Prof. José Ribelles
Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I

Esta práctica pertenece al Bloque 2 de la asignatura y por lo tanto vas a trabajar en lo mismo que se te está explicando en las clases de teoría: normales, modelo de iluminación de Phong, fuentes de luz, modelos de sombreado, coordenadas de textura, técnicas avanzadas de texturas y texturas procedurales. De manera resumida es una práctica que se dirige a mejorar la calidad visual de la escena mediante iluminación, materiales y texturas. A diferencia de la práctica anterior, esta te va a resultar más ligera, más fácil de asimilar y, al mismo tiempo, bastante más agradecida. Se prestará atención especial a cómo modificar los shaders para implementar las diferentes técnicas y soluciones, a veces con mucha ayuda de la GPU pero en otras no tanto.

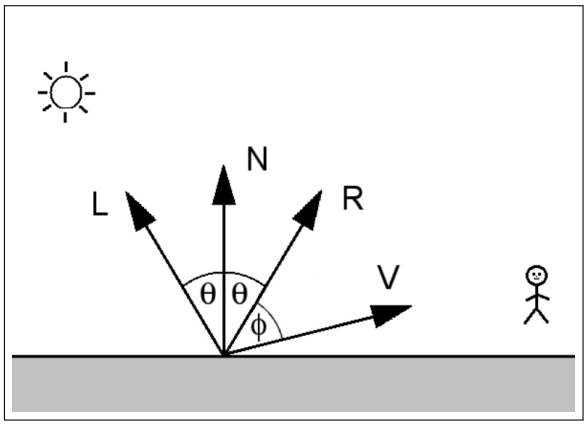
Recuerda que:

- *El boletín es autoguiado, ve a tu ritmo, no tengas prisa por acabar.*
- *Realiza todos los ejercicios que se te indican y contesta a las preguntas que se te plantean.*
- *Entiende lo que estás haciendo y asimílalo, compréndelo.*
- ***Aclara todas las dudas que te surjan.***

El trabajo que has de realizar está organizado en cuatro partes para las que también dispones de cuatro sesiones presenciales de dos horas cada una de ellas. Se utilizará una sesión para cada parte.

Parte I

El modelo de Iluminación de Phong que hemos visto en clase es el que vas a utilizar a partir de ahora para añadir iluminación a tus escenas. La Figura 1 muestra un resumen del modelo, repasa tus apuntes si lo necesitas.



- Iluminación ambiente, $I_a = k_a L_a$
- Reflexión difusa, $I_d = k_d L_d (L \cdot N)$
- Reflexión especular, $I_s = k_s L_s (R \cdot V)^{shininess}$
- Iluminación total en un punto, $I = I_a + I_d + I_s$

Figura 1: Esquema y resumen del modelo de Iluminación de Phong

Vamos a realizar dos simplificaciones en la implementación del modelo:

- solo habrá una única fuente de luz que además será de tipo posicional,
- y no añadiremos el efecto de atenuación.

Como ayuda en el cálculo de la iluminación solo contamos con la siguiente función de la librería *glMatrix* que nos permite obtener la matriz de transformación de la normal a partir de la matriz modelo-vista:

- `mat3.normalFromMat4(mat3.create(),` $M_{modelo-vista}$`)`

Comienza por descargar del *Aula Virtual* los ficheros: `iluminacion.js`, `comun.js`, `materiales.js`, `primitivasGN.js`, `sombreadoPhong.html` y `gl-matrix-min.js`. A continuación carga en tu navegador `sombreadoPhong.html` y comprueba que las primitivas aparecen iluminadas de manera similar a las que se muestran en las imágenes de la Figura 2.

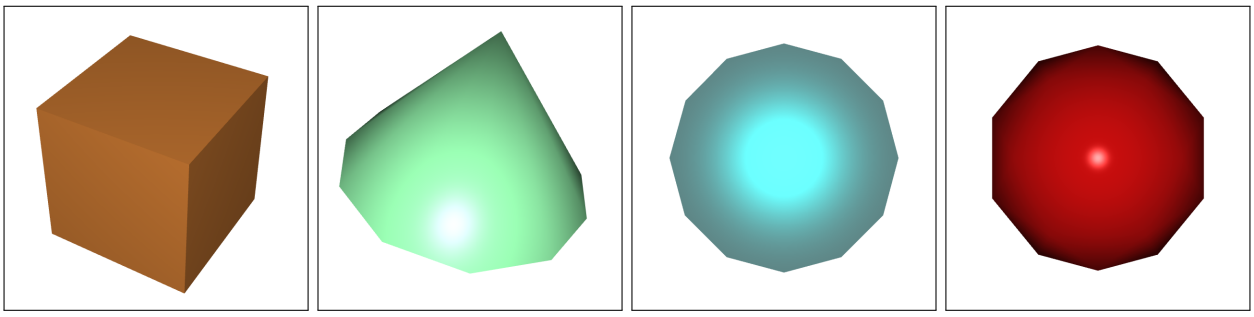


Figura 2: Primitiva y material, izda. a dcha.: cubo (Bronze), cono (Jade), tapa (Cyan rubber), esfera (Ruby)

En efecto, en los ficheros que has descargado ya tienes implementado el **modelo de iluminación** de Phong, donde además se utiliza el **modelo de sombreado** de Phong. Recuerda que no es lo mismo modelo de sombreado que modelo de iluminación (quizá sea un buen momento para que revises tus apuntes).

Veamos ahora qué cambios se han realizado:

- `materiales.js` contiene una pequeña recopilación de materiales para ser utilizados con el modelo de iluminación de Phong. Cada material consta de K_a, K_d, K_s y *shininess*, compruébalo.
- `primitivasGN.js` contiene las mismas primitivas que hay en `primitivasG.js` pero con el nuevo atributo de la normal por vértice. Por lo tanto, los vértices van a tener a partir de ahora dos atributos, **posición** y **normal**. Observa por ejemplo el modelo del plano en el Listado 1 donde a cada vértice se le ha añadido justo a continuación de su atributo de posición su vector normal. En el caso del modelo del plano la normal es la misma para los cuatro vértices. La Figura 3 muestra el modelo del plano con las normales en cada vértice dibujadas.

Listado 1: Vértices del Plano con dos atributos por vértice: posición y normal

```
                        Posición          Normal
"vertices" : [-0.5, 0.0, 0.5,    0.0, 1.0, 0.0,  // v0
               0.5, 0.0, 0.5,    0.0, 1.0, 0.0,  // v1
              -0.5, 0.0,-0.5,    0.0, 1.0, 0.0,  // v2
               0.5, 0.0,-0.5,    0.0, 1.0, 0.0]  // v3
```

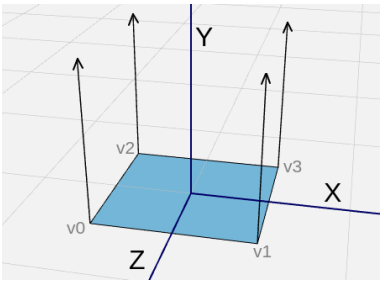


Figura 3: Plano con sus normales dibujadas

- En `iluminacion.js` puedes encontrar los siguientes cambios:
  - La función `initWebGL` se amplía para obtener las referencias al nuevo atributo y a todas las variables que intervienen en el cálculo de la iluminación.
  - Se proporciona la función `setMaterial` para asignar las propiedades de un material.
  - Se proporciona la función `initLight` para asignar las propiedades de la fuente de luz.
  - Y en la función `drawScene`, para cada primitiva, y previo a su dibujado, se establece tanto su matriz de transformación de la normal:
    - `setUniform ("normalMatrix", getNormalMatrix(modelViewMatrix));`

    como su material:
    - `setMaterial (Jade);`
- En `sombreadoPhong.html` está el nuevo *shader*:
  - En el *shader* de vértices aparecen declarados tanto el nuevo atributo, `VertexNormal`, como la matriz de transformación de la normal, `normalMatrix`. En la función `main` se obtiene la normal transformada, `N`, y tanto ésta como la posición del vértice en coordenadas del ojo, `ec`, se van a interpolar por fragmento.

- En el *shader* de fragmentos se calculan los otros dos vectores, L y V, y se llama a la función `phong` que recibe los tres vectores necesarios para realizar el cálculo de la reflexión difusa y la especular. Responde a las siguientes preguntas:
  - ¿Cómo se obtienen los vectores L y V?
  - ¿Cómo se obtiene el vector R?
  - ¿Qué devuelve la función `phong`?

Estudia detenidamente todos los cambios realizados y resuelve las dudas antes de continuar.

El objetivo de esta sesión es añadir iluminación en la escena del flexo (ver Figura 4), haciendo además que la fuente de luz se ubique en el propio foco y que esta se mueva de acuerdo al movimiento del flexo.

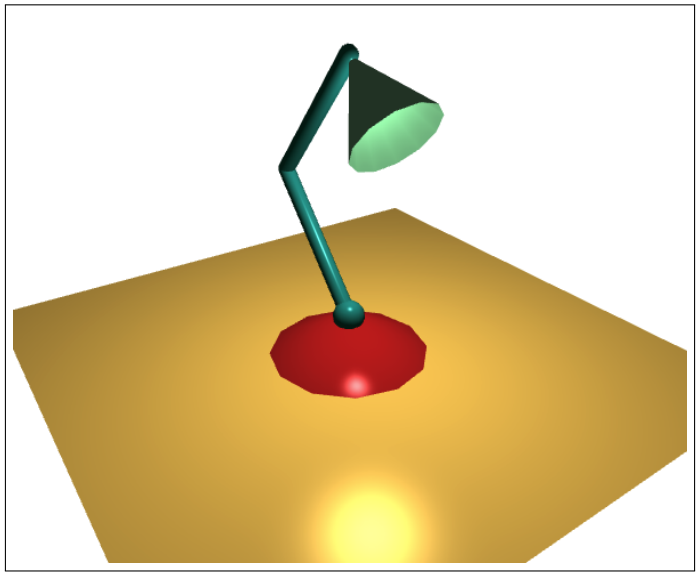


Figura 4: Escena con la fuente de luz ubicada en el foco del flexo

Comienza modificando `iluminacion.js` para que incluya el modelo del flexo de la práctica anterior. De momento no te preocupes por la posición de la fuente de luz pero recuerda que para cada primitiva sí que has de:

- especificar un material, utiliza los que hay en `materiales.js` o prueba a definir los tuyos propios,
- y obtener y proporcionar la matriz de transformación de la normal.

Ahora ya toca colocar la luz en el lugar donde debería ir una bombilla en tu flexo, es decir, dentro del cono que hace de foco. Y además hay que actualizar su posición con el movimiento del flexo. Para resolver el ejercicio piensa primero en cuál es la posición inicial de la fuente de luz. Probablemente sea la de un punto en el eje del cono. Si dicho eje va del punto $(0, 0, 0)$ al punto $(0, 0, 1)$, quizá una buena posición inicial sería el punto $(0, 0, 0.25)$, es decir, ligeramente dentro del cono. Si después simplemente le aplicas a dicho punto las mismas transformaciones que estás aplicándole al cono conseguirás que la bombilla se sitúe y acompañe al movimiento del foco. El Listado 2 recoge la codificación que necesitas para, a partir de la matriz de transformación modelo-vista del cono, `mvCono`, averiguar la posición final de la fuente de luz y actualizarla en el *shader*, utilízalo.

Listado 2: Cálculo de la posición del foco

```
var LpInicial = vec3.fromValues (0.0, 0.0, 0.25);
var LpFinal   = vec3.transformMat4 (vec3.create(), LpInicial, mvCono);
setUniform("Light.Lp", LpFinal);
```

Por último, modifica el *shader* actual para que ilumine los objetos por ambas caras. Esto es muy útil en el caso de que los objetos sean abiertos como por ejemplo ocurre en el foco del flexo. En el Listado 3 tienes el código necesario, que como puedes comprobar únicamente invierte la normal cuando el fragmento pertenece a una cara trasera (respecto a la posición del observador). Decide tú mismo dónde ponerlo. Recuerda que solo verás diferencias si alguna de las primitivas de la escena es abierta.

Listado 3: Fragmento de código necesario para sombrear un modelo por ambas caras

```
if (gl_FrontFacing == false) n = -n;
```

Extensión opcional: implementa un *shader* que te permita iluminar la escena como lo haría un foco de luz (*spotlight*, ver Figura 5). Respecto de una fuente de luz posicional, un foco de luz añade:

- La dirección del foco (`Direction`).
- El ángulo de corte (`Cutoff`).
- Un coeficiente que simula la atenuación de la luz (`Exponent`).

Para hacer el ejercicio más sencillo fija `Cutoff` a 45 grados y `Exponent` a 1.0. El Listado 4 muestra la nueva estructura `LightData`, que solo incorpora el vector de dirección, y la nueva función de *Phong* que ahora comprueba si un fragmento está dentro de la superficie iluminada. Actualiza tu *shader* para que incorpore los nuevos cambios. Después, realiza las modificaciones oportunas en `iluminacion.js` para calcular la dirección correcta del foco de luz y enviársela al *shader*.

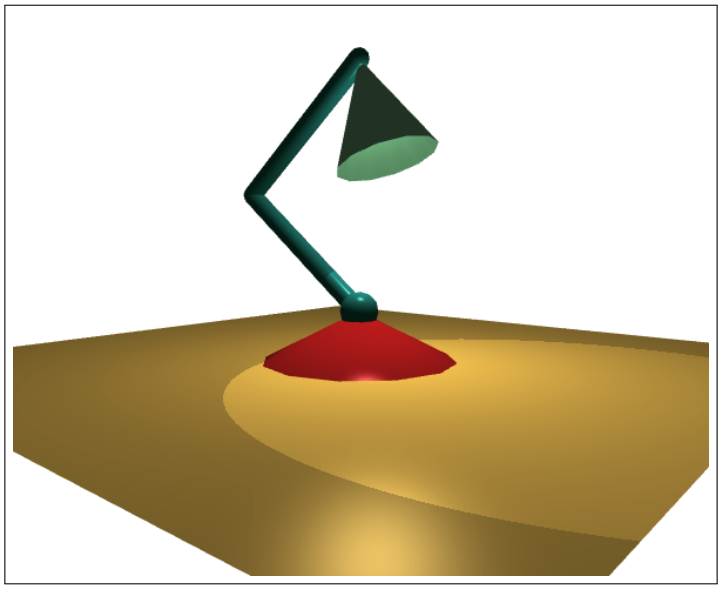


Figura 5: Escena con foco de luz y luz posicional en un mismo punto

Listado 4: Foco de luz o *spotlight*

```
struct LightData {
  vec3 La;          // Ambiente
  vec3 Ld;          // Difusa
  vec3 Ls;          // Especular
  vec3 Lp;          // Posición en coordenadas del ojo
  vec3 Ldir;        // Dirección de la luz en coordenadas del ojo
};

vec3 phong (vec3 N, vec3 L, vec3 V) {

  vec3  ambient  = Material.Ka * Light.La;
  vec3  diffuse  = vec3(0.0);
  vec3  specular = vec3(0.0);

  float NdotL    = dot (N,L);
  float spotFactor = 0.0;

  if (NdotL > 0.0) {
    vec3  S      = normalize (ec - Light.Lp);
    float angle  = acos(dot(S, Light.Ldir));

    if (angle < 0.79) {  // 45 grados en radianes
      spotFactor       = 1.0;

      vec3  R          = reflect(-L, N);;
      float RdotV_n    = pow(max(0.0, dot(R,V)), Material.shininess);

      diffuse  = NdotL   * (Light.Ld * Material.Kd);
      specular = RdotV_n * (Light.Ls * Material.Ks);
    }
  }
  return (ambient + spotFactor * (diffuse + specular));
}
```

Si lo ejecutas verás que solo muestra iluminación ambiente en la parte de la escena en la que no incide el foco. Esto hace que quede demasiado oscura. Con una modificación muy simple en el *shader* puedes conseguir que la misma fuente de luz se comporte tanto como una posicional, que emita luz en todas direcciones, como un foco. Piénsalo y realiza los cambios. El ejemplo que se muestra en la Figura 5 ya incluye el efecto que puedes conseguir.

Cuestiones

1. ¿En qué *shader* se declara el atributo de la normal, vértices o fragmentos? ¿Por qué?
2. Respecto a la matriz de transformación de la normal, ¿cuál de las transformaciones básicas que conoces nunca está incluida en dicha matriz: traslación, giro o escalado?

3. ¿Por qué la matriz de transformación de la normal se suele obtener a partir de la matriz *modelo-vista* y no únicamente a partir de la matriz de transformación del modelo?
4. ¿Qué crees que pasaría con una normal en el caso de que la matriz de transformación del modelo contuviese una operación de escalado en el que uno de sus factores de escala estuviese a cero?
5. ¿Qué representa la K_d para el objeto? ¿Y la K_s? ¿Crees que podrías conseguir que una luz blanca ilumine una esfera blanca y que el brillo observado se aprecie de color azúl?
6. Si añades el factor de atenuación al modelo de iluminación de Phong implementado en esta práctica, ¿a qué componentes del modelo debe afectar?, ¿cómo lo aplicarías en la función `phong`?