Feria de pueblo: LUFER

Manual de técnico

Universidad Nacional
Autónoma de México

Facultad de Ingeniería

Ingeniería en
Computación

Feria de pueblo: LUFER
Manual de técnico

Alumno (a)
Garrido López Luis Enrique
313320727 luispokered@gmail.com
Ortiz Figueroa María Fernanda
313116186 sasferortiz@gmail.com

Asignatura
Laboratorio de Computación Gráfica e
Interacción Humano Computadora

Grupo laboratorio	Grupo teoría
5	3

Profesor
Ing. José Roque Román Guadarrama

Semestre
2019 - 2

Ciudad Universitaria, 23 de Mayo de 2019

Feria de pueblo: LUFER

- **Escenario y objetos en 3D**

- **Atracciones mecánicas principales realizadas por modelado jerárquico, implementación de materiales e interacción con luces cercanas, así como animación.**

** Animación

```
// Movimiento Rueda de la Fortuna
     if (mainWindow.getAnimacionRueda() == 1) {
     // Movimiento general de la rueda de la fortuna
     if (moverRuedaFortuna < 360.0f) {
     moverRuedaFortuna += moverOffsetRuedaFortuna * deltaTime;
     moverRuedaFortunaAsiento -= moverOffsetRuedaFortuna * deltaTime;
     }
     else {
     moverRuedaFortuna = 0.0f;
     moverRuedaFortunaAsiento = 0.0f;
     }
     }

     // Movimiento Carrusel
     if (mainWindow.getAnimacionCarrusel() == 1) {
```

```
// Movimiento general del carrusel
if (moverCarrusel < 360.0f) {
moverCarrusel += moverOffsetCarrusel * deltaTime;
}
else {
moverCarrusel = 0.0f;
}

// Movimiento caballo y avestruz arriba 1
if (avanzaArribaCarrusel1 == 1) {
if (moverArribaCarrusel1 < 0.8f) {
moverArribaCarrusel1 += moverOffsetArribaCarrusel1 * deltaTime;
}
else {
avanzaArribaCarrusel1 = 0;
}
}
else if (avanzaArribaCarrusel1 == 0) {
if (moverArribaCarrusel1 > 0.0f) {
moverArribaCarrusel1 -= moverOffsetArribaCarrusel1 * deltaTime;
}
else {
avanzaArribaCarrusel1 = 1;
}
}

// Movimiento caballo y avestruz arriba 2
if (avanzaArribaCarrusel2 == 1) {
if (moverArribaCarrusel2 < 0.8f) {
moverArribaCarrusel2 += moverOffsetArribaCarrusel2 * deltaTime;
}
else {
avanzaArribaCarrusel2 = 0;
}
}
else if (avanzaArribaCarrusel2 == 0) {
if (moverArribaCarrusel2 > 0.0f) {
moverArribaCarrusel2 -= moverOffsetArribaCarrusel2 * deltaTime;
}
else {
avanzaArribaCarrusel2 = 1;
}
}
}
```

** Caso juego mecánico “Carrusel”

```
// Carrusel*************************************************************************************
 // Parte superior+++++++++++++++++++++++++++++++++++++++++++++++++++++
 modelC = glm::mat4(1.0);
 modelC = glm::translate(modelC, glm::vec3(-10.0f, -2.0f, 0.0f));
 modelC2 = modelC;
 modelC = glm::scale(modelC, glm::vec3(2.0f, 2.0f, 2.0f));
 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelC));
 materialBrillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
 carruselInferiorM.RenderModel();

 // Parte inferior+++++++++++++++++++++++++++++++++++++++++++++++++++++
 modelC = modelC2;
 modelC = glm::translate(modelC, glm::vec3(0.0f, -0.15f, 0.0f));
 modelC = glm::rotate(modelC, moverCarrusel * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
 modelC2 = modelC;
 modelC = glm::scale(modelC, glm::vec3(2.0f, 2.0f, 2.0f));
 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelC));
 materialBrillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
 carruselSuperiorM.RenderModel();
```

```
// Espejos parte superior++++++++++++++++++++++++++++++++++++++++++++++++
modelC = modelC2;
modelC2 = modelC;
modelC = glm::scale(modelC, glm::vec3(2.0f, 2.0f, 2.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelC));
materialBrillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
espejosCarruselM.RenderModel();
```

… **así hasta generar todo el juego mecánico**

```
// Luz carrusel
// Este conjunto de 3 luces rotatorias tambien pueden desplazarse
// Se tiene un contador para el cambio de luces
contLuzCarrusel++;

// Ecuación paramétrica de la circunferencia
// para hacer que roten las luces
zPosLuzCarrusel = 0.0f + (radioLuzCarrusel * cos((moverCarrusel + 0.0f) * toRadians));
xPosLuzCarrusel = -10.0f + (radioLuzCarrusel * sin((moverCarrusel + 0.0f) * toRadians));
spotLights[0].SetPos(glm::vec3(xPosLuzCarrusel + mainWindow.getMovLuzSpotX(),
8.0f + mainWindow.getMovLuzSpotY(),
zPosLuzCarrusel + mainWindow.getMovLuzSpotZ()));

zPosLuzCarrusel = 0.0f + (radioLuzCarrusel * cos((moverCarrusel + 120.0f) * toRadians));
xPosLuzCarrusel = -10.0f + (radioLuzCarrusel * sin((moverCarrusel + 120.0f) * toRadians));
spotLights[1].SetPos(glm::vec3(xPosLuzCarrusel + mainWindow.getMovLuzSpotX(),
8.0f + mainWindow.getMovLuzSpotY(),
zPosLuzCarrusel + mainWindow.getMovLuzSpotZ()));

zPosLuzCarrusel = 0.0f + (radioLuzCarrusel * cos((moverCarrusel + 240.0f) * toRadians));
xPosLuzCarrusel = -10.0f + (radioLuzCarrusel * sin((moverCarrusel + 240.0f) * toRadians));
spotLights[2].SetPos(glm::vec3(xPosLuzCarrusel + mainWindow.getMovLuzSpotX(),
8.0f + mainWindow.getMovLuzSpotY(),
zPosLuzCarrusel + mainWindow.getMovLuzSpotZ()));

// Luces carrusel encendidas
if (mainWindow.getLuzCarrusel() == 1) {
if (tipoColorLuzCarrusel == 0.0f) {
spotLights[0].SetColor(glm::vec3(0.8f, 0.8f, 0.8f)); //Blanco
spotLights[1].SetColor(glm::vec3(0.5f, 0.0f, 0.0f)); //Rojo
spotLights[2].SetColor(glm::vec3(0.0f, 1.0f, 1.0f)); //Cyan
}
else if (tipoColorLuzCarrusel == 1.0f) {
spotLights[0].SetColor(glm::vec3(1.0f, 0.0f, 1.0f)); //Magenta
spotLights[1].SetColor(glm::vec3(0.0f, 0.5f, 0.0f)); //Verde
spotLights[2].SetColor(glm::vec3(0.8f, 0.8f, 0.0f)); //Amarillo
}
else if (tipoColorLuzCarrusel == 2.0f) {
spotLights[0].SetColor(glm::vec3(0.0f, 0.0f, 1.0f)); //Azul
spotLights[1].SetColor(glm::vec3(0.8f, 0.8f, 0.8f)); //Blanco
spotLights[2].SetColor(glm::vec3(0.5f, 0.0f, 0.0f)); //Rojo
}
else if (tipoColorLuzCarrusel == 4.0f) {
spotLights[0].SetColor(glm::vec3(0.0f, 1.0f, 1.0f)); //Cyan
spotLights[1].SetColor(glm::vec3(1.0f, 0.0f, 1.0f)); //Magenta
spotLights[2].SetColor(glm::vec3(0.0f, 0.5f, 0.0f)); //Verde
}
else if (tipoColorLuzCarrusel == 5.0f) {
spotLights[0].SetColor(glm::vec3(0.8f, 0.8f, 0.0f)); //Amarillo
spotLights[1].SetColor(glm::vec3(0.0f, 0.0f, 1.0f)); //Azul
spotLights[2].SetColor(glm::vec3(0.8f, 0.8f, 0.8f)); //Blanco
}
else if (tipoColorLuzCarrusel == 6.0f) {
spotLights[0].SetColor(glm::vec3(0.5f, 0.0f, 0.0f)); //Rojo
spotLights[1].SetColor(glm::vec3(0.0f, 1.0f, 1.0f)); //Cyan
spotLights[2].SetColor(glm::vec3(1.0f, 0.0f, 1.0f)); //Magenta
}
else if (tipoColorLuzCarrusel == 7.0f) {
```

```
spotLights[0].SetColor(glm::vec3(0.0f, 0.5f, 0.0f)); //Verde
spotLights[1].SetColor(glm::vec3(0.8f, 0.8f, 0.0f)); //Amarillo
spotLights[2].SetColor(glm::vec3(0.0f, 0.0f, 1.0f)); //Azul
}

if (contLuzCarrusel == 100.0) {
tipoColorLuzCarrusel = 1.0f;
}
else if (contLuzCarrusel == 150.0) {
tipoColorLuzCarrusel = 2.0f;
}
else if (contLuzCarrusel == 200.0) {
tipoColorLuzCarrusel = 3.0f;
}
else if (contLuzCarrusel == 250.0) {
tipoColorLuzCarrusel = 4.0f;
}
else if (contLuzCarrusel == 300.0) {
tipoColorLuzCarrusel = 5.0f;
}
else if (contLuzCarrusel == 350.0) {
tipoColorLuzCarrusel = 6.0f;
}
else if (contLuzCarrusel == 400.0) {
tipoColorLuzCarrusel = 7.0f;
}
else if (contLuzCarrusel == 450.0) {
tipoColorLuzCarrusel = 0.0f;
contLuzCarrusel = 0.0f;
}
}
// Luces carrusel apagadas
else {
spotLights[0].SetColor(glm::vec3(0.0f, 0.0f, 0.0f));
spotLights[1].SetColor(glm::vec3(0.0f, 0.0f, 0.0f));
spotLights[2].SetColor(glm::vec3(0.0f, 0.0f, 0.0f));
}
```

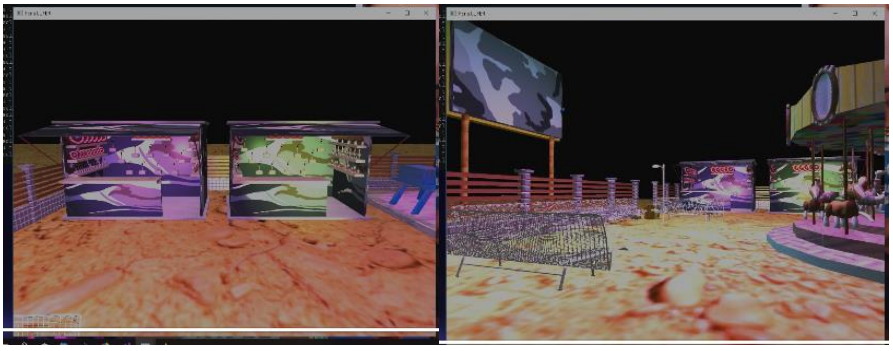
Ambientación de la feria

Cámaras

```
// Teclas para controlar el cambio de camaras
    // Camara general
    if (key == GLFW_KEY_C && action == GLFW_PRESS) {
    theWindow->cambioCamara = 0.0f;
    theWindow->camara = 1.0f;
    }

    // Camara carrusel
    if (key == GLFW_KEY_V && action == GLFW_PRESS) {
    theWindow->cambioCamara = 0.0f;
    theWindow->camara = 2.0f;
    }

    // Camara rueda
    if (key == GLFW_KEY_B && action == GLFW_PRESS) {
    theWindow->cambioCamara = 0.0f;
    theWindow->camara = 3.0f;
    }

    // Camara tiendas
    if (key == GLFW_KEY_N && action == GLFW_PRESS) {
    theWindow->cambioCamara = 0.0f;
    theWindow->camara = 4.0f;
    }

    // Camara aerea
    if (key == GLFW_KEY_M && action == GLFW_PRESS) {
    theWindow->cambioCamara = 0.0f;
    theWindow->camara = 5.0f;
    }


// La primera ves que se ejecuta la vista de la camara se coloca
    // de frente a la feria, es decir, en la entrada
    if (mainWindow.getCamaraGeneral() == 0.0f) {
    camera = Camera(camaraPosicion, glm::vec3(0.0f, 1.0f, 0.0f), camaraYaw, camaraPitch, 5.0f, 0.5f);
    mainWindow.cambioCamara = 1.0f;
    mainWindow.setCamara(1.0f);
    }
    // Solo la camara en primera persona se mueve
    else if (mainWindow.getCamaraGeneral() == 1.0f) {
    // Al regresar a la camara el primera persona, se regresa a la posición en la que estaba
    if (mainWindow.cambioCamara == 0.0f) {
    camera = Camera(camaraPosicion, glm::vec3(0.0f, 1.0f, 0.0f), camaraYaw, camaraPitch, 5.0f, 0.5f);
    mainWindow.cambioCamara = 1.0f;
    camaraSalvar = 1;
    }

    // Movimeinto general de la camara
    camera.keyControl(mainWindow.getsKeys(), deltaTime * 5);
    camera.mouseControl(mainWindow.getXChange(), mainWindow.getYChange());
    }
    // Se posiciona la camara en vista Carrusel
    else if (mainWindow.getCamaraGeneral() == 2.0f && mainWindow.cambioCamara == 0.0f) {
    if (mainWindow.cambioCamara == 0.0f && camaraSalvar == 1) {
    camaraPosicion = camera.getCameraPosition();
    camaraYaw = camera.getCameraYaw();
    camaraPitch = camera.getCameraPitch();
    camaraSalvar = 0;
    }

    camera = Camera(glm::vec3(40.0f, 10.0f, 0.0f), glm::vec3(0.0f, 1.0f, 0.0f), 180.0f, -10.0f, 5.0f,
0.5f);
    mainWindow.cambioCamara = 1.0f;
```

```
    }
    // Se posiciona la camara en vista Rueda de la Fortuna
    else if (mainWindow.getCamaraGeneral() == 3.0f && mainWindow.cambioCamara == 0.0f) {
    if (mainWindow.cambioCamara == 0.0f && camaraSalvar == 1) {
    camaraPosicion = camera.getCameraPosition();
    camaraYaw = camera.getCameraYaw();
    camaraPitch = camera.getCameraPitch();
    camaraSalvar = 0;
    }

    camera = Camera(glm::vec3(-30.0f, 40.0f, 4.0f), glm::vec3(0.0f, 1.0f, 0.0f), -90.0f, -10.0f, 5.0f,
0.5f);
    mainWindow.cambioCamara = 1.0f;
    }
    // Se posiciona la camara en vista Tiendas
    else if (mainWindow.getCamaraGeneral() == 4.0f && mainWindow.cambioCamara == 0.0f) {
    if (mainWindow.cambioCamara == 0.0f && camaraSalvar == 1) {
    camaraPosicion = camera.getCameraPosition();
    camaraYaw = camera.getCameraYaw();
    camaraPitch = camera.getCameraPitch();
    camaraSalvar = 0;
    }
    camera = Camera(glm::vec3(15.0f, 15.0f, 10.0f), glm::vec3(0.0f, 1.0f, 0.0f), 90.0f, -10.0f, 5.0f,
0.5f);
    mainWindow.cambioCamara = 1.0f;
    }
    // Se posiciona la camara en vista aerea
    else if (mainWindow.getCamaraGeneral() == 5.0f && mainWindow.cambioCamara == 0.0f) {
    if (mainWindow.cambioCamara == 0.0f && camaraSalvar == 1) {
    camaraPosicion = camera.getCameraPosition();
    camaraYaw = camera.getCameraYaw();
    camaraPitch = camera.getCameraPitch();
    camaraSalvar = 0;
    }
    camera = Camera(glm::vec3(-20.0f, 180.0f, 0.0f), glm::vec3(0.0f, 1.0f, 0.0f), 0.0f, -90.0f, 5.0f,
0.5f);
    mainWindow.cambioCamara = 1.0f;
    }
```

Objetos con proporciones adecuadas



Prender y apagar por los menos una luz tipo pointligth y una luz tipo spotligth

```
// Movimiento de la luz pointlight
    if (key == GLFW_KEY_R && action == GLFW_PRESS) {
    theWindow->movLuzPointX += 2.0;
    printf("x: %f\n", theWindow->movLuzPointX);
    }

    if (key == GLFW_KEY_F && action == GLFW_PRESS) {
    theWindow->movLuzPointX -= 2.0;
    printf("X: %f\n", theWindow->movLuzPointX);
    }

    if (key == GLFW_KEY_T && action == GLFW_PRESS) {
    theWindow->movLuzPointY += 2.0;
    printf("y: %f\n", theWindow->movLuzPointY);
    }

    if (key == GLFW_KEY_G && action == GLFW_PRESS) {
    theWindow->movLuzPointY -= 2.0;
    printf("y: %f\n", theWindow->movLuzPointY);
    }

    if (key == GLFW_KEY_Y && action == GLFW_PRESS) {
    theWindow->movLuzPointZ += 2.0;
    printf("z: %f\n", theWindow->movLuzPointZ);
    }

    if (key == GLFW_KEY_H && action == GLFW_PRESS) {
    theWindow->movLuzPointZ -= 2.0;
    printf("z: %f\n", theWindow->movLuzPointZ);
    }

    // Movimiento de la luz spotligth
    if (key == GLFW_KEY_U && action == GLFW_PRESS) {
    theWindow->movLuzSpotX += 2.0;
    printf("x: %f\n", theWindow->movLuzSpotX);
    }

    if (key == GLFW_KEY_J && action == GLFW_PRESS) {
    theWindow->movLuzSpotX -= 2.0;
    printf("X: %f\n", theWindow->movLuzSpotZ);
    }

    if (key == GLFW_KEY_I && action == GLFW_PRESS) {
    theWindow->movLuzSpotY += 2.0;
    printf("y: %f\n", theWindow->movLuzSpotY);
    }

    if (key == GLFW_KEY_K && action == GLFW_PRESS) {
    theWindow->movLuzSpotY -= 2.0;
    printf("y: %f\n", theWindow->movLuzSpotY);
    }

    if (key == GLFW_KEY_O && action == GLFW_PRESS) {
    theWindow->movLuzSpotZ += 2.0;
    printf("z: %f\n", theWindow->movLuzSpotZ);
    }

    if (key == GLFW_KEY_L && action == GLFW_PRESS) {
    theWindow->movLuzSpotZ -= 2.0;
    printf("z: %f\n", theWindow->movLuzSpotZ);
    }

    // Luz carrusel tipo spotligth prender o apagar
```

```
 if (key == GLFW_KEY_Q && action == GLFW_PRESS) {
 if (theWindow->luzCarrusel == 1) {
 theWindow->luzCarrusel = 0;
 }
 else {
 theWindow->luzCarrusel = 1;
 }
 }

 // Luz rueda tipo spotligth prender o apagar
 if (key == GLFW_KEY_E && action == GLFW_PRESS) {
 if (theWindow->luzRueda == 1) {
 theWindow->luzRueda = 0;
 }
 else {
 theWindow->luzRueda = 1;
 }
 }

 // Luz letrero tipo pointligth prender apagar
 if (key == GLFW_KEY_P && action == GLFW_PRESS) {
 if (theWindow->luzLetrero == 1) {
 theWindow->luzLetrero = 0;
 }
 else {
 theWindow->luzLetrero = 1;
 }
 }

// Luces carrusel encendidas
 if (mainWindow.getLuzCarrusel() == 1) {
 if (tipoColorLuzCarrusel == 0.0f) {
 spotLights[0].SetColor(glm::vec3(0.8f, 0.8f, 0.8f)); //Blanco
 spotLights[1].SetColor(glm::vec3(0.5f, 0.0f, 0.0f)); //Rojo
 spotLights[2].SetColor(glm::vec3(0.0f, 1.0f, 1.0f)); //Cyan
 }
 else if (tipoColorLuzCarrusel == 1.0f) {
 spotLights[0].SetColor(glm::vec3(1.0f, 0.0f, 1.0f)); //Magenta
 spotLights[1].SetColor(glm::vec3(0.0f, 0.5f, 0.0f)); //Verde
 spotLights[2].SetColor(glm::vec3(0.8f, 0.8f, 0.0f)); //Amarillo
 }
 else if (tipoColorLuzCarrusel == 2.0f) {
 spotLights[0].SetColor(glm::vec3(0.0f, 0.0f, 1.0f)); //Azul
 spotLights[1].SetColor(glm::vec3(0.8f, 0.8f, 0.8f)); //Blanco
 spotLights[2].SetColor(glm::vec3(0.5f, 0.0f, 0.0f)); //Rojo
 }
 else if (tipoColorLuzCarrusel == 4.0f) {
 spotLights[0].SetColor(glm::vec3(0.0f, 1.0f, 1.0f)); //Cyan
 spotLights[1].SetColor(glm::vec3(1.0f, 0.0f, 1.0f)); //Magenta
 spotLights[2].SetColor(glm::vec3(0.0f, 0.5f, 0.0f)); //Verde
 }
 else if (tipoColorLuzCarrusel == 5.0f) {
 spotLights[0].SetColor(glm::vec3(0.8f, 0.8f, 0.0f)); //Amarillo
 spotLights[1].SetColor(glm::vec3(0.0f, 0.0f, 1.0f)); //Azul
 spotLights[2].SetColor(glm::vec3(0.8f, 0.8f, 0.8f)); //Blanco
 }
 else if (tipoColorLuzCarrusel == 6.0f) {
 spotLights[0].SetColor(glm::vec3(0.5f, 0.0f, 0.0f)); //Rojo
 spotLights[1].SetColor(glm::vec3(0.0f, 1.0f, 1.0f)); //Cyan
 spotLights[2].SetColor(glm::vec3(1.0f, 0.0f, 1.0f)); //Magenta
 }
 else if (tipoColorLuzCarrusel == 7.0f) {
 spotLights[0].SetColor(glm::vec3(0.0f, 0.5f, 0.0f)); //Verde
 spotLights[1].SetColor(glm::vec3(0.8f, 0.8f, 0.0f)); //Amarillo
 spotLights[2].SetColor(glm::vec3(0.0f, 0.0f, 1.0f)); //Azul
 }

 if (contLuzCarrusel == 100.0) {
```

```
tipoColorLuzCarrusel = 1.0f;
}
else if (contLuzCarrusel == 150.0) {
tipoColorLuzCarrusel = 2.0f;
}
else if (contLuzCarrusel == 200.0) {
tipoColorLuzCarrusel = 3.0f;
}
else if (contLuzCarrusel == 250.0) {
tipoColorLuzCarrusel = 4.0f;
}
else if (contLuzCarrusel == 300.0) {
tipoColorLuzCarrusel = 5.0f;
}
else if (contLuzCarrusel == 350.0) {
tipoColorLuzCarrusel = 6.0f;
}
else if (contLuzCarrusel == 400.0) {
tipoColorLuzCarrusel = 7.0f;
}
else if (contLuzCarrusel == 450.0) {
tipoColorLuzCarrusel = 0.0f;
contLuzCarrusel = 0.0f;
}
}
// Luces carrusel apagadas
else {
spotLights[0].SetColor(glm::vec3(0.0f, 0.0f, 0.0f));
spotLights[1].SetColor(glm::vec3(0.0f, 0.0f, 0.0f));
spotLights[2].SetColor(glm::vec3(0.0f, 0.0f, 0.0f));
}

// Luz letrero
if (mainWindow.getLuzLetrero() == 1) {
if (contLuzLetrero >= 0.0f && contLuzLetrero < 100.0f) {
pointLights[2].SetColor(glm::vec3(0.0f, 0.9f, 0.9f));
/*pointLights[2].SetPos(glm::vec3(102.8f + mainWindow.getMovLuzPointX(),
23.8f + mainWindow.getMovLuzPointY(),
0.0f + mainWindow.getMovLuzPointZ()));*/
}
else if (contLuzLetrero >= 100.0f && contLuzLetrero < 200.0f) {
pointLights[2].SetColor(glm::vec3(1.0f, 0.0f, 1.0f));
/*pointLights[2].SetPos(glm::vec3(102.8f + mainWindow.getMovLuzPointX(),
23.8f + mainWindow.getMovLuzPointY(),
0.0f + mainWindow.getMovLuzPointZ()));*/
}
else if (contLuzLetrero >= 200.0f && contLuzLetrero < 300.0f) {
pointLights[2].SetColor(glm::vec3(1.0f, 1.0f, 0.0f));
/*pointLights[2].SetPos(glm::vec3(102.8f + mainWindow.getMovLuzPointX(),
23.8f + mainWindow.getMovLuzPointY(),
0.0f + mainWindow.getMovLuzPointZ()));*/
}
else {
contLuzLetrero = -1.0f;
}
contLuzLetrero++;
}
else {
pointLights[2].SetColor(glm::vec3(0.0f, 0.0f, 0.0f));
}
```

Interacciones triggers

Sincronización de las lámparas de la entrada y salida con el Skybox.

```
// Transión del Skybox
 // En caso de querer modificar la velocidad
 // cambiar el valor de la variable velocidadSkybox
 if (tiempoSkybox == (velocidadSkybox * 0.0)) {
 skyboxFaces.clear();
 skyboxFaces.push_back("textures/skybox/arrakisday_0_rt.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_0_lf.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_0_dn.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_0_up.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_0_bk.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_0_ft.tga");
 skybox = Skybox(skyboxFaces);
 prenderApagarLES = 1;
 }
 else if (tiempoSkybox == (velocidadSkybox * 100.0)) {
 skyboxFaces.clear();
 skyboxFaces.push_back("textures/skybox/arrakisday_1_rt.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_1_lf.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_1_dn.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_1_up.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_1_bk.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_1_ft.tga");
 skybox = Skybox(skyboxFaces);
 prenderApagarLES = 1;
 }
 else if (tiempoSkybox == (velocidadSkybox * 200.0)) {
 skyboxFaces.clear();
 skyboxFaces.push_back("textures/skybox/arrakisday_2_rt.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_2_lf.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_2_dn.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_2_up.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_2_bk.tga");
 skyboxFaces.push_back("textures/skybox/arrakisday_2_ft.tga");
 skybox = Skybox(skyboxFaces);
 prenderApagarLES = 1;
 }

// Lampara********************************************************************************************
 // Entrada
 model = glm::mat4(1.0);
 model = glm::translate(model, glm::vec3(47.0f, -3.0f, -48.3f));
 model = glm::scale(model, glm::vec3(1.0f, 1.2f, 1.0f));
 model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
 materialBrillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
 lamparaM.RenderModel();

 // Salida
 model = glm::mat4(1.0);
 model = glm::translate(model, glm::vec3(47.0f, -3.0f, 48.3f));
 model = glm::scale(model, glm::vec3(1.0f, 1.2f, 1.0f));
 model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
 materialBrillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
 lamparaM.RenderModel();

 // Prender y encender luz lampara entrada y salida
 // Sincronización con el skybox
 if (prenderApagarLES == 0) {
 // Luz entrada
```

```
    pointLights[0].SetColor(glm::vec3(0.0f, 0.0f, 0.0f));
    // Luz salida
    pointLights[1].SetColor(glm::vec3(0.0f, 0.0f, 0.0f));
    }
    else {
    // Luz entrada
    pointLights[0].SetColor(glm::vec3(1.0f, 1.0f, 0.5f));
    // Luz salida
    pointLights[1].SetColor(glm::vec3(1.0f, 1.0f, 0.5f));
    }
```

Texturizado

Sonido 3D ambiental

```
// Para el audio
#include <iostream>
#include <irrKlang.h>
using namespace irrklang;

// Música
    ISoundEngine *engine1 = createIrrKlangDevice();
    ISoundEngine *engine2 = createIrrKlangDevice();
    ISoundEngine *engine3 = createIrrKlangDevice();
    ISoundEngine *engine4 = createIrrKlangDevice();

    ISound *desiertoMusic = engine1->play3D("audio/Desierto.mp3", vec3df(8, -4, 0), true, false, true);
    ISound *morningMusic = engine2->play3D("audio/Morning.mp3", vec3df(8, -4, 0), true, false, true);
    ISound *prismoMusic = engine3->play3D("audio/Prismo.mp3", vec3df(8, -4, 0), true, false, true);
    ISound *differentMusic = engine4->play3D("audio/Different.mp3", vec3df(8, -4, 0), true, false, true);



// Reproducción de la música según la posición de la cámara
    // Fuera de la feria
    if (camera.getCameraPosition().x > 80 && camera.getCameraPosition().x <= 200) {
    desiertoMusic->setIsPaused(false);
    prismoMusic->setIsPaused(true);
    differentMusic->setIsPaused(true);
    morningMusic->setIsPaused(true);
    }
    // General
```

```
else if (camera.getCameraPosition().x > 40 && camera.getCameraPosition().x <= 80) {

desiertoMusic->setIsPaused(true);
prismoMusic->setIsPaused(true);
differentMusic->setIsPaused(true);
morningMusic->setIsPaused(false);
}
// Carrusel
else if (camera.getCameraPosition().x >= 0 && camera.getCameraPosition().x <= 40) {
desiertoMusic->setIsPaused(true);
prismoMusic->setIsPaused(false);
differentMusic->setIsPaused(true);
morningMusic->setIsPaused(true);
}
// Rueda
else if (camera.getCameraPosition().x > -60 && camera.getCameraPosition().x < 0) {
desiertoMusic->setIsPaused(true);
prismoMusic->setIsPaused(true);
differentMusic->setIsPaused(false);
morningMusic->setIsPaused(true);
}
// General
else if (camera.getCameraPosition().x >= -100 && camera.getCameraPosition().x <= -60) {
desiertoMusic->setIsPaused(true);
prismoMusic->setIsPaused(true);
differentMusic->setIsPaused(true);
morningMusic->setIsPaused(false);
}
// Fuera de la feria
else if (camera.getCameraPosition().x >= -200 && camera.getCameraPosition().x < -100) {
desiertoMusic->setIsPaused(false);
prismoMusic->setIsPaused(true);
differentMusic->setIsPaused(true);
morningMusic->setIsPaused(true);
}
```