Easy Papervision3D skybox tutorial and source

by Benny Bottema - Friday, May 22, 2009

http://www.bennybottema.com/2009/05/22/easy-papervision3d-skybox-tutorial-and-source/

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

Adding a skybox in Papervision3D is extremely easy. I couldn't find a whole lot of examples/tutorials about skyboxes in Papervision3D, but I managed to find a skybox tutorial source and learned how to do it. The tutorial I got the initial source from is actually about adding stars using particle fields, but I just used the skybox part of it (and used the textures).

I decided to clean it up and make a nice class out of it. I've used the trunk of the papervision code repository, revision 851, but since it's very basic stuff I doubt it'll break with future versions.

- download skybox source

```
package org.codemonkey.papervision3d {
 import org.papervision3d.materials.BitmapMaterial;
 import org.papervision3d.materials.utils.MaterialsList;
 import org.papervision3d.objects.primitives.Cube;

 public class Skybox extends Cube {

  public static const QUALITY_LOW:Number = 4;
  public static const QUALITY_MEDIUM:Number = 6;
  public static const QUALITY_HIGH:Number = 8;

  public function Skybox(bf:Class, bl:Class, bb:Class, bu:Class, br:Cl
ass, bd:Class, skysize:Number, quality:Number):void {
   super(generateMaterials(bf, bl, bb, bu, br, bd), skysize, skysize,
skysize, quality, quality, quality);
  }
```

```
  protected function generateMaterials(bf:Class, bl:Class, bb:Class, b
u:Class, br:Class, bd:Class):MaterialsList {
   var materials:MaterialsList = new MaterialsList();
   materials.addMaterial(new BitmapMaterial(new bf().bitmapData), "fro
nt");
   materials.addMaterial(new BitmapMaterial(new bl().bitmapData), "lef
t");
   materials.addMaterial(new BitmapMaterial(new bb().bitmapData), "bac
k");
   materials.addMaterial(new BitmapMaterial(new bu().bitmapData), "top
");
   materials.addMaterial(new BitmapMaterial(new br().bitmapData), "rig
ht");
   materials.addMaterial(new BitmapMaterial(new bd().bitmapData), "bot
tom");

   for each (var material:BitmapMaterial in materials.materialsByName)
 {
    material.doubleSided = true;
   }

   return materials;
  }
 }
}
```

- download Skybox.as

See the line *material.doubleSided = true;*? That's needed since we're inside the cube that is the skybox. Since normally you can only see objects from the outside we're setting the double-sidedness of the skybox. Perhaps a better solution is to flip the faces of the skybox and leave out the double-sidedness all together:

```
  public function Skybox(bf:Class, bl:Class, bb:Class, bu:Class, br:Cl
ass, bd:Class, skysize:Number, quality:Number):void {
   super(generateMaterials(bf, bl, bb, bu, br, bd), skysize, skysize,
skysize, quality, quality, quality);
   geometry.flipFaces(); // now you can only see the box from the insi
de...
  }
```

To use the Skybox class, here's an example:

```
public class SkyboxTest {

  // skybox images
  [Embed (source="assets/hot_nebula_0.jpg")]
  private var BitmapFront:Class;
  [Embed (source="assets/hot_nebula_270.jpg")]
  private var BitmapRight:Class;
  [Embed (source="assets/hot_nebula_180.jpg")]
  private var BitmapBack:Class;
  [Embed (source="assets/hot_nebula_90.jpg")]
  private var BitmapLeft:Class;
  [Embed (source="assets/hot_nebula_bottom.jpg")]
  private var BitmapDown:Class;
  [Embed (source="assets/hot_nebula_top.jpg")]
  private var BitmapUp:Class;

  public static const SKYSIZE:Number = Number.MAX_VALUE;

  public function loadSkyBox(scene:Scene3D) {
   var skybox:Skybox = new Skybox(BitmapFront, BitmapLeft, BitmapBack,
 BitmapUp, BitmapRight, BitmapDown, SKYSIZE, Skybox.QUALITY_HIGH);
   scene.addChild(skybox);
  }
}
```

Here are the steps you need:

- define a bunch of textures in the top of your classes by embedding them and pass these classes to the Skybox constructor.
- pass in a size for your skybox: I've used *Number.MAX_VALUE* so the skybox won't move or pixelate while I move through it with the camera.
- pass in a quality value: this can be any number actually, I just found 4, 6 and 8 to work nicely. These numbers determine how many segments are used in each of the faces of the skybox to avoid texture distortion and awkward transitions from one texture into another.

For my own convenience I've made a NebulaSkybox which simply encapsulates all the embedded images for the nebula sky:

```
 public class NebulaSkybox extends Skybox {

  // nebula skybox images
  [Embed (source="../../../assets/hot_nebula_0_lowquality.jpg")]
  private var BitmapFront:Class;
```

```
  [Embed (source="../../../assets/hot_nebula_270_lowquality.jpg")]
  private var BitmapRight:Class;
  [Embed (source="../../../assets/hot_nebula_180_lowquality.jpg")]
  private var BitmapBack:Class;
  [Embed (source="../../../assets/hot_nebula_90_lowquality.jpg")]
  private var BitmapLeft:Class;
  [Embed (source="../../../assets/hot_nebula_bottom_lowquality.jpg")]
  private var BitmapDown:Class;
  [Embed (source="../../../assets/hot_nebula_top_lowquality.jpg")]
  private var BitmapUp:Class;

  public function NebulaSkybox(skysize:Number, quality:Number) {
   super(BitmapFront, BitmapLeft, BitmapBack, BitmapUp, BitmapRight, B
itmapDown, skysize, quality);
  }
 }

 public class SkyboxTest {

  public static const SKYSIZE:Number = Number.MAX_VALUE;

  public function loadSkyBox(scene:Scene3D) {
   var skybox:Skybox = new NebulaSkybox(SKYSIZE, Skybox.QUALITY_HIGH);
   scene.addChild(skybox);
  }
 }
```
