now we approximate with a surface $z = \frac{k_1x^2+k_2y^2}{2}$. (Note that there is no k_3xy in the numerator; we can always rotate the coordinate system to eliminate it.) The values k_1 and k_2 are called *principal curvatures*, and they can be of different signs. We will now discuss two results in differential geometry, the *Theorema Egregium* and the *Gauss-Bonnet theorem*. We will include only state the intuitions, formal statements and proofs can be found in textbooks on differential geometry.

Theorema Egregium. Take a piece of paper. Usually, a piece of paper is flat, and thus, both principal curvatures are 0. We can change this, by rolling it into a cylinder of radius r. After this operation, the principal curvatures are 0 at $1/r$.

If we hold a horizontal piece of paper by two corners, in a straight line, it will probably curve itself due to gravity; however, if we curve the paper slightly, this will not happen. Generally, whatever we do (as long as the paper is still smooth), one of the curvatures at every point will be 0. Thus, we cannot smoothly curve the paper to make it into a sphere, whose both principal curvatures are $1/r$.

This was expected, because a piece of paper has intrinsic Euclidean geometry, and a sphere has intrinsic spherical geometry. If we had spherical or hyperbolic paper, we could perform a similar experiment; in all cases, the *product* of k_1 and k_2 does not change; this product is called the Gaussian curvature, $K = k_1 k_2$, and is the same Gaussian curvature as referred to in Chapter **??**.

In some sources the surface $z = \frac{x^2-y^2}{2}$ (a "pringle") is given as an example of hyperbolic geometry; this is not very accurate, because while the Gaussian curvature at $(0, 0, 0)$ is indeed negative, it is not a surface of *constant* negative curvature, and thus it does not have interesting properties of hyperbolic geometry, such as the exponential growth.

Gaussian curvature as defined above is an intrinsic property of (two-dimensional) surfaces. Higher-dimensional hyperbolic, spherical and Euclidean spaces have *constant Gaussian curvature*, which means their intrinsic Gaussian curvature is the same when measured for two-dimensional section (that is not extrinsically curved). An example of a homogeneous three-dimensional geometry which does not have constant sectional curvature is $\mathbb{H}^2 \times \mathbb{R}$, that is, $\mathbb{H}^2$ with an extra dimension added in Euclidean way. Clearly, some sections of it have curvature -1 and some have curvature 0. Note that, for three-dimensional geometries, the space of possibilities is much greater than this example suggests – in *twisted* geometries (Chapter 7.8), equidistant curves might neither converge (as in positively curved spaces) nor diverge (as in negatively curved spaces), but move in the third dimension.

The formula $K = k_1 k_2$ (called *Theorema Egregium* by Gauss) assumes Euclidean geometry. We can similarly define principal curvatures when the ambient space is not Euclidean; in this case, we have $K = k_1 k_2 + K_A$, where K_A is the curvature of ambient space.

The curves of constant curvature in $\mathbb{H}^2$ (straight lines, equidistant curves, horocycles, circles) have their two-dimensional analogs in $\mathbb{H}^3$ (planes, equidistant surfaces, horospheres, spheres). We can use the formula above to compute their Gaussian curvatures. Notably, a horosphere has Gaussian curvature $K = k_1 k_2 + K_A = 1 \cdot 1 - 1 = 0$, while spheres have positive Gaussian curvature (as expected for spheres), equidistant surfaces have Gaussian curvature between 0 and -1, and planes have Gaussian curvature -1. In some sources, a space with constant 0 curvature is called a *flat manifold*; this terminology is "Euclidean-centric", that is, it assumes that the ambient space is Euclidean. We prefer to call such a manifold a *Euclidean manifold* (and "flat" should refer to no extrinsic curvature, i.e., $k_1 = k_2 = 0$; such surfaces are also called *totally geodesic*).

Exercise 4.3.1. In $\mathbb{H}^3$, consider a surface equidistant to the plane $x_3 = 0$, passing through the point $T_3^r C_0$. How would that equidistant surface be perceived by an observer placed in point $T_3^{-p} C_0$ (according to the rules of linear perspective)?

Show that it is a (scaled) general perspective projection, and in particular, (scaled) Klein-Beltrami model for $r = 0$ and Poincaré disk model for $p = r$.

Gauss-Bonnet Theorem. Imagine an observer walking clockwise on some curve c of constant curvature k on some surface S of constant curvature K, and returning back to the starting point and orientation. If l is the length of the curve c, and A is the area of the part of S which is inside c, then we have $kl + KA = \tau$ [1]. For example, for a Euclidean circle of radius r, we have $K = 0$ and $kl + (1/r)2\pi r = 2\pi$; for a great circle on a sphere of radius 1, we have $k = 0$ and $A = \tau$ (half of the sphere), thus $0 + A = \tau$ indeed holds.

This result also generalizes to curves and surfaces of non-constant curvature – formally, we have to replace the products kl and KA by integrals, integrating the curvatures over curves and surfaces. (We need to consider counterclockwise curvature as negative.) It also generalizes to piecewise curves – a sharp clockwise bend of α degrees contributes α to the left side of Equation [1]. Intuitively, in case of such a bend, the whole curvature is concentrated in a single point. (This is a restatement of the property [8] from Section 1.8.)

We can also consider surfaces which are not smooth, for example, surfaces of polyhedra. We can draw curves on polyhedra; if a vertex of such a polyhedron appears in S_1, it contributes $\tau - \alpha$, where α is the sum of angles of faces at this vertex. For example, consider a cube of edge 2 centered in $(0, 0, 0)$, and a loop $(0, 0, 1) - -(0, 1, 1) - -(0, 1, 0) - -(1, 1, 0) - -(1, 0, 0) - -(1, 0, 1) - -(0, 0, 1)$. The loop does not change the direction when it crosses edges, but it rotates $90°$ clockwise in every face center. It contains one vertex, which contributes $\tau - 3 \cdot 90° = 90°$. The total is $3 \cdot 90° + 90° = \tau$.

Exercise 4.3.2. Use this to prove *Euler's polyhedral theorem*: a convex polyhedron with F faces, E edges and V vertices satsfies $V + F = E + 2$.

4.4 Apeirohedra

To conclude this section, we will discuss polygonal and polyhedral approximations of curves of constant curvature. A regular (Platonic) polyhedron can be obtained by taking a regular tessellation of a sphere (embedded in any geometry) and connecting its vertices with straight lines (in the ambient geometry) and replacing spherical tiles with flat faces. A similar thing can be done starting with a regular tessellation of a horosphere or equidistant surface. We use the term *apeirohedron* (*infinite polyhedron*) for the shape obtained; more specifically, *horohedron* when we start with a horosphere, and *pseudohedron* when we start with an equidistant surface. This generalizes to other dimensions (e.g., *apeirogons* for infinite polygons; *apeirochora* for the analog with three-dimensional faces) and more general tessellations (e.g., Archimedean apeirohedra). See figure 4.4 for a tessellation of $\mathbb{H}^2$ using finite polygons of various sizes and apeirogons. We could draw this tessellation on a equidistant surface, and connect its vertices with flat faces, obtaining a pseudohedron.

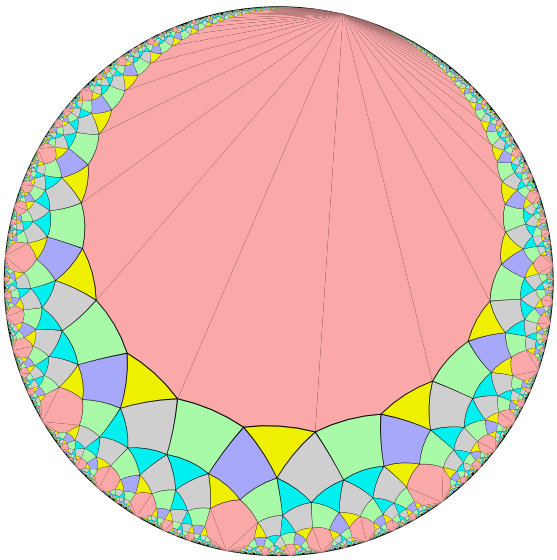

Figure 4.4: A tessellation of $\mathbb{H}^2$ using finite polygons of various sizes and apeirogons.

Chapter 5

Models and Projections

In this chapter we discuss various representations of two-dimensional spherical or hyperbolic geometry, which can be used as *projections* (visualizations) or *models* (representations, essentially, coordinate systems). For simplicity, we will focus on two-dimensional geometries, although we also consider three-dimensional geometries in the concluding section. This subject is especially well researched for spherical geometry. The surface of Earth has approximately spherical geometry, and for practical methods we need maps printed on paper, which has Euclidean geometry. Unfortunately, due to the difference in geometry, every map will have to distort the distances in some way. Cartographers have designed dozens of map projections with various advantages and drawbacks.

Depending on the intended application, we might want our map to have one of the following properties:

- *Conformal* projections map angles faithfully.
- *Equal-area* projections map areas faithfully. In this section, we will assume that *faithfully* means that area A on the projected surface maps to area A in the projection. Of course, for most practical purposes of Earth mapping, surface area A maps to projected area As^2, where s is the scale of the map; in this Chapter, we consider the scaling and the projection itself to be separate components (the projection maps A to A).
- *Equidistant* map projections map distances faithfully. It is impossible to map *all* distances faithfully; we need to specify which distances we are interested in.

These properties are generally exclusive for a (non-trivial) map. Many projections are uniquely defined by two properties: one of the properties above, and another property determining the shape (e.g., *azimuthal* or *cylindrical*).

Many projections are used by map makers. While most projections have been designed as maps from $\mathbb{S}^2$ to $\mathbb{E}^2$, they also have natural analogs which are projections to $\mathbb{H}^2$ to $\mathbb{E}^2$. In some cases, the traditional names of the projections

in spherical/hyperbolic case are different; then, we give both names, in format *spherical/hyperbolic*.

We can generalize even further, by considering arbitrary curvature on both sides: we project $\mathbb{U}^2_{K'}$ to $\mathbb{U}^2_{K''}$. Thus, for example, inhabitants of the hyperbolic plane could use a native Poincaré disk model to map their world to a hyperbolic disk, which corresponds to the case $K' < K'' < 0$. (In this chapter, we will be using the notation $'$ for elements related to the projected geometry, and $''$ for elements related to the target geometry.)

We include the images of all projections discussed in this chapter. However, we would like to emphasize that many projections have interesting effects when animated. We recommend playing with them in HyperRogue/Rogueviz, or watch our video *Hyperbolic analogs of spherical projections*.

Exercise 5.0.1. In this Chapter, for most projections, we provide the properties that a projection should have, but we do not give the formula; finding the formula of a projection is left as an exercise. (It should be straightforward for a reader understanding the models from Chapter 2 and the formulas for lengths and areas of segments of curves of constant curvature, which can be easily found by differentiation or integration.)

5.1 Azimuthal projections

Azimuthal projections are projections that map straight lines going through point X (usually C'_0) into straight lines going through point X' (usually C''_0). Any point x' can be written in *native polar coordinates* (r, ϕ), that is, the point x' is in distance r from C_0, in direction given by the angle ϕ. An azimuthal projection maps it to mative polar coordinates $(r, f(\phi))$, where f is some function, depending on the goal of our projection; in some cases f may also depend on ϕ. Using the notation from Chapter 2, we can write $x' = R^\phi_{1,2} T^r_1 C'_0$ and $x'' = R^\phi_{1,2} T^{f(r)}_1 C''_0$.

Azimuthal equidistant projection The simplest azimuthal projection is the *azimuthal equidistant projection* (Figure 5.2), where f is identity. Thus, $x' R^\phi_{1,2} T^r_1 C'_0$ is mapped to $x'' = R^\phi_{1,2} T^r_1 C''_0$; in other words, native polar coordinates (r, ϕ) are mapped to native polar coordinates (r, ϕ). Azimuthal equidistant projection is considered very natural. It is strongly linked to the polar coordinate model, which is preferred by many researchers working in the Hyperbolic Random Graph model. Furthermore, it can be seen in perspective projections of product geometries such as $\mathbb{H}^2 \times \mathbb{R}$ or $\mathbb{S}^2 \times \mathbb{R}$.

General perspective projection The general perspective projection has already been discussed in Section 3.1. To recall: the point $(x, y, z) \in \mathbb{U}^2_K$ is mapped to $(\frac{x}{z+d}, \frac{y}{z+d})$, where d is the parameter of the projection. See Figure 5.2 for three perspective projections of the sphere $\mathbb{S}^2$, and one new perspective projection of the Minkowski hyperboloid $\mathbb{H}^2$.

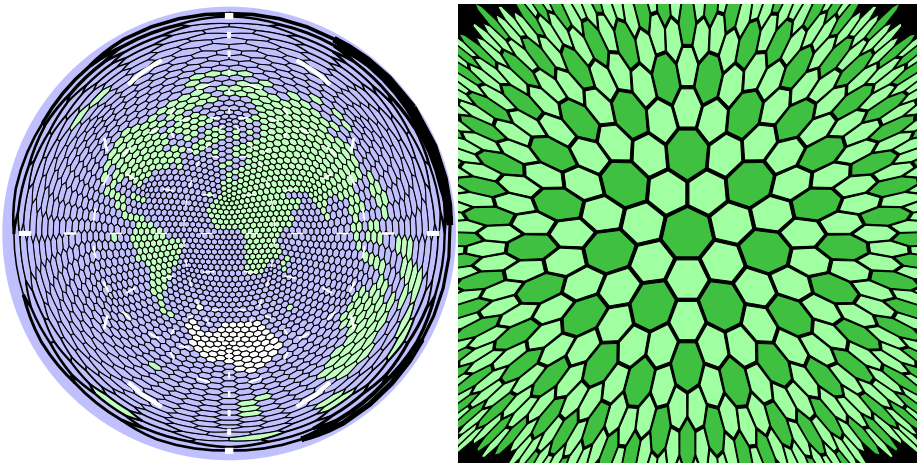

Figure 5.1: Earth map and bitruncated {7,3} in azimuthal equidistant projection.

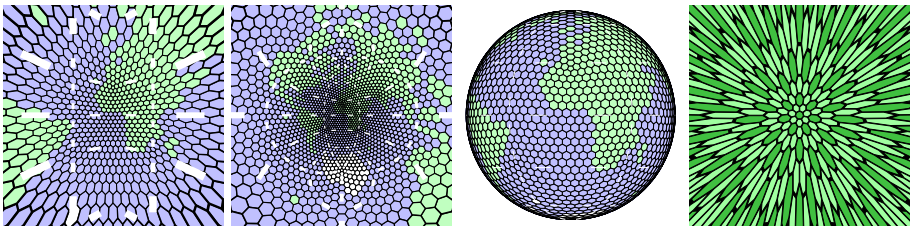

Figure 5.2: Earth map in gnomonic, stereographic, and orthographic projection; bitruncated {7,3} in Gans model.

Gnomonic projection/Beltrami-Klein disk model Case $d = 0$. The Beltrami-Klein model maps straight lines to straight lines, and represents ideal and ultra-ideal points.

Stereographic projection/Poincaré disk model Cases $d = 1$ and $d = -1$ yield conformal models; they will be discussed in Section 5.4.

Orthogonal projection/Gans model The orthogonal projection maps point (x, y, z) to (x, y); it can be considered a special case of general perspective projection where $d = \infty$ (of course we would also need to zoom in the projection obtained). In this book, we tend to use the orthogonal projection for showing spherical geometry, although only the stereographic projection shows the entire sphere (mapped to the infinite plane, though).

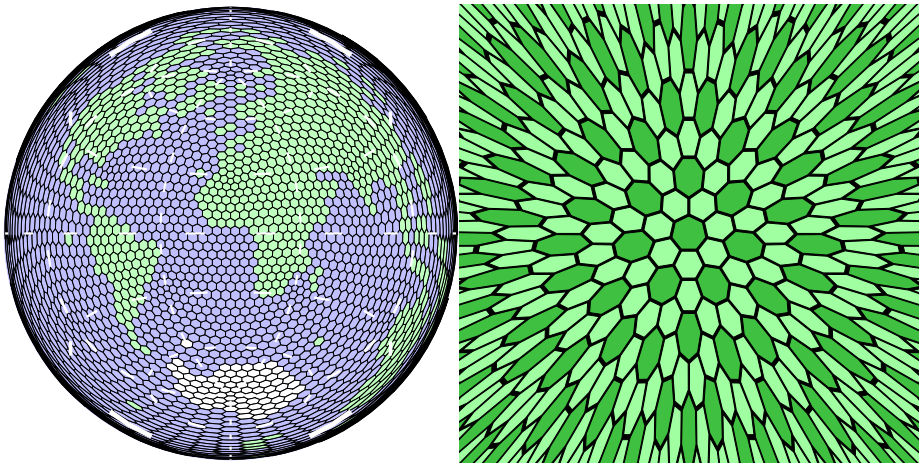

Figure 5.3: Earth map and bitruncated {7,3} in azimuthal equal-area projection.

Azimuthal equal-area projection See Figure 5.3. Equal-area projections are preferred when the mapmaker wants to avoid suggesting that some areas are bigger than they actually are.

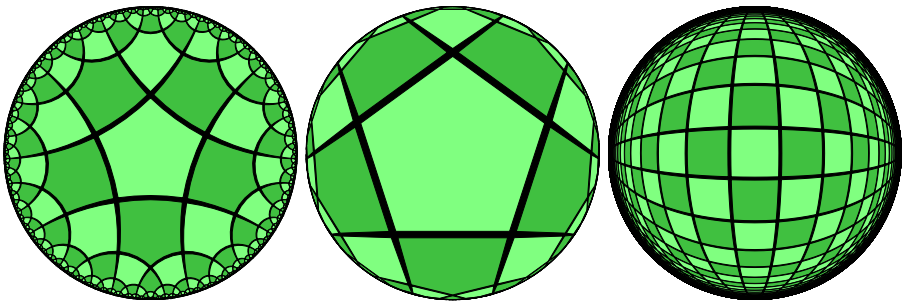

Figure 5.4: {5,4} in Poincaré disk model and Klein-Beltrami model; Euclidean square grid in fish-eye projection.

Fish-eye projection People watching the visualizations of hyperbolic geometry in Poincaré disk model often assume that this is normal Euclidean space, but using some kind of *fish-eye projection*, or possibly a sphere. Thus, we include the fish-eye projection as yet another example of azimuthal projection, and also to show the differences: in each case, angles and straight lines work differently, and also the tessellations are obviously different (Figure 5.4).

The fish-eye projection is a projection of the Euclidean plane, where the plane is projected gnomonically to a hemisphere, and then this hemisphere is projected orthogonally back to the Euclidean plane. We include the fish-eye

projection for completeness.

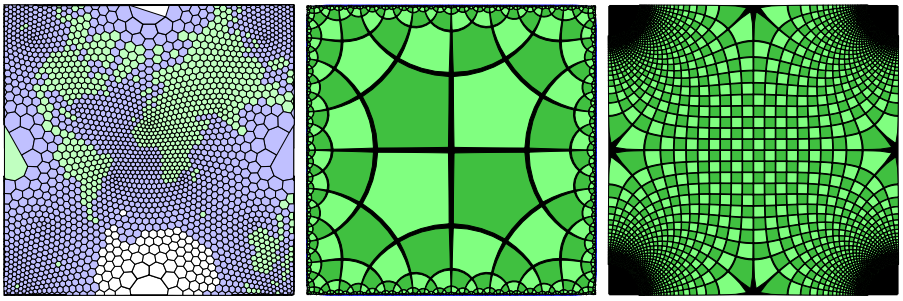

Figure 5.5: Conformal square projection. Earth, hyperbolic plane, and Euclidean plane.

Some 2D video game designers have tried to use fish-eye projection so that the player has one single view which shows both the immediate surroundings of the player character and the whole map. Unfortunately, the approach does not work look very nice, because fish-eye projection is highly not conformal. Changing the map to hyperbolic geometry (as in HyperRogue) is a good way to have a nicely-looking fish-eye-like perspective, although it does not really solve the issue of showing the whole map. A conformal projection could be obtained in somewhat complex way, by projecting the map conformally to a sphere (stereographic projection) and then projecting the sphere to a square (Peirce's quincuncial projection) (Figure 5.5).

Two-point azimuthal projection As mentioned above, and as many examples in this chapter show, two properties are usually needed to define a projection. We can take *azimuthal* twice: that is, a projection azimuthal in *two central points*. We have two centers O'_1 and O'_2, and every point A' is mapped so that both angles $A'O'_1O'_2$ and $A''O''_1O''_2$ are correct. This projection turns out to be simply stretched Beltrami-Klein model; cf. Exercise 3.6.1.

5.2 Cylindrical projections

Azimuthal projections are based on the polar coordinates. Cylindrical projections, on the other hand, are based on the spherical coordinates, latitude ϕ and longitude λ. The point x with coordinates (ϕ, λ) can be written using the notation from Chapter 2 as $x = T_1^\lambda T_2^\phi C_0$. The same formula can be used in Euclidean geometry (yielding Cartesian coordinates) and in hyperbolic geometry (yielding so called *Lobachevsky coordinates*). In general, we will call these coordinates *native equidistant coordinates*. Coordinate ϕ of x determines the (signed) distance from some straight line, which we will call *equator*; coordinate λ measures the (signed) distance of the projection of x on the equator from C_0. Lines of constant ϕ are equidistant to the equator, also called *parallel*; lines of

constant λ are straight lines orthogonal to the equator, also called *meridians*. A cylindrical projection maps a point with native equidistant coordinates (ϕ, λ) to a point with native cylindrical coordinates $(f(\phi), \lambda)$, where f is some function, depending on the goal of our projection.

Cylindrical coordinates are useful in hyperbolic geometry when we want to avoid numerical precision errors, and we know that all points we are interested in are close to some line (which will be chosen to be the equator).

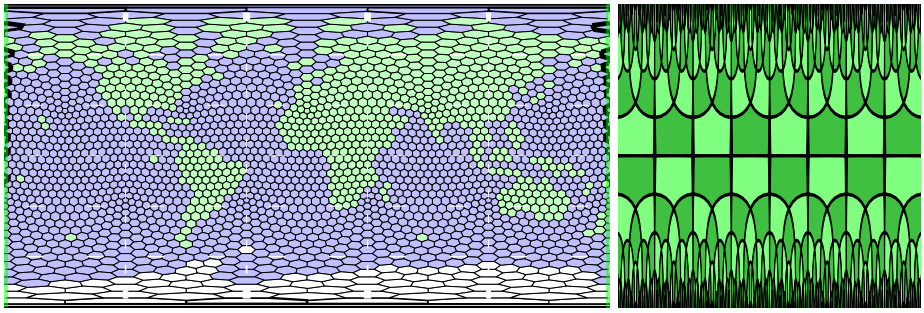

Figure 5.6: Earth map and {5,4} in cylindrical equidistant projection.

Cylindrical equidistant projection (aka equirectangular). In this projection, we simply take f to be the identity (Figure 5.6. This projection is equidistant on the equator and all meridians. By taking $f(\phi) = k\phi$ we can also make a variant that is equidistant on parallel ϕ and all meridians.

Cylindrical conformal projection, aka Mercator projection/band model. This projection will be discussed in Section 5.4.

Cylindrical equal-area projection. See Figure 5.7. Generally, equal-area projections can be stretched vertically without losing the property of being equal-area. In Figure 5.7a, the sphere is projected to a square; cylindrical equal-area projections with various stretch factors have their own names.

Central cylindrical projection. The central cylindrical projection is obtained by projecting the sphere from the central point $(0, 0, 0)$ to the cylinder $\{(x, y, z) : x^2 + y^2 = 1\}$, and then unrolling this cylinder. (Projecting surface S' to S'' from point p means that a point $x \in S'$ maps to $x'' \in S''$ if and only if p, x', and x'' are colinear.) This is a cylindrical projection with meridians mapped according to the gnomonic projection. Mercator projection is sometimes explained this way, but they are in fact different projections. We obtain the hyperbolic analog by mapping the meridians according to the Klein-Beltrami projection.

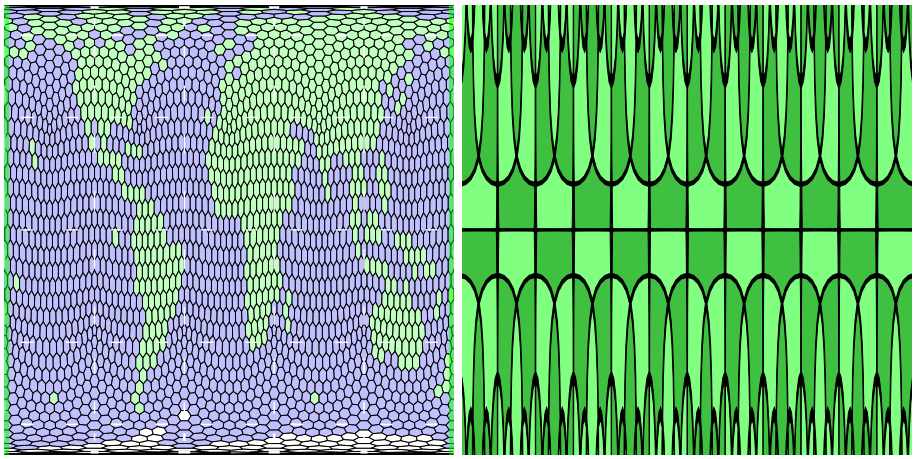

Figure 5.7: Earth map and {5,4} in cylindrical equal-area projection.

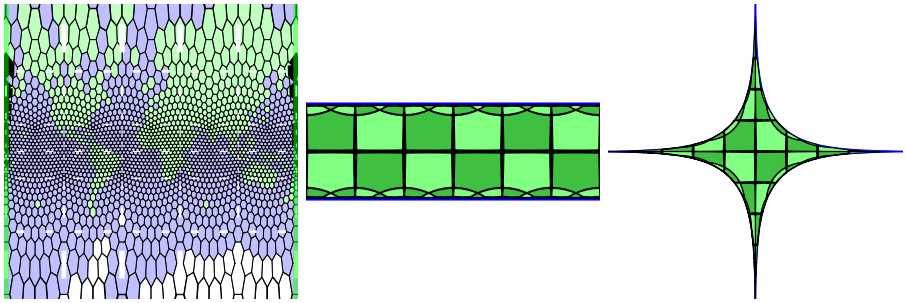

Figure 5.8: Earth map and {5,4} in central cylindrical projection; {5,4} in two-equator cylindrical projection.

Two-equator cylindrical projection. By analogy to the *two-point azimuthal* projection, we can also have a projection that is cylindrical with respect to two (orthogonal) equators. Such a projection is shown in Figure 5.8c. The coordinates of this projection are called *axial coordinates*, and they are one possible Cartesian-like system of coordinates for the hyperbolic plane.

5.3 Horocyclic projections

In Figure 4.3 we have seen concentric curves in $\mathbb{H}^2$ of three types: equidistant curves, circles, and horocycles. Equidistant curves naturally correspond to cylindrical projections, and circles naturally correspond to azimuthal projections. In this section, we will discuss projections based on horocycles. Horocycles have no natural analog in spherical geometry, but they do have a natural analog in Euclidean geometry: straight lines. Thus, horocyclic projections (usually) map

concentric horocycles to horizontal straight lines, and lines orthogonal to them to vertical straight lines.

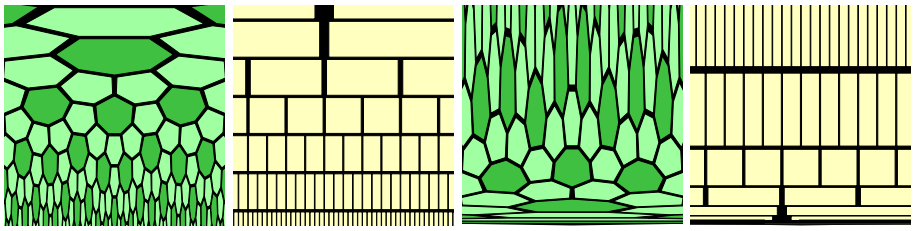

Figure 5.9: {5,4} and the binary tiling in horocyclic equidistant projection, and then in horocyclic equal-area projection.

Horocyclic equidistant projection. See Figure 5.9ab. This projection will be very useful later for introducing Solv geometry (Chapter 7.9).

Horocyclic equal-area projection. See Figure 5.9cd. This projection is included mostly for completeness.

Horocyclic conformal projection. As the Reader might have anticipated, based on azimuthal and cylindrical conformal projections, we leave the description to the chapter on conformal projections.

5.4 Conformal projections

Conformal projections have important and beautiful connections to complex analysis and other areas of mathematics. We will explore these connections in this section; however, in the rest of this book, these connections turn out to be not of much use. Therefore, we will only explain the basic ideas related to complex analysis informally. We refer readers interested in knowing more about complex analysis to other sources.

A projection from A' to A'' will be denoted as $f : A' \nrightarrow A''$; the symbol $\nrightarrow$ means that not every $x \in A'$ needs to mapped to a point in A'', and the projection can be also *multi-valued*, that is, $x \in A'$ can be in fact mapped to many A''.

A projection $f : A' \nrightarrow A''$ is *differentiable* at $a \in A'$ iff, for small x, we have $f(a+x) = f(a) + Tx + \epsilon(x)$, where T is some linear transformation, and $\epsilon(x)$ is small for small x (technically, $\frac{|\epsilon(x)|}{|x|} \to 0$ as $x \to 0$). This T is called a *derivative* of f in a, and called $f'(a)$. A projection is *conformal* iff T is a similarity, that is, a composition of isometry and scaling $x \mapsto kx$ for some scaling factor k. When A' and A'' are curves, surfaces, or higher-dimensional manifolds, we need T to be a scaling transformation only on the tangent space.

As we already know, isometries conserve angles and distances; similarities do not conserve distances, but they conserve angles and shapes. Thus, conformal projections conserve small shapes (intuitively, shapes so small that they are not affected by $\epsilon(x)$).

To study two-dimensional conformal mappings, it is useful to think of $\mathbb{R}^2$ as of complex numbers $\mathbb{C}$. We think of the point (a, b) as of numbers of form $z = a + bi$, where $a, b \in \mathbb{R}$, and i is an imaginary number such that $i^2 = -1$. The real numbers a and b are referred to as $\Re(z)$ (the real part) and $\Im(z)$ (the imaginary part), respectively. Operations such as addition, subtraction, multiplication, and division, and function exp are naturally extended to complex numbers. The function exp can then be used to define other functions, such as trigonometric or hyperbolic functions, log or square root; however, it is good to think of square root and logarithm as multi-valued functions, since $x^2 = (-x)^2$ (for example, i and $-i$ are both square roots of -1) and $\exp(k + 2\pi i) = \exp(k)$.

Thus, a projection $f_r : \mathbb{R}^2 \to \mathbb{R}^2$ can be also thought of as a projection $f_c : \mathbb{C} \to bbC$. However, f_c being differentiable is a stronger condition than f_r being differentiable – in the case of $\mathbb{R}^2$, T has to be a linear transformation, and in the case of $\mathbb{C}$, we need $T(z) = tz$ for some complex number z. For every complex number $t \neq 0$, multiplication $z \mapsto tz$ corresponds to an orientation-preserving scaling of $\mathbb{R}^2$; also, any orientation-preserving scaling of $\mathbb{R}^2$ corresponds to a multiplication by some complex number. Therefore, orientation-preserving conformal projections correspond exactly to projections which are differentiable in the complex sense. A conformal projection that is not orientation-preserving can be made (locally) orientation-preserving by composing it with mirror image, or complex conjugate, $\hbar a + bi = a - bi$.

Functions on complex numbers defined above (arithmetic operations, exp, log) and functions obtained by composing them are differentiable in the complex sense, so generally we can write any formula using them and get a conformal projection – although such a projection might not be very nice due to being not injective or multi-valued. Möbius transformations are functions of form $f(z) = \frac{a+bz}{c+dz}$, where $ad - bc \neq 0$. They have a nice property that they are bijections on $\hat{\mathbb{C}}$, where $\mathbb{C}$ is the set of complex numbers extended with infinity ∞. Möbius transformations map curves of constant curvature to curves of constant curvature.

Stereographic projection/Poincaré disk model. The stereographic projection is the special case of general perspective projection for $d = 1$. It is conformal. When used as a projection from $\mathbb{H}^2$ to $\mathbb{E}^2$, it is called the Poincaré disk model, and it is a very important model of hyperbolic geometry; many people are thinking of hyperbolic geometry in terms of Poincaré disk model. In the sequel, let p be the point of $\mathbb{U}^2_K$ projected stereographically to $\hat{\mathbb{C}}$. If $K = 0$, p can be any number in $\mathbb{C}$. If $K > 0$, the point $-C_0$ is mapped to ∞. If $K = -1$, we get $|p| < 1$. For $K = -1$, points p such that $|p| = 1$ represent *ideal points.*

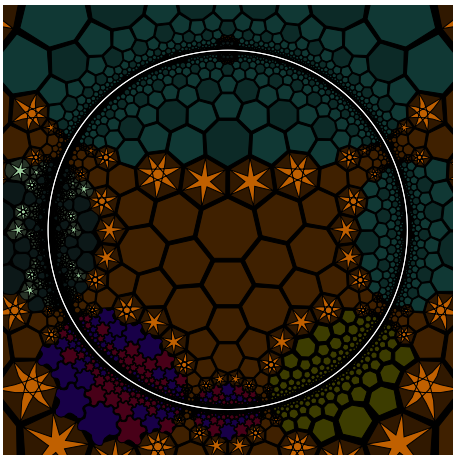

Figure 5.10: HyperRogue map, *Crossroads II*, in Poincaré disk model and inverted Poincaré disk model, on the same picture.

Inverted Poincaré disk model. For $d = -1$ we get the inverted Poincaré disk model, that is related to the Poincaré disk model by $z = 1/\bar{p}$. The mapping $z = 1/\bar{z}$ is an important operation in geometry, called *inversion* in the unit circle. Both Poincaré disk model and inverted Poincaré disk model are shown on one picture in Figure 5.10.

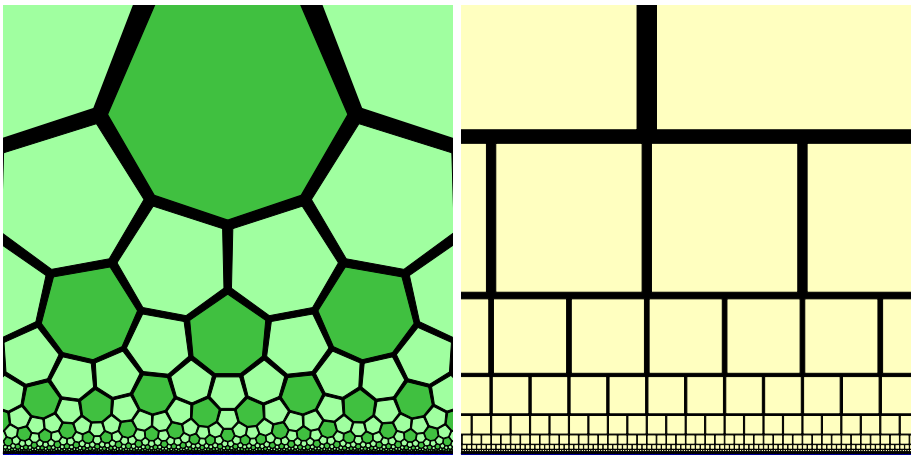

Figure 5.11: {5,4} and the binary tiling in Poincaré upper half-plane model.

Poincaré's upper half-plane model. For $\mathbb{H}^2$, we get the upper half-plane model by applying the Möbius transformation $h = \frac{1}{p+i} - \frac{i}{2}$ to p (Figure 5.11). Geometrically, this is an inversion in circle centered on the boundary on the

Poincaré disk. This Möbius transformation maps points in the disk ($|p| < 1$) to the upper half-plane ($h = a + bi$, where $b > 0$). This model is horocyclic: horocycles centered at the ideal point $h = \infty$ ($p = -i$) are represented as straight horizontal lines. One nice property of this model is that the scaling factor is very simple – the scale at point $h = a + bi$ equals b (that is, a small circle in $\mathbb{H}^2$ of radius r centered at h will be mapped to a circle of radius br). For visualizations, it is often useful to rotate this model; we will call such rotation simply *half-plane model*, possibly with the rotation used, e.g., right half-plane model.

Exercise 5.4.1. Show that the half-plane model is indeed horocyclic.

All three models above map lines of constant curvature to lines of constant curvature in $\hat{\mathbb{C}}$. One surprising application of this is finding Voronoi diagrams. Consider a set of points $P \in \mathbb{U}^2_K$ (usually finite). We assign every point of $\mathbb{U}^2_K$ to the closest point $x \in P$; thus, $\mathbb{U}^2_K$ is split into a tessellation of (possibly unbounded) polygons, called Voronoi cells. We can compute a Voronoi diagram in $\mathbb{U}^2_K$ by mapping all the points P stereographically to $\mathbb{R}^2$, using a Euclidean algorithm to find Euclidean Voronoi cells. The cells of the Voronoi diagram of the original P are not the same as the Euclidean Voronoi cells, because distances are defined differently; however, the diagram will have the same structure – generally, the vertex where Voronoi cells of points x_1, x_2, x_3 meet will be a center of circle containing all three points, and we just have to replace the center of Euclidean circle with the center of native $\mathbb{U}^2_K$ circle. The fact that stereographic projection maps circles to circles is important for this trick.

In all three models above, isometries of $\mathbb{U}^2_K$ are Möbius transformations, which gives us another possible representation of isometries, as an alternative to linear transformations (Möbius transformations are also easy to compose). In particular, the isometries of upper half-plane model are Möbius transformations where $a, b, c, d \in \mathbb{R}$. This will be discussed in Chapter 7.5.

Hemisphere model. Hemisphere model is a stereographic projection from $\mathbb{H}^2$ to $\mathbb{S}^2$. In terms of flat projections, we stereographically project $\mathbb{H}^2$ to $\mathbb{R}^2$, obtaining Poincaré disk of radius 1, and then we project this disk to $\mathbb{S}^2$, obtaining a hemisphere (Figure 5.13). This model highlights beautiful relations between other popular models: as already mentioned, we can obtain Poincaré disk model by projecting stereographically, but we can also obtain Klein-Beltrami disk model by orthographic projection, Gans model by gnomonic projection (in general, general perspective(d) applied to the hemisphere model equals general perspective($1/d$) applied to the hyperboloid model), and the upper half-plane model by rotating the hemisphere before the stereographic projection. This concept is presented very nicely in the video *Illuminating hyperbolic geometry* by Henry Segerman and Saul Schleimer.

We can also do the opposite, that is, a stereographic projection from $\mathbb{S}^2$ to $\mathbb{H}^2$. As seen in Figure 5.13, the "northern" hemisphere maps directly to $\mathbb{H}^2$. The projection also makes sense for the "southern" hemisphere. Recall that $\mathbb{H}^2$ was defined as $\{x : g(x, x) = -1, x_3 > 0\}$; we can also consider the other sheet

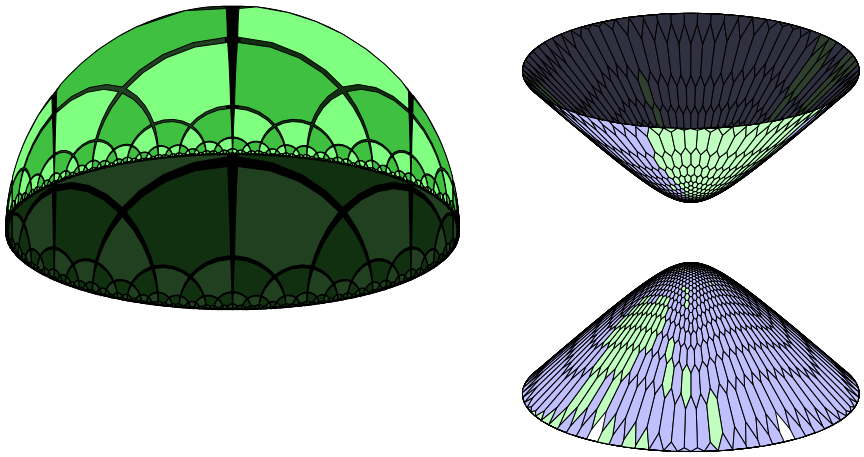

Figure 5.12: {5,4} in hemisphere model, and Earth map in Minkowski hyperboloid model.

of the hyperboloid, obtaining $\pm\mathbb{H}^2 = \mathbb{H}^2 \cup -\mathbb{H}^2 = \{x : g(x,x) = -1\}$. The stereographic projection thus maps the southern hemisphere to $-\mathbb{H}^2$, and the whole sphere to $\pm\mathbb{H}^2$.

The same phenomenon also happens for other conformal projections: the points outside the model represent $-\mathbb{H}^2$. Some artists consider $P\mathbb{H}^2$, that is, $x \in \mathbb{H}^2$ and $-x \in \mathbb{H}^2$ are considered alternative representations of the same point. Rendering both representations yields a picture where the boundary of the disk is a "mirror" (that applies circle inversion). This representation was used in Figure 5.10.

Another way of explaining this is achieved by comparing the respective distance from the boundary of the same point in Beltrami-Klein disk model and Poincaré disk model.

Exercise 5.4.2. Take $x \in \mathbb{H}^2$ which maps to point k in the Beltrami-Klein model that is d_k units from the boundary of the disk, and to the point p in the Poincaré disk model, which is d_p units from the boundary of the disk. The value of d_c is close to $\sqrt{d_k}$. Compute d_p as a function of d_k.

Asymptotically, d_p is on the order of $\sqrt{d_k}$ as $d_k \to 0$; the same asymptotics will be true for other conformal models. For very small d_k, $\sqrt{d_k}$ will be significantly larger than d_k, which, in some situations, makes Poincaré disk model numerically more precise than Beltrami-Klein or hyperboloid model (Chapter 7.5). On the other hand, if d_k is negative, $\sqrt{d_k}$ has no real value, while if d_k is positive, there are two values of d_p such that $d_p^2 = d_k$ – one positive, one negative. Thus, while in the Beltrami-Klein model the points outside ($d_k < 0$) represent ultra-ideal points, in conformal models ($d_p < 0$) they are just other images of the symmetric point with $d_p > 0$.

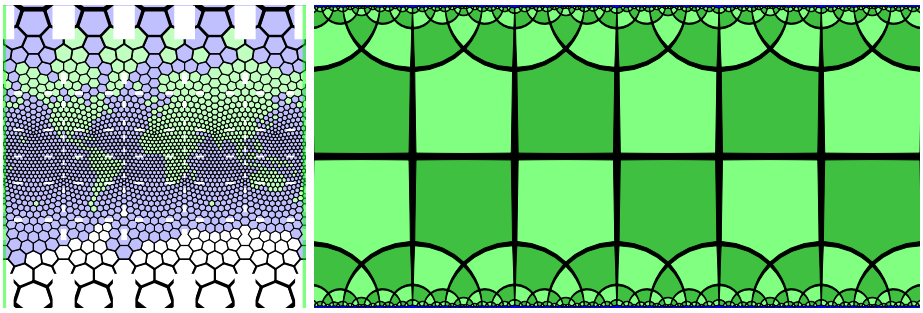

Figure 5.13: Earth in Mercator projection, and {5,4} in band model.

Mercator projection/band model. To obtain the band model, take the half-plane model h and apply the complex logarithm to it: $B = \log h$. We obtain a conformal projection which is cylindrical. It is a hyperbolic analog of the cylindrical conformal projection of the sphere, commonly known as the *Mercator projection*, a very popular projection of the Earth (that is often criticized due to its lack of the equal-area property). The band model has recently gained popularity among artists. It should be probably called the *band projection* (or just Mercator projection), not a model, although as we will see later, it does get some use as an model too.

Exercise 5.4.3. Show that the band model is indeed cylindrical.

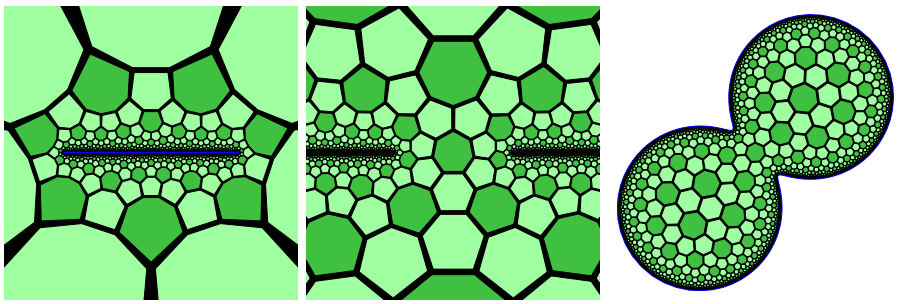

Figure 5.14: Bitruncated {7,3} in Joukowsky, inverted Joukowsky, and dual focus projection.

Applying complex functions. Here we show a few more interesting projections that can be obtained by applying complex functions to the conformal models presented so far. Applying the transformation $j = p+1/p$ to the Poincaré disk model yields *Joukowsky projection*, which maps $\mathbb{H}^2$ to all complex numbers except $\{x \in \mathbb{C} : \Im(x) = 0, |\Re(x)| \leq 1\}$. Applying inversion to it, $j^{-}1 = \frac{1}{p+1/p}$, yields *inverted Joukowsky projection*, which maps $\mathbb{H}^2$ to all complex numbers except $\{x \in \mathbb{C} : \Im(x) = 0, |\Re(x)| \geq 1\}$. On the other hand, $f = \frac{1}{p+\alpha/p}$, produces

a *dual focus projection*, which is a conformal projection of $\mathbb{H}^2$ which focuses on two points; for $\alpha = 0$ it boils down to Poincar'e disk model, and for $\alpha > 0$, as the name suggests, it can be used when we want a projection focusing on two distinct points of the hyperbolic plane (the bigger the α, the bigger the distance of two points).

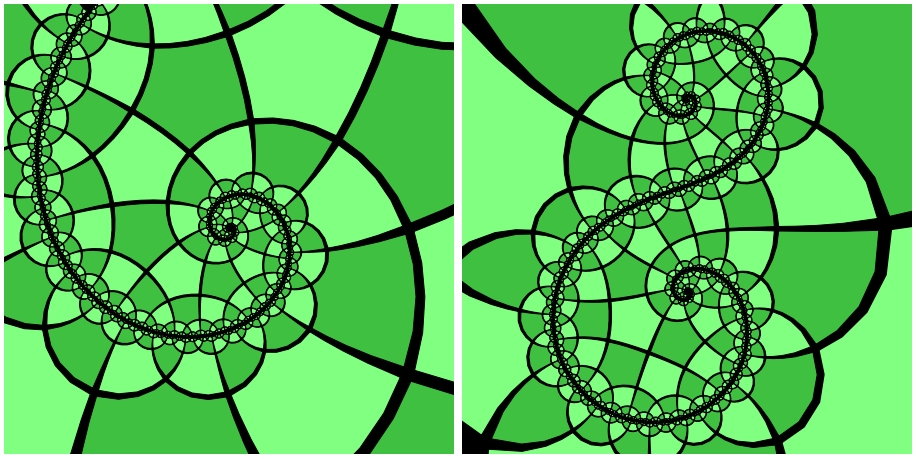

Figure 5.15: {5,4} in spiral projection.

By rotating the band model in such a way that the lower edge of the band is $2\pi i$ below the upper edge of the band, and then applying the complex exponential function to the result, we obtain a nicely looking spiral projection, which we can then modify by applying a Möbius transformation to it (Figure 5.15). These projections have artistic uses.

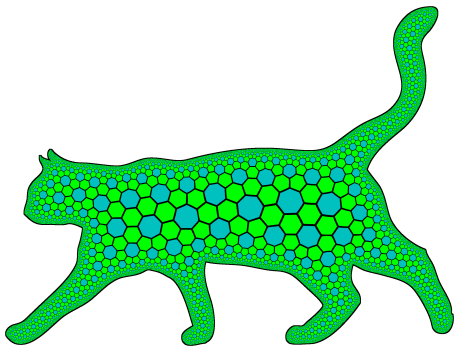

Figure 5.16: Cat model.

Arbitrary shapes. The Riemann mapping theorem says that the hyperbolic plane $\mathbb{H}^2$ can be mapped conformally to any shape $S \subseteq \mathbb{R}^2$ which is bounded,

connected, and has no holes. An example is shown in Figure 5.16. This figure is created using the program *Newconformist*, which finds a conformal mapping from the band model to any shape.

5.5 Other projections

Hundreds of map projections are used in cartography, although some of them are not really applicable for visualizing hyperbolic geometry – for example, they take into account the continents on our planet, or the fact that Earth is not perfeclty spherical. In this section, we will list some more projections that we consider interesting for the purpose of visualizing hyperbolic geometry.

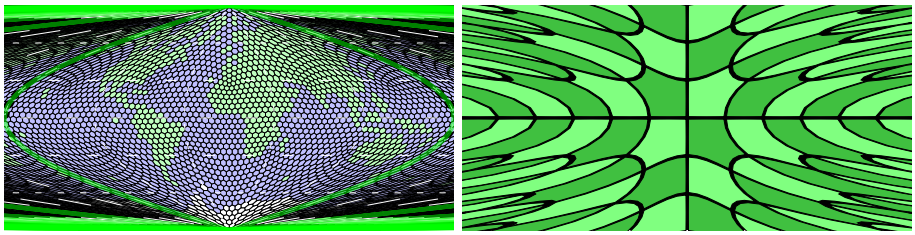

Figure 5.17: Earth map and {5,4} in cosinusoidal projection.

Cosinusoidal projection This projection is obtained by mapping a meridian (equidistantly) to a vertical straight line (orthogonal to the equator), and then, mapping the parallels (also equidistantly) to horizontal straight lines (in $\mathbb{E}^2$; more generally, equidistants to the equator). This is an equal-area projection that maps other meridians to cosinusoids. This is called *sinusoidal projection*, although the name *cosinusoidal projection* is better for our purposes, given that its hyperbolic analog maps meridians to hyperbolic cosinusoids, not hyperbolic sinusoids. While most maps of Earth in this projection show every point once, in Figure 5.17, we continue the projection further to left and right.

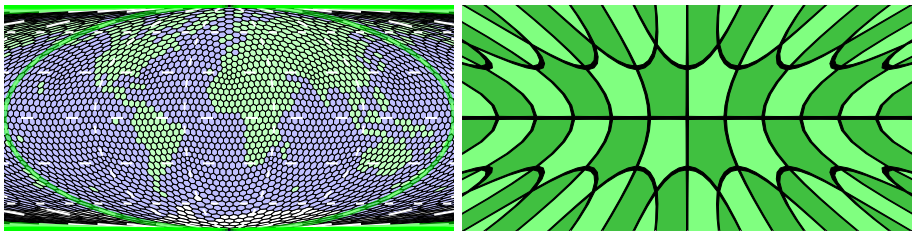

Figure 5.18: Earth map and {5,4} in Mollweide projection.

Mollweide projection This elegant projection is somewhat similar to the cosinusoidal projection. It also maps a meridian to a vertical straight line (orthogonal to the equator), and then, maps the parallels to equidistants to the equator. However, the two mappings are no longer equidistant: the second mapping is proportional (the scale is constant only for the given latitude) and designed so that the meridians are projected to half-ellipses, and the first mapping is chosen so that the whole projection remains equal-area. For the hyperbolic analog, we project the meridians to half-hyperbolas (Figure 5.18).

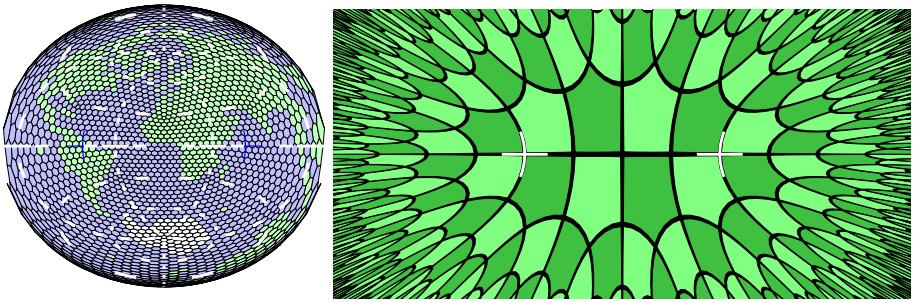

Figure 5.19: Earth map and {5,4} in cosinusoidal projection.

Two-point equidistant projection This projection faithfully maps the distances from two points O_1 and O_2. It maps the neighborhoods of O_1 and O_2 faithfully, but is warped (in an equidistant way) further away from these points. See Figure 5.19.

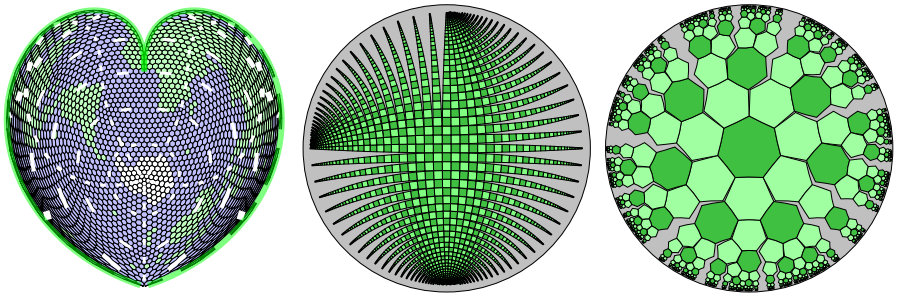

Figure 5.20: Earth map in Werner projection; a net of Euclidean square grid and bitruncated {7,3} in hyperbolic geometry.

Interrupted projections indexinterrupted projection Interrupted projections of $\mathbb{S}^2$ rip the sphere apart, to show the difference between spherical and Euclidean geometry. One example of interrupted projection is obtained by mapping the sphere to a Platonic polyhedron; we will be using an icosahedron as

an example. We project each spherical triangle to an Euclidean triangle, for example, by gnomonic or conformal projection. Then, we unroll the icosahedron into a net, thus obtaining a projection of $\mathbb{S}^2$ to a plane. This projections will have interruptions in the places where we cut the icosahedron.

Another example of interrupted projection is Werner projection (Figure 5.20a). In this projection, a circle of radius r around C_0' is mapped to a circle of radius r around C_0''. Every circle is mapped equidistantly (one half-line starting in C_0' is mapped to a half-line starting in C_0''. In the case of $\mathbb{S}^2 \nrightarrow \mathbb{E}^2$, the spherical circle will have circumference $\tau \sin(r)$ while the Euclidean circle will have radius τr; thus, the spherical circle will have to be interrupted, producing a heart-shaped projection.

Interrupted projections are not very useful for mapping from $\mathbb{H}^2$ to $\mathbb{E}^2$, because what was an interruption when $K' > K''$ becomes an overlap when $K'' > K'$, and overlaps are too confusing for visualization. However, the idea still works when $K' > K''$, for example, when projecting $\mathbb{H}^2$ to $\mathbb{E}^2$. Figure 5.20b visualizes such a projection from $\mathbb{E}^2$ to $\mathbb{H}^2$, and 5.20c visualizes such a projection from $\mathbb{H}^2$ to hyperbolic geometry of lower curvature. These projections are essentially nets of apeirohedra: a regular horohedron and a bitruncated regular pseudohedron, respectively (see Section 4.4). In RogueViz, such interrupted or overlapping views, based on regular tessellations of $\mathbb{U}^d_K$, can be obtained by using the *fake geometry* feature in *experiments with geometry*.

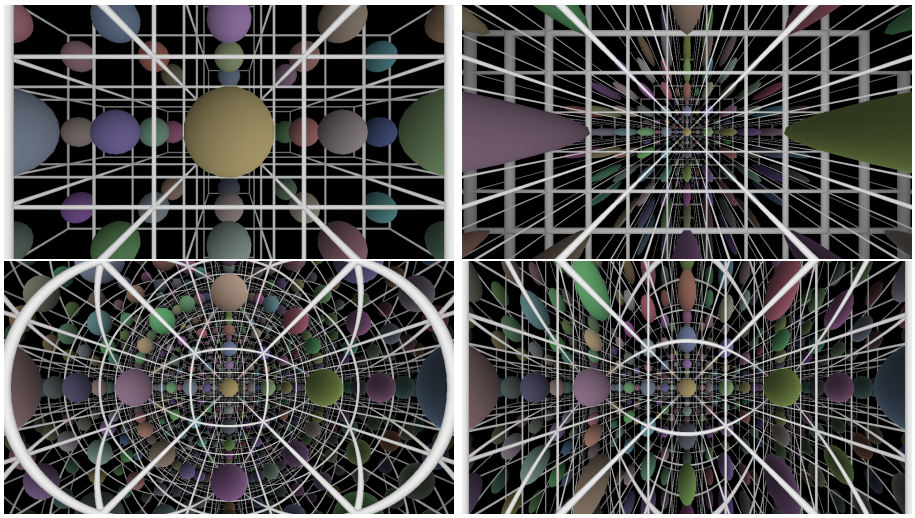

Figure 5.21: Euclidean cube tiling with spheres. Orthographic projection (FOV 90°), orthographic projection (FOV 170°), stereographic projection (FOV 270°), Panini projection (FOV 270°).

Panini projection This projection from $\mathbb{S}^2$ to $\mathbb{E}^2$ is designed not for mapping Earth, but rather an alternative perspective projection of three-dimensional

space. For simplicity, assume that our three-dimensional space is Euclidean. Recall that linear perspective maps point (x, y, z) to screen coordinates $(\frac{x}{z}, \frac{y}{z})$. Suppose the screen uses coordinates $[-x_m, x_m] \times [-y_m, y_m]$; thus, our horizontal field of vision spans from the point $(-x_m, 0, 1)$ to $x_m, 0, 1$, that is, angle $2 \arctan(x_m) < 180°$. However, the field of vision perceived by humans is a bit higher than 180°, due to having two eyes and being able to turn their head slightly. Furthermore, while angles slightly smaller than 180° are technically allowed, the distortion is substantial (Figure 5.21ab). For computer visualizations of non-Euclidean space, we might also want higher field-of-vision to show more of the geometry.

One of the solutions to this is to map the point $v = (x, y, z)$ first to the sphere, $v' = (\frac{x}{r}, \frac{y}{r}, \frac{z}{r})$, where $r = \delta(0, v)$, and then map this sphere stereographically. This allows higher field-of-vision than the standard linear perspective, which corresponds to gnomonic projection in this sense. One nice property of this approach is that it maps spheres to spheres (Figure 5.21c).

Another solution is the Panini projection (Figure 5.21d), named after the italian artist Giovanni Paolo Panini, believed to use it for his paintings. In this projection, we map $v' \in \mathbb{S}^2$ to the cylinder $\{(x, y, z) : x^2 + y^2 = 1\}$, just like in the central cylindrical porjection, and then, we project the cylinder from point $(0, 0, -1)$ to the Euclidean plane $\{(x, y, z) : z = 0$. This projection maps vertical lines to vertical lines and straight lines going through C_0 to straight lines going through C_0, which are desirable properties for street views and similar scenes (the vertical walls of buildings are mapped to vertical lines, and the lines parallel to the street are mapped to radial lines).

Other projections So far, our process for converting a spherical projection $p : \mathbb{S}^2 \to \mathbb{E}^2$ to its hyperbolic analog $p' : \mathbb{H}^2 \to \mathbb{E}^2$ was essentially to list the properties which uniquely identify p, and then to find a unique projection $p' : \mathbb{H}^2 \to \mathbb{E}^2$ that has exactly the same properties. While we have been focusing on hyperbolic projections, $p' : \mathbb{H}^2 \to \mathbb{E}^2$, the same method also works for finding the most general case $p'' : \mathbb{U}^2_{K'} \to \mathbb{U}^2_{K''}$). As mentioned in the Introduction, the inhabitants of the hyperbolic plane could use a native Poincaré disk model to map their world to a hyperbolic disk, which corresponds to the case $K' < K'' < 0$. Another possible use is projecting a sphere to a sphere of different radius, for example, an azimuthal equidistant projection from Moon to Earth.

However, other projections used in cartography are not defined by their properties, but rather by a specific formula that generates a projection that is neither fully conformal nor fully equal-area, but rather a compromise designed to share most of advantages and avoid most of disadvantages of both kinds of projection. What if p is given as a formula? In some cases, we can handle this too. Take a projection $p_{1\to 0} : \mathbb{S}^2 \to \mathbb{E}^2$. What should we do to obtain $p_{K\to 0} : \mathbb{U}^2_K \to \mathbb{E}^2$ for $K > 0$? We should map $\mathbb{U}^2_K$ to $\mathbb{S}^2$ first, by scaling the x and y coordinates by factor $\sqrt{K}$, apply $p_{1\to 0}$ to it, and then scale back, by the factor $1/\sqrt{K}$.

Now, assume that the points in both geometries are modelled with pairs of

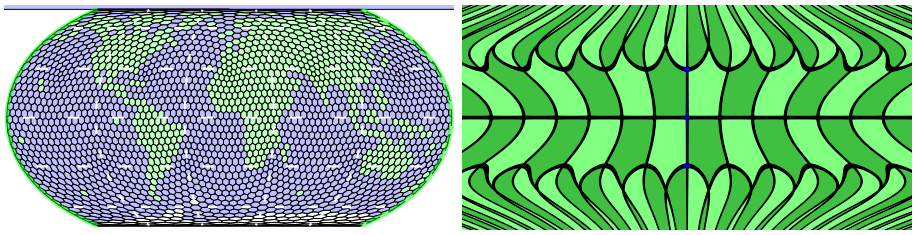

Figure 5.22: Equal Earth projection, applied to Earth map and the hyperbolic plane.

coordinates (take e.g. the stereographic projection in the case of $\mathbb{S}^2$). Thus, we can write $p_{1\to 0} : \mathbb{R}^2 \to \mathbb{R}^2$, and $p_{K\to 0}$ computed according to the recipe in the last paragraph is $p_{K\to 0}(v) = p_{1\to 0}(\sqrt{K}\cdot v)/\sqrt{K}$ for $v \in \mathbb{R}^2$. Assuming that $p_{1\to 0}$ is translated and scaled so that $p_{1\to 0}(0,0) = (0,0)$ and the derivative $p'_{1\to 0}(0,0)$ is an identity, the same will be true for $p_{K\to 0}$.

Now, the same formula can be applied for $K < 0$ – $\sqrt{K}$ is no longer a real number, but if the formula for $p_{1\to 0}$ is an analytic function, it usually can be uniquely extended to a function $p_{1\to 0} : \mathbb{C}^2 \to \mathbb{C}^2$; and if we are lucky, $p_{K\to 0}(v)$ will be in fact not only in $\mathbb{C}^2$, but in $\mathbb{R}^2$. This recipe works, for example, for Eckert IV, Ortelius, Equal Earth, or Wagner VI projection (Figure 5.22). However, it does not work, for example, for the Van der Grinten projection. These projections are available in RogueViz. Also, RogueViz includes more projections which are not discussed in this book.

Exercise 5.5.1. We can also get new projections by composing known projections. One interesting case of such composition is projecting a hyperbolic plane to hemisphere (using the hemisphere model), and then applying a chosen spherical projection π to this hemisphere. If π is conformal, will the composition be conformal too? What about other properties? What do we get if π is orthographic, stereographic, gnomonic, or Mercator projection?

5.6 Three-dimensional geometries

Many of the techniques introduced here naturally generalize to projections from $\mathbb{U}^d_{K'}$ to $\mathbb{U}^d_{K''}$, for arbitrary dimension d. Naturally, we are most interested in the case $d = 3$. While perspective projection is good for immersion, other projections are good, e.g., for creating 3D printed models, or for showing how the space looks from every direction.

Azimuthal and horocyclic projections can be moved to higher dimensions without any change. (Disk models are called ball models in higher dimension, see Figure 5.23.) For cylindrical projections, the natural approach is to use cylindrical coordinates, i.e., $x = T_1^\lambda R_{2,3}^\psi T_2^\phi C_0$; coordinates λ and ϕ correspond to longitude and latitude on a hyperbolic plane rotated around the equator line $T_1^\lambda C_0$ by α. Then, the chosen cylindrical projection is applied to latitude ϕ.

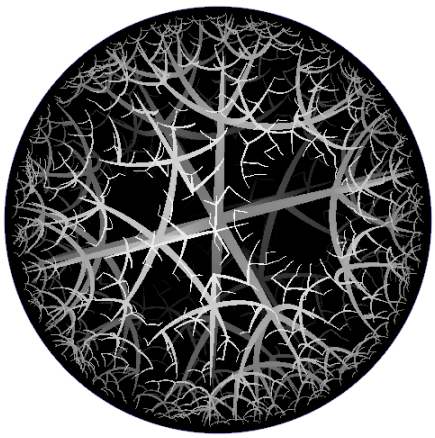

Figure 5.23: Tree for generating the {3,4,4} hyperbolic honeycomb, in Poincaré ball model.

The links between conformal mappings and complex analysis no longer work in higher dimensions. In fact, Möbius transformations (defined as compositions of similarities and sphere inversions) are the only conformal mappings, by by Liouville's theorem. Thus, for example, the three-dimensional analog of band model, constructed as in the last paragraph, will no longer be conformal.

Chapter 6

Topology

In this chapter we will be discussing topological constructions. Recall that topology deals with properties of the space which do not change when the space is stretched or compressed (but may change when the space is cut or glued), and geometry is the opposite: it may change when the space is stretched or compressed, but not when the space is cut or glued.

While in popular mathematics sources, topology is usually described as we did above ("the mathematics of rubber"), the university courses usually start with *general topology*, that is, formal set-theoretic definitions of topological spaces, open/closed sets, compact sets, continuous functions and homeomorphisms. We will not go into such definitions and instead rely on intuition (and our intuitions will work correctly for us, but might no longer be accurate in the more general topological spaces); interested readers may read formal definitions everywhere.

6.1 Surgery

By *gluing* we mean a process which causes two points of the boundary become the same point (usually no longer on the boundary), and *cutting* is the opposite process. We also use the term *surgery* to refer to both cutting and gluing. One can imagine physically cutting and gluing spaces, for example, cutting out a rectangular piece of paper and then gluing its opposite edges to construct a cylinder. We can also imagine it as a purely mathematical construction. For example, in the classic games such as *Pacman* or *Asteroids*, when an object crosses the right edge of the screen, it reappears on the left edge (and vice versa). This means that the left edge and the right edge have been glued. In *Asteroids*, the top and bottom edges are also glued.

6.2 Quotient spaces

As we said, in *Asteroids*, when the player's ship crosses an edge of the screen, it appears at the opposite edge. However, this happens only from the perspective of the player looking at the computer screen. What would the player see if they were actually in the ship? Imagine that, on the screen, the ship is currently close to the top edge of the screen, facing upwards. What does it see? Does it see the edge of the screen, and nothing beyond it?

This of course depends on how we assume the light rays in the world of Asteroids to work, but the logical assumption is that the light rays *also* reappear on the opposite edge when they cross the edges of the screen. So our ship would not actually see the top edge, but it would see what we see on the bottom part of the screen. It is also not hard to see that, over that bottom part of the screen, the ship sees a copy of the top part of the screen (including itself), above that, another copy of the bottom part of the screen, and so on. The ship in Asteroids sees its finite world as an infinitely repeating scene!

Furthermore, the ship will see its world as a Euclidean plane ($\mathbb{E}^2$). The captain can devise a coordinate system where the axes correspond to the observed periods, where 1 denotes the length of the period (screen edge in our world), and 0 is some fixed point. Let us say that $(x_1, y_1) \sim (x_2, y_2)$ if $x_1 - x_2$ and $y_1 - y_2$ are integers. If $(x_1, y_1) \sim (x_2, y_2)$, the ship sees the same object at both coordinates.

This leads us to the concept of a *quotient space*. Take a space X, and a relation $\sim$ on X. The quotient space $X/\sim$ is obtained by identifying the points a, b of X whenever $a \sim b$. In the case of the ship in *Asteroids*, the captain would probably notice that the world repeats, and would guess correctly that the world is actually $\mathbb{E}^2/\sim$. For the quotient space construction to make sense, $\sim$ must be an *equivalence relation*, that is, $a \sim a$ for every point of X, $a \sim b$ iff $b \sim a$, and $a \sim c$ whenever $a \sim b$ and $b \sim c$. We can also phrase this in terms of a function: if Y is our screen, there is a function $c : X \to Y$ which maps a point of X to a matching point of Y.

6.3 Covering spaces

In most cases in this book, the construction is, in a sense, locally consistent. Let us go back to our captain. Imagine a point $x \in X$ in the plane perceived by the captain, and the corresponding point $c(x)$ on screen. Also denote by D the set of all points y such that $c(y) = c(x)$. If we take a small open[1] neighborhood U of the point $c(x)$[2], we can find the matching neighborhood $U_y \in X$ for every point $y \in D$ such that the sets U_y will be disjoint, and for each U_y, the map c restricted to U_y will be a continuous deformation (*homeomorphism*). If such

[1] In a metric space, a set U is *open* iff for every point $x \in U$, there exists $d > 0$ such that the set U contains all points in distance less than d from x.

[2] Note that, if $c(x)$ is on the edge of the screen, the neighborhood will also include the points on the open side.

neighborhoods can be found for every point $x \in X$, we say that X is a *covering space* of Y, and $c : X \to Y$ is a *covering*. Since we care not only about topology but also about geometry, we will usually also want c restricted to U_y to be not only a *homeomorphism*, but also an *isometry* (the distances are faithful).

One possible construction of a covering space of Y is to take a point $y \in Y$, and consider all continuous paths p starting from y. Two paths p_1 and p_2 from y to some $z \in Y$ are homotopic iff one can be continuously deformed into the other without moving the endpoints. Thus, for example, in Figure 1, the paths from y to z of the same color are homotopic, while the paths of different colors are not. In the case of *Asteroids*, we can take a path p in Y and describe it to the ship captain, who will be able to construct the corresponding path p' in X; furthermore, in this case, two paths will be homotopic iff the captain sees them as reaching the same point! Note that our X (the Euclidean plane $\mathbb{E}^2$) is *simply connected*, that is, every two paths with equal endpoint sare homotopic. This suggests a general construction of a covering space X: the set of all paths starting from y, but identifying paths which are homotopic. The space X thus obtained is called an *universal cover* of Y. The universal cover is always simply connected. It is called an *universal cover* because it is, in a sense, the largest covering space – all simpler coverings are not simply connected and thus it is possible to find a larger covering space.

6.4 Summary

To summarize this Chapter so far, a topological construction can be visualized in three ways.

- **Embedding/immersion.** We imagine our space to be embedded in higher-dimensional space. For example, we imagine a *cylinder* (the space obtained by gluing left and right edges of a rectangular piece of paper) as if we physically glued the edges in our world to make a cylindrical shape, and we imagine a *topological torus* (the space where we also glue top and bottom edges) as a donut shape (geometric torus).

  Such pictures are commonly used in topology explanations. However, there are some problems with it. It basically works only for surfaces, not for higher-dimensional spaces. It might not give the correct idea about the *geometry* (a donut shape has smaller inner radius than the outer radius, so it does not correctly represent the geometry of the *Asteroids* world). In some cases, there is no way to place a surface in three dimensions without self-intersections (a mapping that has self-intersections is not an embedding, but an *immersion*).

- **Gluing diagram.** A gluing diagram is based on the idea that the space was obtaining by surgery: *cutting* a part of some larger space, and then *gluing* its edges in some way. In a gluing diagram, we show the part that has been cut, and then, on its edges, we place markers which explain in

which way there are connected. We generally need to specify which edge connects to which (thus, the letters a and b in the torus diagram – a letter connects to the same letter) and also the direction of gluing (with an arrow).

- **Universal cover diagram.** In this visualization method, we show the universal cover of our space Y, and we draw appropriate markers to clarify which points of the universal cover correspond to the same point of Y. As we have seen in the *Asteroids* captain example, one advantage of this is immersiveness: it corresponds to what the creatures living in Y would actually perceive.

In modern 3D video games, immersiveness is usually desired, so the world would be essentially displayed as a universal cover diagram (see e.g. *Manifold garden* in levels without immersive portals – the universal cover interpretation is no longer completely correct in the existence of portals). The method also works in 2D, and is used for the quotient spaces in HyperRogue and the RogueViz visualizations, but it seems that the view in most 2D games (e.g., *Asteroids*) is more in the gluing diagram style – although no arrows are shown because the gluing is to be inferred from the landmarks and how the objects behave. Embeddings are rarely used, but they are also a possibility.

6.5 Spherical and Euclidean surfaces

Armed with the knowledge of surgery and covers, we can now describe the various topologies that surfaces with Euclidean or spherical geometry can have.

In the case of spherical geometry, there is just two choices: the sphere $\mathbb{S}^2$ itself, and the elliptic plane $P\mathbb{S}^2$, which is a quotient space of $\mathbb{S}^2$ where we identify each $x \in \mathbb{S}^2$ with $-x$.

In the case of Euclidean geometry, let us think what spaces we can obtain by surgery from the Euclidean plane. We already know the Euclidean plane $\mathbb{E}^2$ itself, the square torus, and the cylinder. (We see the *square torus* because we can also make a flat torus out of a rectangle, or out of a parallelogram – such tori obtained have the same topology and Euclidean geometry, but they usually will not be isometric to the square torus.)

Cone. (a) We cut out an angle, and then glue its edges. One important thing to observe about this simple construction is that a cone does not locally have Euclidean geometry everywhere, but rather, it has a *cone point* which has less than 360 degrees around it. (It is also possible to obtain a cone point with more than 360 degrees.) Thus, a cone is not a Euclidean manifold, but a Euclidean *cone-manifold*. For the purpose of Gauss-Bonnet theorem (Section 4.3, a cone point should be treated as a single point where non-zero curvature is concentrated.

Möbius strip. This is obtained by cutting a rectangle and gluing its edges, but not like in a cylinder – we rotate the edge by 180° first. The arrows in the gluing diagram are reversed. The Möbius strip has a boundary; we can also

consider an infinite Möbius strip which does not have a boundary, by using not a rectangle, but an infinite strip. That is harder to imagine embedded, but the gluing diagram and the universal cover diagram are quite natural. The Möbius strip is non-orientable, which in the two-dimensional case means that there is no consistent assignment of meaning to *clockwise/right* and emphcounterclockwise/left (when our character in Figure XXX crosses the reversed edge, their mental concept of left and right is reversed).

Klein bottle. A Klein bottle is obtained like the Möbius strip, but one gluing is reversed, just like we did for the Möbius strip. A Klein bottle cannot be embedded in three-dimensional space, although it can be immersed. Immersions can be found online; we do not show such an immersion here, because they are somewhat complex and thus misleading – especially in some sources which call a Klein bottle a four-dimensional object due to it not being embeddable in $\mathbb{E}^3$. Most mathematicians probably think of Klein bottle simply in terms of its (much simpler) two-dimensional gluing diagram, or the universal cover diagram, which is also quite natural.

Both gluings reversed. To construct a Klein bottle, we changed the torus by reversing *one* pair of glued edges. What if we reverse *both* of them? Let us denote the space obtained by B. Such a construction turns out to be less elegant, which can be seen by considering one of the vertices of the original square. Such a vertex has only 180° degrees around it, and thus, B is a cone-surface, not a Euclidean surface.

Some sources call this B the *projective plane* $\mathbb{R}P^2$, that is, two-dimensional projective space. We have already seen the projective space $\mathbb{R}P^d$ defined differently in Section 2.2. That is correct if only the topology is concerned: the definition from Section 2.2 has the same topology as B. However, if we also want to have nice geometry, it is better to consider the elliptic plane $P\mathbb{S}^2$ explained in the beginning of the section. Again, $P\mathbb{S}^2$ and B have the same topology; $P\mathbb{S}^2$ is obtained by changing the geometry by of B spreading the curvature of the two cone points uniformly.

Orbifolds. It is also a good time to intuitively explain the concept of an *orbifold*. An orbifold is obtained by taking a quotient by a relation which does not correspond to a covering. For example, consider a Euclidean plane $\mathbb{E}^2$, and identify points x and y where y is obtained by rotating the point x by a multiple of 90° around the origin. Thus, four points of $\mathbb{E}^2$ are identified as a single point of the quotient space Y – except the origin, which is a cone point with 90° around it. Orbifolds are thus obtained as quotients by local groups of isometries; they arise naturally when we consider periodic tilings, and identify points which are the same relative to the periodic tiling. Contrary to a more general cone-manifolds, a point in orbifold will always have $360^\circ/k$ degrees around it, for some integer k. On the other hand, orbifolds can be also obtained by taking a quotient by a mirror symmetry, for example, identifiying the point (x, y) with $(x, -y)$; a point with $y = 0$ will see only 180 degrees around it, but due to the mirror symmetry, such a mirrored point will work differently than a cone point.

6.6 Euler characteristics

Recall Euler's polyhedral theorem (Exercise 4.3.2): a convex polyhedron with F faces, E edges and V vertices satsfies $V + F = E + 2$. We can generalize this computation to any surface topology: take a compact[3] surface X, mark V points (vertices) on it, connect them with E edges. If we have drawn enough edges, the rest of the surface will be subdivided into some number of *faces*, each homeomorphic to an open disk. Let F be the number of faces.

The number $\chi = V - E + F$ is called the *Euler characteristics* of X. Just like for convex polyhedra, the Euler characteristics does not depend on how exactly have we drawn the vertices and edges. It is also a topological invariant, that is, it does not change when we stretch X.

Exercise 6.6.1. Compute the Euler characteristics of: sphere $\mathbb{S}^2$, elliptic plane $P\mathbb{S}^2$, torus, Klein bottle.

As seen in Exercise 4.3.2, the Euler characteristics is linked to the geometry of X: it is essentially τ times the total Gaussian curvature over X. In particular, Euler characteristics is positive for spherical surfaces, 0 for Euclidean surfaces, and negative for hyperbolic surfaces. We have not constructed compact hyperbolic surfaces yet, but we will do it in the next section, where the Euler characteristics will be useful.

The Euler characteristics is an integer. Furthermore, for orientable surfaces, it must be divisible by 2. For orbifolds, a more general definition of Euler characteristics allows non-integral values.

6.7 Hyperbolic surfaces

Let us try to create a closed hyperbolic surface by surgery.

Note that our tori and Klein bottles could be obtained by taking a number of Euclidean squares and gluing them correctly (four squares meeting in every corner to avoid cone points). A similar construction also works for the sphere (e.g., glue 12 spherical pentagons, three meeting in every corner) and the elliptic plane (glue 6 pentagons, three meeting in every corner). By analogy, we will construct a hyperbolic surface from a regular hyperbolic tessellation $\{p, q\}$. We could use any p and q; let us assume that we use the $\{7, 3\}$. Thus, we want to take a number of hyperbolic heptagons and glue them so that three meet in every vertex. How many heptagons should we take?

The Euler characteristics has told us that counting vertices, edges and faces is important. Since we have a face for every heptagon, let us denote the number of heptagons by F. We also have seven edges for every heptagon, but every edge is shared by two heptagons, so $E = 7F/2$. Furthermore, we have $V = 7F/3$. Thus, $\chi = V - E + F = \frac{7}{3}F - \frac{7}{2}F + F = \frac{14F-21F+6F}{6} = -F/6$. The Euler characteristics should be an integer, so F should be divisible by 6. (A similar

[3] A compact set is closed and bounded.

argument shows that a sphere of Euler characteristics 2 requires 12 pentagons to be glued.)

The smallest possible F is 6. In HyperRogue/Rogueviz, this hyperbolic surface is called the *minimal quotient*. It is not orientable, and cannot be embedded in $\mathbb{E}^3$ (in general, closed non-orientable surfaces cannot be embedded in $\mathbb{E}^3$).

Another hyperbolic surface available in HyperRogue/RogueViz is called the *Zebra quotient*, because it is obtained by identifying the points which are the same with respect to the periodic pattern used in the Zebra land. It is orientable and glued of 12 heptagons, thus its Euler characteristics is -2. An orientable surface of Euler characteristics $2 - 2k$ can be embedded in $\mathbb{E}^3$ as a sphere with k handles (a torus is a sphere with 1 handle). However, as in the case of the torus, such an embedding distorts the geometry.

6.8 Highly symmetric hyperbolic surfaces

Some of our constructions so far have more symmetry than the others.

Chapter 7

To be continued

These chapters are not written yet.

7.1 Tilings of the hyperbolic plane

We discuss the basic tessellations of the hyperbolic plane. Tessellations are useful not only interesting from math art perspective, but they are also good to organize the gameplay of non-Euclidean games and layout of non-Euclidean visualizations, and they are essential to avoid numerical precision issues in large-scale computations.

7.2 Procedural generation

Due to the exponential expansion, hyperbolic geometry is useful in game design. However, we need ways to fill these exponentially expanding worlds. This chapter discusses the methods of doing that.

7.3 Art Models

We discuss constructing models of negatively curved surfaces. It covers crocheting, making models from beads or paper, the Hypersian Rug algorithm which is a computer simulation of hyperbolic crochet, and other isometric immersions of constant-curvature surfaces into $\mathbb{R}^3$.

7.4 Topology

We can use not only the hyperbolic plane, but also hyperbolic surfaces, obtained by cutting a part of the hyperbolic plane and gluing it. In this chapter, we discuss the basics of topology, and the methods of obtaining such surfaces.

7.5 Alternative methods

A chapter about alternative ways of modelling hyperbolic geometry. We discuss their strengths and weaknesses.

7.6 Generating tree structures

This chapter discusses the methods of finding regular tree structures, which were described in Section 3.3.

7.7 Hyperbolic honeycombs

Honeycombs are tessellations of three-dimensional space. We discuss the representation of hyperbolic honeycombs.

7.8 Twisted geometries

We discuss twisted three-dimensional geometries, such as Nil, twisted $\mathbb{H}^2 \times \mathbb{R}$, and Berger sphere.

7.9 Fully anisotropic geometries

We discuss fully anisotropic three-dimensional geometries, such as Solv.

7.10 Fully symmetric spacetimes

We discuss fully symmetric spacetimes, such as de Sitter and anti-de Sitter spacetime.

7.11 Applications

Here we discuss the applications of non-Euclidean geometry in data analysis.

7.12 Distances

Here we discuss efficient algorithms for computing the distances in the tilings of the hyperbolic plane.

7.13 History and Bibliography

Here we briefly discuss the history of concepts and technical solutions discussed in the book, and provide bibliographic references.

Chapter 8

Solutions to Exercises

Solution to Exercise 2.5.1: We will skip the indexes $1, 2$.

The analog of rotation will be again described by some matrix:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

We will call the matrix here L^α. (L stands for Lorentz transformation.) What values should we enter as '?'? Let us try to apply L^α to $e_y = (0, 1)$. We will call the coordinates of this point $(\sinh \alpha, \cosh \alpha)$. This means that

$$L^\alpha = \begin{pmatrix} ? & \sinh \alpha \\ ? & \cosh \alpha \end{pmatrix}$$

Since L^α is supposed to be an isometry, we have $\langle L^\alpha e_y, L^\alpha e_y \rangle = \langle e_y, e_y \rangle = -1$. Thus, we get the following formula: $(\sinh \alpha)^2 - (\cosh \alpha)^2 = -1$.

We also can apply L^α to $e_x = (1, 0)$. We have two formulas: $\langle L^\alpha e_x, L^\alpha e_x \rangle = \langle e_x, e_x \rangle = 1$ and $\langle L^\alpha e_x, L^\alpha e_y \rangle = \langle e_x, e_y \rangle = 0$. By solving the system of equations, we get the remaining values in the matrix L^α:

$$L^\alpha = \begin{pmatrix} \cosh \alpha & \sinh \alpha \\ \sinh \alpha & \cosh \alpha \end{pmatrix}$$

Now, what do we know about functions $\sinh \alpha$ and $\cosh \alpha$?

- Obviously L^0 should be an identity. Therefore, $\sinh(0) = 0$ and $\cosh(0) = 1$.
- How do angles scale to rotation matrices? Well, the rule is that rotation by angle α and then β is the same as rotation by angle $\alpha + \beta$: $R^\beta R^\alpha = R^{\alpha+\beta}$. We want the same to hold for Lorentz transformations.
- We need to choose an unit. Euclidean angles could be measured using any unit: radians, degrees, turns... Radians have a nice property that $\sin'(0) = 1$, that is, rotating the point $(0, 1)$ by small angle α moves it

in distance α. By analogy, we will choose our unit so that we also have $\sinh'(0) = 1$.

- We also know that $|L(\alpha)(e_y)| = 1$. By taking the derivative, we get $\cosh'(0) = 1$.
- Since $L^\beta L^\alpha = L^{\alpha+\beta}$, we have $L'(0)L^\alpha = L'^\alpha$. Therefore:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \cosh\alpha & \sinh\alpha \\ \sinh\alpha & \cosh\alpha \end{pmatrix} = \begin{pmatrix} \cosh'\alpha & \sinh'\alpha \\ \sinh'\alpha & \cosh'\alpha \end{pmatrix}$$

Therefore $\cosh'\alpha = \sinh\alpha$, and $\sinh'\alpha = \cosh\alpha$. We need to solve a system of differential equations to get the full formulas for sinh and cosh. We get: $\cosh(\alpha) = (exp(\alpha) + \exp(-\alpha))/2$, $\sinh(\alpha) = (\exp(\alpha) - \exp(-\alpha))/2$.

Solution to Exercise 4.1.1: Assume that we are moving in a circle of radius r. Specifically, let us assume that, at time t we are in point $c(t) = (r\cos(\alpha t), r\sin(\alpha t))$. The velocity vector at time t is $\dot{c}(t) = (-r\alpha\sin(t), r\alpha\cos(t))$ of length αr, and thus, we are moving at constant speed $v = \alpha r$. The acceleration is $\ddot{c}(t) = (-r\alpha^2\cos(t), -r\alpha^2\sin(t))$, and thus, we have $a_2 = \alpha^2 r$, and the curvature of our track (a circle of radius r) is $a/v^2 = \alpha^2 r/\alpha^2 r^2 = 1/r$. Therefore, a circle of radius r has curvature $k = 1/r$.

Solution to Exercise 4.2.1: Suppose that we, at time t, are in the point $c(t) = T_2^t(C_0) = (0, \sinh(t), \cosh(t))$. Then, the right hand is at point

$$c_1(t) = T_2^t(T_1^x(C_0)) = (\sinh(x), \sinh(t)\cosh(x), \cosh(t)\cosh(x)).$$

Chapter 9

Notation index

The following notation is ubiquitous in the book. For convenience, we also give the counterpart notation in the RogueViz sourcecode.

- $\mathbb{R}^d$ – the set of tuples of d real numbers, without any geometry (Page 15; points are `hyperpoint` and matricrs are `transmatrix`)
- $\mathbb{E}^d$, $\mathbb{S}^d$, $\mathbb{H}^d$ – Euclidean, spherical, and hyperbolic d-dimensional space; we assume curvature 0, 1 or -1, and the coordinates are given in models explained in Chapter 2.
- $\mathbb{U}^n_K$ – a d-dimensional space of constant curvature K; the coordinates are given in the model explained in Chapter 2.
- $P\mathbb{R}^n$ – projective n-dimensional space (Page 17)
- $R^\alpha_{i,j}$ – rotation by α radians in plane i, j (Page 16, `cspin`)
- τ – full turn in radians, i.e., the length of a Euclidean circle of radius 1 (Page 16, `TAU`)
- $x^\circ = \frac{x\tau}{360}$ (Page 16, `x_deg` or `xdegree`)
- $\sum_{i=1}^n x_i = x_1 + \ldots + x_n$ (Page 18)
- $g(v, w)$ – inner product, definition depends on the geometry (Chapter 2)
- $L^\alpha_{i,j}$ – Lorentz boost by α in plane i, j (Page 16, `lorentz`)
- M_i – reflection (mirror image) in the i-th coordinate (Page 17, `cmirror`)
- C_0 – the central point in the chosen model of geometry (Chapter 2, `C0`)
- M, V, P – model, view, and projection transformation (Page 17)
- T^x_i – translation by x in the i-th coordinate, definition depends on the geometry (Chapter 2, `cpush`)

- $\sinh x = \frac{e^x - e^- x}{2}$, hyperbolic sine (Page 21)
- $\cosh x = \frac{e^x + e^- x}{2}$, hyperbolic cosine (Page 21)
- $\tanh x = \frac{\sinh x}{\cosh x}$, hyperbolic tangent
- arc sinh, arc cosh, arc tanh – the inverses of hyperbolic functions (`asinh`, `acosh`, `atanh`)
- $\sin_K, \cos_K$, etc. – curvature-dependent analogs of sinh and cosh (Page 22; `sin_auto`, `cos_auto`; `clamp` versions to protect against numerical errors)
- $\delta(a, b)$ – the distance between a and b (Chapter 2, `hdist`)
- $t \to$ NAME – attribute of object t with the given name (Page 27)
- $M_{t',t}$ – transform t'-relative coordinates to t-relative coordinates (Page 32, `currentmap->relative_matrix`)
- M^{-1} – matrix inverse (Page 33, `inverse`)
- R_v – rotate v to the first coordinate (Page 33, `rspinto`)
- T_v – translate v to C_0 (Page 33, `rgpushxto0`)
- $\mathbb{C}$ – the set of complex numbers (Page 53)

Index