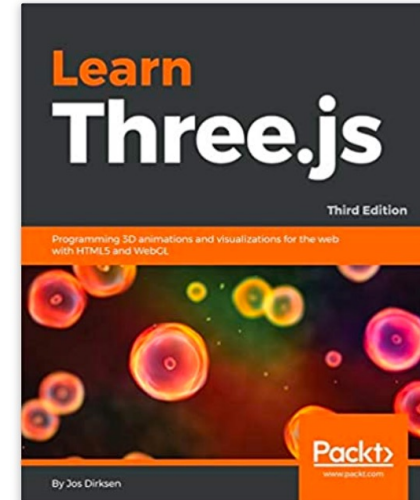


# COMPUTER GRAPHICS

---



\* Adapted from CISC 3326 lecture by Michael Mandel

# PARAMETRIC LINES AND COLOR INTERPOLATION

---

Based on [this CS 307 reading](#) and [this CS 307 lecture](#)\*

\*copyright © Scott D. Anderson and licensed under a [Creative Commons BY-NC-SA License](#)

# THREE.JS EXERCISES

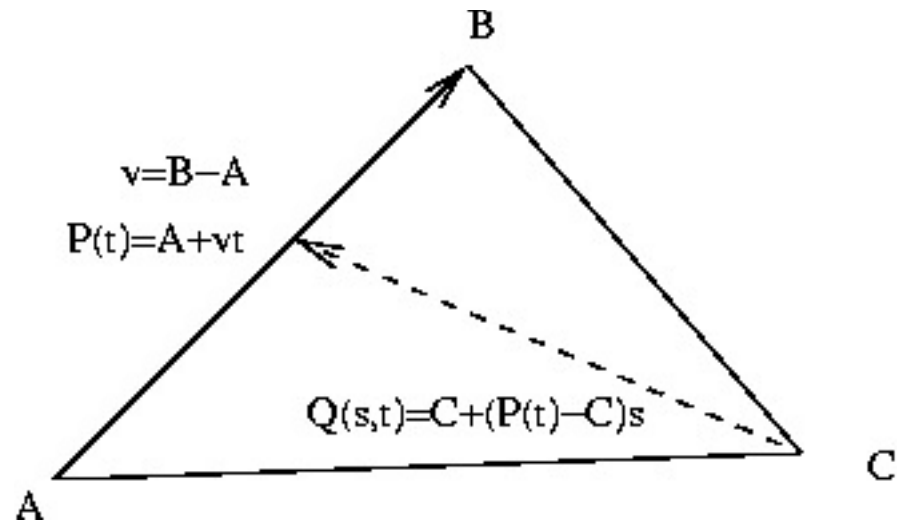
---

# COLOR INTERPOLATION AND GEOMETRY IN THREE.JS

---

# Reminder: Parametric equation for a triangle

- $Q(s,t) = C + (P(t) - C)s$
  - $Q(s,t) = C + (P(t)s - Cs)$
  - $Q(s,t) = [A(1-t) + B(t)]s + C(1-s)$
  - $Q(s,t) = A(1-t)s + Bts + C(1-s)$
- Lines to set up parametric equation of a triangle:

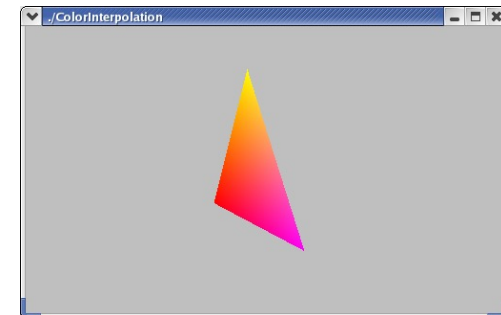


# Color Interpolation in a Triangle

- If the colors of the vertices are different, OpenGL interpolates them for us
  - using the same equations that we used for calculating coordinates.

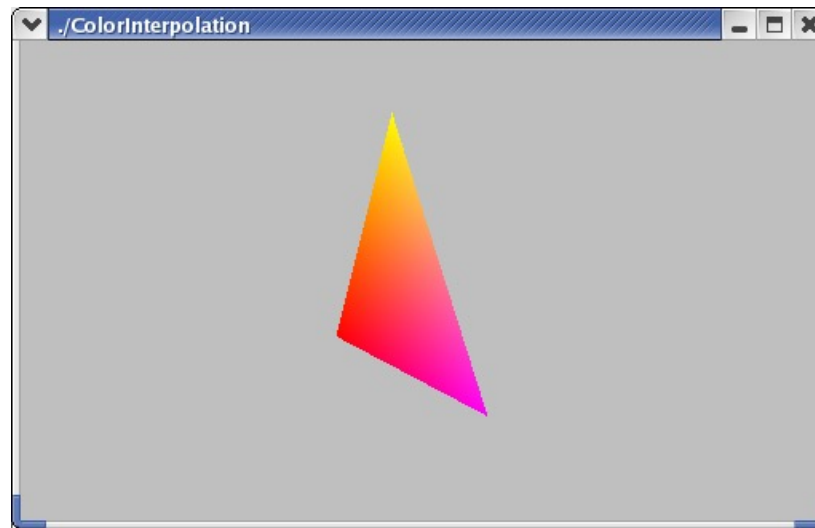
# Color Interpolation in a Triangle

- Suppose A is red (1,0,0), B is magenta (1,0,1), and C is yellow (1,1,0)
- We can compute the color of the middle point,  $Q(0.5,0.5)$ , as:
  - $Q(0.5,0.5) = A(0.25) + B(0.25) + C(0.5)$
  - $Q(0.5,0.5) = (1,0,0)(0.25) + (1,0,1)(0.25) + (1,1,0)(0.5)$
  - $Q(0.5,0.5) = (1, 0.5, 0.75)$
- The triangle as a whole looks like this:



# Color Interpolation in a Triangle

- The triangle as a whole looks like this:



- A triangle with smooth interpolation



# Color Interpolation in Three.js

- To achieve interpolation in Three.js, you need to do the following:
  - Create a colors array, with as many entries as vertices in your mesh.
  - Set the colors array as the **vertexColors** property of the geometry
  - Using **THREE.MeshBasicMaterial**, set the **vertexColors** property to **THREE.VertexColors**

# Example: Color Interpolation in Three.js

- The `THREE.Geometry()` object has a:
  - `vertexColors` property that is an array of colors
  - an array of `THREE.Face3()` objects
- Each `THREE.Face3()` object has a three-element array of colors
  - each is the color of the corresponding face vertex

# Example: Color Interpolation in Three.js

- Using `THREE.MeshBasicMaterial`, we set the `vertexColors` property to `THREE.VertexColors`
  - The value of this property alerts Three.js that the vertices of a face could have different colors
    - The face is a triangle

# Example: Color interpolation RGB triangle

- [Triangle interpolation](#)

# Color interpolation RGB triangle

- Triangle interpolation on a square

# Inconsistent color interpolation

- Inconsistent triangle interpolation on a square

# Inconsistent color interpolation

- Notice that at the lower right we have:
  - vertex B, coordinates (1,0,0), color `THREE.ColorKeywords.lime`
  - vertex B2, coordinates (1,0,0), color `THREE.ColorKeywords.blue`

# EXERCISES

---



# Exercise 1

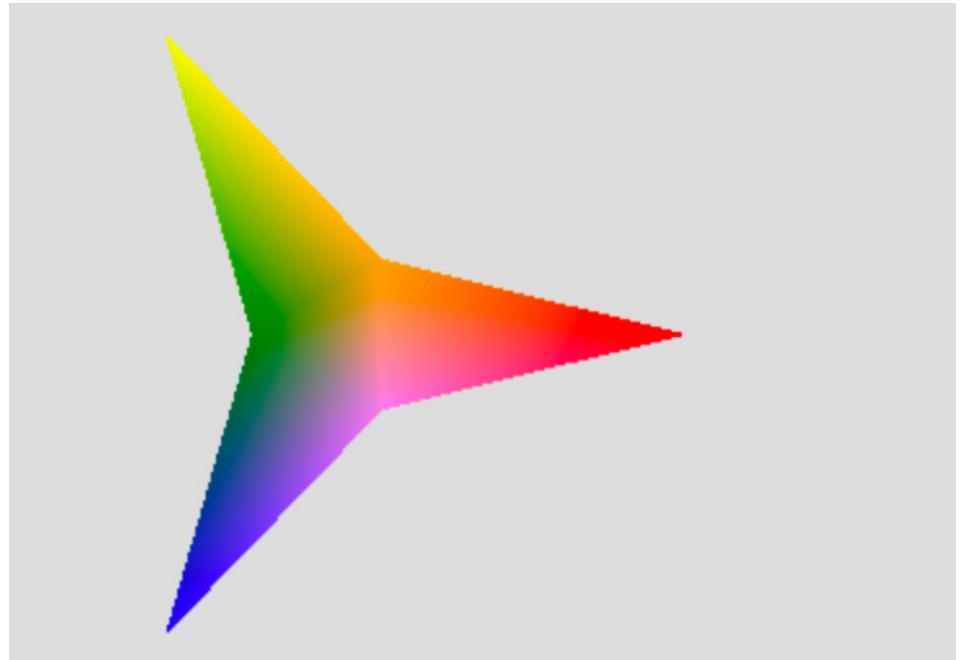
- Start from stars-start
  - pen contains a function starGeometry()
    - that creates and returns a Three.Geometry object for a three-pointed star.

# starGeometry function

- function starGeometry (size) {
- var starGeom = new THREE.Geometry();
- var angle;
- var lens = [size, size/4];
- for (var i = 0; i < 6; i++) {
- angle = i\*(Math.PI/3);
- len = lens[i % 2];
- starGeom.vertices.push(new THREE.Vector3(len\*Math.cos(angle), len\*Math.sin(angle)));
- }
- starGeom.faces.push(new THREE.Face3(0,1,5));
- starGeom.faces.push(new THREE.Face3(1,2,3));
- starGeom.faces.push(new THREE.Face3(3,4,5));
- starGeom.faces.push(new THREE.Face3(1,3,5));
- return starGeom;
- }

# starGeometry function

- How does the geometry work?



# starGeometry function

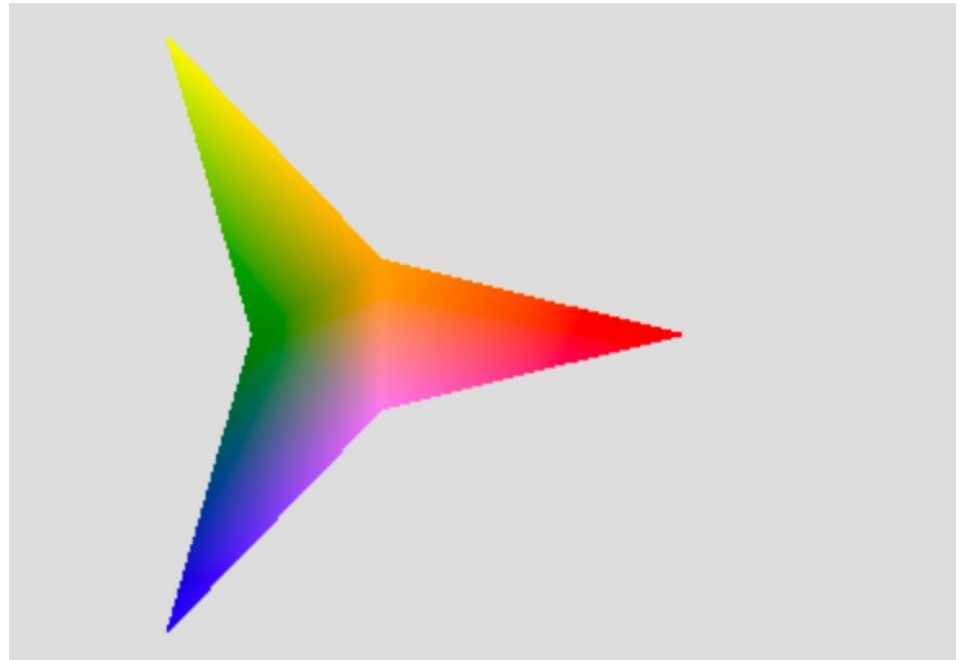
- Triangle Vertices
- Adding triangles

# Exercise 1

- Exercise: Modify this code to create a star that uses color interpolation of the triangular faces
  - and adds it to the scene.

# Exercise: Colorful Stars

- Your result might look like this:

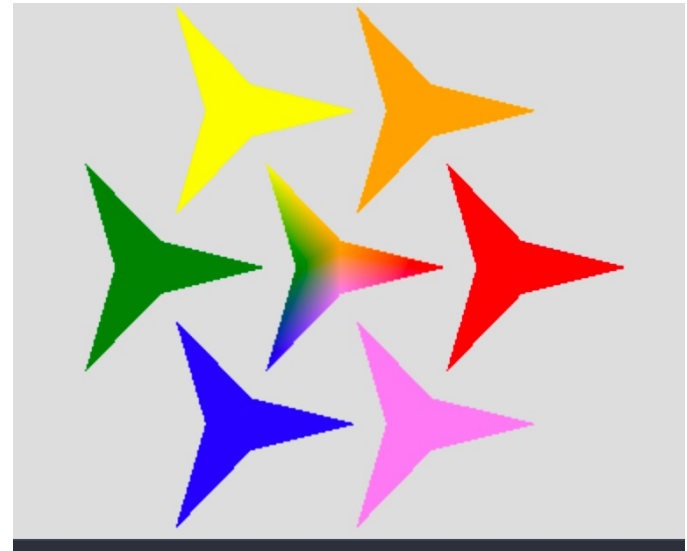


# Exercise: Colorful Stars

- Suggestions:
  - The starting code includes an array of `THREE.Color` objects named `colors`
    - You can change the colors to whatever you want!
      - `Colors` array is defined in the starter lab
  - Create the material for the star using `THREE.MeshBasicMaterial`:
    - add a second property to the input object
      - in addition to the `vertexColors` property
    - Property should tell Three.js to render both sides of the triangular faces:
      - `side: THREE.DoubleSide`

# Exercise: Add stars to the scene

- Add six additional stars to the scene that each have a uniform color
  - placed around the central star
- Something like this:





# Exercise: Add stars to the scene

- Suggestions:
  - Think about how this can be done with a loop
  - Use the same array of colors that you used for the central star
  - Recall that `position.set()` can be used to place a mesh at a desired location
  - Remember to adjust the bounding box supplied to `TW.cameraSetup()` to see the additional stars

# Exercise: Add stars to the scene

- Suggestions:
  - Inside a loop, you may want to include code similar to:
    - for (i = 0; i < 6; i++) {
    - ...
    - angle = i\*(Math.PI/3);
    - x = 1.5\*size\*Math.cos(angle);
    - y = 1.5\*size\*Math.sin(angle);
    - starMesh.position.set(x,y,0);
    - ...
    - }

Questions?

