Papervision3D star (sun) tutorial and source

by Benny Bottema - Thursday, February 11, 2010

http://www.bennybottema.com/2010/02/11/papervision3d-star-sun-tutorial-and-source/

If you are looking for a cool sun effect (no pun intended) with flaring corona, you're in the right place. This post isn't about some practical sun in some shooter though: it would require some tweaking to make it useful in such context, as the animation is rather cpu heavy due to its massive corona noise textures.

No, this post is about combining cool PV3D features in order to achieve a nice looking animated sun. I've used the trunk of the papervision code repository, revision 851 (it works up to r910, after that a bug was introduced). Note that I'm no PV3D guru; I'm just sharing what I've learned so far.

- download sun source

```
/wp-content/uploads/2016/02/Suntutorial.swf, 400, 400
```

(note: this sun uses the skybox from the skybox tutorial)

Closely watch that corona for a while!

Contents

How it works

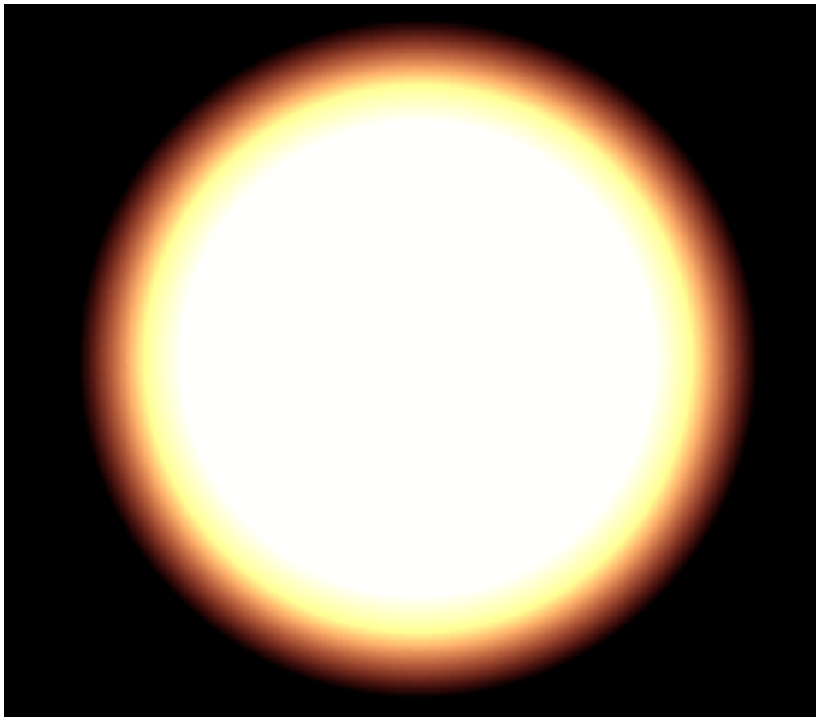
In short: the sun exist entirely of planes, controlled by alpha ratios and always facing the camera:

- one plane with a gradient texture white to red with glow
- two planes with a perlin texture, that resize and tag eachother as to provide a continuous outwards motion
- one plane that acts as a mask for the other planes to provide a perfect circle giving the illusion of a round star

The star’s gradient

Actually, it took quite some tweaking to get a nice gradient of white in the middle and yellow to red on the edge. It’s a basic Sprite gradient fill, using three colors, alpha layering and a specific color ratio array to get the last two colors (yellow and red) on the edge. Then I’m applying this gradient *twice* on the same canvas to get the result I want when the flaring corona’s are added. I couldn’t get it the way I wanted with the alpha array alone (let me know if you got a better way), but this worked and the gradients are only calculated and drawn once.

Drawing it twice means less transparency and that means more dense flares as we’ll see later on when we’re combining the star glow layer and corona layer using BlendMode.ADD (which is also the reason why we need MovieMaterial here: otherwise they won’t interact on layer-level).



```
// pseudo code
var sunColors:Array = [0xfefefe, 0xFAEB61, 0xff0000]; // white, orange
/yellow, red
var sunAlphas:Array = [1, .8, 0]; // white part opaque, 20% from the e
dge out fading out
var sunRatios:Array = [0x85, 0xAA, 0xDD]; // all white except for the
edges

var size:Number = 1000;
var sunglowMaterial:MovieMaterial = createSunglowMaterial(size, sunCol
ors, sunAlphas, sunRatios);
var sun:Plane = new Plane(sunglowMaterial, size, size, null, null);
```

```
/**
 * Creates a disc texture using radial alpha. Draw twice for extra int
ensity of the colors.
 */
private function createSunglowMaterial(size:Number, colors:Array, alph
as:Array, ratios:Array):MovieMaterial {
 var mat:Matrix = new Matrix();
 mat.createGradientBox(size, size);
 var sunTexture:Sprite = new Sprite();
 for (var i:Number = 0; i
  sunTexture.graphics.beginGradientFill(GradientType.RADIAL, colors, a
lphas, ratios, mat);
  sunTexture.graphics.drawRect(0, 0, size, size);
  sunTexture.graphics.endFill();
 }
 return new MovieMaterial(sunTexture, true);
}
```

The star's flaring corona

The corona was tricky. I needed to find some way to create a seamless and continuous outward motion of a Perlin noise map that would function as a flare animation. In the end I settled for a two plane solution where one slowly grows and finally fades out at which point the second fades in, rotated randomly and grows until it fades out again. Rinse and repeat. When calibrated carefully, the resizing, rotating and fading is imperceptible to the eye, due to the naturally distributed Perlin noise:

```
/wp-content/uploads/2016/02/imperceptible-corona.swf, 300, 300
```

(imperceptible version)

```
/wp-content/uploads/2016/02/perceptible-corona.swf, 300, 300
```

(perceptible version)

```
/wp-content/uploads/2016/02/sun-final-corner.swf, 200, 200
```

(result)

The Perlin noise is added with a simple Perlin noise command as follows:

```
var coronaMaterial1:BitmapMaterial = createCoronaMaterial(size / 1.25,
                        size / 100, SEGMENTS);
var coronaMaterial2:BitmapMaterial = createCoronaMaterial(size / 1.25,
                        size / 100, SEGMENTS);
            var corona1:Plane = new Plane(coronaMaterial1);
            var corona2:Plane = new Plane(coronaMaterial2);

                                    /**
 * Creates a perlin noise texture to simulate the flaring flames on th
                        e sun's outer glow.
                                    */
private function createCoronaMaterial(textureSize:Number, noiseSize:Nu
                mber, segments:Number):BitmapMaterial {
 var coronaBitmap:BitmapData = new BitmapData(textureSize, textureSize
                                    );
 coronaBitmap.perlinNoise(noiseSize, noiseSize, 4, 61, true, true, 1,
                                  false);
             return new BitmapMaterial(coronaBitmap, true);
                                    }
```

The algorithm to animate the two planes is included in the attached sources.

The corona and star glow combined and masked by a perfect circle

The planes are all on the same depth and level. This works because we're using alpha ratios to determine where one plane's visibility starts and the other disappears. The star glow textures stands on its own, it is inherently radial and as such doesn't need a circle mask. This leaves the Perlin noise textures which need masking. To do this, I've put them in separate layers, which are then both masked by a third layer.

```
                                    /*
 * use root corona layer so we can perform an alpha mask to both child
                                  layers at once
 *     - contains child corona layers so we can control individual alp
                                  ha properties
 *         - contains child corona planes so we can apply a perlin noi
                      se texture to it for 'flaring' effect
                                    */
  var coronaLayer:ViewportLayer = new ViewportLayer(viewport, null);
        coronaChildLayer1 = new ViewportLayer(viewport, null);
        coronaChildLayer2 = new ViewportLayer(viewport, null);
              coronaLayer.addLayer(coronaChildLayer1);
```

```
            coronaLayer.addLayer(coronaChildLayer2);
        coronaChildLayer1.addDisplayObject3D(corona1);
        coronaChildLayer2.addDisplayObject3D(corona2);
       viewport.containerSprite.addLayer(coronaLayer);
      // give the corona of the sun a 'flaring' effect
           coronaLayer.blendMode = BlendMode.ADD;

   // only show the corona part of the flaring perlin noise planes
var coronaMaskLayer:ViewportLayer = viewport.getChildLayer(coronaMask)
                                          ;
            coronaLayer.cacheAsBitmap = true;
        coronaMaskLayer.cacheAsBitmap = true;
         coronaLayer.mask = coronaMaskLayer;
```

On a final note, I'm using the skybox from the skybox tutorial and also the RealtimeDisplayObject3D helper baseclass for the sun's scene updates (which is why the coronas keep animating smoothly even at low FPS).
