

Tikz For More Than Just Math

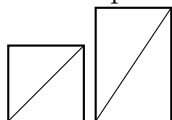
Daniel Alvarez

April 03, 2017

1 Introduction

The sole purpose of this article is to demonstrate the use of *Tikz* for projects other than math. Usually, *Tikz* would be used by professionals in math and the many various fields of science and engineering to make diagrams. These are usually technical diagrams, for example user manuals, build-it-yourself instructions and the diagrams on many exams that we have all experienced at some point. But, unfortunately for those of you who'd like a demonstration for using *tikz* for "mathy" projects this is not the article for you. This article will introduce how *tikz* can be used to make patterns by repeating objects and shapes. In particular my favorite shape the, triangle. I will make the argument that triangles are the only closed shape that we "humans" need to make any other object, at least draw any other object.

Squares: What do I mean by this ridiculous claim? Well think about it, a square can be made up if two right angle triangles, therefore any rectangular object (a square is a rectangle with equal sides) can be broken up into triangles of either equal or unequal length sides. For example:



As you can see, by drawing a diagonal line we can break a square or rectangle into two similar triangles.

Circles: A circle can be approximated by drawing straight lines (edges) to many points (vertices) a circle can be made or at least something very near to a perfect circle. Fun fact, any circle you see on a screen is a not perfect circle. Zoom into any curved surface and you will see that it is not "perfectly" curved. This is because circle's in math are continuous and have infinite lines that can divide them. Computers understand 1's and 0's which are discrete therefore a computer as we know it cannot understand the concept of continuity. Added to that a computer cannot ever calculate numbers to infinity and that includes the infinite decimal numbers between whole numbers like 1.00000..." and whole number infinities. Example:

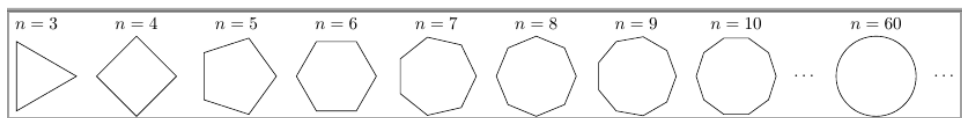


Figure 1: Polygon to n

From the figure above it is clear that as more vertices are added the more the polygon "looks" like a circle. Draw a line from each vertex to the center of the "circle" and it would make a circle approximated by triangles. In reality, or at least in a computer's reality, this is how 2D and 3D objects are made. This concept is heavily used in the video game and graphics industries.

1.1 Making a triangle

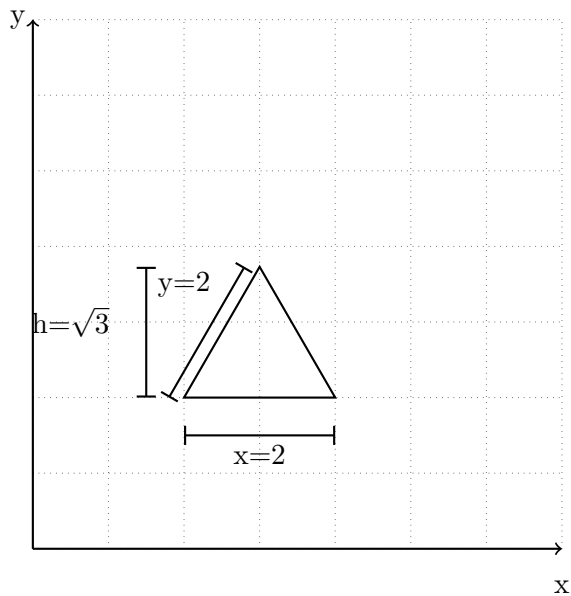
Now that you’ve been convinced that triangles are awesome let me now demonstrate how to make one. This article will focus on equilateral triangles.

1.1.1 Equilateral Triangle

An equilateral Triangle is one with equal length sides that meet at 60 degree angles. Simple enough however to draw it in Tikz requires a little knowledge of geometry, particularly one Theorem and that is good ole Pythagorus’ Theorem.

$$x^2 + y^2 = h^2$$

In the case of our triangles x is our base, y is our height and h the hypotenuse. To make these triangles the I used an x = 2 and h = 2 which implies that the height is $\sqrt{3}$ or 1.72 which is the approximation used in the loops to make multiple triangles.



code for one triangle:

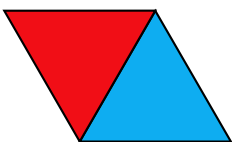
```
%triangle
\draw [thick] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
```

2 Adding Triangles

In this section we’ll look at shapes and patterns that can be made by adding together equilateral triangles together at angles.

2.1 By Edges

To have one triangle rest against another perfectly the second triangle will have to be drawn about 60 degrees to any one point. This will add the triangles together by their edges. For example:

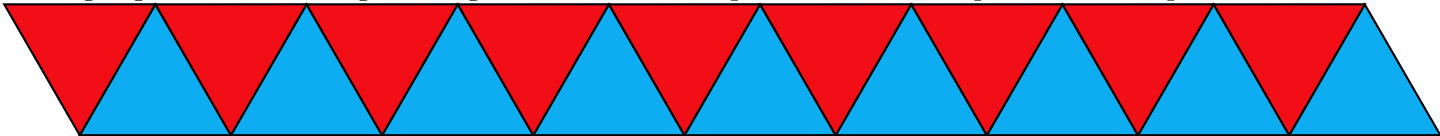


code:

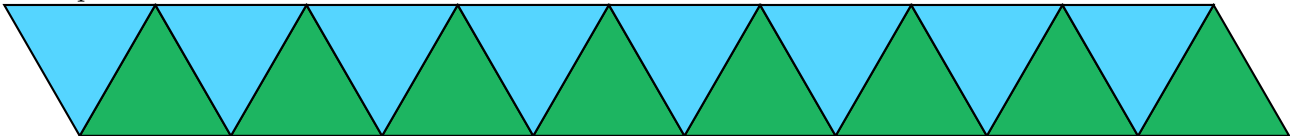
```
\draw [thick,fill=cus-blue1] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\draw [thick,rotate around ={60:(0 ,0)},fill=cus-red1] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
```

From the figure above, the red triangle is the same as the one in blue but it is rotated about the point (0,0) which in this case is the lower left corner of the blue triangle. Rotation in Tikz are done as they are in math which is that it rotates counterclockwise. Hence the red triangle is to the left of the blue.

Adding together these triangles along a vertical axis can produce some nice patterns. example:

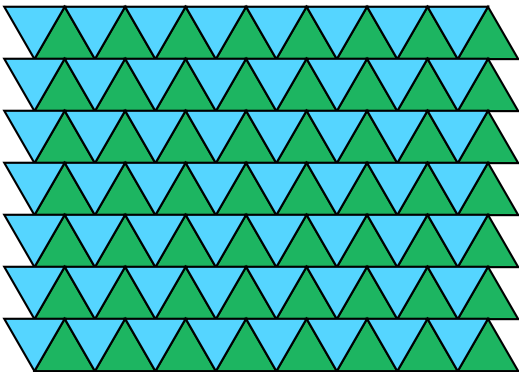


Of course depending on how it's colored various pictures can be drawn. For example coloring a light blue with green can convey mountains in your favorite retro NES game (if any of you were alive for that. Fun fact, that was my first console). Example:



```
code:
\foreach \x in {2,4,...,16}{
\begin{scope}[xshift=\x cm]
\draw [thick,fill=cus-green] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\draw [thick,rotate around ={60:(0 ,0)},fill=cus-sky] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\end{scope}
}
```

And of course by adding another we can draw these in both directions:



```
code:
\foreach \x in {2,4,...,16}{
\foreach \y in {0,-1.72,...,-10.38}{
\begin{scope}[xshift=\x cm, yshift=\y cm]
\draw [thick,fill=cus-green] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\draw [thick,rotate around ={60:(0 ,0)},fill=cus-sky] (0 ,0)--(1,1.73)
--(2 ,0)--cycle;
\end{scope}
}
}
```

NOTE: Notice that the inner loop controls the y axis translations of the object and it goes in a range that is a multiple of 1.72 or $\sqrt{3}$ which is the height of each triangle. Not doing this will either leave space between the rows of triangles or place them one on top each other. This is of course up to the artist but for the purpose of this demonstration we require them to touch.

2.1.1 Triforce

Anyone who really knows me knows that I am a huge fan of video game franchise The Legend of Zelda TM . The trademark



symbol for this video games is what's referred to as the "Triforce" which looks like this: . Of course as you make many of the same objects it's better to wrap them in `\begin{scope}` environments then making a new command in the preamble and setting the possible parameters to whatever you you'd like to manipulate. Therefore the code for this one triforce is:

```
code:

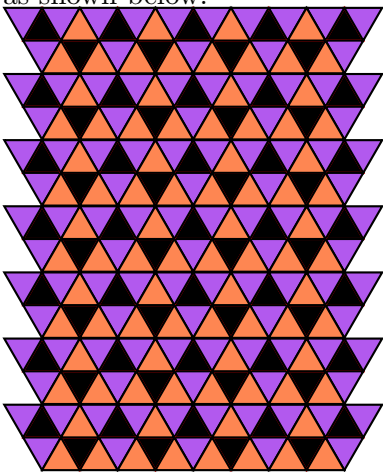
\begin{scope}[yscale = -1]
\draw [thick,fill=black] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\draw [thick,rotate around = {60:(0 ,0)},fill=yellow] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\draw [thick,rotate around = {60:(1 ,1.73)},fill=yellow] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\draw [thick,rotate around = {60:(2 ,0)},fill=yellow] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\end{scope}
```

We then take this code and make a new command called `\triforce{arg1}{arg2}{arg3}{arg4}{arg5}` like so:

```
code:

\newcommand{\triforce}[5]{
\begin{scope}[xshift=#1 cm, yscale=#2,yshift=#5 cm]
\draw [thick,red,fill=#3] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\draw [thick,rotate around = {60:(0 ,0)},fill=#4] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\draw [thick,rotate around = {60:(1 ,1.73)},fill=#4] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\draw [thick,rotate around = {60:(2 ,0)},fill=#4] (0 ,0)--(1,1.73)--(2 ,0) -- cycle;
\end{scope}
}
```

Adding these together in various ways can produce some interesting patterns. I particularly like this for tiling patterns as shown below:



```
code:

\foreach \x in {-10,-6,...,6}{
\foreach \y in {6,2.5,-1,...,-15}{
\triforce{\x}{1}{black}{cus-purp}{\y}
}
}
\foreach \x in {-8,-4,...,4}{
\foreach \y in {-6,-2.5,...,15}{
\triforce{\x}{-1}{black}{cus-orange}{\y}
}
}
```

NOTE: Notice that the Triforce object is created using `\triforce{\x}{1}{black}{cus-purp}{\y}` which makes the object with parameters: xshift, yscale, center triangle fill color, outer triangle fill color and yshift. It's good practice to be

consistent in the order of arguments but sometimes different objects may require a different order. A neat trick is using `xscale` and `yscale` to flip objects. Setting `xscale = -1` flip horizontally and of course `yscale = -1` flips vertically.


2.2 By Vertices

Now let’s take a look at what is possible when triangles are added by there vertices or simply there corner points.

Adding at Vertices Creating these patterns was done in a very object oriented way. The result was that it made things easier when wanting to do use the patterns for different purposes. I may need just one line for a table like table 2 or for an entire as seen in the last section of this article. This process involved make an object that is a shape made by `n` Triangles. I originally made a new object for each shape. Starting with `n = 2`.

```
code:
\newcommand{\duo}[5]{
\begin{scope}[xshift = #1 cm, yshift = #2 cm, rotate = #5]
\foreach \c [count=\r from 1] in {\#3,\#4}{
\draw [thick,rotate around ={\r*180:(0,0)},fill=\c] (0,0)--(1,1.73)--(2,0) -- cycle;
}
\end{scope}
}
```

NOTE: I wanted to do this using only one for loop for a better run time yet still be able to which sort of manipulate color which sort of forced me to make the number of triangles that are created depend on the colors that are passed in. As is it now arguments `#3` and `#4` are the colors that are passed in. Then for every triangle that is created the next will be created at the point (0,0) and rotated by $\frac{360}{n}$ degrees. To add more triangles just add more arguments for the colors i.e `\foreach \c [count=\r from 1] in {\#3,\#4,\#3,\#4}` for 4 triangles.

The code above produces: . I then created an object called `\ntwo` containing that object to make it easier to make many more single patterns at different angles as seen in table 1 below. I then created an object for each pattern. Again there is a better way of doing this.

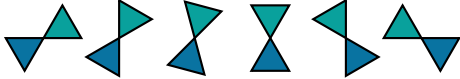



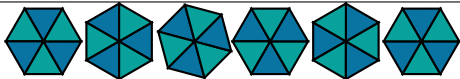
n	angles{0,30,45,60,90,120}
n = 2	
n = 3	
n = 4	
n = 5	
n = 6	

Table 1: Table of triangle up to n=6

It can be seen by the table that as more triangles are added it becomes difficult to notice small changes in angle of rotation does not change by much. In fact had they been all one color it would be difficult so see how they are rotated especially with the close form $n=6$ pattern.

3 Combining Patterns

In this section we'll see how adding together patterns can make more complicated patterns that once colored in can produce an entirely new visual experience. However, for this to work at least for the purpose of this article, the shapes must be touching at least in one direction. To accomplish this yet another object "command" was made to store the shape but this time arguments were passed in to better manipulate the size, position, rotation and background color of the overall pattern. For example:

```
code:
\newcommand{\tduo}[7]{
\begin{tikzpicture}[scale = #1,background rectangle/.style={fill=#5}, show background rectangle]
\foreach \x in {#2}{
\foreach \y in {#3}{
\duo{\x}{\y}{#6}{#7}{#4}
}
}
\end{tikzpicture}
}
```

NOTE: Notice where the arguments #2 and #3 are, this technique proved to be very helpful for changing the range when displaying the pattern for different purposes. For example, I may need just one line for a table like table 2 below.

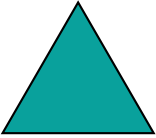
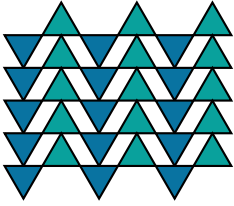
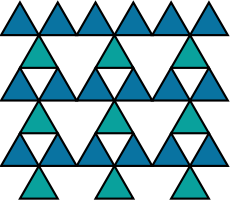
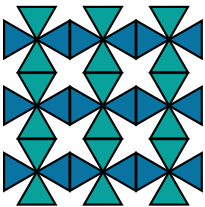
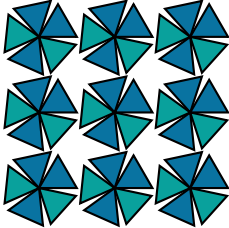
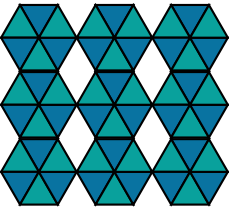
n	1	2	3
Pattern			
n	4	5	6
Pattern			

Table 2: Examples of patterns with n triangles

The commands used in this table are

```
\single{cus-tur}
\tduo{.25}{0,4,...,8}{3.46,1.73,...,-5.17}{0}{white}{cus-blue2}{cus-tur}
\trio{.25}{0,4,...,8}{0,-3.46,...,-10.38}{0}{white}{cus-blue2}{cus-tur}
\quadr{.25}{0,3.5,...,7}{0,-3.5,...,-7}{60}{white}{cus-blue2}{cus-tur}
\pent{.25}{0,4,...,8}{0,-4,...,-8}{0}{white}{cus-blue2}{cus-tur}
\hex{.25}{0,4,...,8}{0,-3.55,...,-10.65}{0}{white}{cus-blue2}{cus-tur}
```

Notice some of the ranges are in multiples of $\sqrt{3}$. This is where knowing the dimensions of one singular triangle (other any object) is important.

4 Patterns

Below are some examples of larger patterns, for the intentions of this article I chose patterns created at angles I thought looked better but please play the angle of rotation:

4.1 Dual Triangles

Rotation	0°	30°
Pattern		

Table 3: Dual triangle patterns

4.2 3 Triangles

Rotation	0°	30°
Pattern		
Rotation	45°	60°

Table 4: Triple triangle patterns

4 Triangles

0 degrees

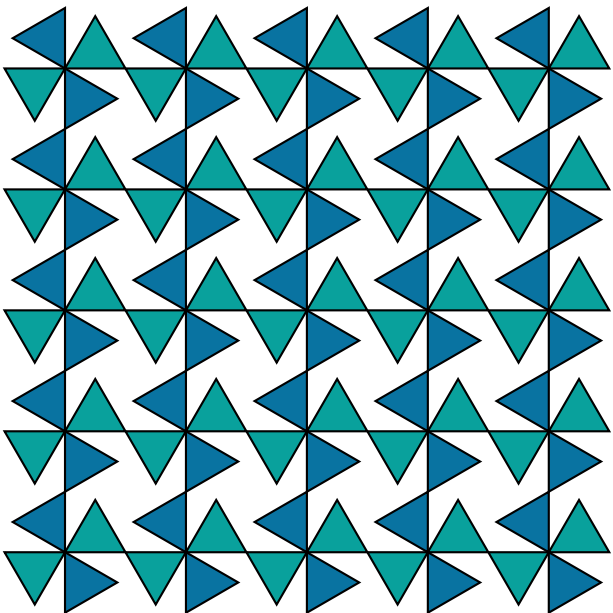
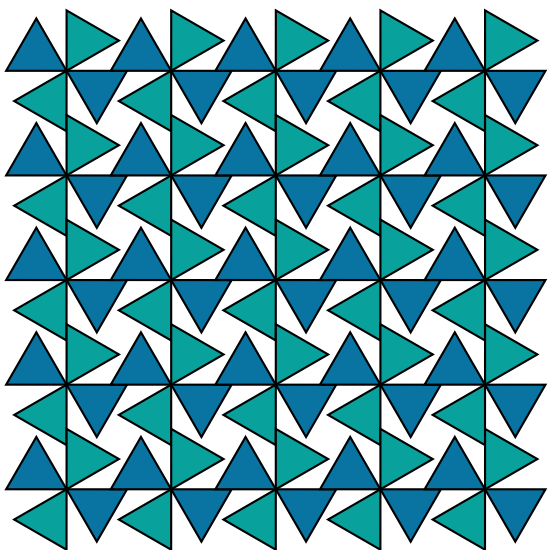
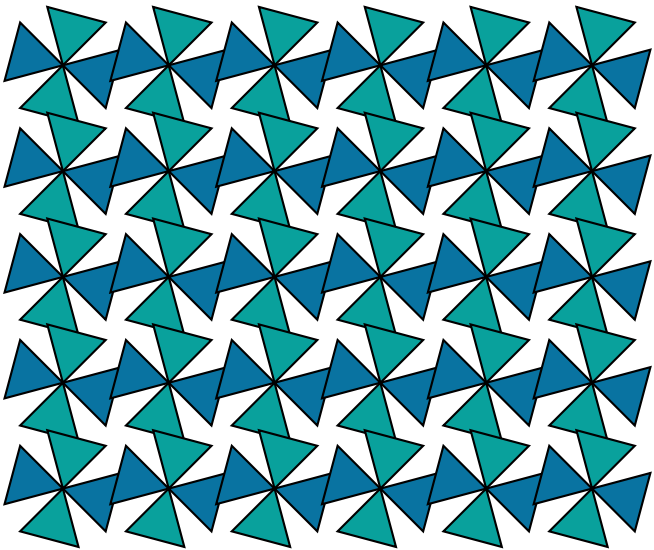
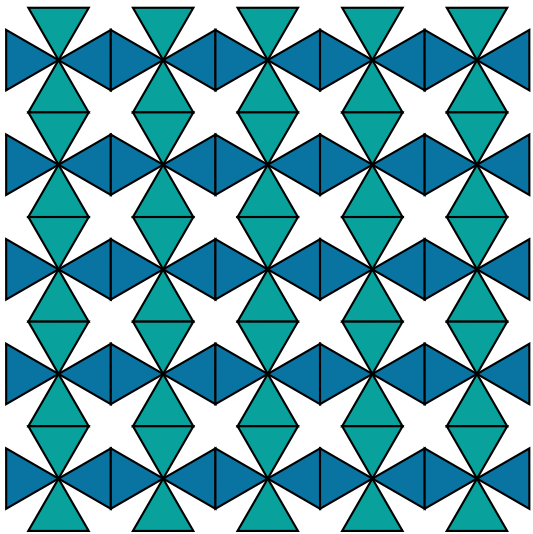
Rotation	0°	30°
Pattern		
Rotation	45°	60°
Pattern		

Table 5: Quadruple triangle patterns

5 Triangles

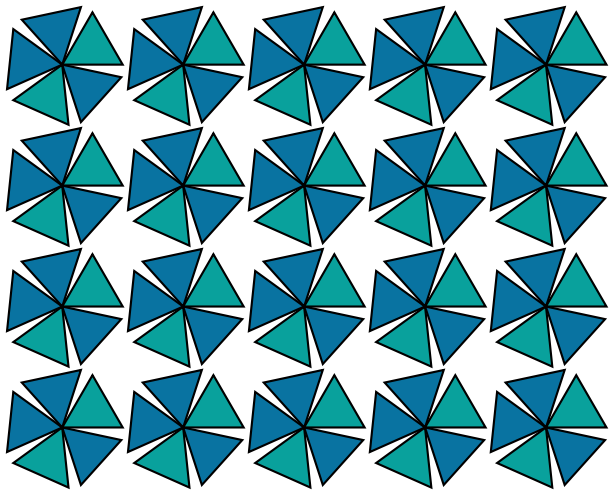
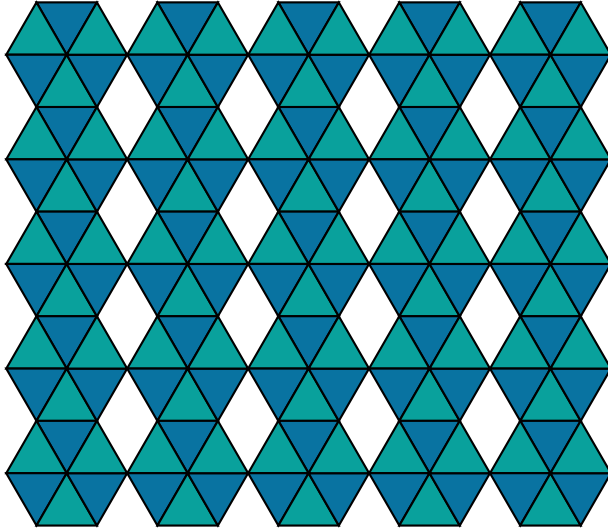
n Triangles	Five	Six
Patterns		

Table 6: Five and six triangle patterns

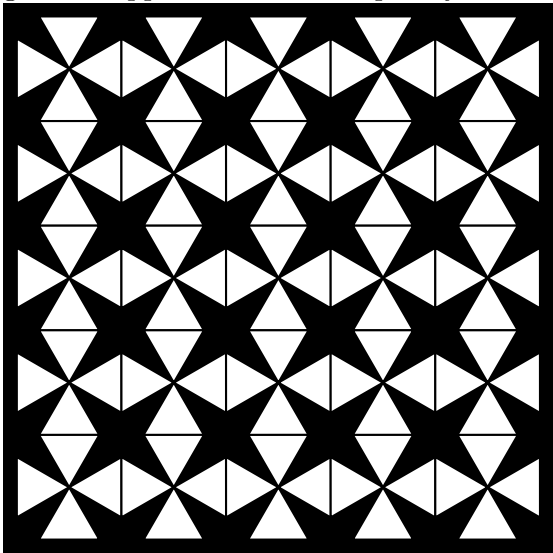
As seen in table 1 patterns with five and six triangles do not look much different when the angle is changed. This is however only true because they are being drawn with only two colors.

5 Using backgrounds

Notice that each shape takes in an argument for the background. by doing this patterns may be created using the negative spaces of the shape. Make sure to include `\usetikzlibrary{decorations, decorations.text,backgrounds}` to get access to the background color and type (rectangle, circle,etc).

5.1 Black and White

The table below shows my personal favorite patterns when drawn in black and white. Notice how the negative spaces can give the appearance of a completely different pattern being drawn. This is especially the case for $n = 4$ as seen below.



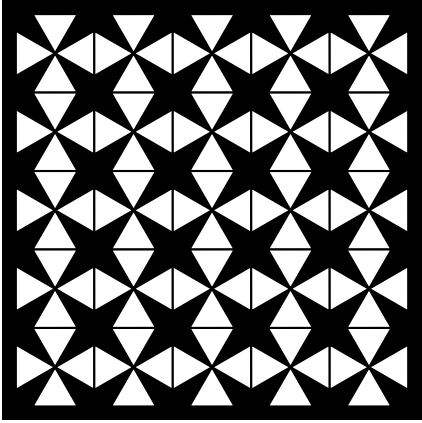
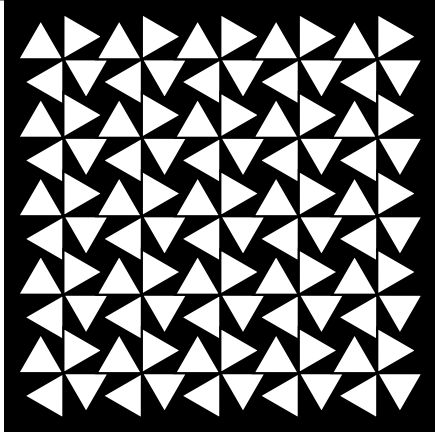
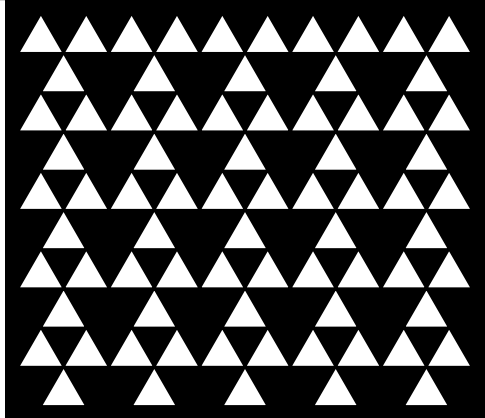
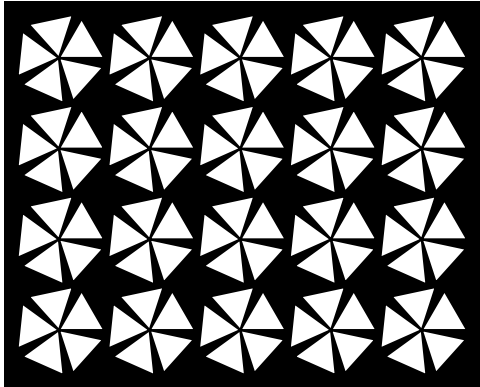
4 triangles at 60°	4 triangles at 30°
	
3 triangles at 0°	5 triangles at 72°
	

Table 7: Favorite Black and White patterns

5.2 Multi-Color

Here are the same patterns in different colors

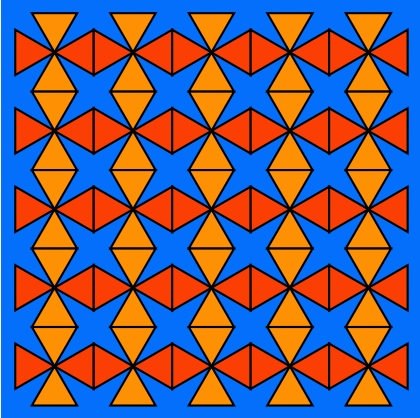
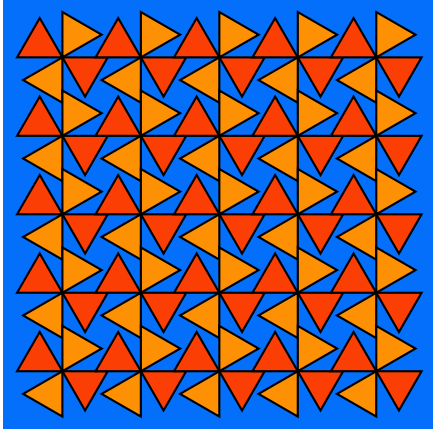
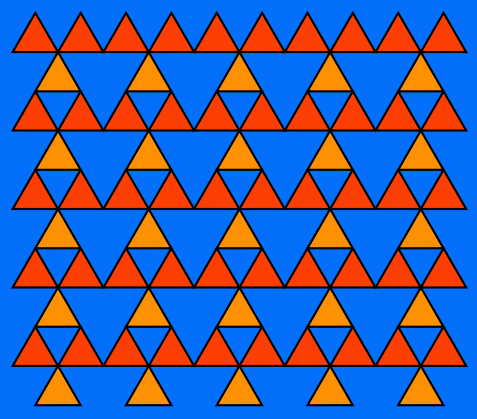
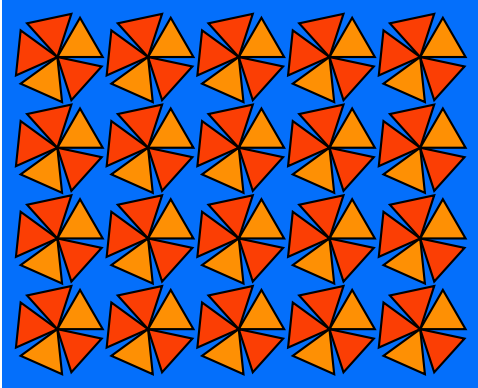
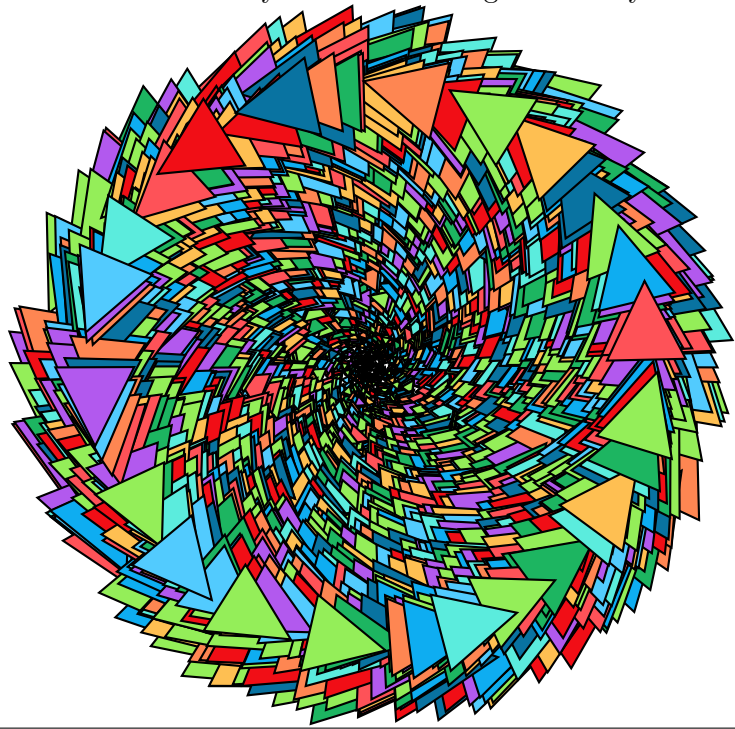
<p>4 triangles at 60°</p> 	<p>4 triangles at 30°</p> 
<p>3 triangles at 0°</p> 	<p>5 triangles at 72°</p> 

Table 8: Multi-Color patterns

6 Extra:Infinite Triangle

I love the idea of infinity and I love triangles. So why not mix them?



code:

```
\newcommand{\circctri}[1]{  
  
\begin{scope}[xshift=#1 cm]  
\foreach \x in {-8,-7.95,...,7}{  
\foreach \c in { cus-blue,cus-red,cus-orange,cus-yellow,cus-blue1,cus-red1,cus-green,  
cus-aqua,cus-purp,cus-blue2,cus-green1,cus-violet}{  
\draw [thick,rotate=rand*360,xshift=\x*.6 cm,fill=\c,scale=\x*.1] (0 ,0)--(1 ,2)--(2 ,0) -- cycle;  
}  
}  
\end{scope}
```

Making this was quite a challenge. I honestly just played around with it's xshift values and using a random rotation to get get the triangles to move in a circle. Decreasing the scale as triangles are being created collapsed the picture to the center, that was the key to get the infinity look. Notice that for every xshift value a triangle is created for every color for every x. Therefore the amount of triangles created per rotation is based on the number of colors, a similar technique used earlier. Thus there are $(.05x15)x12 = 3,600$ triangles being created.