**Iluminación**

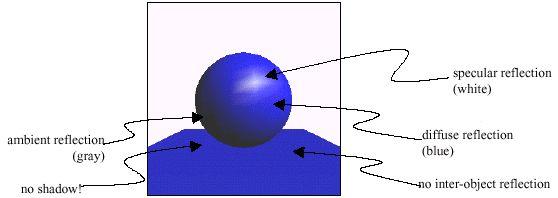
**Caracteristices Básicas**

Hasta el momento todolo que se ha visto a nivel de pintado de las figuras es l a información que se da en el *ColoringAttributes* de la Apariencia. De esta forma no se tiene un buen nivel de realismo, Es por eso que en en esta sección Veremos como Java3D maneja los efectos de la Luz.

Java3D basa completamente sus modelos de 'Shading' en OpenGL, asi que mas información se puede encontrara en la documentación de OpenGL.  
  
**Modelo de Iluminación**

Primero que nada se debe decir que al igual que en OpenGL, el modelo sólo simula los efectos de la luz sobre objetos independientes, por lo que no hay sombras ni reflexiones; para lograr estos efectos se deben simular dibujando poligonos que aparezcan en la escena como sombras o reflejos.

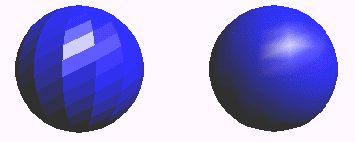
El modelo de iluminación tres tipos de reflexiones de la luz, la *ambiental*, la *difusa* y la *especular*. Para entender mejor estos terminos se presenta la siguiente ilustración



**Colores y Sombreado**

Los colores en Java3D se especifican usando los valores RGB (red, green, blue), y para casos esciales se puede utililzar el RGBA (RGB mas Alpha, transparencia).

Los tipos de sombreado aceptados por Java3D son *Flat* y *Gouraud*; esta caracteristica se debe setear en el *ColoringAttributes*, de la apariencia de los objetos. Las diferencias son claras en la siguiente figura.



**Receta para crear un objeto iluminado**

1. Especificar la fuente de luz
   1. Setear Límites
   2. Ponerlo en el Grafo de la Escena
2. Objeto
   1. Generar normales
   2. Ponerle un *Material* a la apariencia

A continuación un ejemplo mínimo usando la receta

Appearance createAppearance() {

Appearance appear = new Appearance();

Material material = new Material();

appear.setMaterial(material); //2.2 poner material en la apariencia

return appear;

}

BranchGroup createScene (){

BranchGroup scene = new BranchGroup();

scene.addChild(new Sphere(0.5f, Sphere.GENERATE\_NORMALS, //2.1 generar normales del objeto

createAppearance()));

AmbientLight lightA = new AmbientLight();

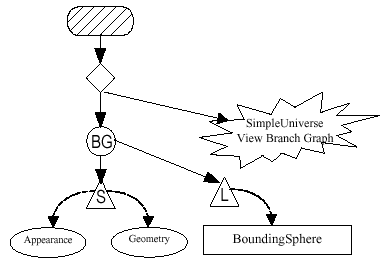
lightA.setInfluencingBounds(new BoundingSphere()); //1.1 Bounds

scene.addChild(lightA); //1.2 ponerlo en el grafo

return scene;

}

Y el grafo correspondiente.



**Fuentes de Luz**

Existen cuatro tipos de fuentes de luz, *AmbientLight*, *PointLight*, *DirectionalLight*y *SpotLight*; para cada una de ellas veremos sus constructores.

AmbientLight(boolean lightOn, Color3f color)

*Crea una luz ambiental con los parametros especificados*

DirectionalLight(boolean lightOn, Color3f color, Vector3f direction)

*Crea la luz direccional con los parametros indicados y en la dirección dada*

PointLight(boolean lightOn, Color3f color, Point3f position, Point3f attenuation)

*Crea una luz puntual con los datos entregados; la atenuación contiene los parametros de una ecuación cuadratica para indicar como la luz pierde intensidad con la distancia*

SpotLight(boolean lightOn, Color3f color, Point3f position, Point3f attenuation, Vector3f direction, float spreadAngle, float concentration)

*Crea la luz 'spot' con los parametros dados*

**Material**

Los materiales presentan las cracteristicas reflectivas de un objeto, el cual es sumado a la información de colores dada por las fuentes de luz. A cotinuacion un resumen de los constructores y métodos de la clase *Material*

Material()

*Crea el material con parametros por default*

Material(Color3f ambientColor, Color3f emissiveColor, Color3f diffuseColor, Color3f specularColor, float shininess)

*Crea el material con los parametros dados*

void setAmbientColor(Color3f color)

*cambia el color de ambiente para el material*

void setDiffuseColor(Color3f color)

*cambia el color de reflexión difusa para el material*

void setEmissiveColor(Color3f color)

*cambia el color emitido por el material*

void setEmissiveColor(Color3f color)

*cambia el color de ambiente para el material*

void setShininess(float shininess)

*cambia el brillo(reflectividad especular) del material*

void setSpecularColor(Color3f color)

*cambia el color de reflexión especular para el material*