# MATH089 Project 2 - Patterns, Tilings, Crystals

Posted: 09/15/21

Due: 10/04/21, 11:55PM

- Formulate a short introduction to the problem of tiling space in 2D, 3D, with some typical applications
- Classify types of tiling
- Implementation of typical tiling approaches
- Discussion of your observations on tiling

# 1 Introduction

# 2 Models and methods

## 2.1 Two-dimensional tilings by regular polygons

### 2.1.1 Tiling with squares

```
Julia (1.6.1) session in GNU TeXmacs
```

```
∴ function square(x0,y0)
    x=[x0,x0+1,x0+1,x0,x0];
    y=[y0,y0,y0+1,y0+1,y0];
    plot(x,y,"-k");
  end;
```

```
∴ figure(1); clf(); square(0,0); axis("equal");
```

```
∴ function xtile(m,y)
    square.(0:m-1,y)
  end;
```

```
∴ clf(); xtile(10,0); axis("equal");
```

```
∴ function tile(m,n)
    xtile.(m,0:n-1)
  end;
```

```
∴ clf(); tile(8,8); axis("equal");
```

```
∴ savefig("/home/student/courses/MATH089/images/chessboard.eps");
```

```
∴
```

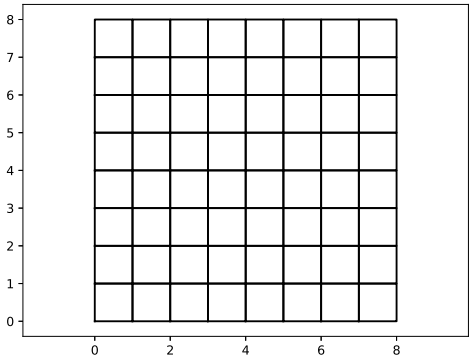


**Figure 1.** Tiling of the plane through squares

### 2.1.2 Tiling with triangles by translation

```
∴ function triangle(x0,y0)
    x=[x0,x0+1,x0+0.5,x0];
    y=[y0,y0,y0+sqrt(3.0)/2,y0];
    plot(x,y,"-k");
  end;
```

```
∴ clf(); triangle(0,0); axis("equal");
```

```
∴ function xtile(m,x,y)
    triangle.(x:x+m-1,y)
  end;
```

```
∴ clf(); xtile(10,0,0); axis("equal");
```

```
∴ function tile(m,n)
    xtile.(m,(0:n-1)*0.5,(0:n-1)*sqrt(3.0)/2)
  end;
```

```
∴ clf(); tile(8,8); axis("equal");
```

```
∴ savefig("/home/student/courses/MATH089/images/goboard.eps");
```

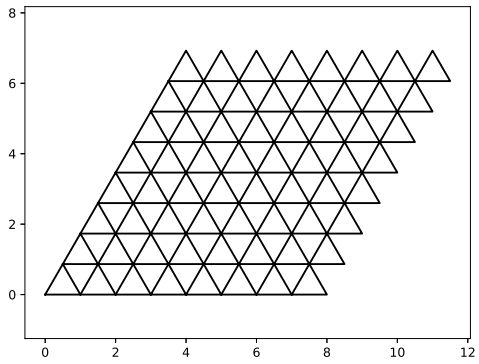


**Figure 2.** Tiling of the plane through triangles

### 2.1.3 Tiling with triangles by translation along normals and rotation or reflection

```
∴ clf(); triangle(0,0); axis("equal");
```

```
UndefVarError(:triangle)
```

```
∴ function PlotTri(X)
    x=[X[1,:]; X[1,1]]
    y=[X[2,:]; X[2,1]]
    plot(x,y,"-k");
  end;
```

```
∴ X=[0 1 0.5; 0 0 sqrt(3)/2];
```

```
∴ clf(); PlotTri(X); axis("equal");
```

```
∴ function TileTri(X)
    PlotTri(X)
    Xm = 0.5*(X[:,2]+X[:,3]); R=Xm-X[:,1]
    Xr = Xm + R; Y=copy(X); Y[:,1]=Xr; PlotTri(Y)
    Xm = 0.5*(X[:,3]+X[:,1]); R=Xm-X[:,2]
    Xr = Xm + R; Y=copy(X); Y[:,2]=Xr; PlotTri(Y)
    Xm = 0.5*(X[:,1]+X[:,2]); R=Xm-X[:,3]
    Xr = Xm + R; Y=copy(X); Y[:,3]=Xr; PlotTri(Y)
  end;
```

```
∴ clf(); TileTri(X);
```

```
∴ function TileTri(X,n)
    if (n==0) return end
    PlotTri(X)
    Xm = 0.5*(X[:,2]+X[:,3]); R=Xm-X[:,1]
    Xr = Xm + R; Y=copy(X); Y[:,1]=Xr; PlotTri(Y); TileTri(Y,n-1)
    Xm = 0.5*(X[:,3]+X[:,1]); R=Xm-X[:,2]
    Xr = Xm + R; Y=copy(X); Y[:,2]=Xr; PlotTri(Y); TileTri(Y,n-1)
    Xm = 0.5*(X[:,1]+X[:,2]); R=Xm-X[:,3]
    Xr = Xm + R; Y=copy(X); Y[:,3]=Xr; PlotTri(Y); TileTri(Y,n-1)
  end;
```

```
∴ clf(); TileTri(X,6);
```

```
∴ function TileTri(X,n,s)
    if (n==0) return end
    PlotTri(X)
    Xm = 0.5*(X[:,2]+X[:,3]); R=Xm-X[:,1]
    Xr = Xm + R; Y=copy(X); Y[:,1]=Xr;
    Yc=(Y[:,1]+Y[:,2]+Y[:,3])/3; Y=s*(Y-Xm)+Xm;
    PlotTri(Y); TileTri(Y,n-1)
    Xm = 0.5*(X[:,3]+X[:,1]); R=Xm-X[:,2]
    Xr = Xm + R; Y=copy(X); Y[:,2]=Xr;
    Yc=(Y[:,1]+Y[:,2]+Y[:,3])/3; Y=s*(Y-Xm)+Xm;
    PlotTri(Y); TileTri(Y,n-1)
    Xm = 0.5*(X[:,1]+X[:,2]); R=Xm-X[:,3]
    Xr = Xm + R; Y=copy(X); Y[:,3]=Xr;
    Yc=(Y[:,1]+Y[:,2]+Y[:,3])/3; Y=s*(Y-Xm)+Xm;
    PlotTri(Y); TileTri(Y,n-1)
  end;
```

```
∴ clf(); TileTri(X,3,1.0/3.0)
```

```
DimensionMismatch("dimensions␣must␣match:␣a␣has␣dims␣(Base.OneTo(2),␣Base.OneTo(3)),␣mu
```

```
∴
```

## 2.2 Tiling with concave polygons

### 2.2.1 Maltese cross

## 2.3 Three-dimensional tilings by regular polyhedra

### 2.3.1 Tilings by cubes

```
∴ function cube(x0,y0,z0,c)
    h=0.025;
    x=[x0+h; x0+1-h; x0+1-h; x0+h; x0+h]
    y=[y0+h; y0+h; y0+1-h; y0+1-h; y0+h]
    z=ones(5)*(z0+h)
    plot3D(x,y,z,c); plot3D(x,y,z.+(1-h),c);
    x=ones(5)*(x0+h)
    y=[y0+h; y0+h; y0+1-h; y0+1-h; y0+h]
    z=[z0+h; z0+1-h; z0+1-h; z0+h; z0+h]
    plot3D(x,y,z,c); plot3D(x.+(1-h),y,z,c);
  end;
```

```
∴ clf(); cube(0,0,0,"g"); cube(1,0,0,"r");
```

```
∴ function xtile(x0,y0,z0,m,c)
    cube.((0:m-1).+x0,y0,z0,c)
  end;
```

```
∴ clf(); xtile(0,0,0,5,"r");
```

```
∴ function xytile(x0,y0,z0,m,n,c)
    xtile.(x0,(0:n-1).+y0,z0,m,c)
  end;
```

```
∴ clf(); xytile(0,0,0,5,5,"r");
```

```
∴ function xyztile(x0,y0,z0,m,n,p,c)
    xytile.(x0,y0,(0:p-1).+z0,m,n,c)
  end;
```

```
∴ clf(); xyztile(0,0,0,5,5,5,"g");
```

```
∴
```

### 2.3.2 Tetrahedral tiling

Define a function to draw a tetrahedron given the coordinates of its four vertices

```
∴ function PlotTet(X)
    for k=0:3
      PlotTri3D( circshift(X,(0,k))[:,1:3] )
    end
  end;
```

Define a function to draw each face triangle in 3D

```
∴ function PlotTri3D(X)
   for k=0:2
     Y=circshift(X,(0,k))[:,1:2]
     x=Y[1,:]; y=Y[2,:]; z=Y[3,:]
     plot3D(x,y,z,"-k")
   end
  end;
```

Test the trianle drawing function

```
∴ X=[1 0 0; 0 1 0; 0 0 1];
```

```
∴ clf(); PlotTri3D(X);
```

```
∴ BaseDir="/home/student/courses/MATH089/images/";
```

```
∴ savefig(BaseDir*"P02FigTri.eps");
```

```
∴
```

Test the tetrahedron drawing function

```
∴ s3=sqrt(3); s23=sqrt(2/3); s13=sqrt(1/3);
```

```
∴ T=[0 0 -0.5 0.5; 0 s3/3 -s3/6 -s3/6; s23 -s13 -s13 -s13];
```

```
∴ clf(); PlotTet(T);
```

```
∴ savefig(BaseDir*"P02FigTet.eps");
```

```
∴
```

Define a function t otile 3D-space by tetrahedra

```
∴ function TileTet(X,n)
  end;
```

```
∴
```

**Figure 3.**

## 2.4 Polygons with fractal boundaries

Repeated subdivision of polygon edges combined with geometric transformations can lead to non-intuitive results. Consider the process depicted in Fig. 4, described mathematically by taking the coordinates

$$X = \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \end{bmatrix},$$

forming the end-to-end vector

$$R = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} - \begin{bmatrix} x_1 \\ y_1 \end{bmatrix},$$

and drawing as output four line segments, that are modifications of $R$ by scaling and rotation

$$X_a = X_1 + \frac{1}{3} M(0) R, X_b = X_a + \frac{1}{3} M\left(\frac{\pi}{3}\right) R, X_c = X_b + \frac{1}{3} M\left(-\frac{\pi}{3}\right) R$$

$$M(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

The above is implemented in the `ScRot` function:

```
Julia (1.6.1) session in GNU TeXmacs
```

```
∴ function ScRot(X0, s, θ, R)
      X0 + s*[cos(θ) -sin(θ); sin(θ) cos(θ)]*R
  end;
```

```
∴ ScRot([1; 0], 1/3, π/3, [1; 0])
```

$$\begin{bmatrix} 1.1666666666666667 \\ 0.28867513459481287 \end{bmatrix} \tag{1}$$

```
∴
```

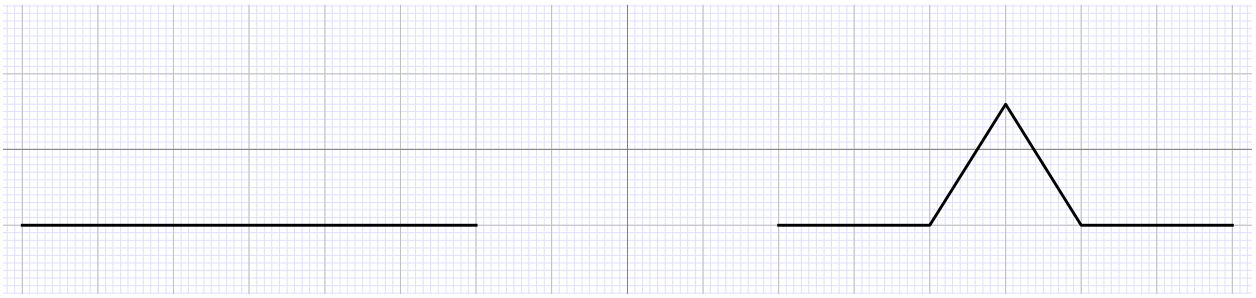

**Figure 4.**

```
∴ function PlotSeg(X)
  end function
```

### 2.4.1 Koch snowflake

```
∴ function star(x0,y0,c)
      b0=0.5; z=0.0
      c0=sqrt(3.0); c1=c0/3; c2=c0/6
      x = [z,b0,-b0,z]
      y = [c1,-c2,-c2,c1]
      plot(x,y,c,x,-y,c);
  end;
```

```
∴ clf(); star(0,0,"r-"); axis("equal");
```

```
∴ star(0,0,"w-");
```

```
∴
```

```
∴ function Koch(m)

  end;
```

```
∴
```

# 3 Results

# 4 Discussion and conclusion