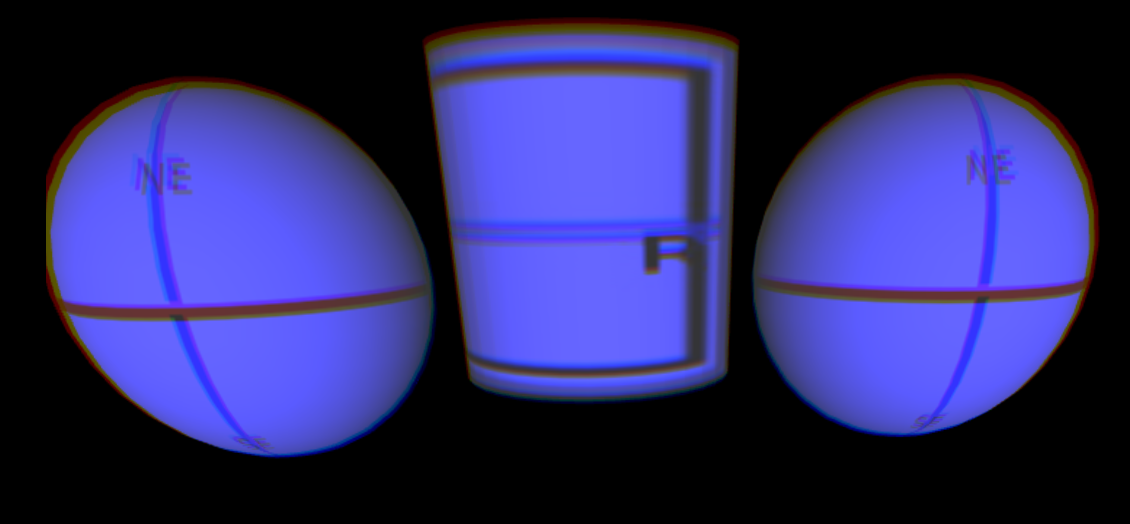
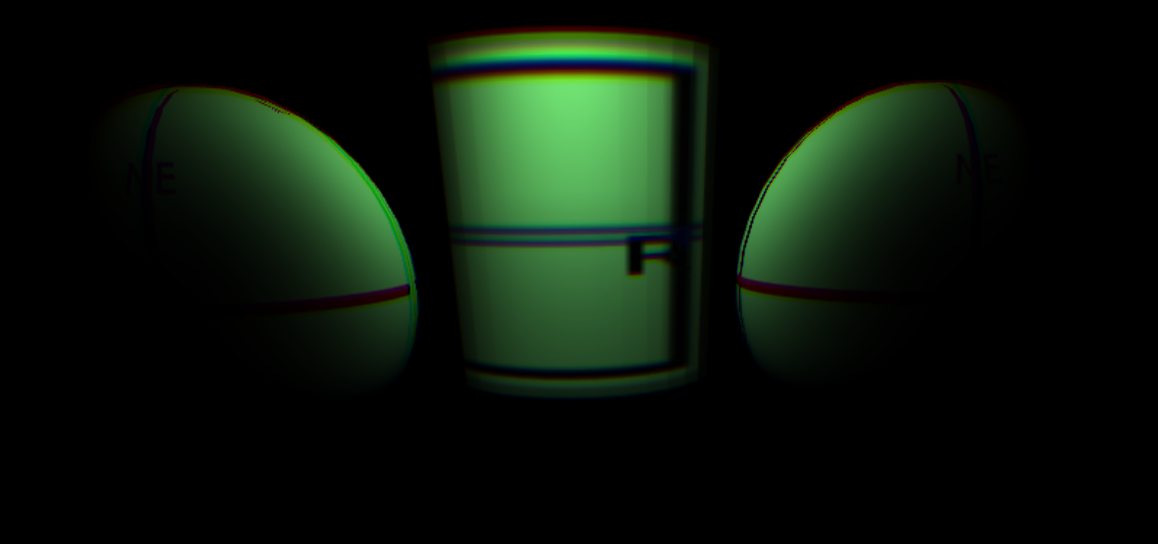
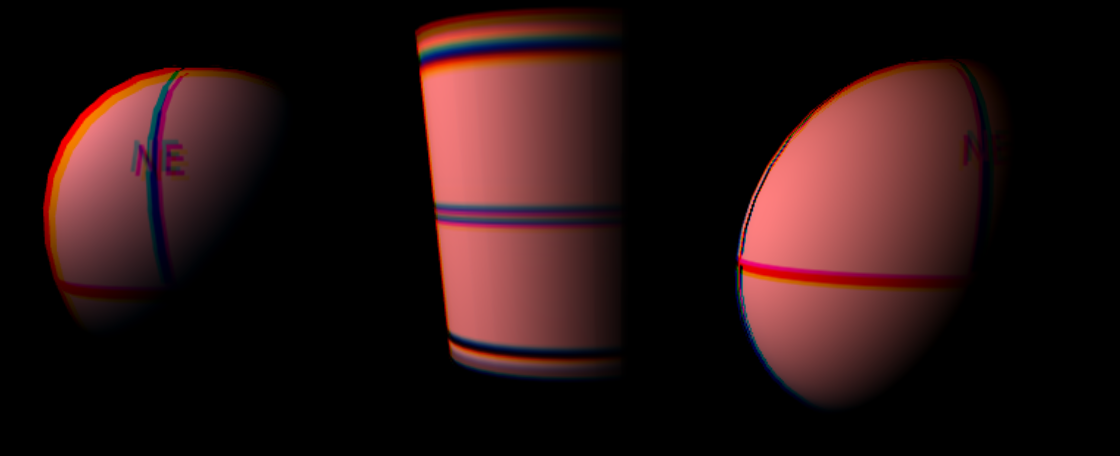
**Scripting**

These are demonstrations with the <SCRIPT> node that contains JavaScript. Users are encouraged to modify the JavaScript code.

**JavaScriptLightsOn.x3d**

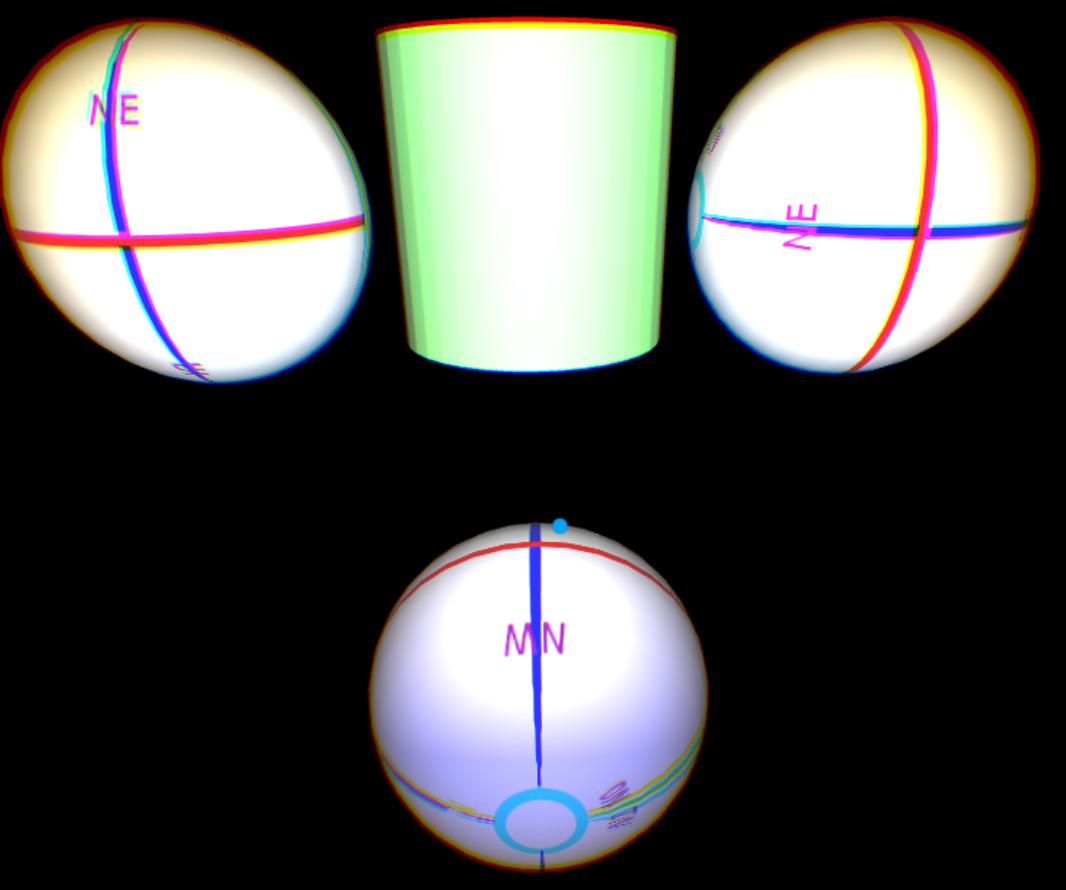
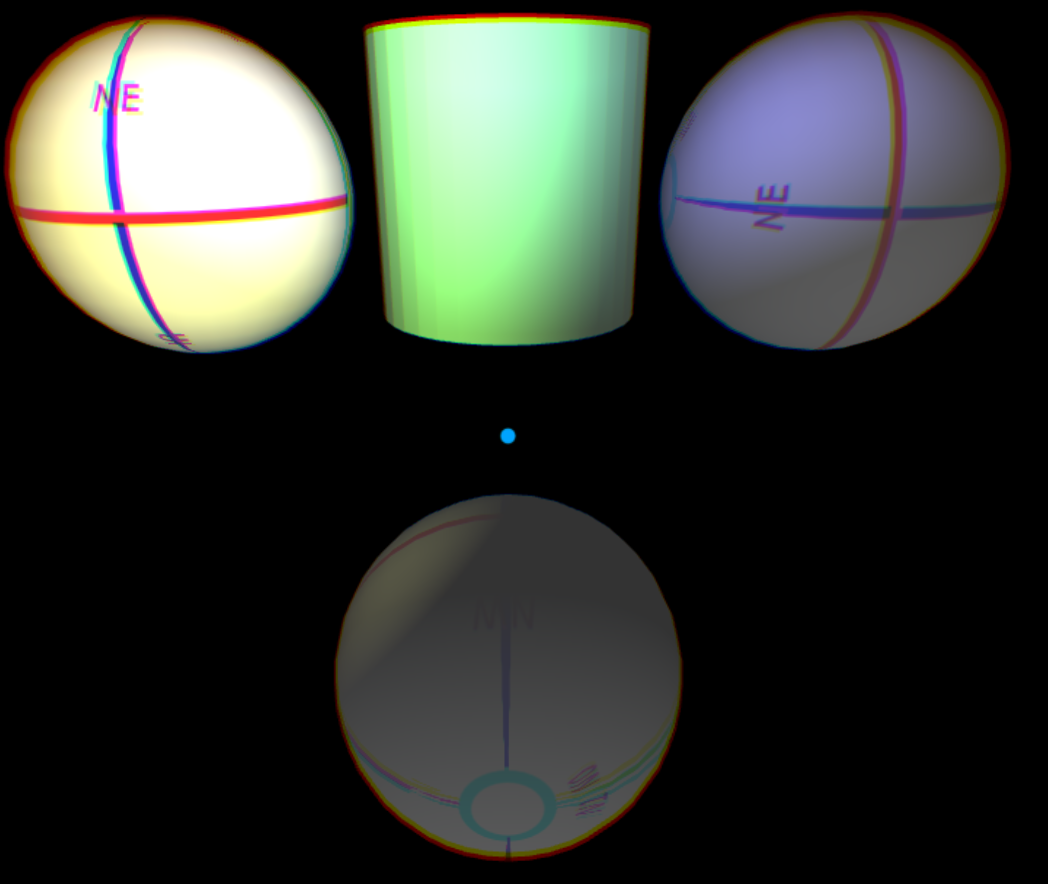
Demonstration has 3 lights – a red point light on the left at (-3, 1, -4); a green spot light at (0, 1, -4); and a blue directional light. Placing the cursor over the center cylinder will cycle through the red point light, the green spot light, and then the blue directional light.



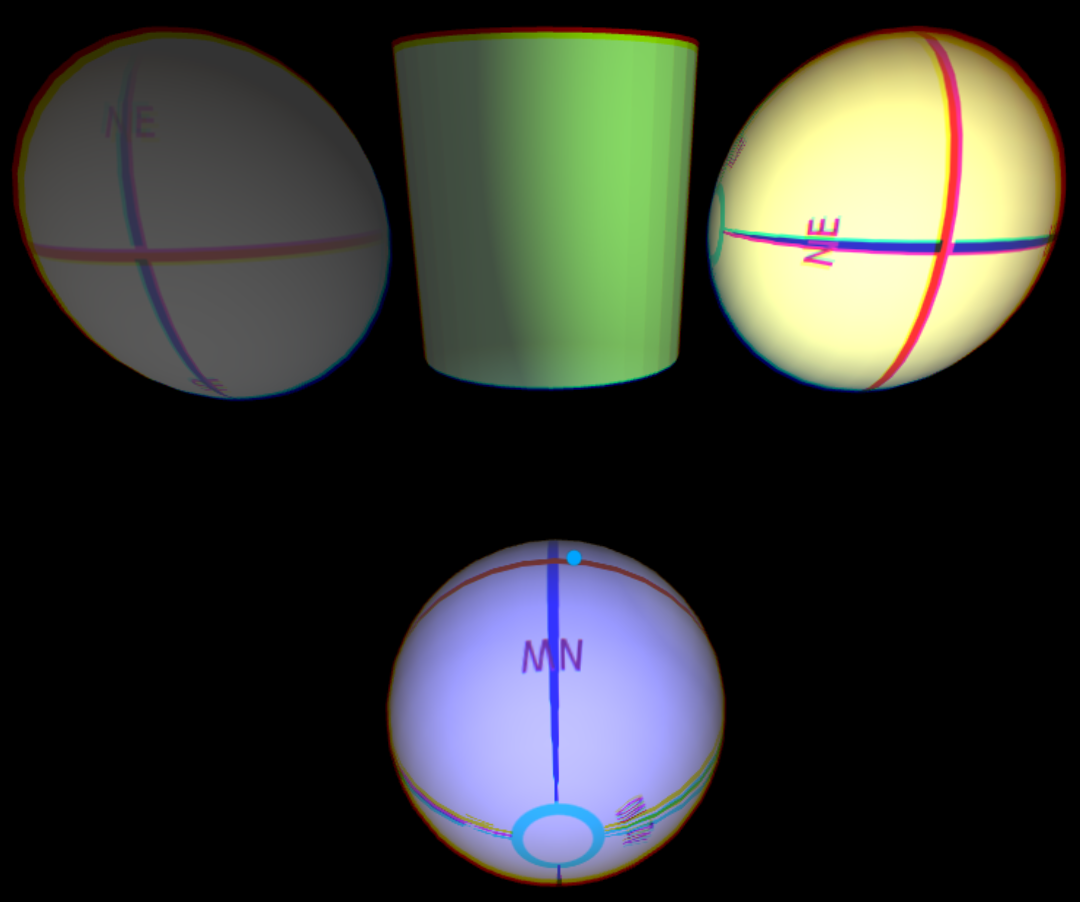
**JavaScriptSpotLightDirection.x3d**

Scene has two spot lights at the origin. The yellow spot light begins pointing toward the left at (-.6, 0, -.8) and the blue spot light begins pointing up toward (0, .5, -.866). While rolling over the center green Cylinder, the red spot light increases its x-direction value by .1, moving left to right, and the blue spot light decreases its y-direction value by .15, and progressively pointing down. Incidentally, the spheres each have their own rotations.

In the opening scene, the yellow spot points toward the left and the blue spot light points upward, though the spotlights overlap creating some white light areas. In the second image, the two spot lights are overlapping.



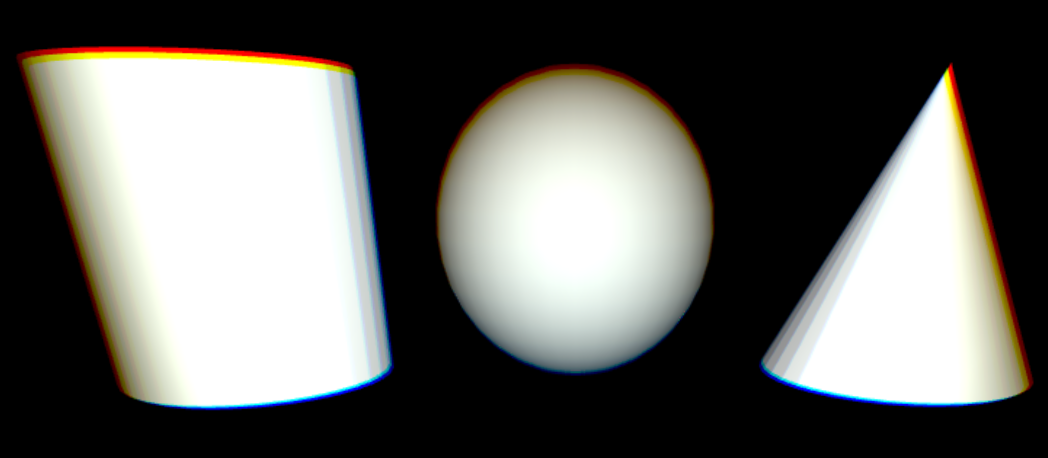
In this final scene, the yellow spot light has progressively pointed toward the right, and the blue spot light is now pointing down.



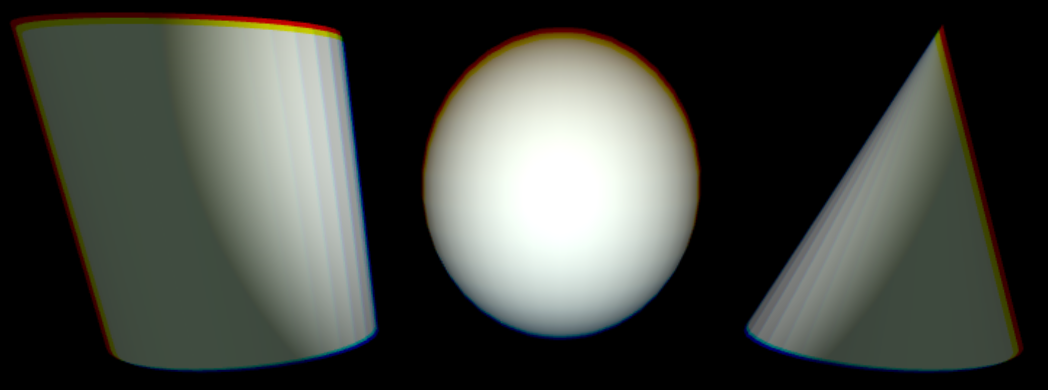
**JavaScriptSpotLightBeamWidthCutOffAngle.x3d**

Scene begins with a spotlight with default BeamWidth at .785 radians (PI/4) and CutOffAngle at the default 1.57 radians (PI/2). Rolling over the center Sphere decreases the BeamWidth and CutOffAngle by 20%. When either of those values gets lower than .2 radians, either is reset to its default values.

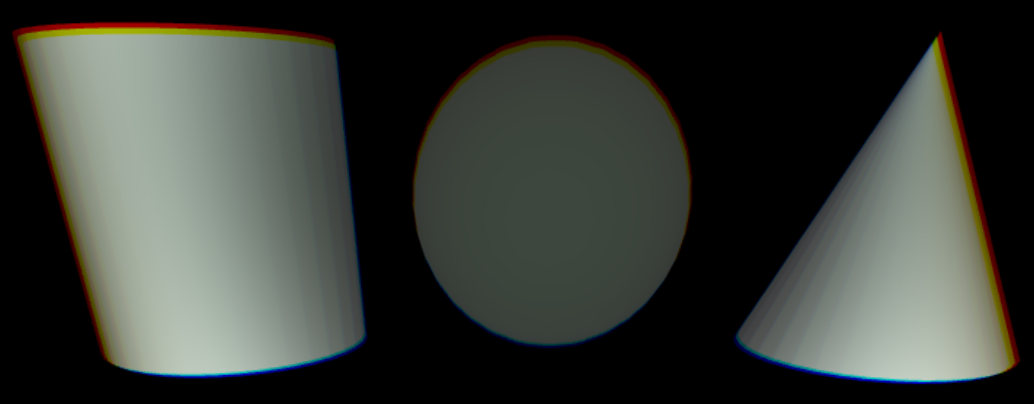
Here is the opening scene:



After rolling over the center Sphere 3 times, meaning the BeamWidth is at .4 radians (PI/4 \* .83) and the CutOffAngle is at .8 radians (PI/2 \* .83).



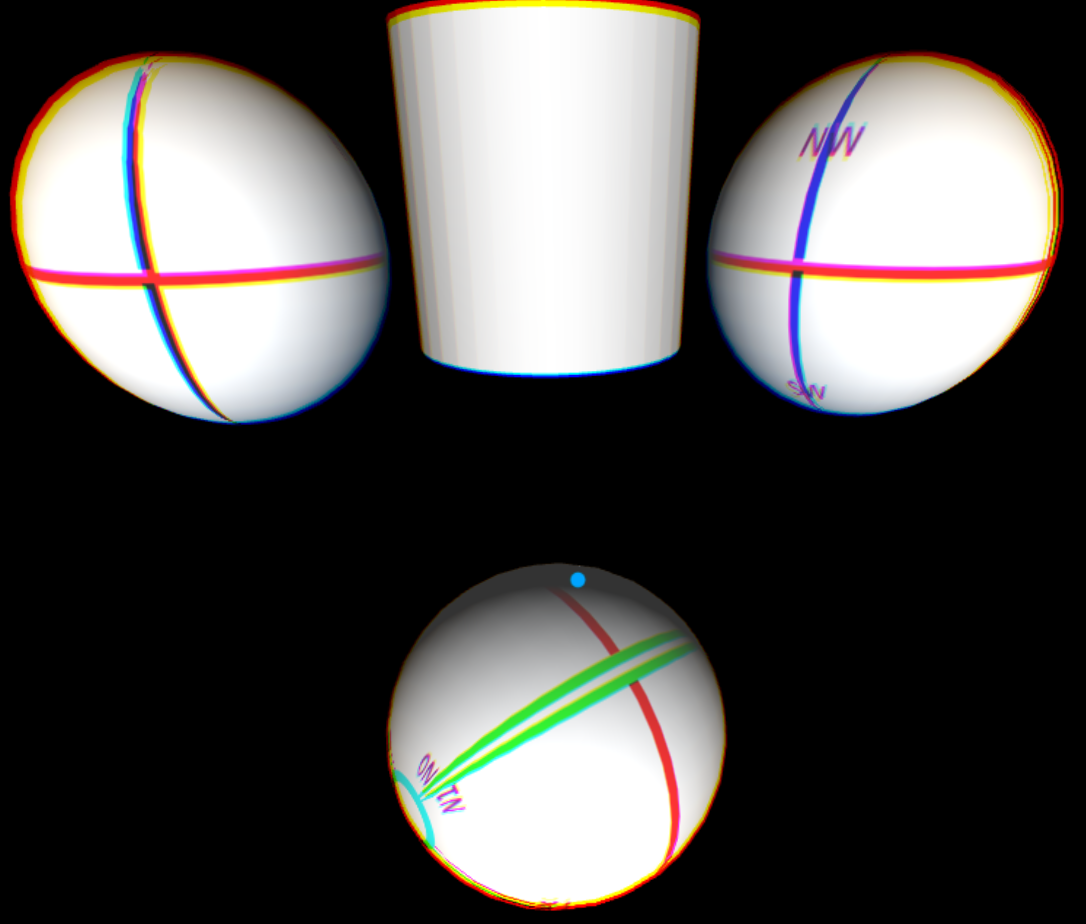
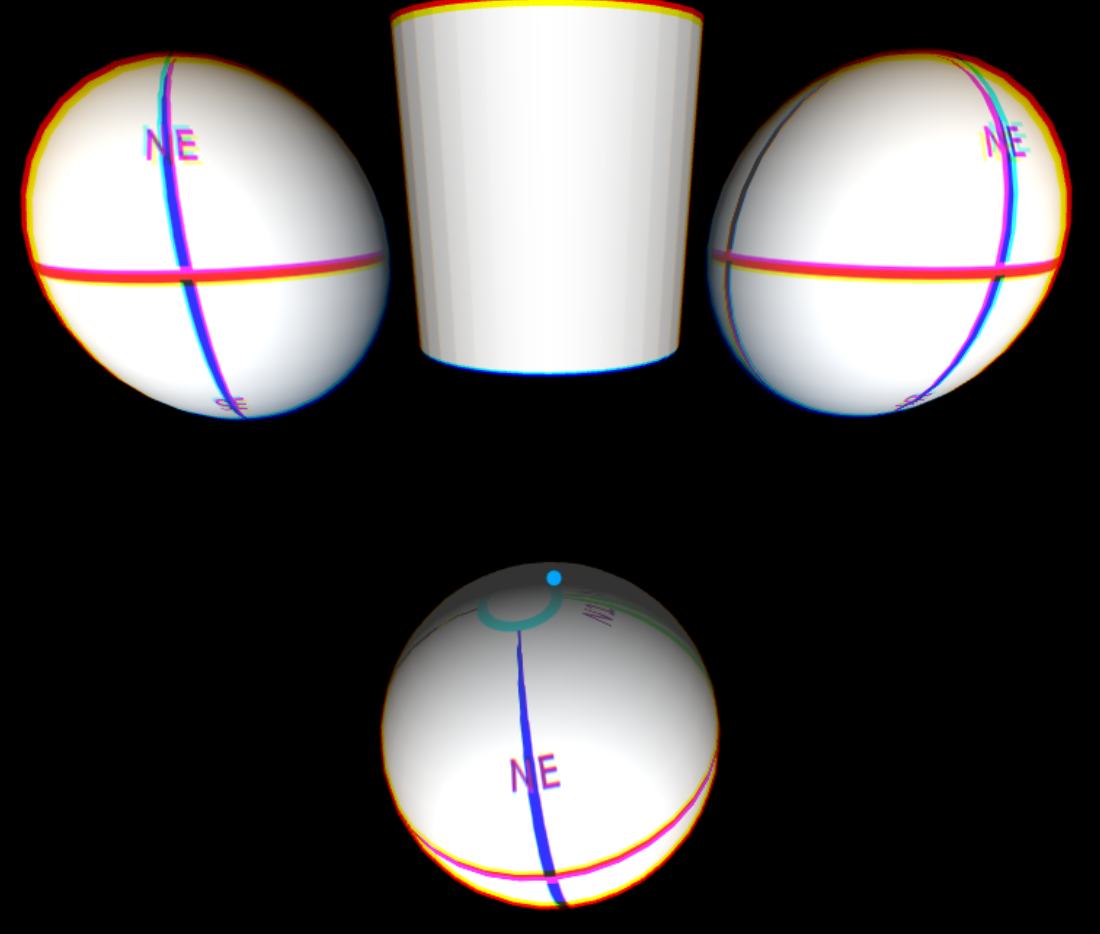
The scene below the center Sphere is rolled over 6 times and thus the BeamWidth was reset to its default of PI/4, and now larger than the CutOffAngle. X3D specification states the BeamWidth should be capped at the CutOffAngle, however, OpenGL renders it this way.



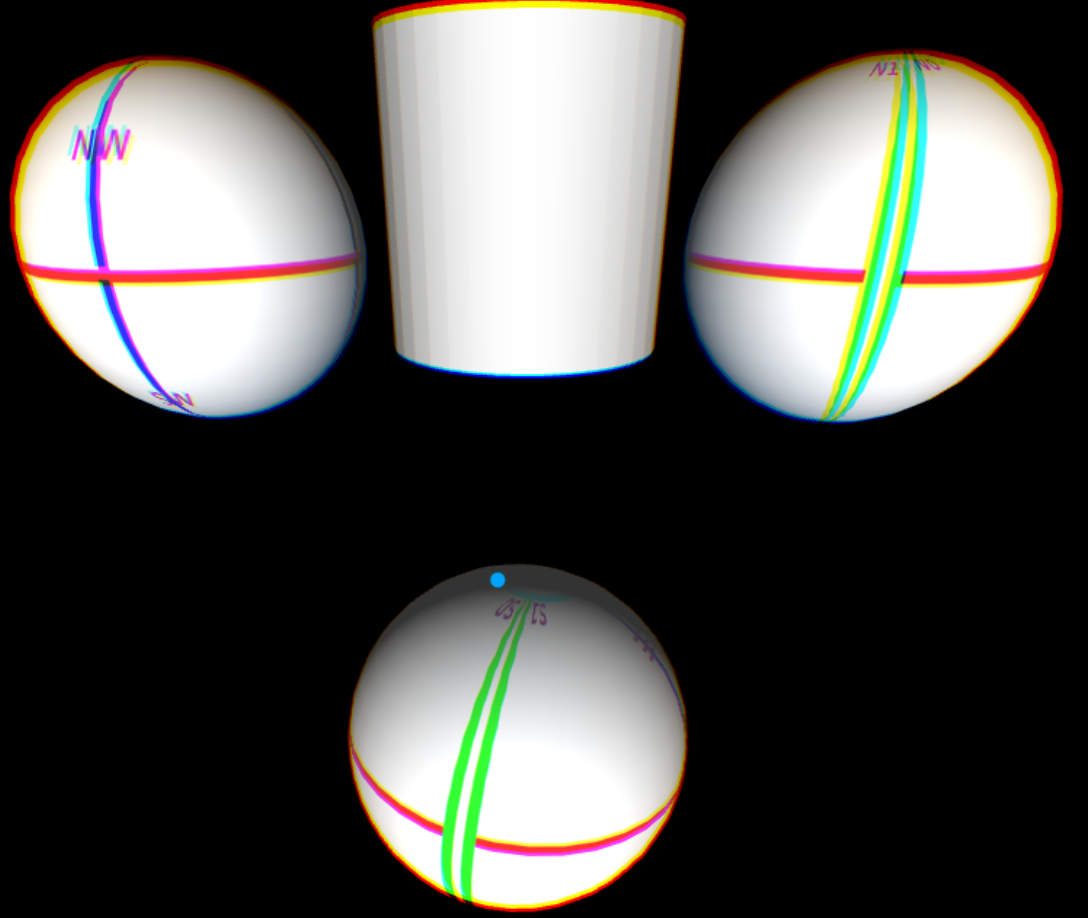
**JavaScriptsfRotation.x3d**

Scene contains 3 Spheres that rotate when rolling over the center Cylinder. The left Sphere rotates .2 radians around the y-axis. The right Sphere rotates .3 radians around the y-axis. The bottom Sphere begins with an Axis-Angle rotation of (.6, 0, .8, 0 radians). The bottom Sphere updates its Rotation on a roll over of the Cylinder, and a roll off.

The second scene is after 7 roll-overs. The left Sphere rotated 1.4 radians around the y-axis, the right axis rotated 2.1 radians around the y-axis, and the bottom Sphere has rotated (.6, 0, .8, 1.4 radians).

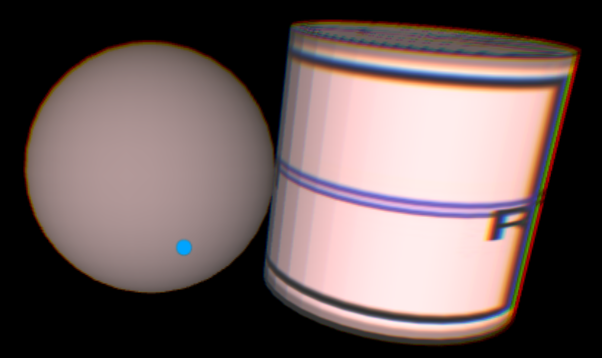
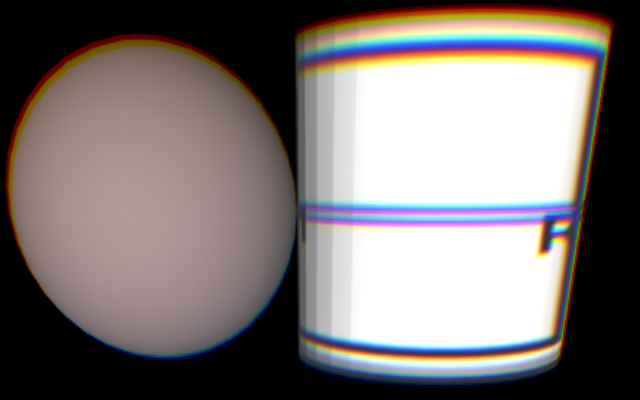


This scene is after 14 clicks. The bottom Sphere has rotated (.6, 0, .8, 2.8 radians).

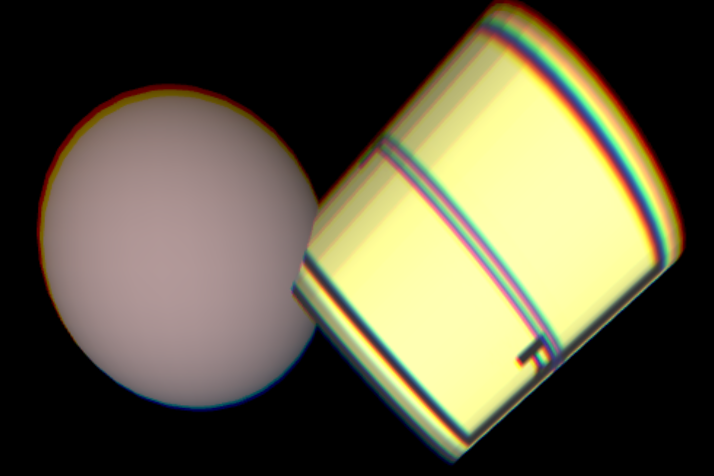


**JavaScriptSFRotationVectorMultiply.x3d**

This demo tests a method of the SFRotation object where two vectors can be used to create an Axis-Angle rotation. After rolling over, and then off, the Sphere, the Cylinder is colored partially red and rotated -.8 radians around the Z-axis.



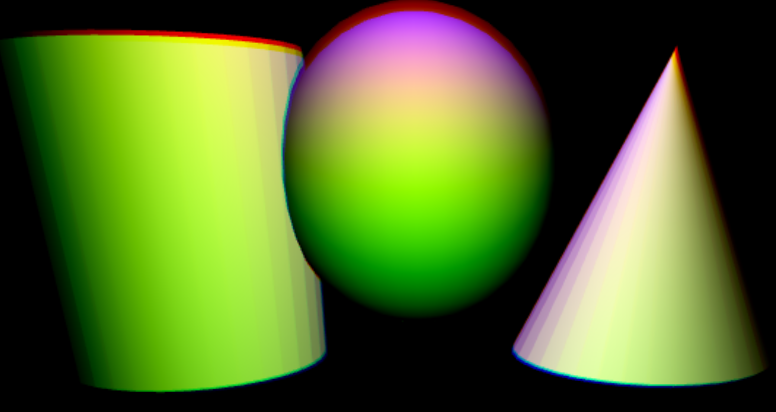
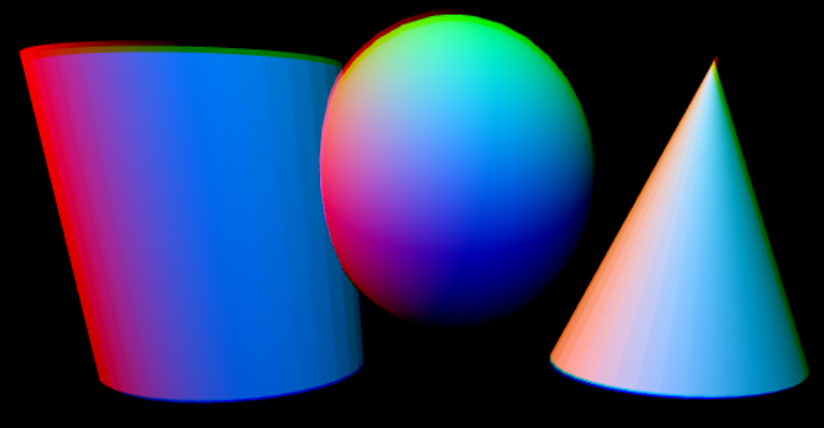
Rolling over the Sphere, Cylinder is yellow and rotated from two vectors (0, .8, -.6) and ( -.5, -.707, -.5).



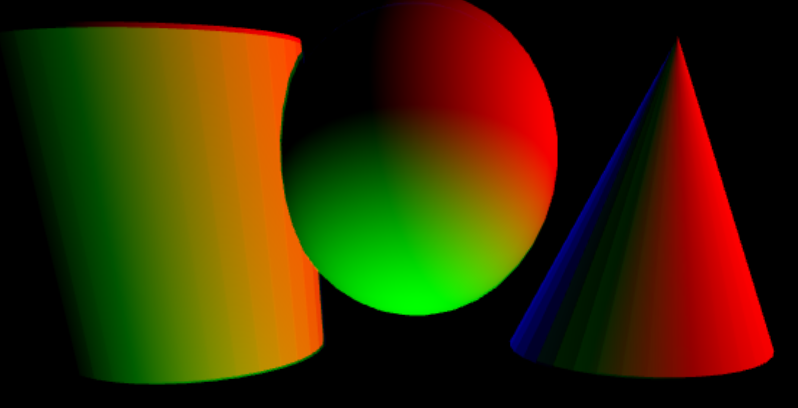
**JavaScriptMoveColorPointLights.x3d**

The scene has a red point light is on the left at (-5, 1, -4), a green point light above at (0, 5, -3), and a blue point light above the camera at (0, 2, 0). When rolling over the center Sphere the red point light increases its x-value by 1 and thus moves from left to right, the green point light deducts its y-value by 1 and moves from above to below, and the blue point light deducts its z-value by 1 and thus moves from above the camera toward and then behind the scene objects.

The second scene is after 5 clicks.

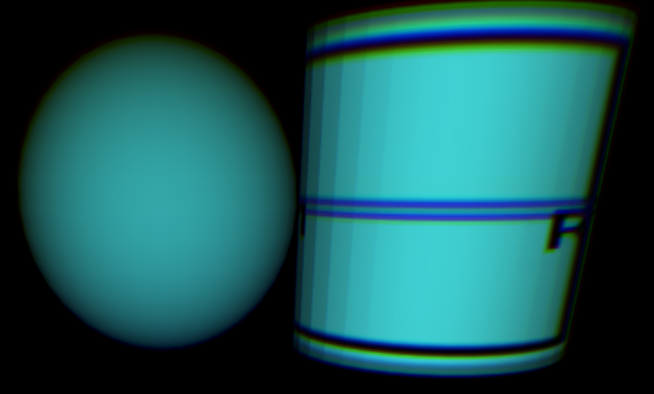
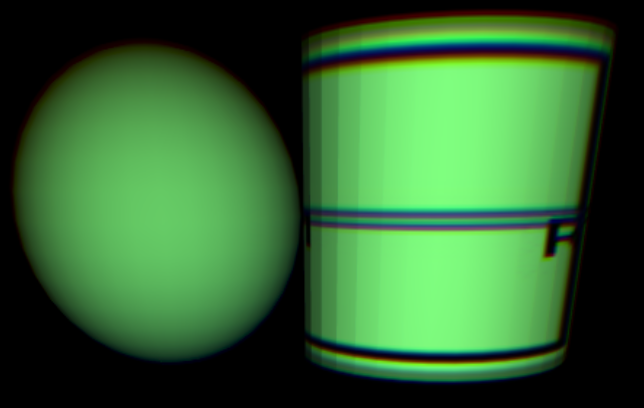


This final scene is after 10 clicks. The blue light is above the Sphere, the red light is on the right, and the green light is below the Sphere.



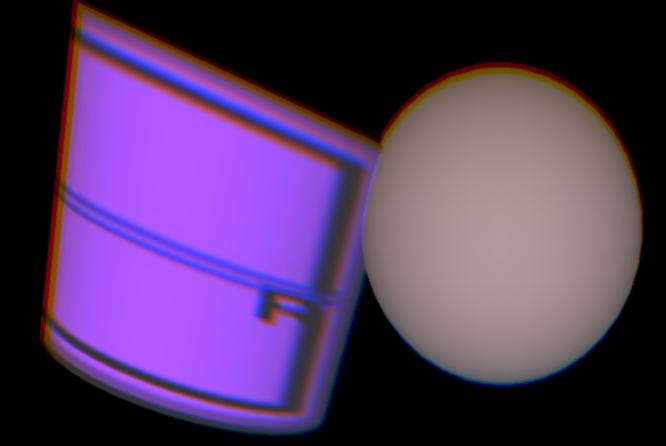
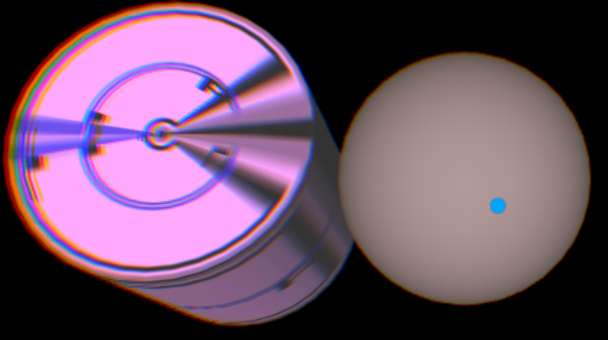
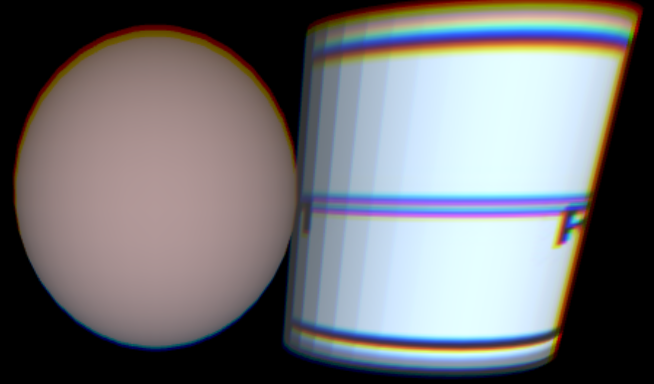
**JavaScriptLightColorModified.x3d**

Scene has a point light with color initialized to light green (.5, 1, .5). Each cursor over the sphere, it adds .13, .19 and .29 respectively to the red, green and blue. If the color exceeds 1, then it subtracts 1 from that color, but sets a minimum value of .25 for any color.



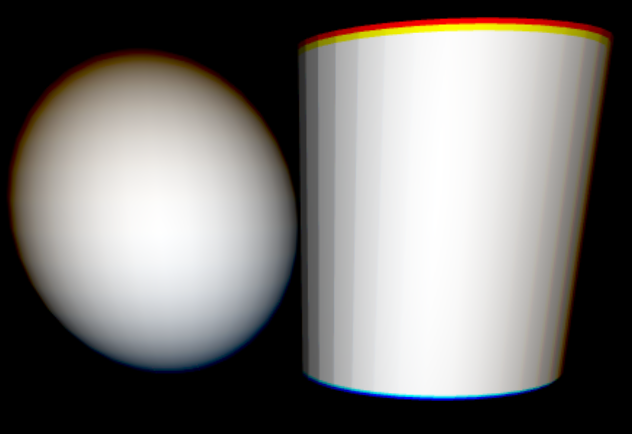
**JavaScriptSFColorInputOutput.x3d**

This tests internal programming where we send the script a color parameter *objColor*, and output another *myColor*, but it modifies the same Cylinder material diffuse color. The position and orientation of the Cylinder is also modified when rolling over and off the Sphere.

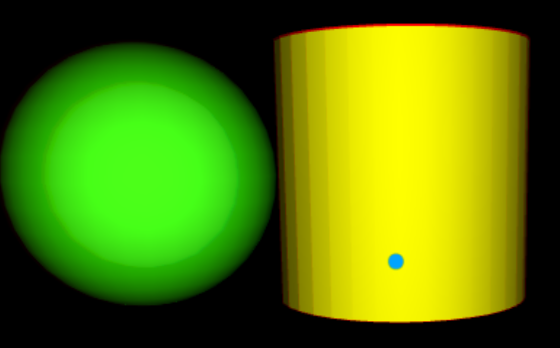
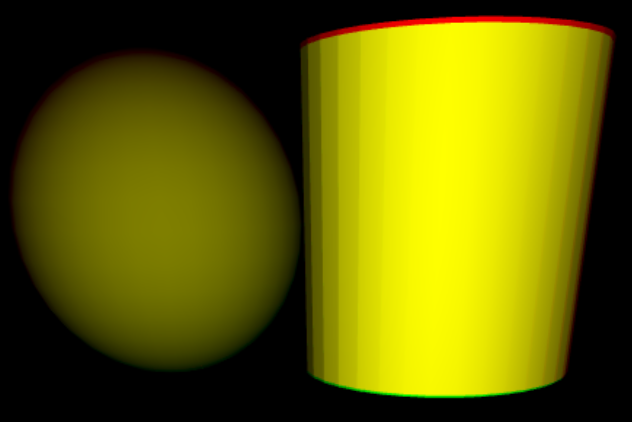


**JavaScriptMultipleScripts.x3d**

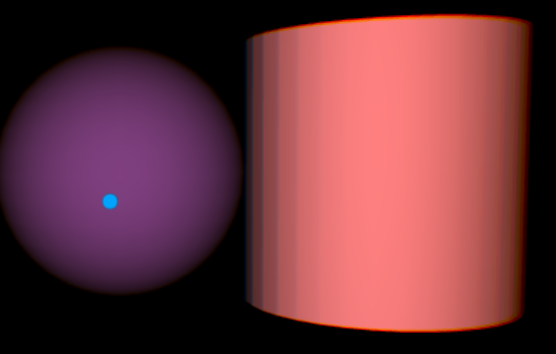
Tests multiple methods in a single SCRIPT.



Rolling over the Cylinder changes the Sphere’s color to green and rolling off to yellow.

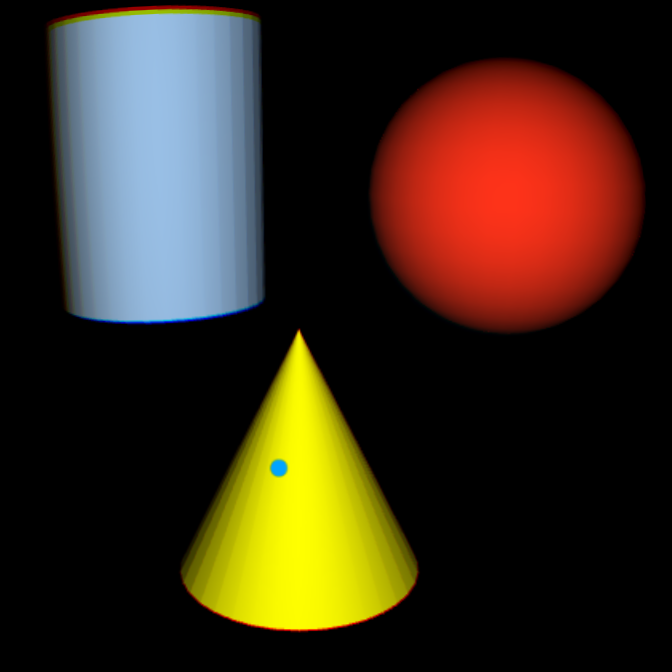


Rolling over the Sphere changes the light’s color to red, and rolling off back to yellow.



**JavaScriptInitialize.x3d**

Tests the SCRIPT’s initialize() method, which is called upon initialization of the scene. Originally, the Sphere is gray, but that will never be seen because the initialize() method changes it to red.



Clicking on the gray blue sphere will change the Sphere from green to blue and the Cone from magenta to cyan.

