Understanding Geometric Pattern and its Geometry Part 6 – Using Geometer's Sketchpad for designing sizeable geometric projects

Mirosław Majewski

mirek.majewski@gmail.com
New York Institute of Technology, School of Arts & Sciences,
Abu Dhabi campus, UAE

Abstract: *This document aims to explain a few aspects of dealing with geometric objects in Geometer's Sketchpad that are not entirely related to geometry. We discuss how one can reduce the amount of data needed to create bulky geometric constructions without losing the quality and accuracy of the final design. We use a real complex geometric pattern to demonstrate our conclusions.*

Introduction

Designing geometric patterns with GSP (Geometer's Sketchpad) or any other geometry software[1] means two things – creating geometric objects and dealing with the software's non-geometric features. We know a lot about geometry, but we often get into trouble with its non-geometric aspects.

While drawing complex geometric patterns using GSP, I observed that sometimes my computer slows down, and it takes a longer time to redraw the image on the screen. Some of my students, using older machines, reported that their computers crashed while producing such designs. In this paper, we deal with this issue. We will show how one can design even very bulky geometric patterns without slowing the performance of their computers and still obtaining good quality designs.

While working with the program, GSP keeps its sketch entirely memory- resident. No part of the drawing, selecting, or dragging objects require access to the file whatsoever. The file on the disk is completely independent and is used for long-term storage and data transfer. Thus, in reality, we have two documents – one the computer file and another one the in-memory sketch.

Generally, GSP files aim to be small. They contain a mathematical description of objects used in the sketch. The in-memory sketches aim to be as fast as possible. While loading a file from the disk, GSP will allocate, compute, and store the equations and parameters of each object most recently sampled to rapidly redraw the picture without having to perform any arithmetic every time it needs to redraw even a small portion of the window. Thus, we have a reasonably small file on the computer disk, while the in-memory sketch is a maximal object.

[1] Most of the information in this paper is valid also for GeoGebra.

How much information is needed for an object?

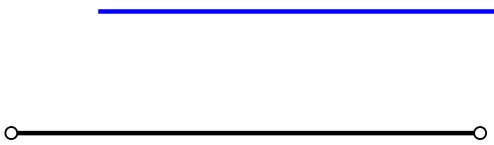
Let us examine some of the objects we deal with in Geometer's Sketchpad.

A point from the geometric point of view is a small dot. A point means much more in GSP (and any other geometry software) – its coordinates, color, size, and flag for object's features – hidden or visible. All this information is stored in the in-memory sketch. There is also a visual representation of a point – two small circles, one black and another smaller using a specific color.

Thus, if we have a few thousand points in our design, we already have a considerable amount of hidden data. But points are not the bulkiest objects.

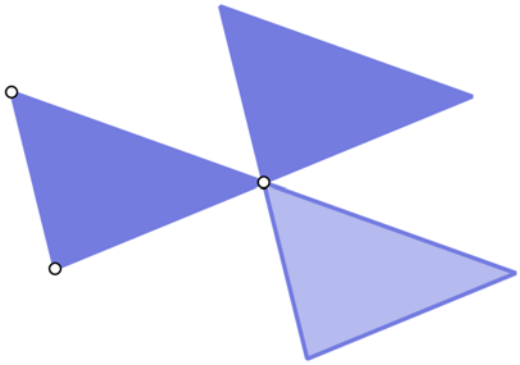
A segment is a collection of two points used to create it, the equation of the line passing through these points, the color of the segment, its thickness, and its style. This means that for every segment, we have about 3 times more information than for a point. But there is something hidden that we didn't notice. The information about the endpoints of a segment is doubled. Here is an example:

The black segment below was created by connecting the two points shown here. The blue segment is a translation of the black one. I changed color to distinguish them. The top segment does not have visible endpoints, but they still exist. This means that if we draw a segment, information about its ends is stored twice – (1) in coordinates of points used to create a segment and (2) in the description of the segment (look at the blue part).



All this means again a lot of hidden, sometimes unnecessary information is stored in the in-memory sketch.

A polygon should be understood as a collection of many objects: vertices, sometimes (!) segments joining these vertices, and filling of the polygon area. This is a lot. Here are also a few things that we should consider. While creating a polygon, we use points for its vertices, but while creating a polygon, we also create coordinates of its vertices. The next illustration shows what we really create.



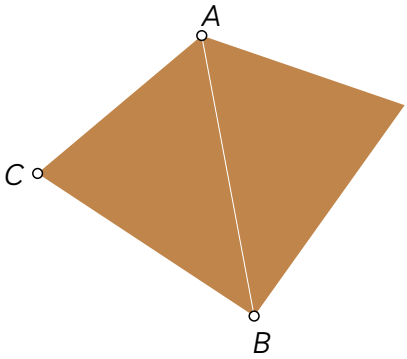
Anatomy of a polygon

In this drawing, the left triangle was created using the three points shown here. The top-right polygon is its translation, and the drawing right-down is another translation of the original polygon. For the last one, we used 50% opacity to show its real structure.

These two right images show that any polygon itself contains information about its vertices even if they are not visible, and (this is important) the polygon has in its definition also edges (let us call them **frames**). They are clearly seen on the right-bottom triangle.

The above analysis concludes that a simple triangle like the one on the drawing contains a large amount of data outside of our awareness. Thus the question is: how can we reduce all this information to absolutely necessary elements? This way, we could reduce the number of objects and data that, in reality, we do not need and thus make in-memory sketches smaller and faster displaying their content. Here are some conclusions:

1. We could reduce the size of our constructions and consequently in-memory sketches by not drawing edges of polygons. They are already there. Unfortunately, they have the same color as the whole polygon, which cannot be changed.
2. If we want to reduce the number of points, we can consider translating or rotating polygons and segments without their endpoints.
3. We can also remove the frames of polygons like the one seen in the drawing below. But this possibility has some drawbacks.

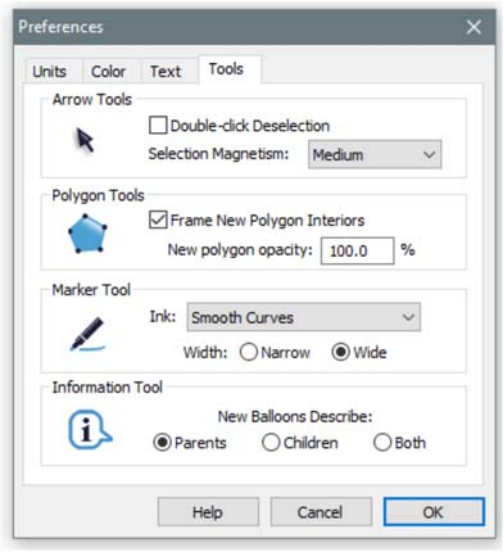


Frames of polygons

In this illustration, the left triangle was created by drawing a polygon with vertices ABC. The right one is its reflection about a line passing through points A and B. In both cases, we removed the frames of each polygon. The drawback is this tiny gap between them.

This means that we can remove these frames without any problem in some cases, and in some others, we will get unpleasant artifacts in the form of a gap between polygons.

How do we fix this issue? There are two ways.

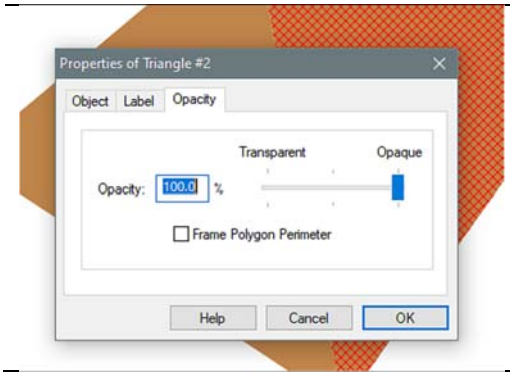


The preferences panel

In GSP's Edit menu, at the very bottom, there is the [Preferences] option. The illustration to the left shows its appearance in the Windows version. Here we can change several parameters but the most important for us, at the very moment, are polygon tools. Here we can decide if we want polygon frames created for us and no white gaps between polygons, and here we also determine if we want to have 100% opaque (non-transparent polygons).

This way, we decide the features of our polygons globally. But of course, later, we can select any polygon and change its own specific features.

Important – there is a bug in GSP. Suppose that we created an unframed polygon with the Frame New Polygon option checked. Then each copy of it obtained by translations, rotations, or reflections will get a frame.



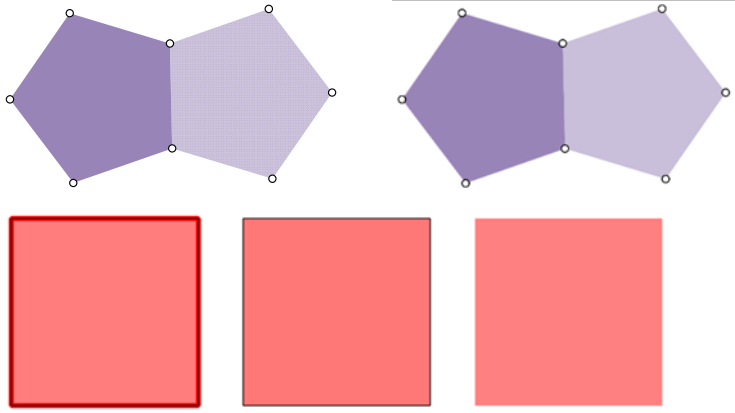
The properties of a polygon

By the right-mouse click on any polygon, we get several options, and one of them is the Properties panel. Here we also can apply opacity and framing but only for the selected polygons. All other polygons will not be changed at all.

All this discussion brings us to the two questions: should polygons be created with frames or without, transparent or not? You can decide on your own what you prefer and see how this fits into your needs.

In all my works, when I deal with tangent polygons, I use polygons with frames. If I do not have tangent polygons or a few tangent only, I prefer to have polygons without frames, and these few I change to framed by hand. I rarely use polygons with lower than 100% opacity.

The opacity of polygons brings us to a few unexpected problems. Transparent polygons at, say, 50% opacity are great at showing an image behind the polygon. But they are awful when we take an image from GSP, paste it into MS Word, and then print the Word file. In such cases, I make a right-click on this image in Word, and I ask to save it as a PNG file. I then take this PNG file and copy and paste it into the Word document, replacing the original image.



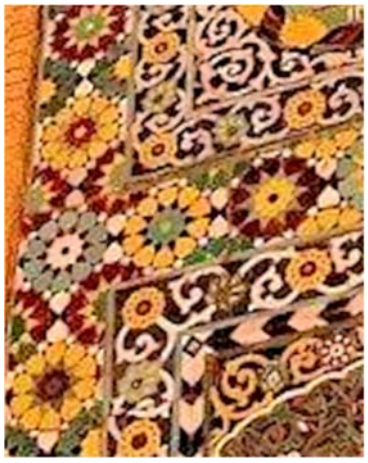
Left – original drawing from GSP (obtained by cut and paste), right is its copy created by MS Word (save as Picture). We do not see any difference between these two images in an MS Word document, but we see the difference after printing it.

In this image, we show three 50% transparent squares (from the left): a polygon with thick frames and edges drawn by us; a polygon without frames but with edges drawn by us; a polygon with no frames and no extra edges. The left one uses maximum data, the right one uses the minimum of necessary data.

A few more conclusions – we can avoid framed polygons if we use contrasting colors, one dark and another light. This way, the gaps between polygons with two contrasting colors will still exist, but they will be less visible. Tangent polygons with the same dark color should always be framed. If we use contrasting colors, we do need to draw extra edges of polygons at all. A good solution is to choose one color as a background for the whole design and draw it in the form of a large polygon, usually a rectangle.

Custom color palettes

The original set of colors in GSP may not be enough for a more sophisticated design. Thus we can create our own color scheme/palette and save it as a GSP tool. Here I show step by step how one can do it.



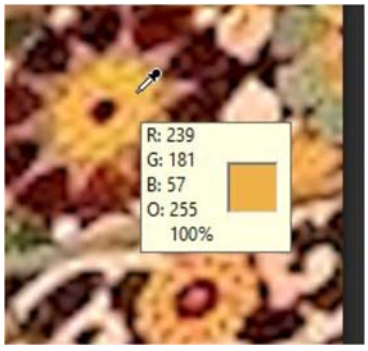
A sample photo

The photo shown here has a very bad quality but an excellent color scheme. We will try to develop a color scheme based on this photo.

For this reason, we will need any graphics program for photo editing. It can be the Paint program from Windows, or Photoshop, or anything else. In most of these programs, we have a Color Picker tool that looks like the one shown in the next image. Positioning it on a particular area will display the color parameters of the given pixel.

Here we see that the point has color parameters R:239, G:191, B:57, and O:255. This is the amount of each color component for the given point in the image: red, green, blue, and opacity. Ignore the last one.

Now, we have to choose color parameters for a few different and characteristic for this image places. Write them down on a piece of paper and go with them to GSP. In Sketchpad, we can create, for example, a rectangle divided into smaller cells, and each of them fill with a polygon. To each of these polygons, we apply color parameters taken from our photograph. Such an object can be saved as a new tool. The next drawing shows the color palette that I created for this image. I selected each polygon and through the menu



Display > Color > Other

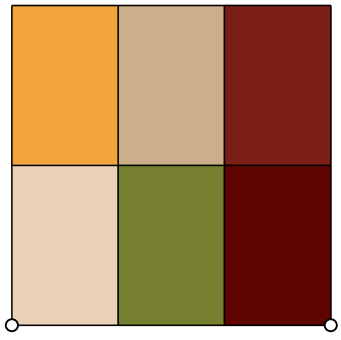
I was able to change the color of each polygon.

In this color palette, the following colors were used:

[R:242, G:165, B:59], [R:203, G:175, B:138], [R: 123, G:30, B:22]

[R:235, G:209, B:184], [R:119, G:128, B:49], [R:94, G:5, B:1]

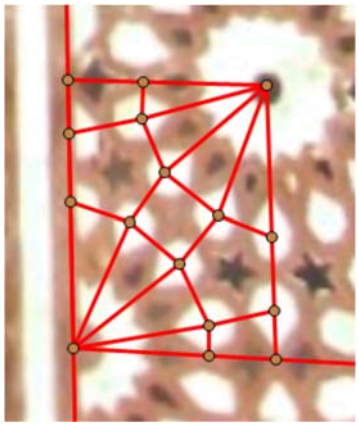
If we want to use this color palette in our design, we have to put it on the screen and select an object in our design, then with [shift] down, select the desired color from the palette and again through the menu:



Display > Color > Other

apply the new color. We do not need to type color parameters again. They are already there. This way, we can change the colors of any number of objects in our design. Important – while changing colors, the undo operation does not work. We cannot get back old colors by using the ctrl-z key.

A real example using the user-created color palette

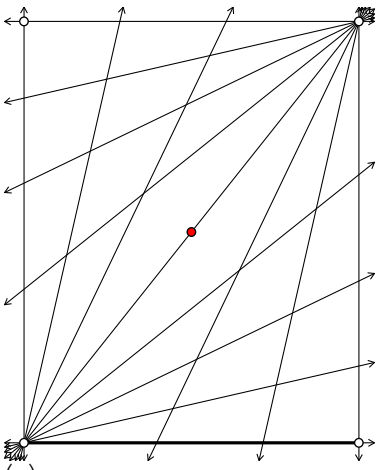


Pattern from Fatehpur Sikri

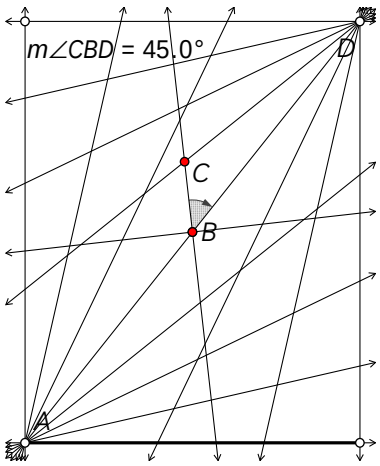
In this example, we will use an existing pattern from Mughal architecture, from Fatehpur Sikri. I got it from Richa Raut, an Indian architect, already suggesting a tessellation for it. This is our starting point. We will develop this pattern using the color palette created a while ago.

The pattern itself is very interesting. It contains regular stars with 14 vertices and pentagonal stars with one symmetry line only. A significant feature of this pattern is the middle star with four arms. It is an object with four-fold symmetry. Thus we may assume that it can be wrapped in a square as it is shown here.

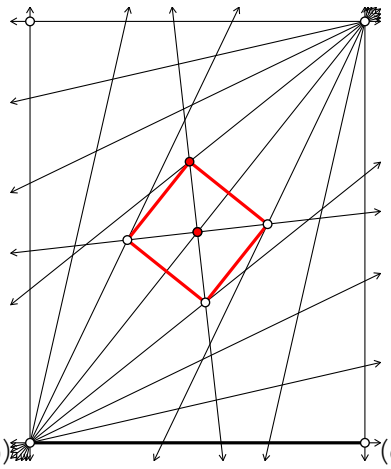
The next drawings show the step-by-step construction of the contour, tessellation, and pattern.



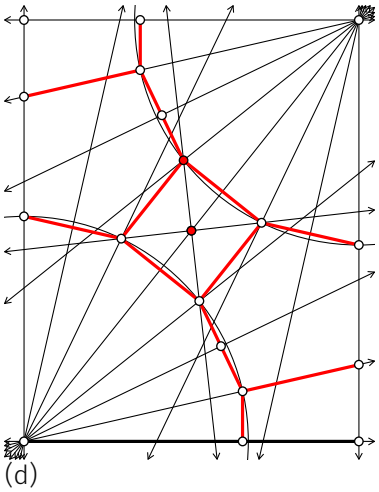
(a)



(b)



(c)

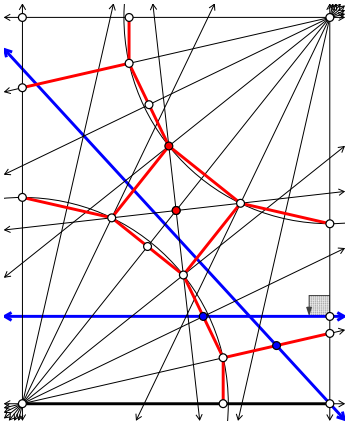


(d)

In drawing (a), we start from the thick horizontal segment and two vertical lines perpendicular to it. After dividing the left-bottom angle 90 degrees into 7 equal parts (in GSP, use 90/7 without bothering about the angle's accurate value), we take the 4^{th} section line to produce the contour's top edge. In the same way, we divide the top right angle into 7 equal parts.

In drawing (b), we rotate the contour's diagonal about 45 degrees in each side. This way, we get the central square (c). The red point is the center of rotation.

Drawing (d) shows how to complete the tessellation. The two pentagons are not regular, but they have one symmetry line. Can you draw it?



(f)

Here we show how one can approach the construction of the pattern on this tessellation. The slant blue line passes through the midpoint of the pentagon's edge (blue point) and the contour's corner. This part is evident from the photograph.

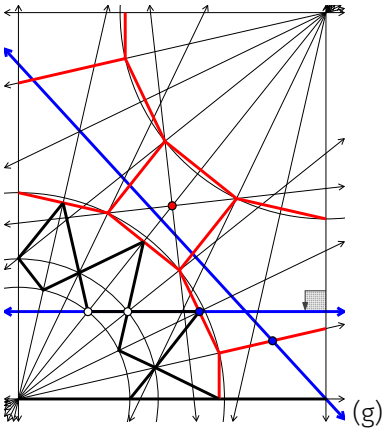
The second blue line can be drawn in a few different ways. Here I made it horizontal, but it can be slightly slanted in both ways. The photograph does not show the pattern accurately.

The selection of both blue lines determines the construction of the pattern.

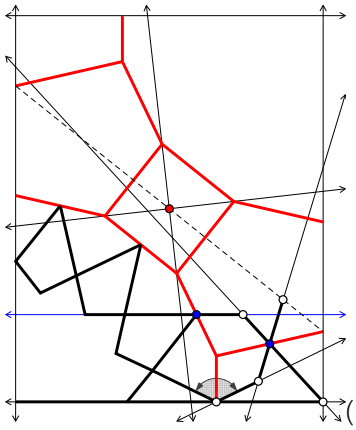
Drawing (g) shows how we can create the quarter of the large star.

In (h), we see how we could develop the pattern in half of the trapezium (bottom-right).

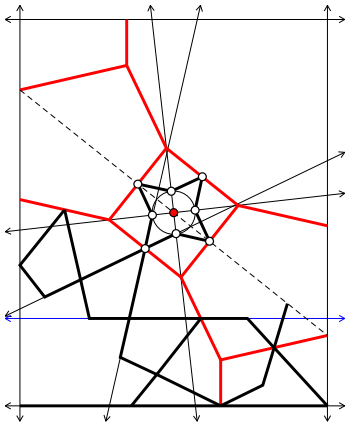
In (i), we see how we can draw the pattern inside the central square.



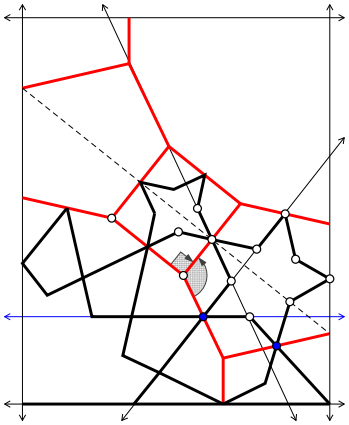
(g)



(h)



(i)

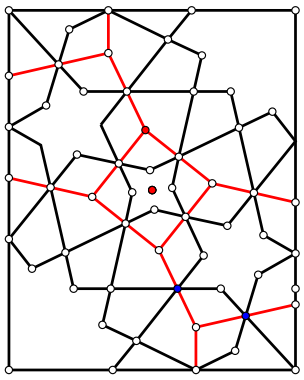


(j)

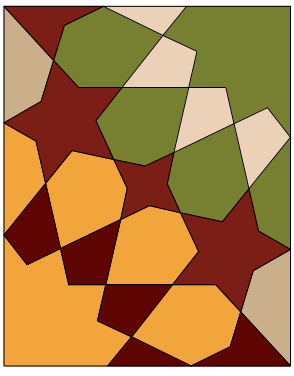
In this drawing (j), we see how we can develop the pentagonal star. It will never be regular. The reason for this fact is that the two angles shown here are different.

Below – the three drawings show the complete pattern with tessellation; the pattern using black segments for edges of polygons; the pattern using only polygons with contrasting colors.

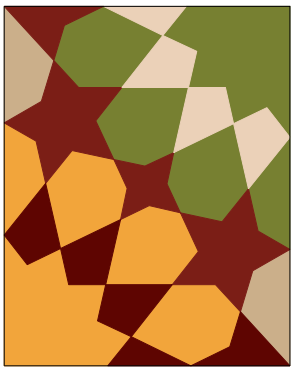
It is essential to notice that the number of segments in drawing (k) is not very large, but we may get a large set of edges when we make a large pattern out of it. Thus using the version with framed polygons only (drawing (l)) we can produce a design with a significantly smaller number of elements – no points, no edges, frames, and polygons only.



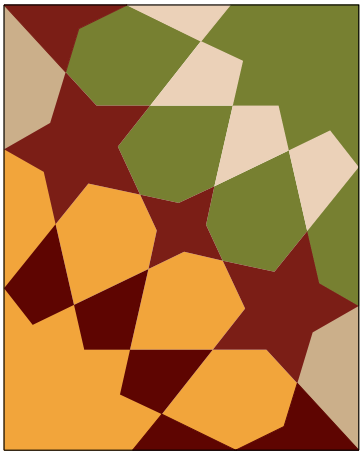
(j)



(k)



(l)



An image with an absolute minimum of data

This image uses tangent polygons with contrasting colors. There are no borders created by the user.

Polygons inside the contour are without frames. Polygons tangent to the contour edges have frames. There are 39 objects in this creation. In the above images, the central pattern (k) has 116 objects, and the right one (l) has 54 objects.

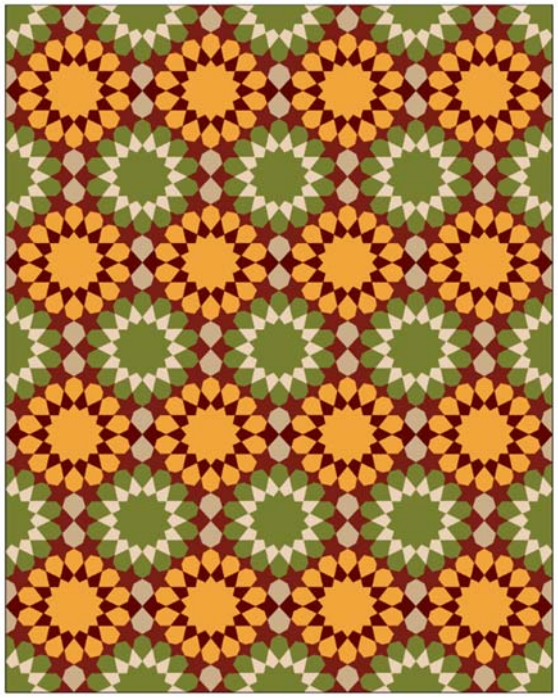
Thus if we create a pattern using 8x8 copies of a template, we will produce objects with

8x8x(number of objects in the template) – 7x7

This gives us numbers 575, 815, and 1807. The 7x7 is for removing edges of the contour and each it's copy.

Here we have the final version of a pattern using framed polygons only. This is important to notice that after creating a large design from the template using any type of polygons, we do not replicate anything that is not visible.

Another important fact – a geometric pattern with a large number of elements copied into an MS Word document will slow it down. The best solution is to paste it into MS Word, ask MS Word to save it as a picture. We will get a bitmap – not vector graphics. Then replace the image in the MS Word document with the bitmap file. In the MS Word file and then in the PDF file, we will not see a difference between these two images.



Size of graphic files

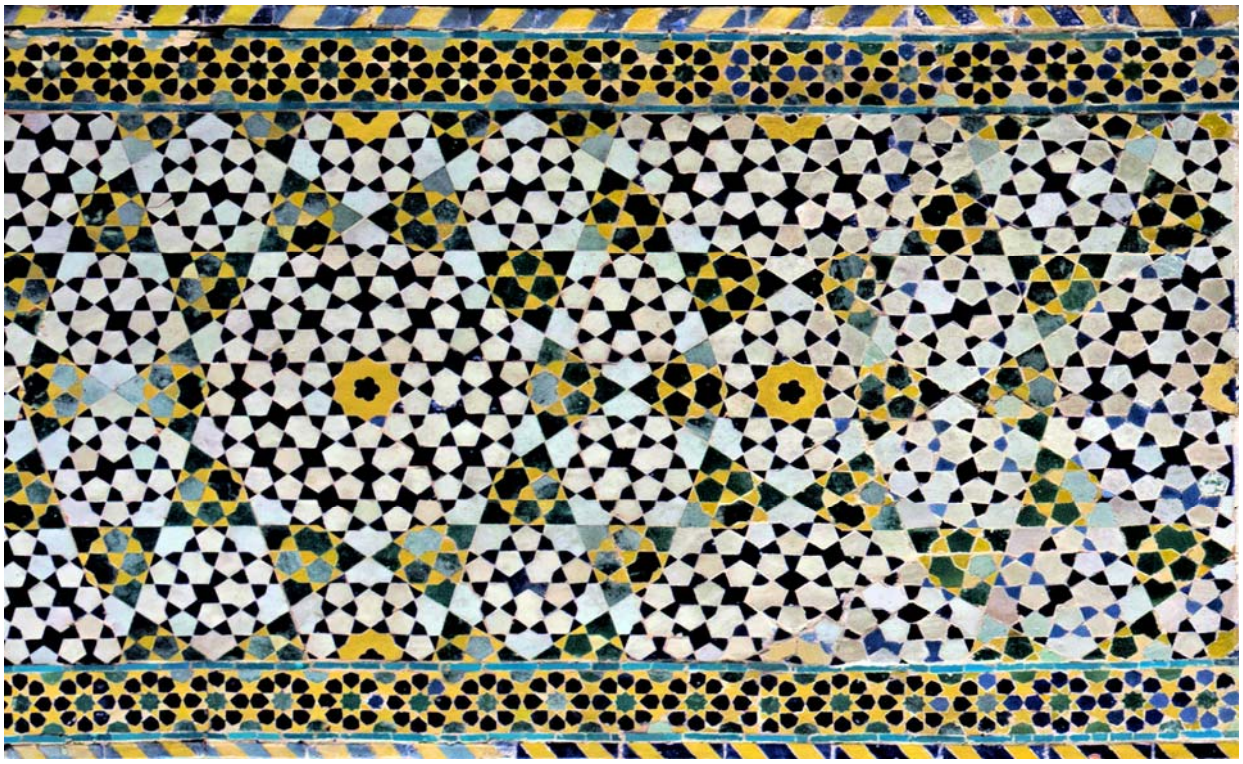
GSP graphics are always in vector format. This means the size of graphics copied from the GSP screen is dependent on the number of elements used in the construction and their parameters. In designs with a lot of details, this can be a huge number. While saving an image from MS Word as a picture, we produce a bitmap image. The size of such an image depends on the number of pixels in it. The bitmap file can be much smaller than the original GSP vector file for the constructions with a vast number of elements and complex parameters.

For smaller designs, it can be much better to use images copied directly from the GSP notebook. They are smaller than their bitmap versions.

GSP for Windows can save a construction in the EMF format. This is also a vector graphics file. Lines are very crisp, but they can be bulky for large designs.

A bulky example of Persian origin

In this example, we will show how one can create a very complex geometric pattern following the principles discussed in this paper.



A decagonal pattern with two color schemes and a large number of elements

The photo shows a typical Persian design using two different color schemes. We have dark greenish and yellow ceramic tiles and areas with black and very light blue slightly pinkish ceramic tiles.

In western literature, design from the photo is described as a mysterious pattern with some symbolic features and as a self-similar fractal-like object. A person with some basic knowledge of fractals will notice that there is nothing fractal-like in it. In the next few pages, we will reconstruct this design using a new custom-made color palette. We will show that there is nothing mysterious in it, only pure decagonal geometry. Our main goal will be to produce an optimal design with possibly the lowest number of elements. Thus we will simplify everything possible. We will cut most of the unnecessary details.

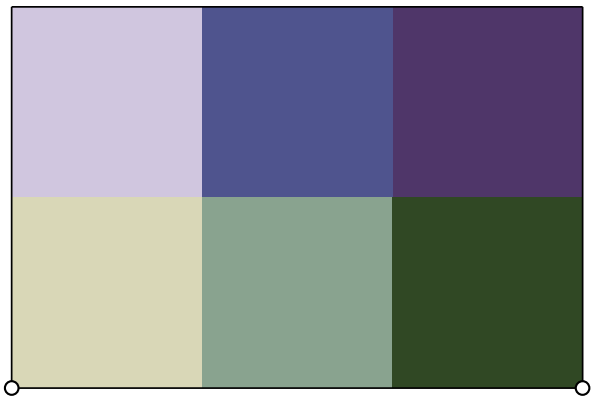
Pattern analysis

By analyzing the colors used in this design, we can notice that we have four significant areas: kites (dark green and yellow parts), regular pentagons, halves of regular pentagons, and large stars (black and bluish

tiles). These figures form the so-called upper-level pattern. These figures are filled with Persian style patterns using elements that we created already in our earlier works (see, for example, [2] or [3]). This is the lower-level pattern. Here are the following steps in this project:

1. Create two distinct color palettes using contrasting colors.
2. Construct the overall structure of the template – contour, tessellation, and upper-level pattern.
3. Construct the lower-level pattern for each mentioned shape – kites, pentagons, etc. There are only four of them.
4. Create a large top-level pattern and fill it with the lower-level pattern.

Color palettes



For this project, we will use two color palettes – one with a blue component with colors:

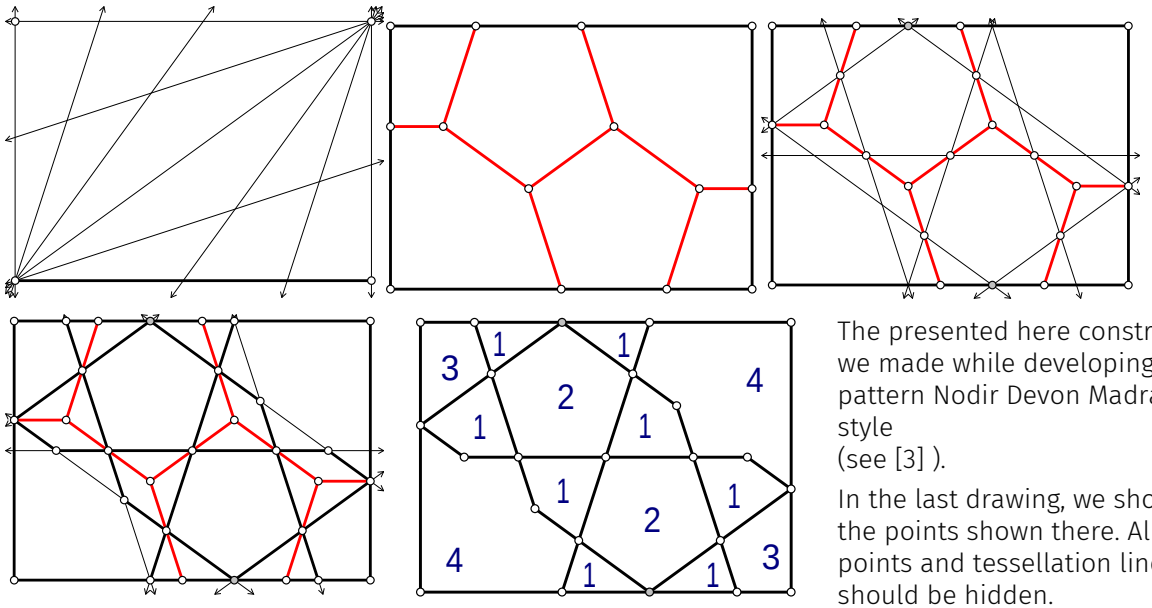
[R:208, G:198, B:223],
[R:79, G:84, B:142],
[R:79, G:54, B:105],

and another one with green component:

[R:217, G:215, B:183],
[R:137, G:163, B:143],
[R:48, G:72, B:36].

Template for the upper-level pattern

This part is easy. We did it already a few times on various occasions ([3]).



The presented here construction we made while developing the pattern Nodir Devon Madrasah style
(see [3]).

In the last drawing, we should keep the points shown there. All other points and tessellation lines should be hidden.

We have here only 4 different shapes numbered from 1 to 4.

Multilevel patterns in Persian art

A particular feature of Persian art are designs with two or more patterns overlapping each other. In Iran, we often see designs where a large pattern forms a framework for another more detailed pattern. Usually, there are two levels. We will call them upper-level and lower-level. However, it is possible creating patterns with three levels. These designs will be dense-packed with a pattern that may be very difficult to render in any type of material.

The example discussed in this text contains two levels. The upper-level pattern is shown in the above drawing. The lower level will be created on the next few pages.

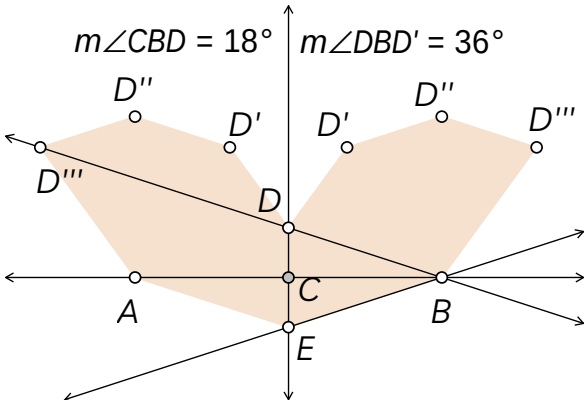
It is essential to notice that nodes of the upper-level pattern usually are centers of stars or rosettes of the lower-level pattern.

Tools for this design

In this paper, we will make intensive use of two kinds of GSP custom tools. There will be basic tools representing copies of elements of the lower-level pattern and tools for pattern filling large blocks of the upper-level pattern.

A tool to draw tangent decagons

This design will need a tool to draw parts of two tangent decagons provided that their centers are given.



Start with line AB and create the midpoint C. Rotate the line 18 degrees up and down about point B. Create the line perpendicular through point C. This way, we will obtain points D and E. All other points in this drawing are rotations of the point D around A and B at an angle of 36 degrees.

NOTE – usually, in similar situations, I construct two tangent decagons. I this example, we do not need complete decagons. We will need only parts of them. It is also essential to notice that, shown here, polygons are required only for a brief moment, and we can delete them. External points D''' and D'' can be deleted (deleted, not hidden) if we do not need them.

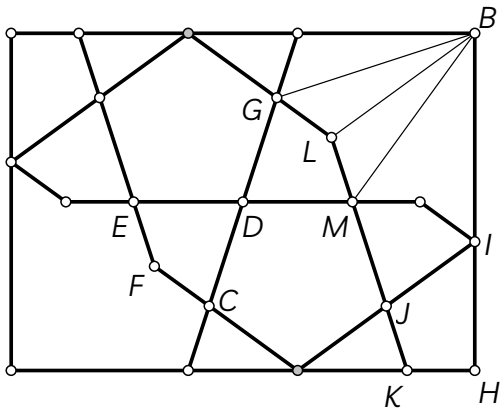
Hide or delete

In Sketchpad and in any other geometry software, we are often tempted to delete unwanted objects. Sometimes after deleting an item, the whole construction is still not affected, but in some other cases deleting a single point or a segment may destroy the entire construction. All geometry programs use the parent-child model. Every element that was used to create a new object is a parent for this object. For example, a line B created as a parallel to a given line A and passing through the given point *a* is a child of A and *a*. These two objects are parents of line B.

Elements that are parents of other objects cannot be deleted by deleting these objects. But we can always hide them, and this will not affect the rest of the construction.

Angles and lengths

The drawing presented below displays all necessary angles and lengths needed in further steps of this design. I assume that the reader knows how to construct the golden section of a segment.

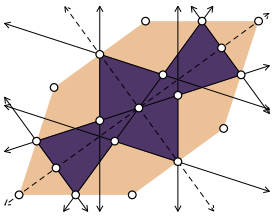
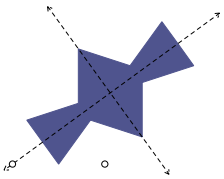
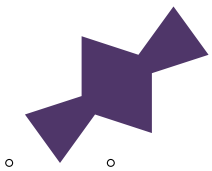
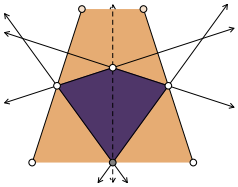
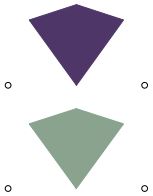
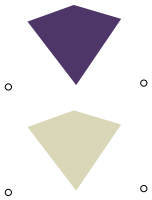


$m\angle CDE = 72°$
$m\angle DEF = 72°$
$m\angle EFC = 144°$
$m\angle EDG = 108°$
$m\angle HIJ = 54°$
$m\angle IJK = 108°$
$m\angle LMB = 54°$
$m\angle BLM = 108°$
$\frac{m\,\overline{IJ}}{m\,\overline{JK}} = 1.61803$

Tools for basic elements

There are only two basic elements that we need for creating lower-level. These are the black shapes in the light areas and yellow kites in the yellow/green areas. Thus we have two shapes: a small kite and a flower vase-like shape. The first one is the fill for a long triangle tile or a tall trapezium. The second one is the fill for the long hexagon. All other elements can be considered as a background.

In the table, we show steps in constructing each of them.

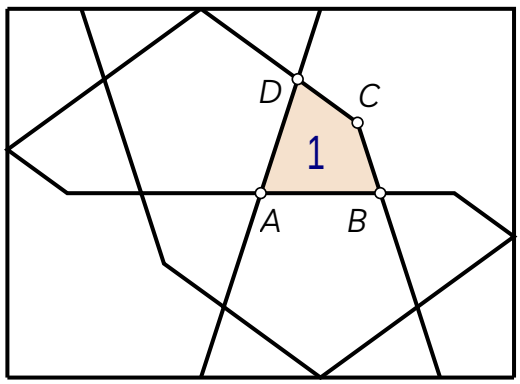
Construction	Framed shapes using multiple polygons	Unframed shapes using single polygon	
			The **vase-like shape** was created on a long hexagon with all edges equal and lines forming 54 degrees angle. We need two tools: a tool with four identical parts (middle). The symmetry lines separate them. Each polygon in this tool is framed, with thin frames, to avoid gaps between them; a tool with one unframed polygon (right).
			We create four tools for **kites**. Kites in the middle are split into two tangent framed triangles (thin lines) Kites in the right are created in the form of one unframed polygon.

The shapes in the middle column have a slightly different color than those in the right column. This way, we can easier show where we use them. Later we can adjust their colors if we want this.

Now we can start developing patterns for the shapes in the upper-level design.

A tool for the pattern inside the kite (number 1)

Our first step is to produce a pattern for the quadrilateral ABCD and other identical shapes.



$m\angle CBA = 72°$ $m\angle BCD = 144°$
$m\angle DAB = 72°$

Left – the upper-level pattern with angles for the kite ABCD.

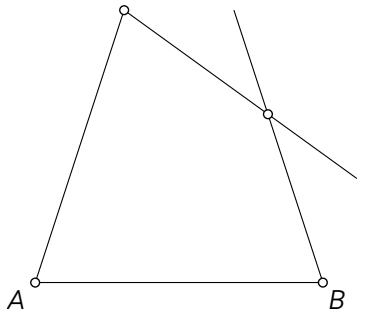
Below – construction of the kite and the pattern for it.

Start with segment AB and rotate it about points A and B according to the angles shown here (a). Use twice the tool to draw tangent decagons (b)

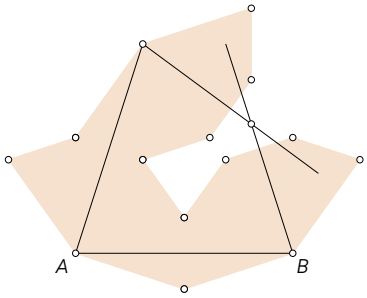
Fill the beige polygons with two versions of the small kite. These crossed by the kite's edge should be made out of small kites with two framed polygons. These inside the large kite should be one-piece polygons unframed (c).

The drawing (d) shows how to fill the gap in the wide corner of the large kite.

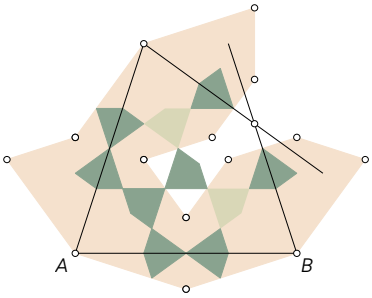
All polygons outside the large kite should be deleted. Some but not all points outside can also be deleted.



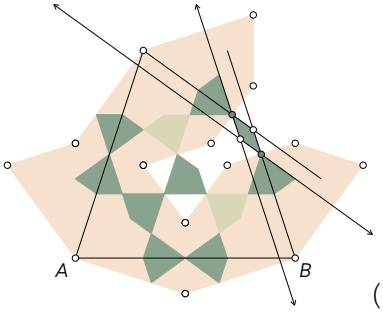
(a)



(b)



(c)



(d)

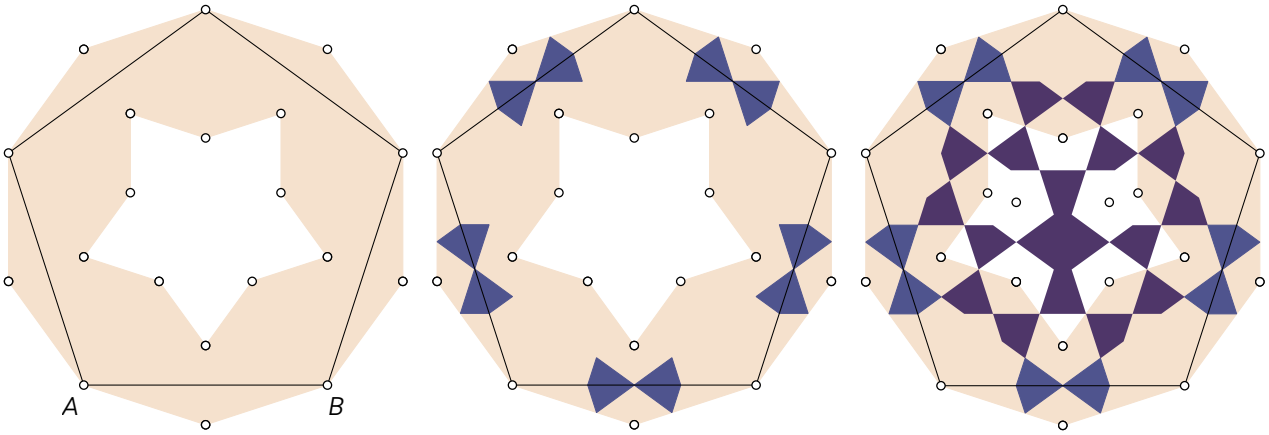

(e)

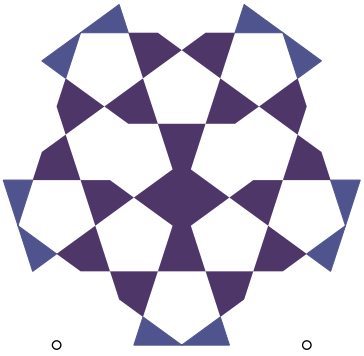
Left – the final pattern for the large kite. This is important – along the kite symmetry line, we also have small kites created out of two tangent triangles. Why we did this?

Select the two points below the pattern (e), and then all polygons and create a new tool 'kite pattern.'

The pattern for large kites is the simplest one. Now we can design the pattern for pentagons. This is a slightly more complex task.

A tool for the pattern inside the large pentagon (number 2)





Construction of pattern for the large pentagon

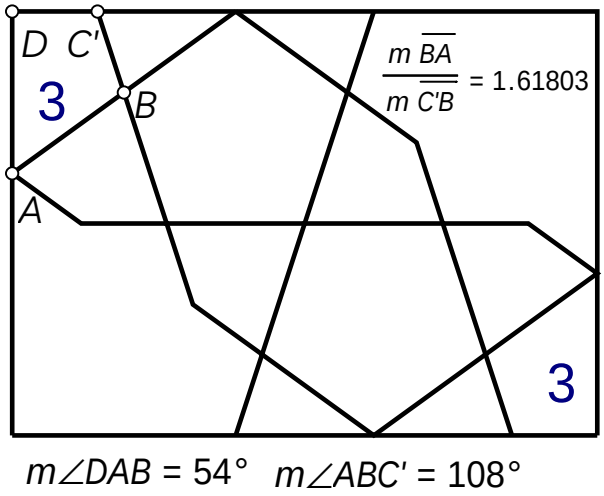
Start from segment AB and by rotating it 108 degrees about its endpoints, create the pentagon. For each edge of the pentagon, use the tool to draw tangent decagons. This is what is shown in the first image. Then fill it with kites and vase-like shapes. Start filling the central white area by inserting the bottle shape (one framed polygon).

Delete all polygons that are outside of the pentagon. Hide all points leaving only points AB. We need them to create the tool for filling with the pattern for the pentagon.

Select the two points shown here and then select the pattern and create a new tool, 'pentagon pattern.'

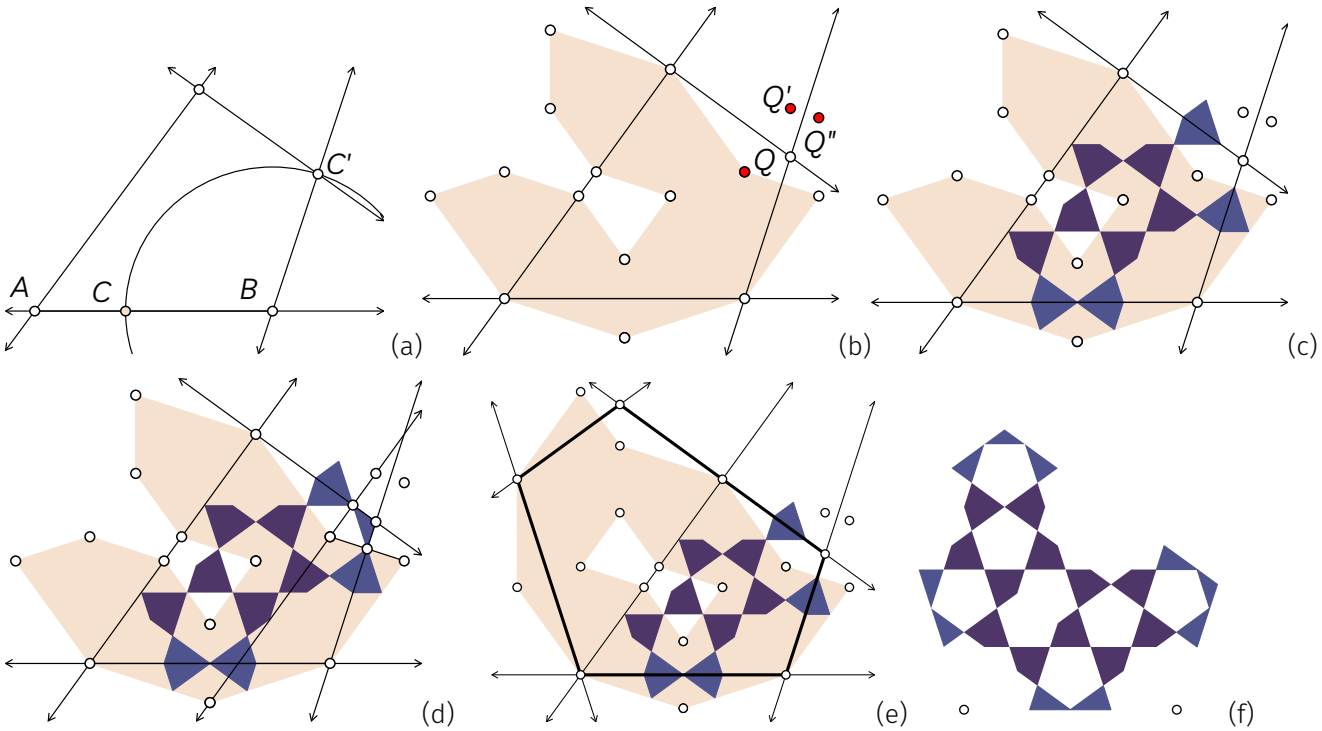
Next comes the pattern for shape in the top-left and bottom-right corners of the upper-level pattern – the figure with number 3.

A tool for pattern filling corners (number 3)



The corner polygon is half of a figure that can be easily obtained by cutting a regular pentagon. However, the lower-level pattern filling it will be different than for the pentagon.

The drawing shows all necessary angles and the ratio of the two edges DE and EF. We can recognize here the golden ratio. Thus we have an easy way to construct the shape and the pattern for it.



A brief description

In the drawing (a), we start with the two points A and B. Point C is the point marking the golden section of AB.

In (b) point Q' is a mirror reflection of the point Q, and point Q" is a mirror reflection of Q'.

Drawing (d) shows how to fill the space in the top-right corner.

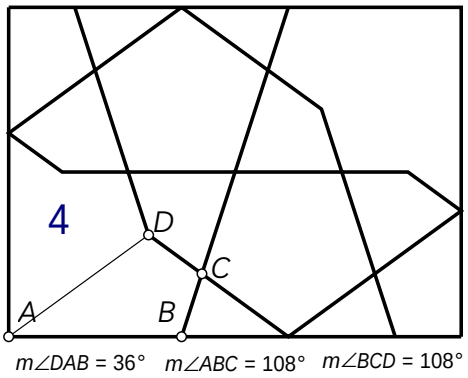
Drawing (e) shows the complete shape. We developed a pattern for only half of it. In the same way, we develop a pattern for the other half.

Drawing (f) is the final pattern for the filling shape number 3 and its reflected part. All kites around the edge are framed. All other kites are single polygons, unframed.

Select the two points shown here and then all polygons and create a new tool, 'corner pattern'.

A tool for a pattern for the big star (shape number 4)

The last part of this long example is constructing a pattern for the big star (shape number 4). This design can be developed in a few different ways. We can split the big star into 10 segments and create a pattern for one of them. We can also develop a complete design for a quarter of the star.

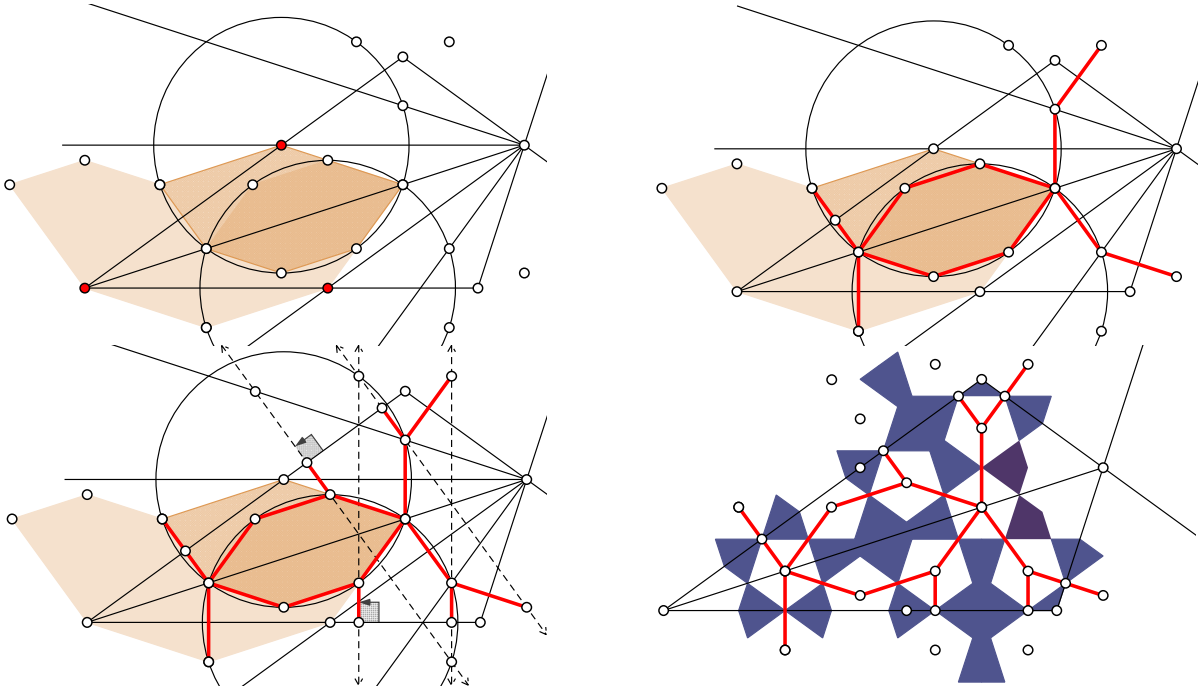


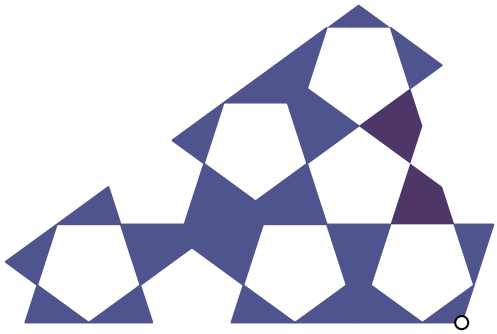
Angles and properties of the big star

If we split the star into 10 identical kites, we will get a polygon that can be easier to fill with a pattern. It can be convenient to start from segment AB and then obtain the remaining edges by rotations 36 and 108 degrees.

There will be a few shapes cut along their symmetry lines. Thus we will have to use for them two or four framed polygons.

Below – construction of the long kite and constructing a tessellation using twice the tool for creating tangent decagons.





Above - left finishing tessellation, right the tessellation and the pattern. See how we created the pattern for the corners.

Left – complete pattern for the long kite. Each of the light blue figures uses two or four framed polygons. Why? There are only two figures made out of a single unframed polygon.

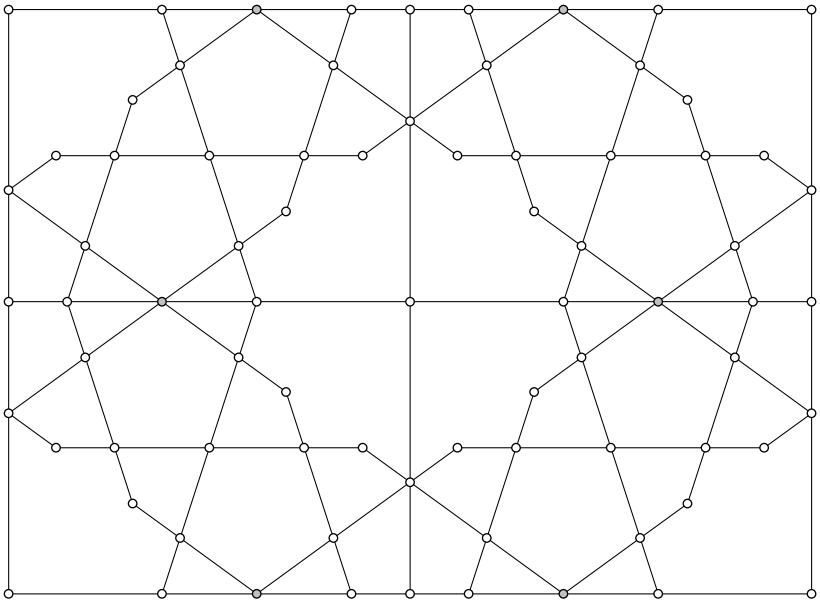
Select the two points with all polygons and create a new tool, ‘long kite pattern’.

Assembling the complete design

Assembling the complete design is the very last and the least complicated task. We start from a large template created out of four copies of the upper-level pattern template. This is our master template. Now we have to fill it with all the elements that we made.

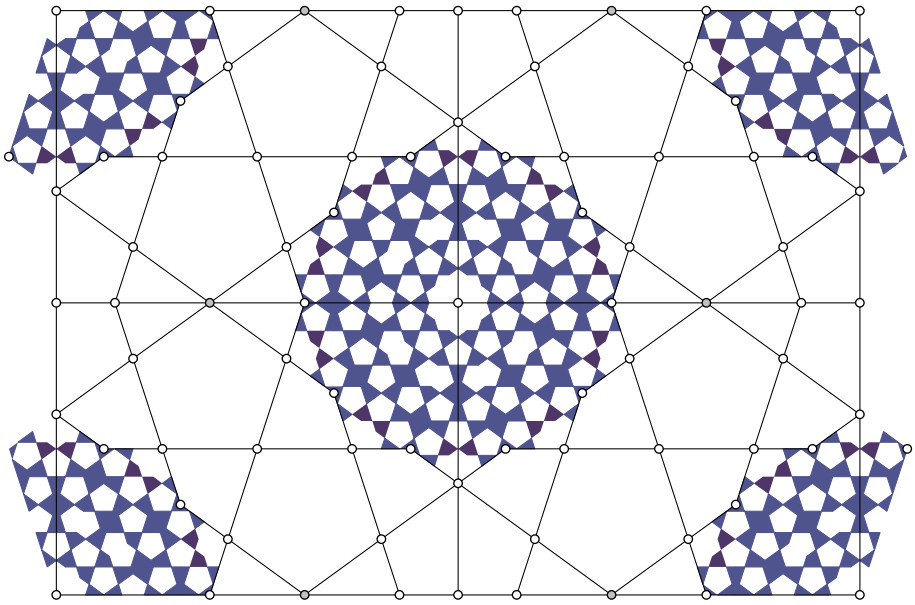
STEP 1 – complete the upper-level pattern

We have here four copies of the upper-level pattern template created at the beginning of this project.



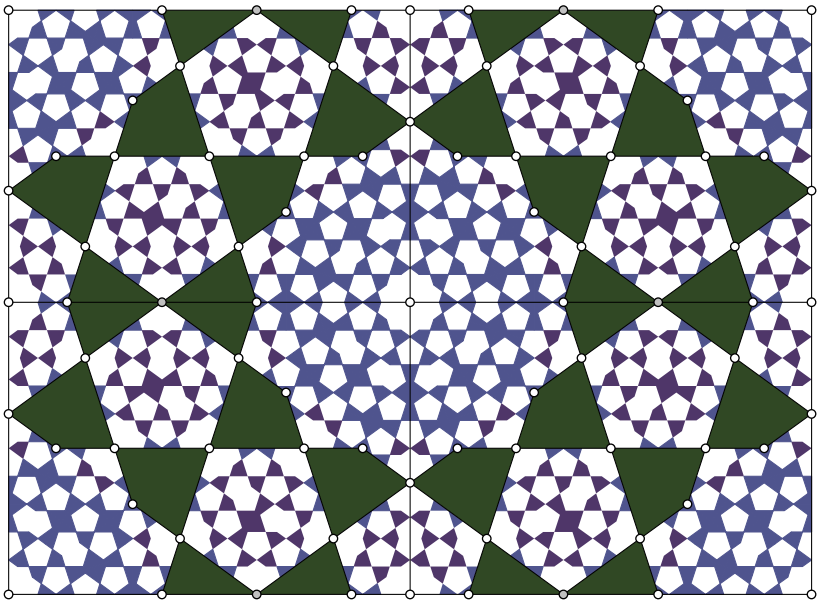
STEP 2 – add fills for the big star

We use the 'long kite pattern tool' to fill the center and all four corners.

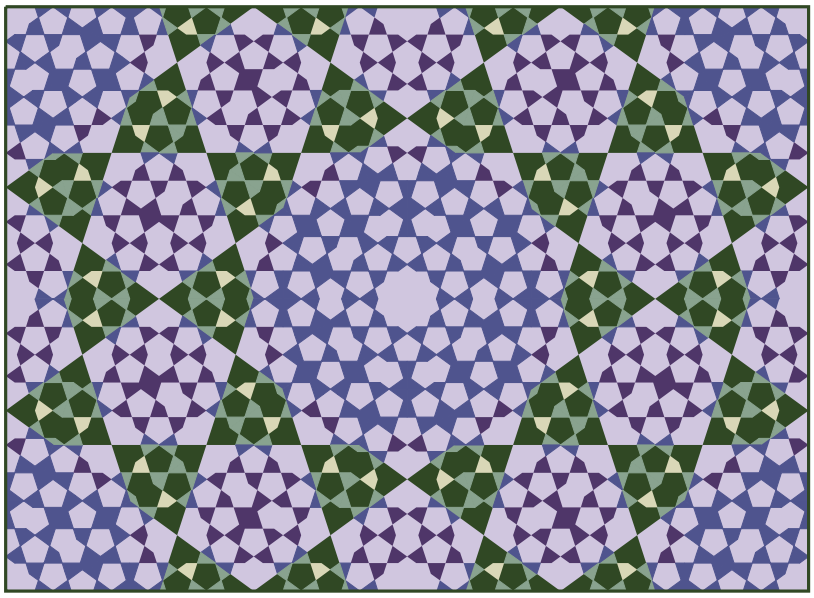


STEP 3 – fill pentagons with the pattern

Before filling kites with the pattern, create there the dark green background from the second color palette. Use unframed polygons. Now, use the 'pentagon pattern' tool to fill each pentagon shown on the drawing. Note how the pattern for pentagons was inserted. Its direction is different than in the original design.



Step 4 – fill remaining shapes with the pattern, hide the upper-level pattern, hide all points



The design we created contains 1784 elements. If we use all framed polygons, then this number will be at least 4 times higher. The most significant saving we get through unframed polygons and one global background for the whole design. The light blue color is one rectangle – not a few hundreds of pentagons, etc. Each small kite also has one background (see step 3).

In this design, we use polygons on a few layers. The top layer contains small shapes. The middle layer form the dark green kites. The bottom layer is one large light blue rectangle.

Final comments

In this text, we discussed methods for optimizing large geometric patterns. The final result looks exactly like the pattern created without using the techniques presented here. However, it contains much fewer elements. The example discussed in this paper is relatively simple. The number of elements may grow to 50 thousand or more elements – segments and polygons in more complex examples. Such a quantity of objects is still acceptable for Geometer's Sketchpad. But it will slow down its work. However, if we copy a GSP design to any other vector graphics program, we may wait a long time to render our graphics.

In this text, we did not discuss the differences between vector graphics files and bitmap files. Such a discussion can be found in any introductory textbook of graphics design or an appropriate Wikipedia page.

References[2]

[1] Majewski. M. (2020). *Practical Geometric Pattern Design: Geometric Patterns from Islamic Art*. Kindle Direct, Independently published (February 10, 2020).

[2] Majewski. M. (2020). *Understanding Geometric Pattern and its Geometry (Part 1)*, eJMT, vol. 14, Nr 2, pages 87-106.

[3] Majewski. M. (2020). *Understanding Geometric Pattern and its Geometry (Part 2) – Decagonal Diversity*, eJMT, vol. 14, Nr 3, pages 147-161.

[4] Majewski. M. (2021). *Practical Geometric Pattern Design: Decagonal Patterns in Persian Traditional Art*. Kindle Direct, Independently published (March 29, 2021)

[2] Papers [2] and [3] can be downloaded from https://www.researchgate.net/profile/Miroslaw-Majewski