Morphing Tilings of the Plane into Tilings of Surfaces

Mark L. Loyola
mloyola@ateneo.edu
Department of Mathematics
Ateneo de Manila University
Philippines

Ma. Louise Antonette N. De Las Peñas
mdelaspenas@ateneo.edu
Department of Mathematics
Ateneo de Manila University
Philippines

Abstract: *This work discusses a procedure to generate a tiling* $\mathcal{T}_{S_f}$ *of a* 2*-dimensional surface* S_f *embedded in the Euclidean* 3*-space* $\mathbb{R}^3$ *from a tiling* $\mathcal{T}$ *of the Euclidean plane* $\mathbb{R}^2$*. We employ the computer algebra system Mathematica to generate 3D graphical images of* $\mathcal{T}$ *and* $\mathcal{T}_{S_f}$ *and render animations which create the effect of* $\mathcal{T}$ *morphing into* $\mathcal{T}_{S_f}$*.*

1. Introduction

Tilings of surfaces, space forms, or quotient spaces have been interesting subjects of study in discrete geometry and computer graphics [10, 13, 14, 15]. These geometric structures find applications as artworks or as mathematical models of chemical structures such as analogs of carbon nanotubes [6] or carbon nanotori [11, 12].

In recent years, technological tools have played an important role in the teaching and research of tilings. Dynamic geometry software, for example, have allowed for the investigation of properties and relationships of tilings [3], and the discovery of new families of tilings [2, 5]. Computer algebra systems have facilitated an efficient way of rendering tilings in surfaces and quotient spaces [7].

One of the important benefits of a technological tool is being able to facilitate the visualization of geometric representations. Geometric concepts can be visualized with dynamic geometric representations such as graphical images, animations, videos, applets, or with constructed or drawn representations made with a computer program [18]. Interactive applets, for example, can facilitate the deeper understanding of otherwise static geometric images, as applets allow users to explore, conjecture, make discoveries on these geometric figures, and even arrive at mathematical proofs. Computer programs can produce visual outputs, such as visualization of tilings of higher dimensions and other spaces, facilitating mathematical modeling and simulation of real-life scenarios [4].

In this paper, we present our work on tilings of surfaces, which are realized as geometric representations with the use of technology. We organize the paper as follows. In the next section, we enumerate different parametric systems of equations that define surfaces embedded in the Euclidean 3-space and discuss a procedure to generate a tiling of a given surface from a tiling of the Euclidean plane. In Section 3, we implement the said procedure in a computer algebra system to generate 3D graphical images of such tilings. We present two different ways of implementing this procedure and discuss the programming code on how each can be accomplished. In the final section, we demonstrate a technique to produce animations with a slider and animated gif files that show a tiling of the plane transforming into a tiling of a surface.

2. Tilings of Surfaces

A *tiling* or *tessellation* $\mathcal{T}$ of a point space $\mathbb{X}$ is a collection of subsets of $\mathbb{X}$ called *tiles* whose interiors do not intersect and whose union is the entire space $\mathbb{X}$. Although $\mathbb{X}$ can, in principle, be any set, it is usually endowed with a geometric structure. Various examples of tilings of the Euclidean plane $\mathbb{R}^2$ and their classifications in terms of symmetry and topological considerations are found in the classic text *Tilings and Patterns* [9]. One may also consult the searchable online databases *Tiling Search* [16] and *Tilings Encyclopedia* [8] to download images of tilings and retrieve useful information about them.

Tilings with rich mathematical structures and at the same time aesthetically pleasing are those consisting of tiles which either belong to a single congruence or similarity class or to a finite number of such classes. We present in Figure 2.1 examples of tilings of two point spaces. Figure 2.1(a) depicts a square patch of the famous *Ammann-Beenker tiling* $\mathcal{A}$ of $\mathbb{R}^2$ [8]. This tiling consists of congruent squares and rhombi which are arranged around the center of the plane to form a pattern with an 8-fold rotation symmetry. A star-shaped patch of an analog of this tiling on the *unit* 2-*sphere* $\mathbb{S}^2$ is shown in Figures 2.1(b) [front view] and 2.1(c) [back view].

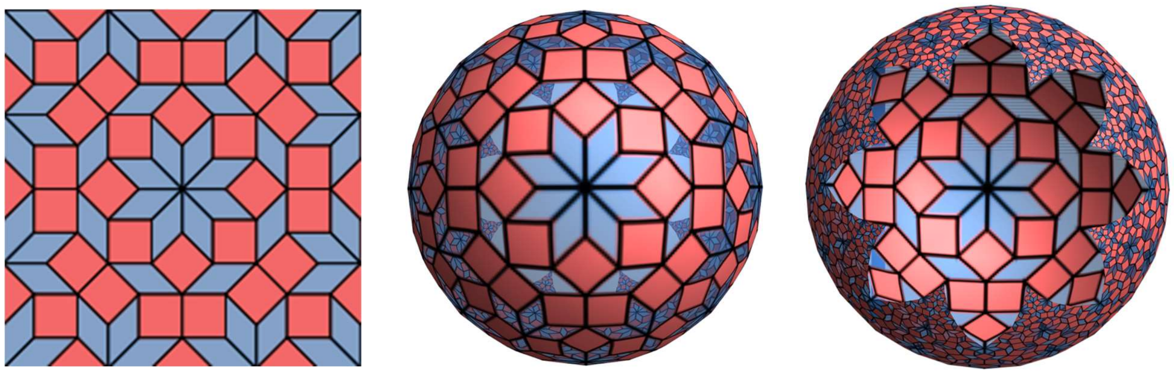


Figure 2.1 Amman-Beenker tilings of the point spaces (a) $\mathbb{R}^2$ and (b, c) $\mathbb{S}^2$.

In this work, we shall restrict ourselves to point spaces $\mathbb{X}$ that correspond to 2-dimensional *surfaces* embedded or lying in the Euclidean 3-space $\mathbb{R}^3$. Formally, we define a *surface* S_f to be the image of a continuous map $f : \mathbb{R}^2 \to \mathbb{R}^3$ in two variables u and v. If we denote by (x, y, z) the image in $\mathbb{R}^3$ of a point (u, v) in $\mathbb{R}^2$, then x, y, z may be identified with three continuous maps f_x, f_y, f_z, respectively, of u and v. These identifications make S_f a *parametric surface* and allow us to describe the surface conveniently using the parametric system of equations

$$\begin{aligned} x &= f_x(u, v) \\ y &= f_y(u, v) \\ z &= f_z(u, v) \end{aligned} \tag{2.1}$$

with parameters $u, v \in \mathbb{R}$. For example, if we take the maps $f_x(u, v) = u,\ f_y(u, v) = v,\ f_z(u, v) = 0$, we obtain the standard embedding of the Euclidean plane $\mathbb{R}^2$ into $\mathbb{R}^3$. This trivially implies that the Euclidean plane may also be viewed as a surface. As a matter of fact, the point space $\mathbb{S}^2$, which is the image of $\mathbb{R}^2$ under the inverse *stereographic projection*, is also a surface. Observe that under this map, which is an *inversion relative to* $\mathbb{S}^2$, the image of the small square patch of $\mathcal{A}$ in Figure 2.1(a) covers a large portion of the sphere as shown in Figure 2.1(b, c). The parametric equations defining these two and other surfaces, which we shall encounter later, are listed in Table 2.1 below.

Table 2.1 Surfaces defined by a parametric system of equations with parameters $u, v \in \mathbb{R}$, where $a, b > 0$ are constants and $\theta_{u,v} \in [0, 2\pi]$ is the angle measured in the counterclockwise direction between the vectors $(1, 0)$ and (u, v).

Surface	f_x	f_y	f_z
Plane $\mathbb{R}^2$	u	v	0
Unit 2-Sphere $\mathbb{S}^2$	$\frac{2u}{1+u^2+v^2}$	$\frac{2v}{1+u^2+v^2}$	$\frac{-1+u^2+v^2}{1+u^2+v^2}$
Egg Carton $\mathbb{C}_e$	u	v	$\sin u + \cos v$
Cone $\mathbb{C}_o$	$\frac{a\sqrt{u^2+v^2}}{2\pi}\cos\left(\frac{\theta_{u,v}}{a}2\pi\right)$	$\frac{a\sqrt{u^2+v^2}}{2\pi}\sin\left(\frac{\theta_{u,v}}{a}2\pi\right)$	$\sqrt{u^2+v^2}$
Cylinder $\mathbb{C}_l$	$\frac{a}{2\pi}\cos\left(\frac{u}{a}2\pi\right)$	$\frac{a}{2\pi}\sin\left(\frac{u}{a}2\pi\right)$	v
Torus $\mathbb{T}$	$\frac{b\cos\left(\frac{v}{b}2\pi\right)}{\sqrt{a^2+b^2}-a\cos\left(\frac{u}{a}2\pi\right)}$	$\frac{b\sin\left(\frac{v}{b}2\pi\right)}{\sqrt{a^2+b^2}-a\cos\left(\frac{u}{a}2\pi\right)}$	$\frac{a\cos\left(\frac{u}{a}2\pi\right)}{\sqrt{a^2+b^2}-a\cos\left(\frac{u}{a}2\pi\right)}$

Given a tiling $\mathcal{T}$ of $\mathbb{R}^2$ and a parametric surface S_f, the collection $\mathcal{T}_{S_f} := \{f(t) \mid t \in \mathcal{T}\}$ of images of tiles in $\mathcal{T}$ forms a cover for S_f. If, in addition, the interiors of these images do not intersect, which depends ultimately on the map f, then $\mathcal{T}_{S_f}$ becomes a tiling of S_f. This happens, for instance, when f is a homeomorphism between $\mathbb{R}^2$ and its image $f(\mathbb{R}^2)$. Such is the case with the maps that define the surfaces $\mathbb{R}^2$, $\mathbb{S}^2$, and $\mathbb{C}_e$ in Table 2.1.

Illustration 2.1 We illustrate in Figure 2.2(a) the tiling of the *egg carton surface* $\mathbb{C}_e$ arising from $\mathcal{A}$.

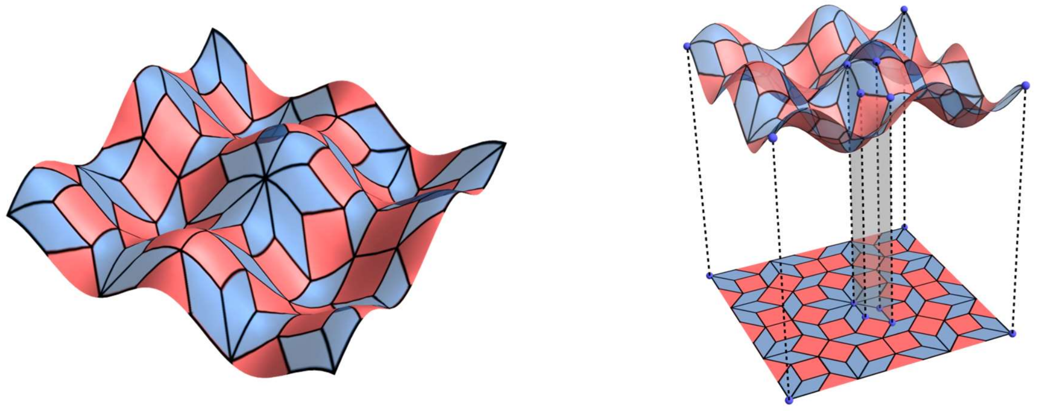

(a) (b)

Figure 2.2 (a) Tiling $\mathcal{A}_{\mathbb{C}_e}$ of $\mathbb{C}_e$ arising from $\mathcal{A}$. (b) The image in $\mathbb{C}_e$ of a rhombic tile in $\mathcal{A}$.

In Figure 2.2(b), eight selected vertices of $\mathcal{A}$ are marked and connected by dashed line segments with their images in $\mathbb{C}_e$. Notice that the marked rhombic tile of $\mathcal{A}$ underwent a deformation after it was mapped to $\mathbb{C}_e$. This is because the map used to define $\mathbb{C}_e$ is neither length-preserving (*isometric*) nor angle-preserving (*conformal*). Contrast this with the tiling $\mathcal{A}_{\mathbb{S}^2}$ in Figure 2.1(b) in which the angles (but not the lengths) of the tiles are preserved. This is a consequence of the fact that the inverse stereographic projection is conformal [1].

The maps in Table 2.1 may be broadly classified as either injective or non-injective. As implied earlier, the first three belong to the first class. On the other hand, the presence of the periodic sine and cosine functions in the formulas for f_x and f_y when f defines either a *cone* $\mathbb{C}_o$, a *cylinder* $\mathbb{C}_y$, or a *torus* $\mathbb{T}$ hints at non-injectivity. Although this classification may seem purely algebraic at first glance, it has an important geometric implication when it comes to the construction of tilings of surfaces.

The cone, cylinder, and torus maps, and hence the surface they define as well, are compatible only with tilings of $\mathbb{R}^2$ that satisfy some symmetry properties. More particularly, a tiling $\mathcal{T}$ of $\mathbb{R}^2$ gives rise to a tiling of

- $\mathbb{C}_o$ if $\mathcal{T}$ possesses a $\frac{2\pi}{a}$-fold rotation symmetry r_a about the origin;
- $\mathbb{C}_l$ if $\mathcal{T}$ possesses a horizontal translation symmetry t_x with vector of length a; and
- $\mathbb{T}$ if $\mathcal{T}$ possesses both a horizontal and a vertical translation symmetry t_x and t_y with vectors of lengths a and b, respectively.

The effect of any of these non-injective maps is to identify points in $\mathbb{R}^2$ that are equivalent with respect to the group of symmetries generated by r_a, t_x, or t_x and t_y. Identified points are then "glued" together to become a single point in the resulting surface, which may now be referred to as a *space form* or a *quotient space*. Consequently, identified tiles of the plane become a single tile under the map. We discuss a specific example in the next illustration.

Illustration 2.2 Since the Ammann-Beenker tiling $\mathcal{A}$ remains invariant under the 2-fold rotation r_2

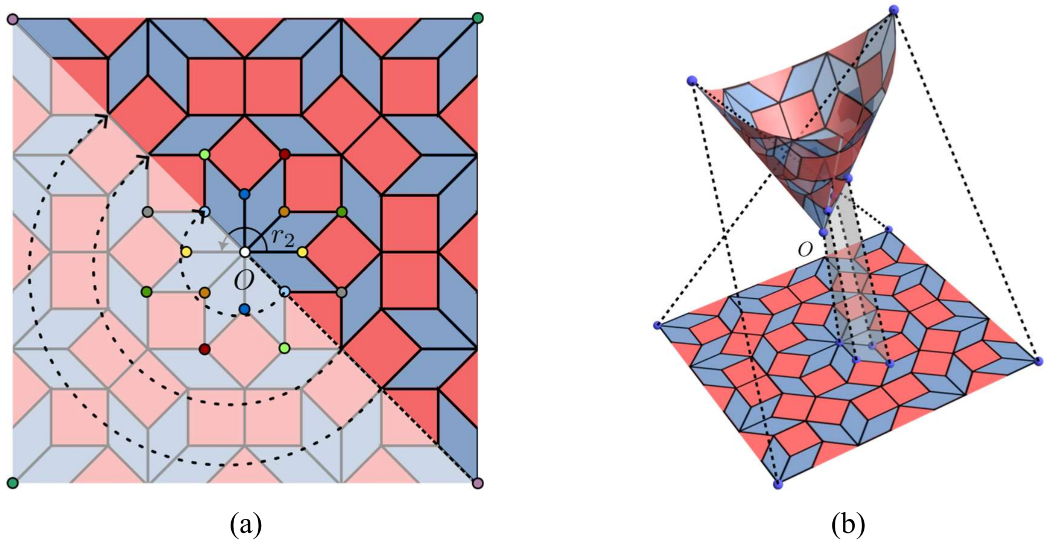


Figure 2.3 (a) Points in $\mathbb{R}^2$ are identified with respect to r_2 to obtain (b) a tiling of $\mathbb{C}_o$.

about its center, which we assume to be the origin O, we can map it to the cone $\mathbb{C}_o$ by setting $a := 2$. Under this map, points in $\mathbb{R}^2$ that are sent to each other by the rotation r_2 are considered equivalent and get identified as a single point in the resulting cone. In Figure 2.3(a), vertices of $\mathcal{A}$ that share the same color become a single vertex in the resulting tiling of the cone. This process may be visualized as the following sequence of actions: cut the square patch in Figure 2.3(a) along the dashed segment from a corner point up to the center to form two flaps; hold one of the flaps at the corner point; and then rotate it towards the opposite corner point so that vertices of the same color are glued together to form the cone whose apex is O (see Figure 2.3(b)). Observe that, as expected, points such as the opposite corner points in Figure 2.3(a) are sent to the same points of $\mathbb{C}_o$. In addition, the angles of the tiles in $\mathcal{A}$ are preserved in $\mathcal{A}_{\mathbb{C}_o}$. This is because the cone map together with the cylinder and torus maps is conformal. We note, however, that the latter two maps are incompatible with $\mathcal{A}$ precisely because this tiling is *non-periodic* [8]. That is, $\mathcal{A}$ does not admit translation symmetries in its symmetry group.

3. Generating 3D Images of Tilings of Surfaces

Our main objective in this and the next section is to illustrate a generic procedure to generate a 3D image of a tiling $\mathcal{T}_{S_f}$ of a surface S_f from a given tiling $\mathcal{T}$ of $\mathbb{R}^2$ and render a simple animation that shows $\mathcal{T}$ morphing into $\mathcal{T}_{S_f}$. We shall implement this procedure in Wolfram Mathematica [16], a powerful computer algebra system (CAS) developed by Wolfram Research. We take advantage of this system's functional programming paradigm and its fast 2D and 3D graphics and animation rendering capabilities.

Illustration 3.1 To illustrate the steps of the procedure, we use the face-colored *truncated quadrille tiling* $\mathcal{Q}$ of $\mathbb{R}^2$, a patch of which is shown in Figure 3.1(a). This tiling is one of the eight semiregular *Archimedean tilings* [9] and consists of congruent regular squares and octagons with one square and two octagons arranged around each vertex. We introduce colors to the tiles of $\mathcal{Q}$ so we can easily see and analyze the effect of a map on the tiles. Observe that the uncolored tiles of $\mathcal{Q}$ can be generated from a single square tile $\mathcal{s}$ and a single octagon tile $\mathcal{o}$ (see Figure 3.1(b)) by successively translating them along two linearly independent directions determined by the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. In this case, we begin with the tile $\mathcal{s}$ whose vertices we assign to be $(1,0)$, $(0,1)$, $(-1,0)$, $(0,-1)$. Basic geometry and trigonometry computations yield the vertices

$$\left(2+\sqrt{2},1+\sqrt{2}\right), \left(1+\sqrt{2},2+\sqrt{2}\right), \left(1,2+\sqrt{2}\right), \left(0,1+\sqrt{2}\right), (0,1), (1,0), \left(1+\sqrt{2},0\right), \left(2+\sqrt{2},1\right)$$

of the tile $\mathcal{o}$. With these vertices, we deduce the coordinates of the vectors $\mathbf{v}_1 = \left(2+\sqrt{2},0\right)$ and $\mathbf{v}_2 = \left(2+\sqrt{2},2+\sqrt{2}\right)$. To generate a patch of the uncolored tiling, we translate $\mathcal{s}$ and $\mathcal{o}$ along vectors of the form $\mathbf{v} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2$, where the coefficients c_1, c_2 are integers at most some pre-assigned bounding values. The checkerboard coloring of the tiles is achieved by introducing a coloring rule based on the parity of c_1. More specifically, for a square or octagon tile, we assign yellow or blue, respectively, when c_1 is even; or orange or pink, respectively, otherwise. The Mathematica implementation of this construction appears in Code 3.1. Note that we utilized the built-in **`Polygon`** function to construct a tile from its list of vertices and the **`Table`** function to generate translates of $\mathcal{s}$ and $\mathcal{o}$. We also converted all point and vector coordinates into their numerical or decimal equivalent using the function **`N`**. This makes computations and graphics rendering in

Mathematica significantly faster as the system does not need to deal with exact algebraic values which often involve radical or trigonometric expressions.

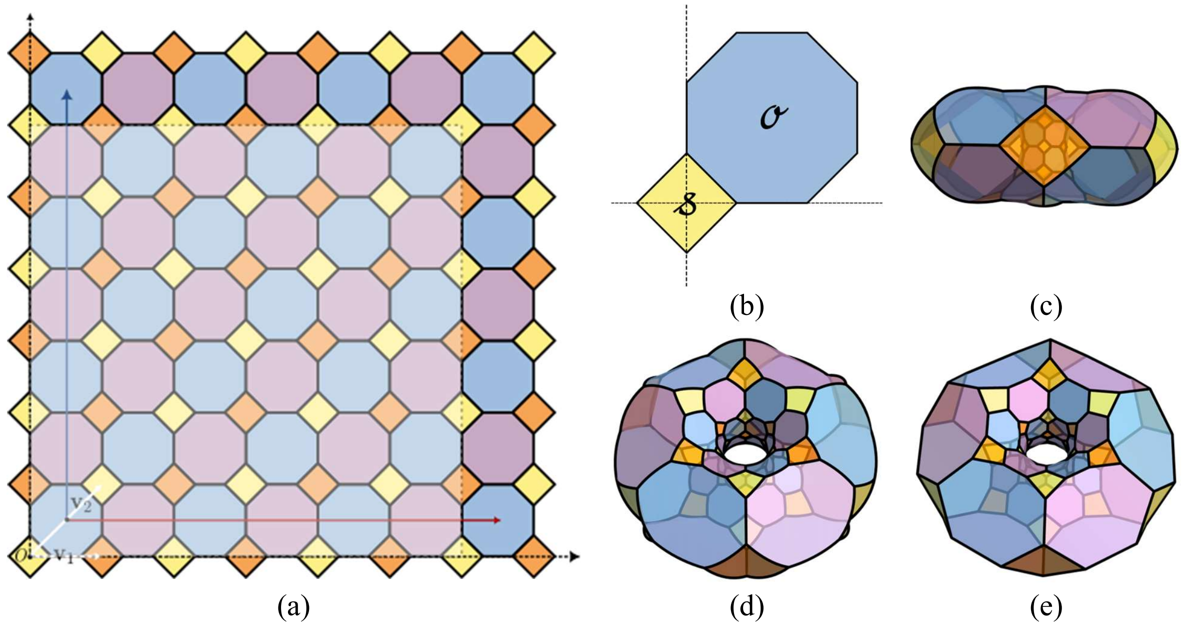


Figure 3.1 (a) A face-colored tiling $\mathcal{Q}$ of $\mathbb{R}^2$ and (b) two tiles used to generate $\mathcal{Q}$ by translations. (c – e) Images of the tiling $\mathcal{Q}_{\mathbb{T}}$ using the torus map with $a = \|6\mathbf{v}_1\|$ and $b = \|6(\mathbf{v}_2 - \mathbf{v}_1)\|$.

```
yellow = RGBColor[252/256,233/256,79/256,67/100];
blue = RGBColor[114/256,159/256,208/256,67/100];
orange = RGBColor[245/256,122/256,0/256,67/100];
pink = RGBColor[173/256,128/256,168/256,67/100];

v1 = N[{2+Sqrt[2],0}];
v2 = N[{2+Sqrt[2],2+Sqrt[2]}];

sqVerts = N[{{1,0},{0,1},{-1,0},{0,-1}}];
sqPolyYellow = Flatten[Table[Polygon[Table[sqVert+2i*v1+j*v2,{sqVert,sqVerts}]],{i,-4,4},{j,-1,7}],1];
sqPolyOrange = Flatten[Table[Polygon[Table[sqVert+(2i+1)*v1+j*v2,{sqVert,sqVerts}]],{i,-4,4},{j,-1,7}],1];

octVerts = N[{{2+Sqrt[2],1+Sqrt[2]},{1+Sqrt[2],2+Sqrt[2]},{1,2+Sqrt[2]},{0,1+Sqrt[2]},{0,1},{1,0},
    {1+ Sqrt[2],0},{2+ Sqrt[2],1}}];
octPolyBlue = Flatten[Table[Polygon[Table[octVert+2i*v1+j*v2,{octVert,octVerts}]],{i,-4,4},{j,-1,6}],1];
octPolyPink = Flatten[Table[Polygon[Table[octVert+(2i+1)*v1+j*v2,{octVert,octVerts}]],{i,-4,4},
    {j,-1,6}],1];

tiling = Graphics[{EdgeForm[{Black,Thickness[0.005]}],FaceForm[yellow],sqPolyYellow,FaceForm[orange],
   sqPolyOrange,FaceForm[blue],octPolyBlue,FaceForm[pink],octPolyPink},
  PlotRange->{{1.01*Norm[v1],7.01*Norm[v1]},{-0.021Norm[v1],4.23*Norm[v2]}}]
```

Code 3.1 Mathematica code to generate the face-colored tiling of $\mathcal{Q}$ in Figure 3.1(a).

Generating an image of a tiling of S_f from a tiling $\mathcal{T}$ then becomes a matter of replacing the coordinates of the vertices of the planar tiling with their images under the map f. Doing this directly, however, produces a tiling of S_f whose tiles have straight edges rather than smooth curves. To remedy this, we first subdivide each edge of $\mathcal{T}$ into smaller sub-edges by introducing new subdivision points; treat the new subdivision points as the new vertices of the tiling; and then apply f to these new vertices. We discuss this method for the tiling $\mathcal{Q}$ in the next illustration.

Illustration 3.2 We first remark that Q is an example of a *periodic* tiling since it possesses both horizontal and vertical translation symmetries. Consequently, it is compatible with a torus surface. For this illustration, we consider the surface $\mathbb{T}$ defined by the torus map with $a \coloneqq \|6\mathbf{v}_1\|$ and $b \coloneqq \|6(\mathbf{v}_2 - \mathbf{v}_1)\|$, the length of the red and blue vectors, respectively, in Figure 3.1(a). The additional step of subdividing each edge of the tiles in Q to obtain new vertices is captured in the first five lines of Code 3.2. These lines replace the definitions of **`sqVerts`** and **`octVerts`** in Code 3.1. Here, **`n`** refers to the number of new subdivision points, including the original endpoints, that will be introduced per edge. The greater this number is, the smoother the edges of the resulting tiling $Q_{\mathbb{T}}$ of the torus $\mathbb{T}$ will be. The succeeding lines of code define the torus map; employ this map to obtain the vertices of $Q_{\mathbb{T}}$; and render a 3D image $Q_{\mathbb{T}}$. Two perspective views of $Q_{\mathbb{T}}$ are shown in Figure 3.1(c – d). Compare these with Figure 3.1(e) corresponding to the case **`n = 1`** in which no new subdivision points are introduced to smoothen the resulting bounding curves of the tiles of $Q_{\mathbb{T}}$. We remark that Code 3.2 can be easily modified to produce a tiling of another compatible surface by simply replacing **`torusMap`** with another map.

```
n = 10;

oldSqVerts = N[{{1,0},{0,1},{-1,0},{0,-1}}];
sqVerts = N[Flatten[Table[oldSqVerts[[Mod[k,4]+1]]+(i/n)*(oldSqVerts[[Mod[k+1,4]+1]]-
           oldSqVerts[[Mod[k,4]+1]]),{k,0,3},{i,0,n-1}],1]];
oldOctVerts =  N[{{2+Sqrt[2],1+Sqrt[2]},{1+Sqrt[2],2+Sqrt[2]},{1,2+Sqrt[2]},{0,1+Sqrt[2]},{0,1},{1,0},
    {1+Sqrt[2],0},{2+ Sqrt[2],1}}];
octVerts = N[Flatten[Table[oldOctVerts[[Mod[k,8]+1]]+(i/n)*(oldOctVerts[[Mod[k+1,8]+1]]-
            oldOctVerts[[Mod[k,8]+1]]),{k,0,7},{i,0,n-1}],1]];

torusMap[a_,b_,{u_,v_}] := (1/(Sqrt[a^2+b^2]-a*Cos[u*2π/a]))*{b*Cos[v*2π/b],b*Sin[v*2π/b],a*Sin[u*2π/a]};

sqPolyYellow = Flatten[Table[Polygon[Table[torusMap[6*Norm[v1],6*Norm[v2-v1],sqVert+2i*v1+j*v2],
       {sqVert,sqVerts}]],{i,-4,4},{j,-1,7}],1];
sqPolyOrange = Flatten[Table[Polygon[Table[torusMap[6*Norm[v1],6*Norm[v2-v1],sqVert+(2i+1)*v1+j*v2],
       {sqVert,sqVerts}]],{i,-4,4},{j,-1,7}],1];

octPolyBlue=Flatten[Table[Polygon[Table[torusMap[6*Norm[v1],6*Norm[v2-v1],octVert+2i*v1+j*v2],
       {octVert,octVerts}]],{i,-4,4},{j,-1,6}],1];
octPolyPink=Flatten[Table[Polygon[Table[torusMap[6*Norm[v1],6*Norm[v2-v1],octVert+(2i+1)*v1+j*v2],
       {octVert,octVerts}]],{i,-4,4},{j,-1,6}],1];

Graphics3D[{EdgeForm[{Black,Thickness[0.01]}],FaceForm[{yellow,Opacity[0.72],
     Specularity[yellow, 30]}],sqPolyYellow,FaceForm[{orange,Opacity[0.72],Specularity[orange,30]}],
  sqPolyOrange,FaceForm[{blue,Opacity[0.72],Specularity[blue,30]}],octPolyBlue,
  FaceForm[{pink,Opacity[0.72],Specularity[pink, 30]}],octPolyPink},Lighting->"Neutral",Boxed->False,
 Axes->False]
```

Code 3.2 Mathematica code to generate the face-colored tiling of $Q_{\mathbb{T}}$ in Figure 3.1(d – e).

An alternative, more straightforward, and simpler way of generating a tiling of a surface in Mathematica is to use the **`Texture`** graphics directive within the built-in **`ParametricPlot3D`** function. The runtime of this second method is significantly faster and its implementation can reduce Code 3.2 to just a single line of code! We discuss how this is done in the illustration below.

Illustration 3.3 The second method involves plotting the parametric surface by supplying its parametric system of equations and then applying an image of the tiling as texture of the surface. In the case of the tiling Q and the surface $\mathbb{T}$ in Illustration 3.2, the implementation is found in Code 3.3.

```
ParametricPlot3D[torusMap[6,6,{u,v}],{u,0,6},{v,0,6},PlotStyle->Directive[Texture[tiling]],
 Lighting->"Neutral", Mesh-> None,Boxed->False,Axes->False]
```

Code 3.3 Mathematica code to generate the face-colored tiling of $Q_{\mathbb{T}}$ in Figure 3.2(a – e).

The **`torusMap[6, 6, {u, v}]`** input in **`ParametricPlot3D`** instructs Mathematica to draw the parametric surface defined by the torus map introduced earlier in Code 3.2. The bounds **`0`** and **`6`** for both the parameters **`u`** and **`v`** indicate that the lightly shaded patch of the tiling in Figure 3.1(a) which was supplied as the sole input to **`Texture`** must be treated as an image consisting of pixels located within a $[0,6] \times [0,6]$ grid. Mathematica will then automatically perform any rescaling necessary for the image. Note that with this given rescaling, both the constants a and b get the value 6. What Mathematica will do essentially is "wrap" the patch around the surface according to the rule defined by the supplied parametric equations. That is, it assigns the color or texture of the point (u,v) to be the color or texture of the image point $f(u,v)$ in the surface. This method produces the much smoother and rounder images of $\mathcal{Q}_{\mathbb{T}}$ shown in Figure 3.2(a – d). When the torus map in Code 3.3 is replaced by the cylinder map with $a = 6$, we obtain the images for $\mathcal{Q}_{\mathbb{C}_l}$ shown in Figure 3.2(e – g).

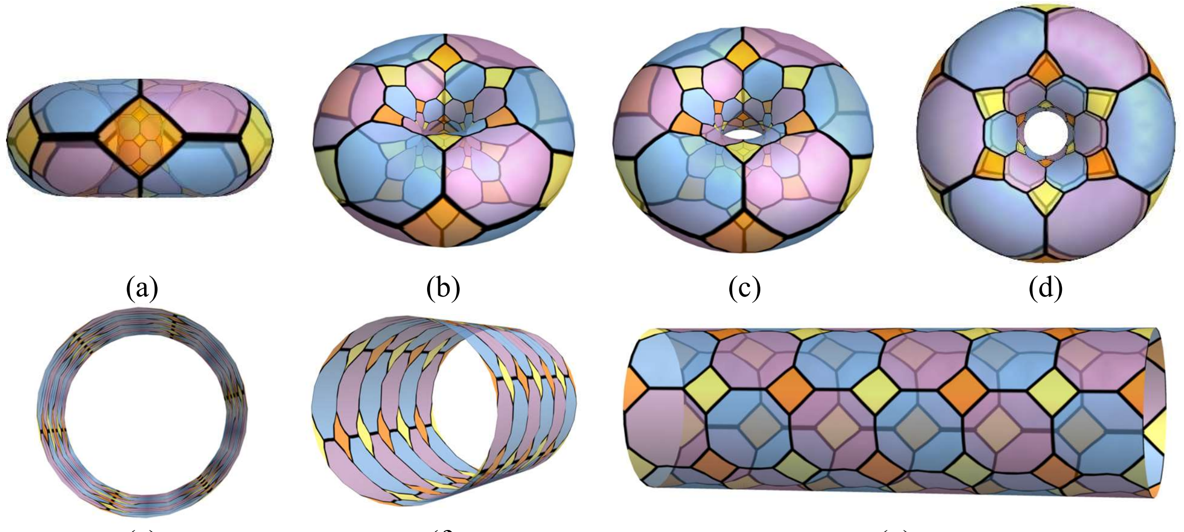

(a) (b) (c) (d)
(e) (f) (g)

Figure 3.2 Views of the truncated quadrille tilings of the point spaces (a – d) $\mathbb{T}$ and (e – g) $\mathbb{C}_l$.

We remark that while the second method discussed above is much faster and easier to implement than the first, the first provides more flexibility and adaptability when it comes to modifying components (vertices, edges, faces) of the tiling separately or independently from one another. This is because the second method requires a 2D image of the whole planar tiling to be used as basis for texture while the first converts each component of this planar tiling directly into a 3D graphics object. Another disadvantage of the second method is that it requires a high-resolution image for texture since the map that defines the surface stretches or deforms the image in various directions. Minimizing deformations is precisely the reason why we opt to use conformal maps, whenever possible, in lieu of ordinary maps.

4. Rendering Animated Tilings of Surfaces

Now that we have discussed completely how to generate an image of a tiling $\mathcal{T}_{S_f}$ of a surface in the previous section, we can now demonstrate how to render an animation that creates the effect of $\mathcal{T}$ morphing or transforming into $\mathcal{T}_{S_f}$. Mathematica offers at least two options for this to be accomplished. We can either create an animation with a slider that allows a user to manually control the animation or create an animated gif file that automatically runs the animation when opened. In either case, we create m still frames for the animation where the ith animation frame, which records the ith state in the entire morphing process, shows the tiling $\mathcal{T}$ on the parametric surface

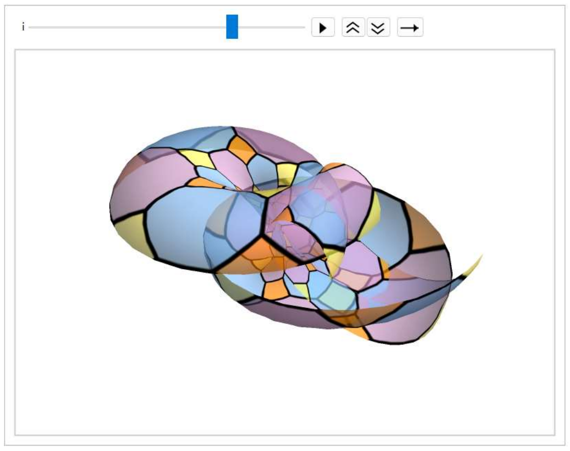
$$\left(1-\frac{i-1}{m-1}\right)\cdot(u,v,0)+\left(\frac{i-1}{m-1}\right)\cdot\left(f_x(u,v),f_y(u,v),f_z(u,v)\right), \qquad (4.1)$$

for $1 \leq i \leq m$. Observe that when $i = 1$, we just obtain $\mathcal{T}$ and when $i = m$, we obtain $\mathcal{T}_{S_f}$.

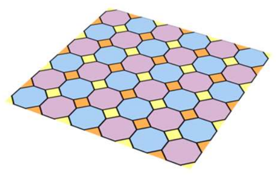
Illustration 4.1 To produce an animation showing Q morphing into $Q_{\mathbb{T}}$, we run Code 4.1, which simply implements the procedure discussed earlier. In this implementation, we created the function **`frame`**, which generates the ith animation frame, and used this as an input within Mathematica's **`Animate`** function to produce the animation with slider in Figure 4.1(a). The **`animation`** list in the penultimate line creates a list of eight frames, four of which are shown in Figure 4.1(b – e). When this list is called within the **`Export`** function, it transforms into an animated gif file which is saved in the user's directory for later retrieval and use. The optional argument **`DisplayDurations`** specifies the number of seconds delay between showing successive frames.

```
m = 8;

frame[i_] := ParametricPlot3D[(1-(i-1)/(m-1))*{u,v,0}+((i-1)/(m-1))*torusMap[6,6,u,v],{u,0,6},{v,0,6},
   Background->Opacity[0], PlotStyle->Directive[Texture[tiling]],Lighting->"Neutral",Mesh->None,
   Boxed->False,Axes->False];

Animate[frame[i],{i,1,m}]
Animation = Table[frame[i],{i,1,m}];

Export["animation.gif",animation,"DisplayDurations"->0.5]
```

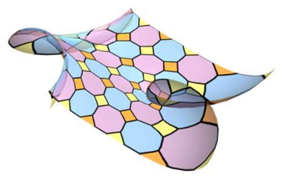
Code 4.1 Mathematica code to render an animation which shows Q morphing into $Q_{\mathbb{T}}$.



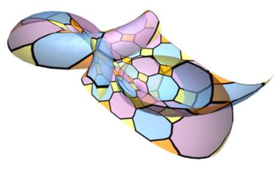
(a)



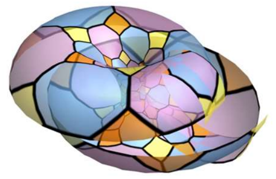
(b)



(c)



(d)



(e)

Figure 4.1 (a) Animation with slider showing Q morphing into $Q_{\mathbb{T}}$. (b – e) Four still frames captured from the animation in (a).

Acknowledgements The authors thank the Ateneo Symmetries in Algebra and Geometry Laboratory (SAGE) where this research was carried out. They are grateful to the Ateneo Department of Mathematics, the Office of the Dean of the School of Science and Engineering, the Ateneo de Manila University Research Council, and the Office of the Assistant Vice President for Research, Creative Work, and Innovation for support to present this work in the 2023 Asian Technology Conference in Mathematics.

References

[1] Brannan, D.A., Esplen, M.F., and Gray, J.J. (2012). *Geometry*, 2nd Ed. Cambridge: Cambridge University Press.

[2] Breda, A. and Dos Santos, J. (2018). *Spherical geometry and spherical tilings with Geogebra.* J. Geom. Graph. 22(2), 283–299.

[3] Breda, A. and Dos Santos, J. (2019). *Spherical tilings with Geogebra*. Resonance 24, 861–873.

[4] Davis, P.J. (1993). *Visual theorems*. Educ. Stud. Math. 24(4), 333–344.

[5] De Las Peñas, M.L.A.N. and Taganap, E. (2017). *Discovering new tessellations using dynamic geometry software*. Proceedings of the 22nd Asian Technology Conference in Mathematics, 153–162.

[6] De Las Peñas, M.L.A.N. *et al.* (2014). *Symmetry groups of single-wall nanotubes*. Acta Cryst. A70, 12–23.

[7] Feijs, L.M.G. (2018). *Torus and Klein bottle tessellations with a single tile of Pied de poule (houndstooth).* Bridges Stockhom 2018: Mathematics, Art, Music, Architecture, Education, Culture, 115–122.

[8] Frettlöh, D., Harriss E., and Gähler, F. *Tilings encyclopedia*. https://tilings.math.uni-bielefeld.de.

[9] Grünbaum, B. and Shephard, G.C. (1987). *Tilings and patterns*. New York: W.H. Freeman and Company.

[10] Kaplan, C.S. (2009). *Semiregular patterns on surfaces*. Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering, 35–39.

[11] Loyola, M.L. *et al.* (2014). *A quotient space approach to model nanotori and determine their symmetry groups.* AIP Conf. Proc. 1602(1), 620–626.

[12] Loyola, M.L. *et al*. (2015). *Symmetry groups associated with tilings on a flat torus*. Acta Cryst. A71, 99–110.

[13] Senechal, M. (1988) *Tiling the torus and other space forms.* Discrete Comput. Geom. 3 55–72.

[14] Sullivan, J.M. (2011). *Conformal tilings on a torus*. Bridges Coimbra 2011: Mathematics, Music, Art, Architecture, Culture, 593–596.

[15] Thomassen, C. (1991) *Tilings of the torus and the Klein bottle and vertex transitive graphs on a fixed surface.* Trans. Am. Math. Soc. 323(2), 605–635.

[16] Wichmann, B. *Tiling search*. https://tilingsearch.mit.edu/.

[17] Wolfram Research, Inc. (2023), Mathematica, Version 13.3. https://www.wolfram.com/mathematica.

[18] Žakelj, A. and Klancar, A. (2022). *The role of visual representations in geometry learning.* Eur. J. Educ. Res. 11(3), 1393–1411.