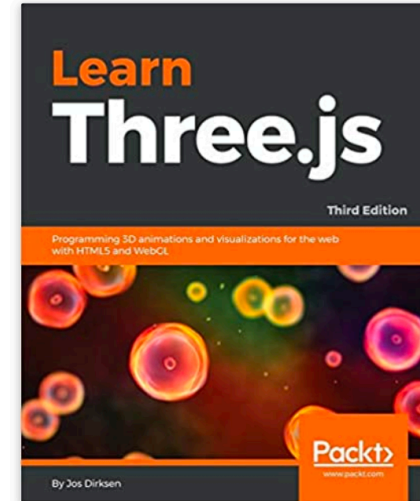


COMPUTER GRAPHICS



* 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 SUPPORT FOR COLOR INTERPOLATION

Reminder: Color mixing

- Problem: Suppose that vertex A is red $(1,0,0)$ and vertex B is magenta $(1,1,0)$.
 - What is the color of the point that is $2/3$ of the way from A to B?



Solution

- Solution: We can use the same mixture equation that we just used to find coordinates:
 - $P(2/3) = A(1/3) + B(2/3)$
 - $P(2/3) = (1,0,0)(1/3) + (1,1,0)(2/3)$
 - $P(2/3) = (1,2/3,0)$
- **Note:**
 - if a problem calls for more than one line, each line gets its own parameter, such as r , s , or u .
 - Each parameter has a meaning:
 - $t=0$ means the initial point of the line
 - so $s=0$ would be the initial point of the other line.

Interpolation

- “a method of constructing new *data points* within the range of a discrete set of known data points”
- Linear interpolation:
 - Finding the points between points p_s and p_f
 - Parametric equations enable doing this
 - $B(t) = p_s + (p_f - p_s)t$

Three.js Support for Interpolation

- Three.js has some useful functions for doing interpolation on [Vector3](#) objects.
- Suppose v_1 , v_2 , and v_3 are all Vector3 objects:
 - $v_1.add(v_2)$ adds vector v_2 to v_1
 - $v_1.addVectors(v_2, v_3)$ sets v_1 to the sum $v_2 + v_3$
 - $v_1.multiplyScalar(s)$ multiplies v_1 by a scalar s
 - $v_1.sub(v_2)$ subtracts vector v_2 from v_1
 - $v_1.subVectors(v_2, v_3)$ sets v_1 to the difference $v_2 - v_3$
 - $v_1.lerp(v_2, \theta)$ moves v_1 towards v_2 , by a fraction θ , using linear interpolation

Three.js Support for Interpolation

- Warning, these methods all modify the object
 - so if you want to compute a new vertex, it's best to `.clone()` the vertex first.

Computing with Three.js interpolation

- Here's an example of:
 - computing a new point B , given a point A and a vector V
 - then computing the midpoint of the segment from A to B
 - `var A = new THREE.Vector3(1,3,5);`
 - `var V = new THREE.Vector3(10,20,30);`
 - `var B = A.clone();`
 - `B.add(V);`
 - `alert("B is " + JSON.stringify(B));`
 - `var Mid1 = A.clone();`
 - `Mid1.lerp(B,0.5); alert("midpoint is " + JSON.stringify(Mid1));`

Computing with Three.js interpolation

- Here's another way to do the same thing
 - `var A = new THREE.Vector3(1,3,5);`
 - `var V = new THREE.Vector3(10,20,30);`
 - `var B = A.clone();`
 - `var Mid2 = A.clone();`
 - `var Vhalf = V.clone();`
 - `Vhalf.multiplyScalar(0.5);`
 - `Mid2.add(Vhalf);`
 - `alert("midpoint is also " + JSON.stringify(Mid2));`

Interpolating the steeple coordinate using Three.js

- The code below carries out our earlier computation of the coordinates of the point B for the steeple, given the vertices R and S.
 - `var R = new THREE.Vector3(15,55,0);`
 - `var S = new THREE.Vector3(0,30,0);`
 - `var Ans1 = S.clone();`
 - `Ans1.lerp(R,0.8);`
 - `alert("Ans1 is " + JSON.stringify(Ans1));`

Interpolating the steeple coordinate using Three.js

- An alternative method:
 - `var R = new THREE.Vector3(15,55,0);`
 - `var S = new THREE.Vector3(0,30,0);`
 - `var V = new THREE.Vector3();`
 - `V.subVectors(S,R); // vector down the roof`
 - `V.multiplyScalar(0.2);`
 - `var Ans2 = R.clone();`
 - `Ans2.add(V);`
 - `alert("Ans2 is " + JSON.stringify(Ans2));`

Colors, Interpolation and RGB

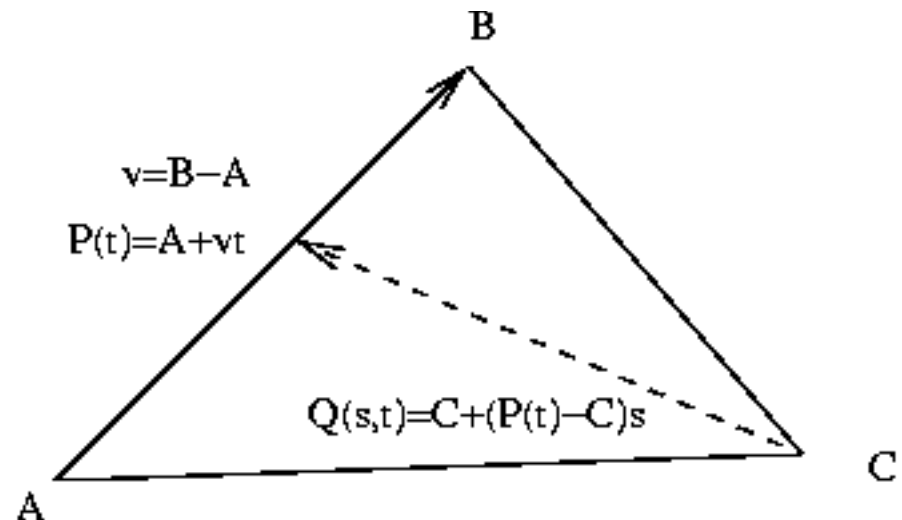
- **RGB color** is a three-dimensional system just like our 3D spatial coordinates.
- Color interpolation works pretty much the same way as spatial interpolation.
- Assuming the vertex S is cyan, and R is red, what color is B?
 - `var cyan = new THREE.Vector3(0,1,1);`
 - `var red = new THREE.Vector3(1,0,0);`
 - `var mix = cyan.clone();`
 - `mix.lerp(red,0.8);`
 - `alert("mix is " + JSON.stringify(mix));`

Parametric Equation for a Triangle

- Since a triangle is a 2D thing, the parametric equation for a triangle will have *two* parameters
 - the first parameter, say t , moves you along one side of the triangle, from vertex A to vertex B.
 - Let $P(t)$ be that point along the AB edge of the triangle.
 - The second parameter, say s , is the parameter of a line from vertex C to $P(t)$.
 - That is, the endpoint of the second line is a moving target.
 - The point $Q(s,t)$ is a point in the triangle, on a line between C and $P(t)$.

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:



Choices

- Notice that we have several choices:
 - The line from A to B could instead go from B to A.
 - Similarly, the line from C to $P(t)$ could go from $P(t)$ to C.
- These yield equivalent equations
 - just as the equation of a line from A to B is equivalent to the equation of a line from B to A.

Parametric triangle as a weighted sum

$$Q(s, t) = A(1 - t)s + Bts + C(1 - s)$$

- This is a three-way mixture of the vertices.
- Meaning a triangle is all points in the *convex sum* of the vertices.
- A *convex sum* is a weighted sum of N things, where the weights all add up to 1.0:
- $S = w_1 \cdot A + w_2 \cdot B + w_3 \cdot C$
- $1 = w_1 + w_2 + w_3$

Parametric triangle as weighted sum

- Do the weights sum to 1 for

$$Q(s, t) = A(1 - t)s + Bts + C(1 - s)$$

- Let's see

$$(1 - t)s + ts + (1 - s) = 1$$

- Incidentally, the center of the triangle is where all the weights are equal: one-third.

Example: Equation of a Triangle from Three Points

- Suppose we have a triangle ABC whose vertices are:
 - $A = (1,2,3)$
 - $B = (2,4,1)$
 - $C = (3,1,5)$
- We could write down the following equation for the triangle:
 - $Q(s, t) = A(1 - t)s + B(ts) + C(1 - s)$
 - $Q(s, t) = (1,2,3)(1 - t)s + (2,4,1)ts + (3,1,5)(1 - s)$

Example: Equation of a Triangle from Three Points

- We could write down the following equation for the triangle:
 - $Q(s, t) = A(1 - t)s + B(t)s + C(1 - s)$
 - $Q(s, t) = (1, 2, 3)(1 - t)s + (2, 4, 1)ts + (3, 1, 5)(1 - s)$
- Each coordinate separately is:
 - $x(s, t) = (1 - t)s + 2ts + 3(1 - s)$
 - $y(s, t) = 2(1 - t)s + 4ts + (1 - s)$
 - $z(s, t) = 3(1 - t)s + ts + 5(1 - s)$

Example: Equation of a Triangle from Three Points

- We can simplify this algebraically to:

- $x(s, t) = (1 - t)s + 2ts + 3(1 - s) =$
- $= s - ts + 2ts + 3 - 3s =$
- $= ts - 2s + 3$

- $y(s, t) = 2(1 - t)s + 4ts + (1 - s) =$
- $= 2s - 2ts + 4ts + 1 - s =$
- $= 2ts + s + 1$

- $z(s, t) = 3(1 - t)s + ts + 5(1 - s) =$
- $= 3s - 3ts + ts + 5 - 5s =$
- $= -2ts - 2s + 5$

Example: Equation of a Triangle from Three Points

- Suppose a point's parameters with respect to that triangle are $(0.5, 0.5)$.
 - What does that mean?
 - It means that the point is halfway between C and the midpoint of AB.
- The coordinates are:
 - $x(0.5, 0.5) = (0.5)(0.5) - 2(0.5) + 3 = 2.25$
 - $y(0.5, 0.5) = 2(1 - 0.5)0.5 + 4(0.5)(0.5) + (1 - 0.5) = 2$
 - $z(0.5, 0.5) = 3(1 - 0.5)0.5 + (0.5)(0.5) + 5(1 - 0.5) = 3.25$
 - \Rightarrow the coordinates of $Q(0.5, 0.5)$ are $(2.25, 2, 3.25)$

Equivalent solution

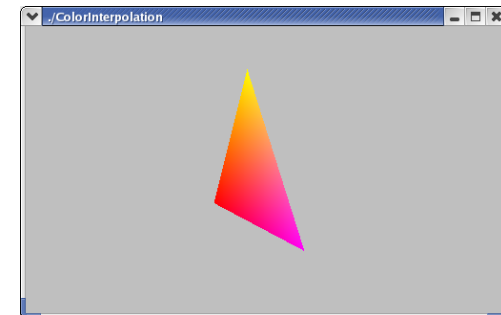
- $Q(s,t)$ can be computed as a weighted sum of the triangles' vertices:
 - $Q(s,t) = A(1-t)s + B(t)s + C(1-s)$
 - $Q(0.5,0.5) = A(1-0.5)(0.5) + B(0.5)(0.5) + C(1-0.5)$
 - $Q(0.5,0.5) = A(0.25) + B(0.25) + C(0.5)$
- To compute the coordinates of $Q(0.5,0.5)$:
 - substitute the coordinates of ABC
 - calculate the weighted sum

Color Interpolation in a Triangle

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

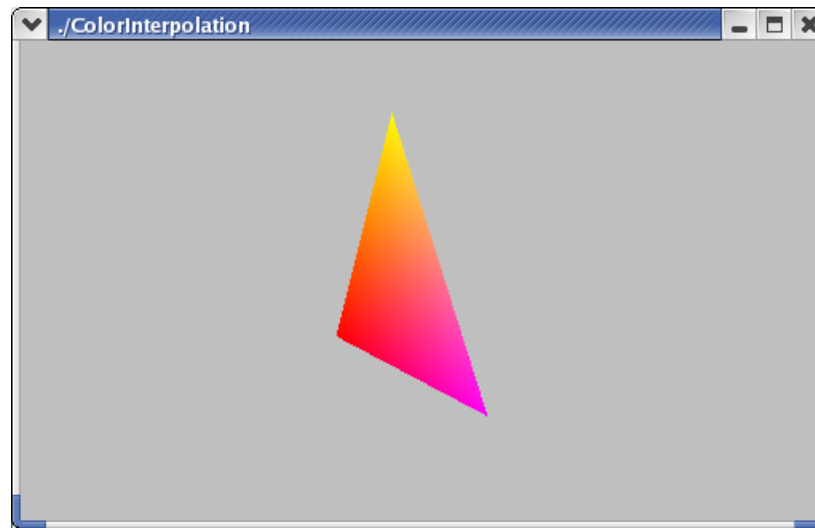
Color Interpolation in a Triangle

- 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`

Questions?

