Turtle Graphics — Stars & Fireworks

After learning about the basics of 'Turtle Graphics' in the previous lesson, let's try to make a more complex display of stars and fireworks!

1. **Part I: Drawing Stars**
   When making polygons using the turtle, you use the Geometric property that the sum of the exterior angles of a polygon is 360 degrees. When making a regular polygon the turning angle is therefore 360/n since all the angles are equal.

   In this first part of the activity, you will use a similar property to produce star polygons like the one shown at the right.

2. First let's take a look at the basic regular polygon program from the previous activity.
   The program shown here makes a regular *septagon*.

3. Change the code to make a star…
   In the turn function, instead of 360, use an *integer multiple* of 360.

   To get this image we used **2***360.

4. You might try a higher multiple of 360 This star uses **3***360 instead of 2*360.

   The **multiple** represents the number of revolutions that the turtle makes to complete the star. This is called a **(7/3) star polygon** since it has seven vertices and it takes three revolutions to complete.

5. When you add polygon *filling* to a star polygon, the results can be surprising.

   Add **t.fillcolor(, ,)**, **t.begin_fill( )** and **t.end_fill( )** to your program.

   The entire star is not filled. There are places that the fill algorithm skips because it is looking for 'borders'.

6. **Part II: The Fireworks!**
   In this section of the activity you will use lots of random values to generate these awesome fireworks on the screen.

7. In the star-making loop shown here **n** is the number of vertices and **r** is the number of revolutions. What values for **n** and **r** make for pleasing stars?

   We will limit **n** to be random *odd* numbers: 5,7,9,11…
   For 'pointy' stars we will make **r = int(n/2)**

   We will also use a *random* position, heading, color, and side length for each star in this project.

8. First test these star-making options:

```
n = 2 * randint(2, 10) + 1 #  odd numbers
r = int(n / 2)             #  or   r = n // 2
```

Running this program will make a pointy star with an odd random number of vertices.

9. Now add code to the program to:
   - Hide the turtle
   - Hide the grid
   - Make the turtle move fast: **t.speed(10)** (or 0)
   - Put the star-making code in a **while not escape():** loop
     - Assign a random *odd* value to **n**
     - Calculate **r**
     - Draw the star using the for loop
   - Clear the screen to see the different stars. *This is temporary.*

Test your program to see various stars displayed like this one.

10. ***We're ready for the fireworks:***
    - each star will be drawn in a different position on the screen using a different heading with different colors and sizes.
    - assign random values to appropriately-named variables to achieve the fireworks effect as seen here.
    - #comment the **t.clear( )** statement since we want to fill the screen with stars.
    - Run the program and enjoy the show!

Teacher Tip: Odd numbers to avoid possible disconnected polygons. The number of revolutions gives the pointiest stars

Teacher Tip: Possible **Fireworks** solution:

```
from turtle import *
from random import *
t=Turtle()
t.hidegrid()
t.hideturtle()
t.speed(0)
while get_key() != "esc":
  n=2*randint(2,9)+1
  r=int(n/2)
  a=randint(-90,90)
  s=randint(10,100)
  t.penup()
  t.goto(randint(-179,159),randint(-106,106))
  t.pendown()
  t.setheading(a)
  t.fillcolor(randint(0,255),randint(0,255),randint(0,255))
#  t.pencolor(255,255,255)
  t.begin_fill()
  for i in range(n):
    t.forward(s)
    t.left(r*360/n)
  t.end_fill()
```