1 Introduction

The following exercises are updates of the Hello World of Graphics program. Put each project in its own directory and zip the entire set of directories into one zip file. Upload the zip file to the Homework #1 page of the MyClasses site for this class.

2 Generic Python Exercises

1. Write a program that will take one input integer from the user, n. You may assume that this number is large then 1. The program will then generate the $3n+1$ sequence, print it to the screen along with the length of the sequence. Recall that the $3n+1$ sequence is defined as a list of numbers that starts with a positive integer larger than 1. Each number in the sequence is determined by the previous number. If the previous number is even then the next number is the previous divided by 2. If the previous number is odd then the next number is the previous times 3 and add one. We stop when the last number is 1. A couple runs are below,

   ```
   Input an integer between 2 and 1000000: 17
   Sequence: 17  52  26  13  40  20  10  5  16  8  4  2  1
   Number of integers in list =  13

   Input an integer between 2 and 1000000: 1
   Invalid Input.
   Input an integer between 2 and 1000000: 100000000000000
   Invalid Input.
   Input an integer between 2 and 1000000: 42
   Sequence: 42  21  64  32  16  8  4  2  1
   Number of integers in list =  9
   ```

   In this exercise create a function named `nifty` that takes in a single integer parameter n and returns the next number in the $3n+1$ sequence. That is, if the input n is a number in the sequence then the function returns the next number. So `nifty(17)` is 52, `nifty(52)` is 26, and so on. The main should call this function in a loop until the value 1 is reached in the sequence.

2. Write a program that will take a filename from the user and a target string that is a single word. The program is to find all of the occurrences of the input word in the file. The searching must be case insensitive. I have included two text files for this exercise, `DATHT.txt` and `HGGT.txt`. These files contain the Hitchhikers Guide to the Galaxy trilogy, which is actually 5 books. In the file `DATHT.txt` the text is broken into over 26,000 lines as one would expect to read and the file `HGGT.txt` is the same words but is on a single line. Your program must work for both files without alteration.

   For this exercise read the entire file in and make a list of each separate word. That is, each element of the list is a single word. Also, for each of these words remove any punctuation and convert the word to lower case. Then count the number od occurrences of the target word. An example run is below.

```
Input text filename: DATHT.txt
Input word to search for: Arthur
The number of words in the text is  268108
The number of times that  arthur  appears is  1506
```

For this exercise you will want to look over the functionality of the list and strings. In particular, the split function of strings may come in handy as well as the append or extend functions for the list.

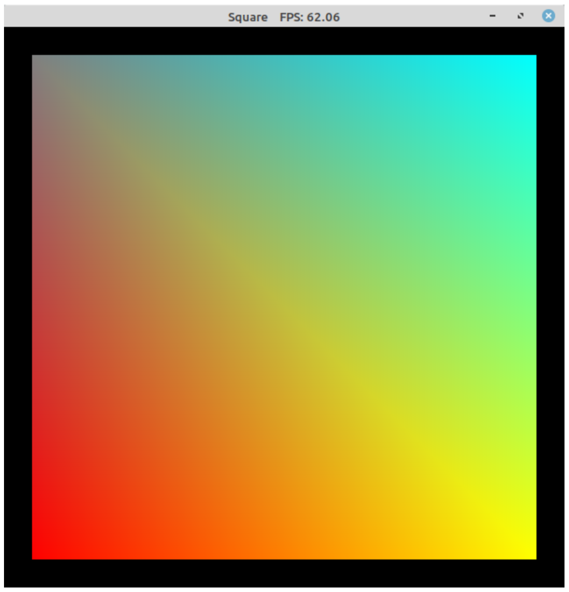
3. Write a program that will display some statistics about the stock prices from the Intel Corporation, IBM, and Microsoft. There are three files included in this exercise, `IBM.txt`, `INTC.txt`, and `MSFT.txt`. The files contain the daily stock prices from the time the company went public up until early February 2021. Each line of the file contains a date followed by the opening price for that day followed by the closing price for that day. The file is tab delimited, so the values on each line have a tab character between them. Note that the first line is a header line. Read in the entire file, and format it as a list of lists. The lists inside the containing list are each data line of the file, that is a list of the date, opening price, and closing price for that day. So the first entry is a string of the date, the second is a numeric value as is the third. Have program ask for the filename, do its data reformatting, print each day to the screen on its own line and then display the number of days in the file, the average opening stock price, the minimum and maximum opening stock price, the number of up days (closed at a higher price than it opened), and the number of down days (closed at a lower price than it opened). A run is below.

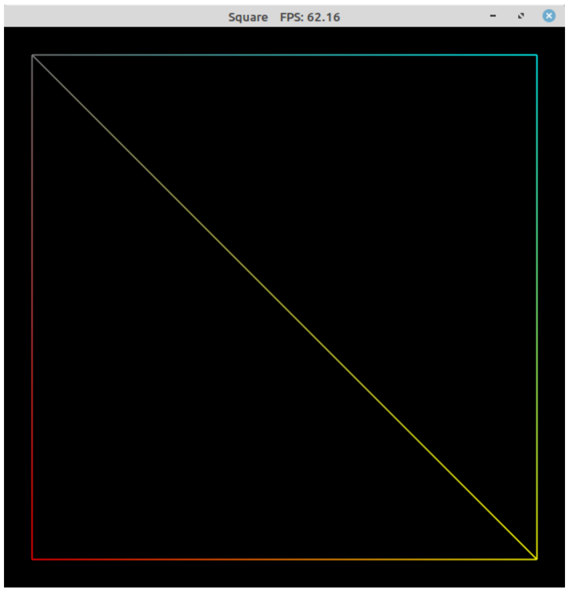
```
Input data filename: MSFT.txt

['1986-03-14', 0.097222, 0.100694]
['1986-03-17', 0.100694, 0.102431]
['1986-03-18', 0.102431, 0.099826]
.
.
.
['2021-02-04', 242.660004, 242.009995]
['2021-02-05', 242.229996, 242.199997]
['2021-02-08', 243.149994, 242.470001]

Number of stock values in the file:  8798
Average opening stock price:  33.46342192862014
Minimum opening stock price:  0.090278
Maximum opening stock price:  243.149994
Number of up days:  4339
Number of down days:  4205
```

For this exercise you will want to look over the functionality of the list and strings. In particular, the split function of strings may come in handy as well as the append or extend functions for the list.

3 Graphics Exercises

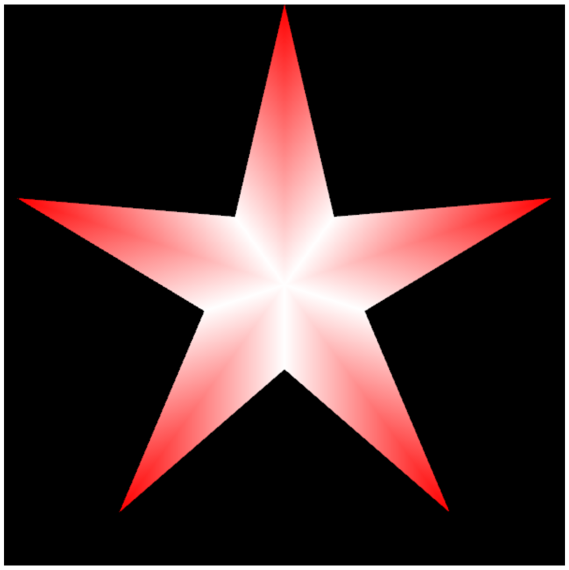
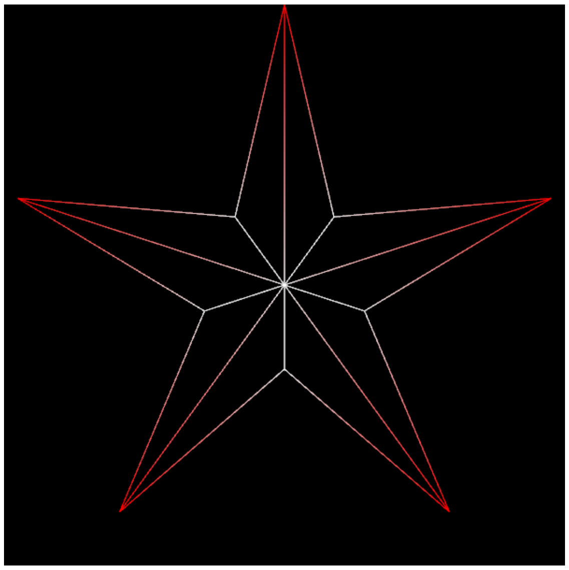
1. For the first program make the initial size of the window 600×600 and make the reset option resize to 600×600 as well. Make the title bar display "Square" and incorporate the FPS tracking in the title bar. Have the image on the screen be a square like the image below. Keep the rest of the features the same as with the triangles program. Hint: A square can be produced using two triangles. Below are two screen shots, one in fill mode and one in line mode.



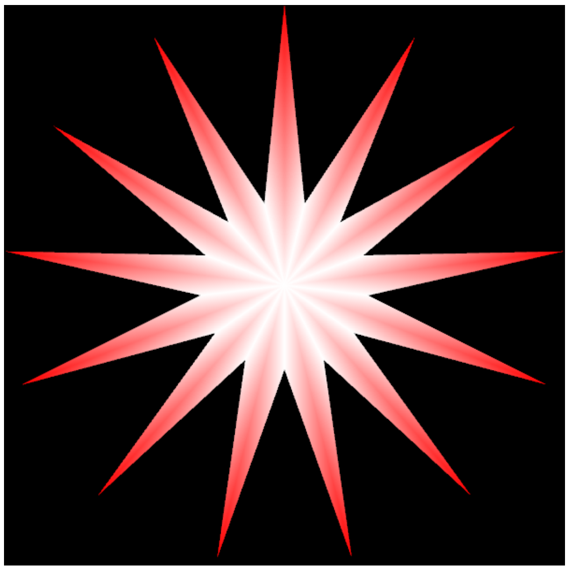
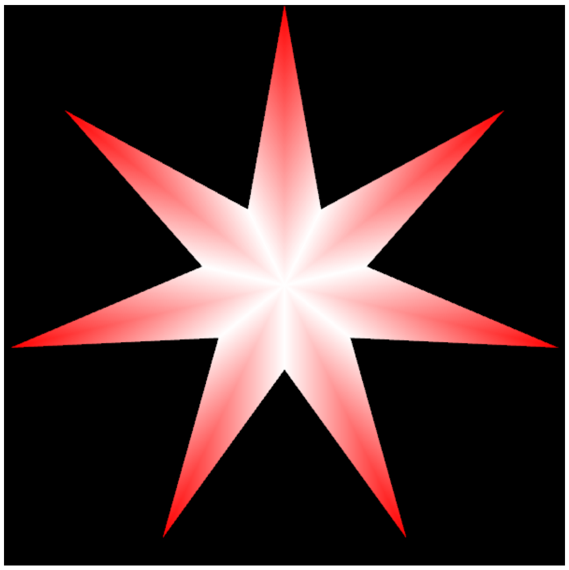


2. Crete a program to produce a 5-point star with the coloring of the star being like the image below, that is, red at the points and white in the center. The title bar should also display the title "Star" with FPS in the title bar. As with the previous exercise, make the initial size of the window 600×600 and make the reset option resize to 600×600 as well. Keep the rest of the features the same as with the triangles program. Below are two screen shots, one in fill mode and one in line mode. Note that one point of the star is always pointed straight up.

   Hints: Each point of the star can be done with two triangles. Also recall from trigonometry that a point on the unit circle can be represented by $(\cos(\theta), \sin(\theta))$. So if we wanted to space n points around the unit circle evenly we could use the points $(\cos(i \cdot 2\pi/n), \sin(i \cdot 2\pi/n))$ where $0 \leq i < n$. We can also scale the circle by multiplying by a constant. So n evenly spaced points around a circle of radius r would be $(r \cdot \cos(i \cdot 2\pi/n), r \cdot \sin(i \cdot 2\pi/n))$ where $0 \leq i < n$.

3. Update the above star program to produce an n-point star with the coloring of the star being like the image below where n is chosen randomly between 5 and 15. So on each run the number of points to the star could be different. Hint: Use the random package and look up the documentation on the randint function. Also, unlike C++, you do not need to set a seed for the random number generator. Then generalize what you did for the 5-point star to the n-point star. Note that one point of the star is always pointed straight up.

4. Update the above n-star program to add in some user interface from the keyboard. Have the program start by displaying a 5-point star, when the user hits 2 replace the star by a 2-point star, when the user hits 3 replace the star by a 3-point star, when the user hits 4 replace the star by a 4-point star, when the user hits 5 replace the star by a 5-point star, and so on up to 9 creates a 9-point star. I would suggest creating a new function for the graphics engine called `loadStarData(self, numpoints)` which takes in the number of points to be used, creates the star, and loads it to the graphics card.

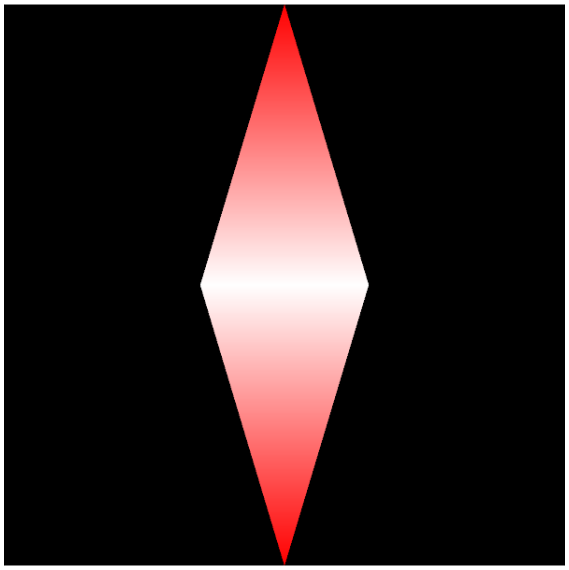
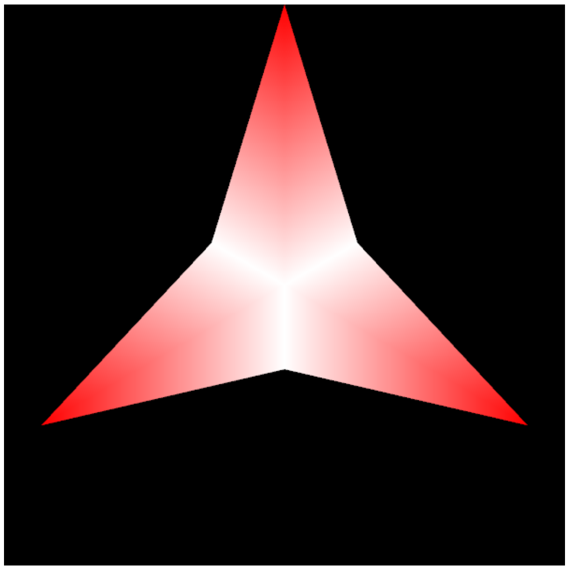
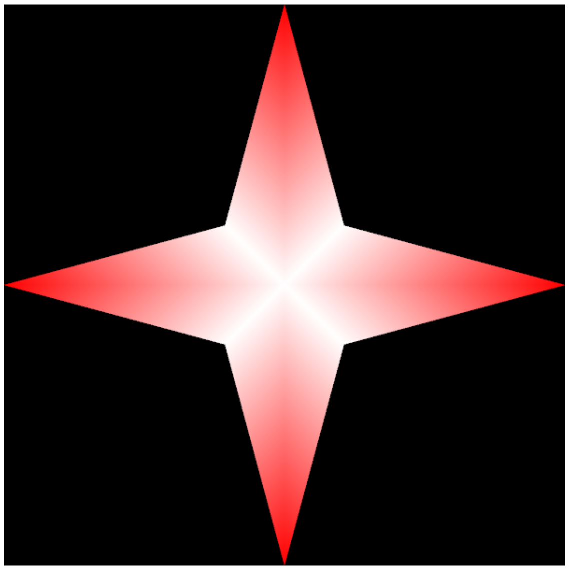
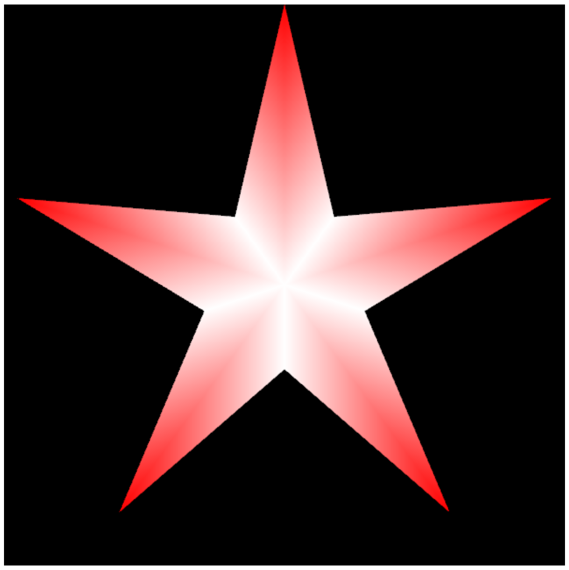
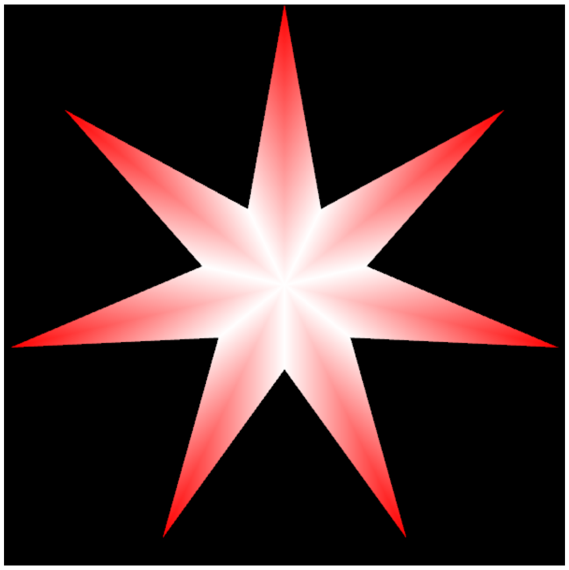
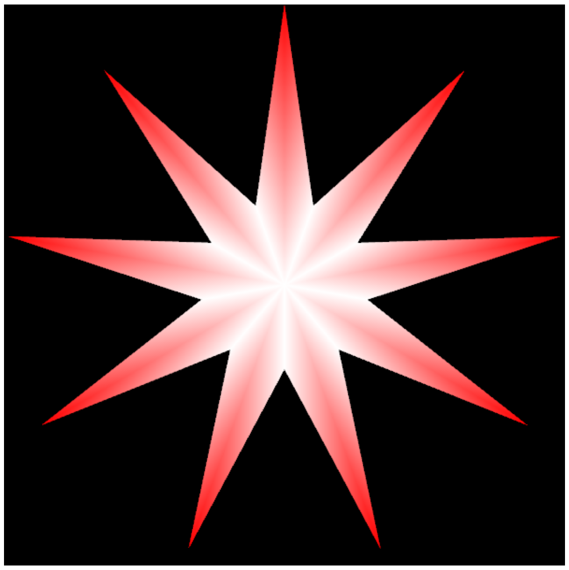
   A note on memory management on the graphics card. If you keep generating new vertex array objects and buffers inside them, it is like doing the new (or malloc) command over and over again. That is, you will have a memory leak on the graphics card. It is unlikely that you will run out of memory with this program but we would like to be as efficient as possible. The first time you call `loadStarData` you will need to generate your arrays and buffers. Then for every subsequent call to `loadStarData` you should delete your VAO and Buffer before you generate another one. This is like doing the delete (or free) command to release the memory locations for use. Another method would be to define the VAO and the Buffer in the constructor of the graphics engine and set them to None (the Python version of NULL)

```
self.VAO = None
self.Buffer = None
```

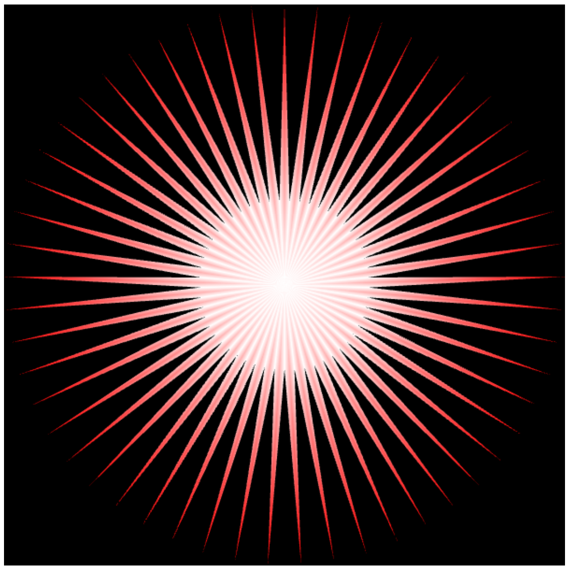
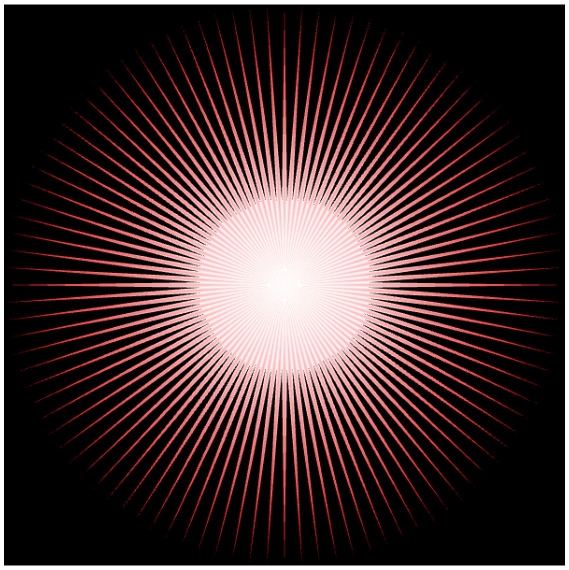
   Then in the loadStarData function test if either of them is not None and delete the objects. Do this right before you generate the VAO and buffer.

```
if self.VAO or self.Buffer:
    glDeleteBuffers(1, self.Buffer)
    glDeleteVertexArrays(1, self.VAO)
```

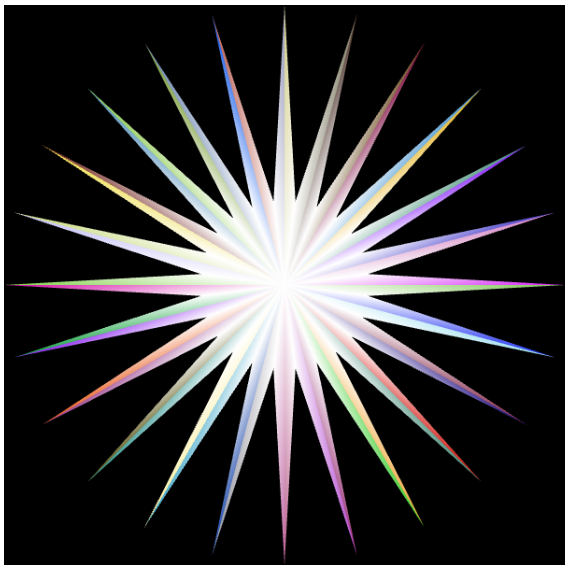
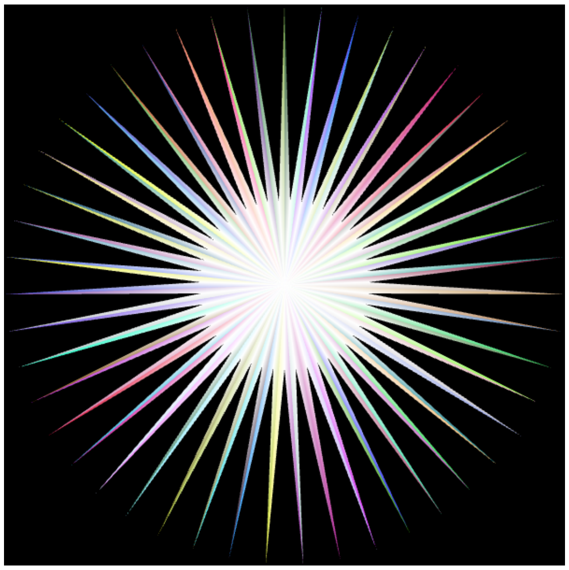
   Note, just like the other programs, one point of the star is pointed straight up.

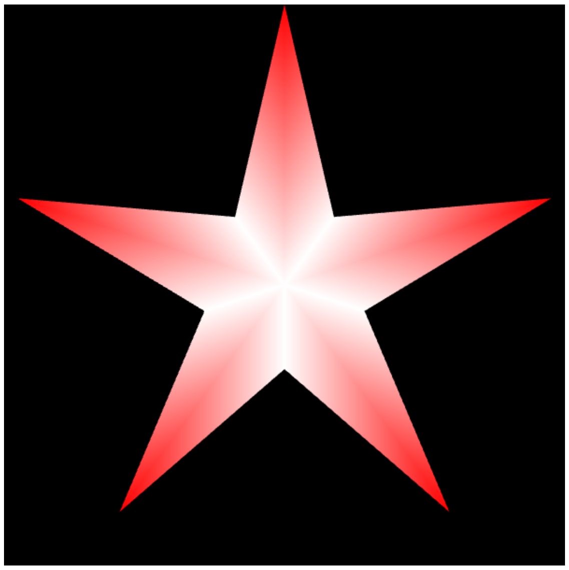
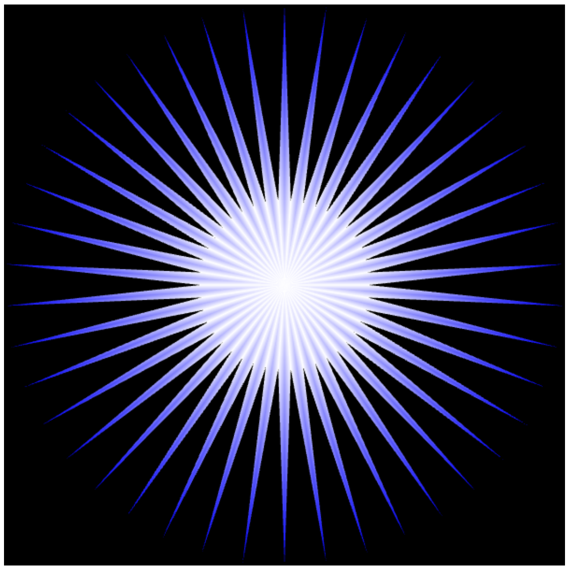
5. Update the above n-star program to add in one more user interface option. If the user presses the up arrow key the number of points will be incremented by 1 and if they press the down arrow key the number of points will be decremented by 1. The minimum number of points should be 2 and the maximum number of points should be 100.

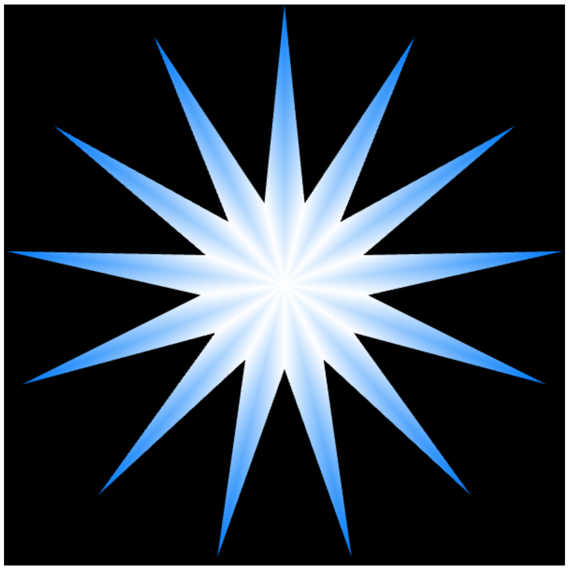
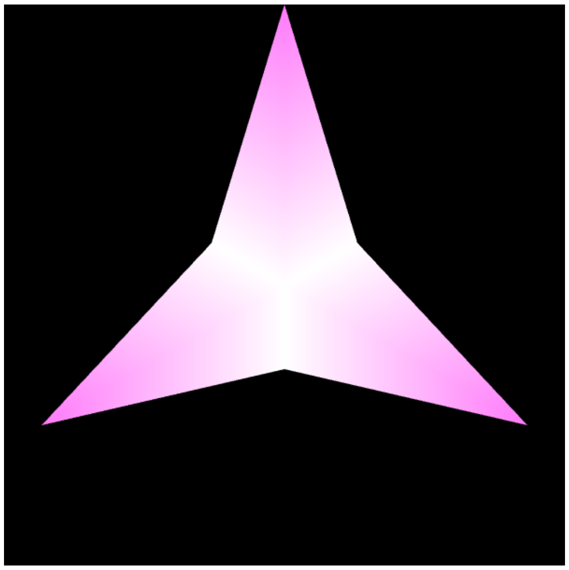



6. Update the above n-star program to randomize the colors of the points. The center should still be white. Specifically, any time that `loadStarData` is called load in random colors for the points.

7. Update the n-star program from #5 to let the user change the color of the points. If the user presses the r key the amount of red will be increased by 0.01 until the value is 1, when t is pressed the red will be decreased by 0.01 until the value is 0. Similarly, g will increase the green and h will decrease the green, b will increase the blue and n will decrease the blue. The program should start out with red as the point color.

8. Update the n-star program from above to alter the size of the star points. Have the right arrow make the points longer and the left arrow key make them shorter. The maximum length should be 1 so that the star stays inside the window. The minimum value should be 0. The user color selection and the up and down arrows for increasing and decreasing the number of points should also be implemented and the program should start out with red as the point color.