

6. TIPOS DE FUENTES DE LUZ

Se puede decidir tener una fuente de luz que sea tratada como si estuviera localizada en infinitamente lejos de la escena o una que sea cercana a la escena.

6.1.1.1. Direccional

El primer tipo referido, se conoce como fuente de luz *direccional*; el efecto de una localización en el infinito es que los rayos de luz pueden considerarse paralelos en el momento que alcanzan un objeto.

A la función **glLight*()** se le pasa un vector de cuatro valores (x,y,z,w) para el parámetro GL_POSITION. Si el último valor, w , es cero, la fuente de luz correspondiente es de tipo direccional y los valores (x,y,z) describen su dirección.

6.1.1.2. Posicional

El segundo tipo referido se conoce como fuente de luz *posicional*, ya que su posición exacta dentro de la escena determina el efecto que tiene sobre esta, específicamente, la dirección desde la cual vienen los rayos de luz.

A la función **glLight*()** se le pasa un vector de cuatro valores (x,y,z,w) para el parámetro GL_POSITION. Si el último valor, w , es distinto de cero, la fuente de luz correspondiente es de tipo posicional y los valores (x,y,z) especifican la localización de la fuente de luz.

Para luces en el mundo real, la intensidad de la luz decrece a medida que aumenta la distancia desde la fuente de luz. Ya que una luz direccional está infinitamente lejos, no tiene sentido atenuar su intensidad con la distancia, luego la atenuación está desactivada para luz direccional. OpenGL atenúa una fuente de luz multiplicando la contribución de esa fuente por un factor de atenuación:

$$\text{factor de atenuación} = \frac{1}{k_c + k_l d + k_q d^2}$$

donde

d : distancia entre la posición de la luz y el vértice

k_c : GL_CONSTANT_ATTENUATION

k_l : GL_LINEAR_ATTENUATION

k_q : GL_QUADRATIC_ATTENUATION

6.1.1.3. Spotlight

Una luz posicional irradia en todas las direcciones, pero se puede restringir esto de forma que se produzca un cono de luz definiendo lo que se conoce como una luz tipo *spotlight* (un ejemplo puede ser una lámpara tipo flexo).

Por lo tanto, para definir un spotlight hay que determinar la apertura deseada del cono de luz. (Recordar que, puesto que las spotlights son luces posicionales, hay que darles una localización). Para especificar el ángulo entre el eje del cono y un rayo a lo largo del borde del cono, usar el parámetro GL_SPOT_CUTOFF.

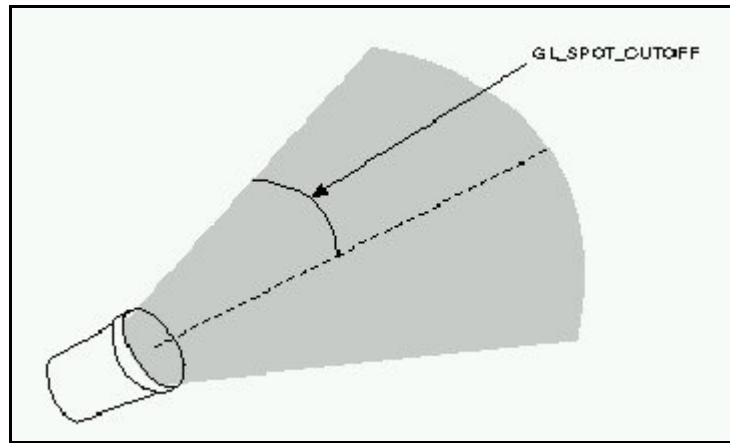


Figura 1. Significado del parámetro GL_SPOT_CUTOFF

Por defecto el valor del parámetro es 180° , es decir, irradia en todas las direcciones (360°). Los valores que puede tomar, a parte del particular 180, están comprendidos en el intervalo $[0,90]$.

También hay que especificar la dirección del eje del cono de luz, para lo cual se utiliza el parámetro GL_SPOT_DIRECTION.

Hay dos formas de controlar la intensidad de la distribución de la luz dentro del cono:

- Fijando el *factor de atenuación* descrito anteriormente, el cual es multiplicado por la intensidad de la luz.
- Fijando el parámetro GL_SPOT_EXPONENT que controla cómo es la luz de concentrada. La intensidad de la luz es más alta en el centro del cono. La luz se atenúa hacia los bordes del cono luz, luego a mayor exponente resulta en una fuente de luz más focalizada.

6.2. INTRODUCIENDO DIFERENTES LUCES AL PROGRAMA TECNUNLOGO

6.2.1. Tipos de luces

Las tres luces de que dispone el programa tecnunLogo, actualmente las tres del mismo tipo, se van a cambiar para que cada una sea de un tipo: una de tipo direccional, una de tipo posicional y una de tipo spotlight. Esto permitirá estudiar las diferentes características de los tipos de luces y cuales son sus efectos sobre la escena tanto al cambiar su posición o dirección como al cambiar las componentes propias de la luz. Además, permitiendo que puedan ser encendidas o apagadas de forma independiente se pueden estudiar los efectos de combinaciones entre ellas.

6.2.2. Interface de usuario

El interface de usuario actual permite pasar al Modo Luces y pasar el control de una luz a otra, encender y apagar cada una de las luces e indicar la dirección de las luces direccionales. Se va a añadir la posibilidad de controlar la dirección y ángulo de la luz de tipo spotlight (Tabla 6.1).

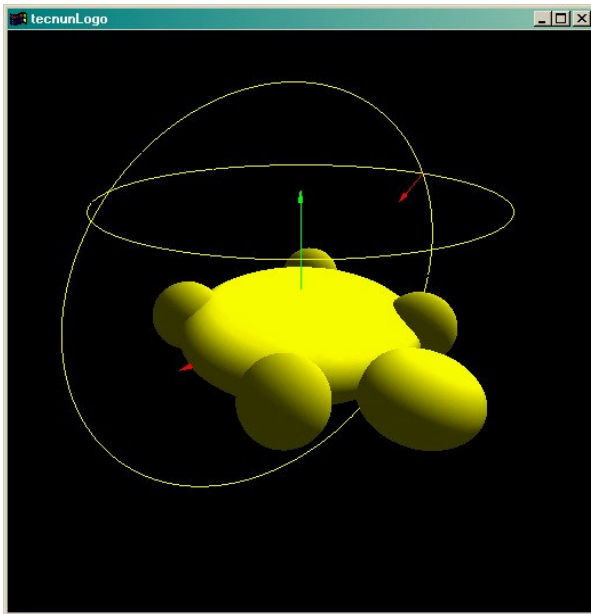


Figura 2. Luz Direccional

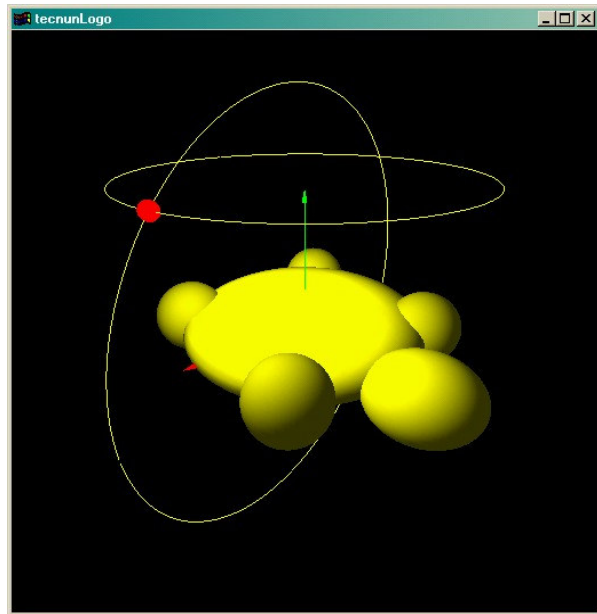


Figura 3. Luz Posicional

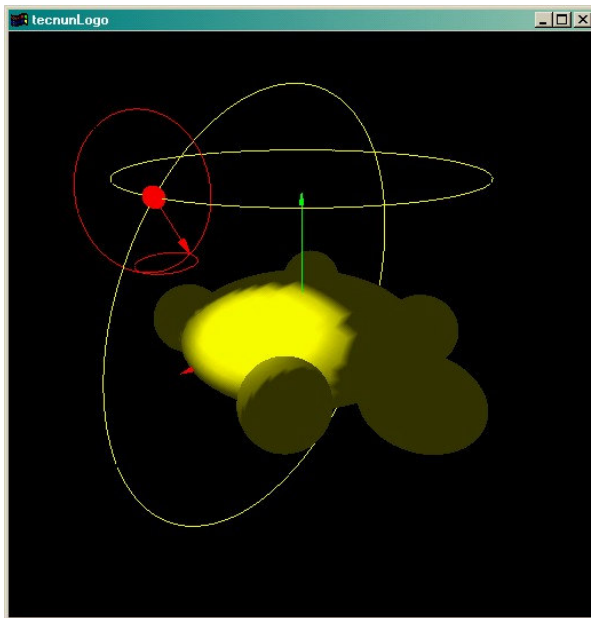


Figura 4. Luz Spotlight

En las figuras 2, 3 y 4 se muestra cómo aparecerán representados en el programa tecnunLogo cada uno de los tipos de luz.

La luz direccional se indica su dirección mediante un paralelo y un meridiano y viene representada con un vector dirigido siempre hacia el centro de la escena que indica su dirección. En el caso de luz posicional aparecerá una esfera y en el caso de luz tipo spotlight su representación es a través de una esfera que indica la posición y un vector con otro paralelo y meridiano que indica la dirección del cono de luz que emite.

Mediante el ratón será posible cambiar la posición de las luces (Tabla 6.2) y para el caso de la luz tipo spotlight cambiar también la dirección de la luz y el ángulo de apertura del cono de luz (Tabla 6.3).

6.2.3. Modificaciones en el código

Se va a dotar al programa de los tres tipos de luces expuestas en el apartado anterior que permitirán observar las diferencias entre ellas y su efecto en la escena. Para ello se permitirá interactuar con la posición y dirección de cada una de ellas y mantenerlas encendidas o apagadas independientemente una de otra. Para ello se utilizará el interface de luz definido en la práctica 5 y se realizarán modificaciones al código.

Tecla	Acción
F1	Desactivar luces (en general, desactivará cualquier modo)
F8	<p>Modo luces. Cada vez que se pulsa se pasa de una luz a otra:</p> <pre> graph LR A(Direccional) --> B(Posicional) B --> C(Spotlight) C --> A </pre>
F9	ON/OFF la luz con la cual se está interactuando.
F10	<p>Cuando se está interactuando con la luz tipo spotlight, pulsando F10 se pasa de controlar la posición de la luz a controlar la dirección de esta y viceversa:</p> <pre> graph LR A(Posición) --> B(Dirección) B --> A </pre>

Tabla 6.1. Teclas del Modo Luces

Movimiento del Ratón	Acción
Adelante/Atrás	Movimiento de la luz con la cual se está interactuando a lo largo de un <i>meridiano</i> con centro en el origen de coordenadas de la escena.
Izquierda/Derecha	Movimiento de la luz con la cual se está interactuando a lo largo de un <i>paralelo</i> con centro en el eje Y.
Adelante/Atrás con botón izquierdo pulsado	Movimiento de la luz a lo largo del vector que une la posición de la luz con el centro de coordenadas de la escena.

Tabla 6.2. Movimiento de la posición de las luces

Movimiento del Ratón	Acción
Adelante/Atrás	Movimiento del extremo del vector de dirección del spotlight a lo largo de un <i>meridiano</i> con centro en la posición de la luz.
Izquierda/Derecha	Movimiento del extremo del vector de dirección del spotlight a lo largo de un <i>paralelo</i> con centro en la vertical que pasa por la posición de la luz.
Adelante/Atrás con botón izquierdo pulsado	Incremento/Decremento del ángulo de apertura del spotlight..

Tabla 6.3. Control de la dirección y apertura del cono de luz del spotlight

Para la luz del tipo spotlight hará falta una variable más que indique si se está interactuando con la posición de la luz o con el vector de dirección.. Se incluyen las siguientes líneas en el fichero `tecnunLogo.c` inmediatamente después de los ficheros de cabecera:

```
static int spot_move = 0;
```

Se deben dar las características a cada una de las luces. Esto lo realizamos modificando las siguientes sentencias en la función **main()** del programa:

```
//Creamos las luces y damos a cada una sus características
//DIRECCIONAL
LOCAL_MyLights[0] = CreateDefaultLight();
LOCAL_MyLights[0]->type = AGA_DIRECTIONAL;
LOCAL_MyLights[0]->id = GL_LIGHT0;
LOCAL_MyLights[0]->position[0] = 1.0f;
LOCAL_MyLights[0]->position[1] = 1.0f;
LOCAL_MyLights[0]->position[2] = 1.0f;
LOCAL_MyLights[0]->position[3] = 0.0f;
LOCAL_MyLights[0]->pointAtInfinity[0] = LOCAL_MyLights[0]->position[0];
LOCAL_MyLights[0]->pointAtInfinity[1] = LOCAL_MyLights[0]->position[1];
LOCAL_MyLights[0]->pointAtInfinity[2] = LOCAL_MyLights[0]->position[2];
//POSICIONAL
LOCAL_MyLights[1] = CreateDefaultLight();
LOCAL_MyLights[1]->type = AGA_POSITIONAL;
LOCAL_MyLights[1]->id = GL_LIGHT1;
LOCAL_MyLights[1]->position[0] = 1.0f;
LOCAL_MyLights[1]->position[1] = 1.0f;
LOCAL_MyLights[1]->position[2] = -1.0f;
LOCAL_MyLights[1]->position[3] = 1.0f;
//SPOT
LOCAL_MyLights[2] = CreateDefaultLight();
LOCAL_MyLights[2]->type = AGA_SPOT;
LOCAL_MyLights[2]->id = GL_LIGHT2;
LOCAL_MyLights[2]->position[0] = -1.0f;
LOCAL_MyLights[2]->position[1] = 1.0f;
LOCAL_MyLights[2]->position[2] = 1.0f;
LOCAL_MyLights[2]->spotDirection[0] = 1.0f;
LOCAL_MyLights[2]->spotDirection[1] = -1.0f;
LOCAL_MyLights[2]->spotDirection[2] = -1.0f;
```

De esta forma quedan definidos los interfaces para las tres luces en un array. Con el índice 0 se tiene una luz direccional, con el índice 1 se tiene una luz posicional y con el índice 2 se tiene una luz tipo spotlight.

En la función **SpecialKey()** se deberán incluir las sentencias necesarias para definir las teclas que permiten pasar al modo de movimiento de la dirección del spotlight:

```
case GLUT_KEY_F10:
    if (current_light == 2){
        if ( spot_move == 0 ){
            glutPassiveMotionFunc(Mouse_Spot);
            spot_move = 1;
        }else{
            glutPassiveMotionFunc(Mouse_Luces);
            spot_move = 0;
        }
    }
    break;
```

La función **Mouse_Spot(int x, int y)** moverá el vector de dirección de la luz tipo spotlight, cuando se esté interactuando con él, a lo largo de un paralelo o un meridiano según los movimientos del ratón definidos con anterioridad:

```
void Mouse_Spot(int x, int y){
    float rot_x, rot_y;

    rot_y = (float)(old_y - y);
    rot_x = (float)(x - old_x);
    Rotar_Spot_Latitud(LOCAL_MyLights[current_light], rot_y*DEGREE_TO_RAD);
    Rotar_Spot_Longitud(LOCAL_MyLights[current_light], rot_x*DEGREE_TO_RAD);

    old_y = y;
    old_x = x;
    glutPostRedisplay();
}
```

La función **mouse()** quedará de la siguiente forma después de incluir las sentencias necesarias para interactuar con las luces:

```
void mouse(int button, int state, int x, int y){

    old_x = x;
    old_y = y;

    switch(button){
    case GLUT_LEFT_BUTTON:
        if(current_light > 0){
            if(current_light == 2 && spot_move == 1){
                if (state == GLUT_DOWN)
                    glutMotionFunc(Mouse_Spot_Abrir_Cerrar);
                if (state == GLUT_UP){
                    glutPassiveMotionFunc(Mouse_Spot);
                    glutMotionFunc(NULL);
                }
            }else{
                if (state == GLUT_DOWN)
                    glutMotionFunc(Mouse_Luces_Acercar_Alejar);
                if (state == GLUT_UP){
                    glutPassiveMotionFunc(Mouse_Luces);
                    glutMotionFunc(NULL);
                }
            }
        }
        }else{
        switch(LOCAL_MyCamera->camMovimiento){
        case CAM_EXAMINAR:
            if (state == GLUT_DOWN) glutMotionFunc(Zoom);
            if (state == GLUT_UP){
                glutPassiveMotionFunc(Examinar);
                glutMotionFunc(NULL);
            }
            break;
        case CAM_PASEAR:
            if (state == GLUT_DOWN) glutMotionFunc(Andar);
            if (state == GLUT_UP) glutMotionFunc(NULL);
            break;
        }
    }
}
```

```

        break;
    case GLUT_RIGHT_BUTTON:
        if (state == GLUT_DOWN) ;
        break;
    default:
        break;
    }
    glutPostRedisplay();
}

```

Se ve que se realizan llamadas a la función callback **glutMotionFunc()** que responde a los movimientos del ratón cuando se tiene pulsado algún botón de este. Si se está interactuando con la dirección de la luz tipo spotlight, se le pasará por ventana la función que se encarga de abrir o cerrar el ángulo del cono de luz, ángulo que variará entre 0 y 90 grados:

```

void Mouse_Spot_Abrir_Cerrar(int x, int y){
    float step;

    step = (float) (y - old_y) ;
    old_y = y;
    if(LOCAL_MyLights[current_light]->spotCutOff + step < 90 &&
        LOCAL_MyLights[current_light]->spotCutOff + step > 0)
        LOCAL_MyLights[current_light]->spotCutOff += step ;

    LOCAL_MyLights[current_light]->needsUpdate = TRUE;
    glutPostRedisplay();
}

```

La función **display()** quedará de la siguiente forma:

```

void display(void) {
    float At[3];
    float Direction[3];

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);

    SetGLCamera( LOCAL_MyCamera );

    SetLight( LOCAL_MyLights[0] );
    SetLight( LOCAL_MyLights[1] );
    SetLight( LOCAL_MyLights[2] );

    glPushMatrix();
    glColor3f(1.0,1.0,0.0);
    drawSphereTurtle();
    switch( current_light ){
    case 0:
        At[0] = LOCAL_MyLights[current_light]->position[0];
        At[1] = LOCAL_MyLights[current_light]->position[1];
        At[2] = LOCAL_MyLights[current_light]->position[2];
        Direction[0] = - LOCAL_MyLights[current_light]->position[0];
        Direction[1] = - LOCAL_MyLights[current_light]->position[1];
        Direction[2] = - LOCAL_MyLights[current_light]->position[2];
        Draw_Parallel(At);
        Draw_Meridian(At);
        Draw_Vector(At, Direction);
    }
    glPopMatrix();
}

```

```

        break;
    case 1:
        At[0] = LOCAL_MyLights[current_light]->position[0];
        At[1] = LOCAL_MyLights[current_light]->position[1];
        At[2] = LOCAL_MyLights[current_light]->position[2];
        Draw_Parallel(At);
        Draw_Meridian(At);
        glTranslatef(At[0],At[1],At[2]);
        glColor3f(1.0,0.0,0.0);
        glutSolidSphere(0.05,28,28);
        break;
    case 2:
        At[0] = LOCAL_MyLights[current_light]->position[0];
        At[1] = LOCAL_MyLights[current_light]->position[1];
        At[2] = LOCAL_MyLights[current_light]->position[2];
        Direction[0] = LOCAL_MyLights[current_light]->spotDirection[0];
        Direction[1] = LOCAL_MyLights[current_light]->spotDirection[1];
        Direction[2] = LOCAL_MyLights[current_light]->spotDirection[2];
        Draw_Parallel(At);
        Draw_Meridian(At);
        glColor3f(1.0,0.0,0.0);
        Draw_Vector(At, Direction);
        Draw_Sphere_Spot(At, Direction);
        glTranslatef(At[0],At[1],At[2]);
        glutSolidSphere(0.05,28,28);
        break;
    default:
        break;
}
glPopMatrix();

glutSwapBuffers();
}

```

6.3. TRABAJOS PROPUESTOS

- *En la luz tipo spotlight, que se muestre el cono de luz.*
- *Definir las propiedades del material de la tortuga.*
- *Proporcionar al programa alguna forma amigable de interactuar con las propiedades del material (interface de usuario).*