Universidad Nacional Autónoma de México

Facultad de Ingeniería

Computación Grafica Avanzada

Practica 5. Múltiples Luces.

Alumna: Anahi Hillary Gil González.

Profesor: Reynaldo Martell Ávila.

Edo. De México a 1 de diciembre de 2020

Desarrollo.

1. Utilizando la textura del Blend map, agregar 3 luces más del tipo de faro 2 en los senderos del terreno **(25 Puntos)**.

Se agregaron las posiciones y orientaciones de las otras 3 luces del modelo 2:

```
162    std::vector<glm::vec3> lamp2Position = { glm::vec3(-36.52, 0, -23.24), glm::vec3(-52.73, 0, -3.90),
163        glm::vec3(-50.5859, 0, 14.4531), glm::vec3(-47.4609, 0, 29.4921), glm::vec3(-36.1328, 0, 34.5703)};//------------>Lamparas agregadas
164    std::vector<float> lamp2Orientation = { 21.37 + 90, -65.0 + 90,
165                                            23.50, 237.26, 90};//---------------------------------------------->Orientación de las lamparas
```

Luego se declaran las características de la luz de las lámparas del modelo 2:

```
1062        for (int i = 0; i < lamp2Position.size(); i++) { //------------------------------Luz del segundo modelo de lampara.
1063            glm::mat4 matrixAdjustLamp = glm::mat4(1.0f);
1064            matrixAdjustLamp = glm::translate(matrixAdjustLamp, lamp2Position[i]);
1065            matrixAdjustLamp = glm::rotate(matrixAdjustLamp, glm::radians(lamp2Orientation[i]), glm::vec3(0, 1, 0));
1066            matrixAdjustLamp = glm::scale(matrixAdjustLamp, glm::vec3(1.0, 1.0, 1.0));
1067            matrixAdjustLamp = glm::translate(matrixAdjustLamp, glm::vec3(0.759521, 5.00174, 0));
1068            glm::vec3 lampPosition = glm::vec3(matrixAdjustLamp[3]);
1069            shaderMulLighting.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + i) + "].light.ambient", glm::value_ptr(glm::vec3(0.2, 0.16, 0.01)));
1070            shaderMulLighting.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + i) + "].light.diffuse", glm::value_ptr(glm::vec3(0.4, 0.32, 0.02)));
1071            shaderMulLighting.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + i) + "].light.specular", glm::value_ptr(glm::vec3(0.6, 0.58, 0.03)));
1072            shaderMulLighting.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + i) + "].position", glm::value_ptr(lampPosition));
1073            shaderMulLighting.setFloat("pointLights[" + std::to_string(lamp1Position.size() + i) + "].constant", 1.0);
1074            shaderMulLighting.setFloat("pointLights[" + std::to_string(lamp1Position.size() + i) + "].linear", 0.09);
1075            shaderMulLighting.setFloat("pointLights[" + std::to_string(lamp1Position.size() + i) + "].quadratic", 0.01);
1076
1077            shaderTerrain.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + i) + "].light.ambient", glm::value_ptr(glm::vec3(0.2, 0.16, 0.01)));
1078            shaderTerrain.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + i) + "].light.diffuse", glm::value_ptr(glm::vec3(0.4, 0.32, 0.02)));
1079            shaderTerrain.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + i) + "].light.specular", glm::value_ptr(glm::vec3(0.6, 0.58, 0.03)));
1080            shaderTerrain.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + i) + "].position", glm::value_ptr(lampPosition));
1081            shaderTerrain.setFloat("pointLights[" + std::to_string(lamp1Position.size() + i) + "].constant", 1.0);
1082            shaderTerrain.setFloat("pointLights[" + std::to_string(lamp1Position.size() + i) + "].linear", 0.09);
1083            shaderTerrain.setFloat("pointLights[" + std::to_string(lamp1Position.size() + i) + "].quadratic", 0.02);
1084        }
```

Las lámparas agregadas se pueden ver en la siguiente imagen:

2. Utilizar un modelo de lámpara (No se permite usar los que ya se encuentran, bajar o modelar uno sencillo) agregar un modelo más con su luces, cómo se agrego con la lámpara 2 en el ejercicio de la práctica. **(35 Puntos)**.

Agregando el tercer modelo de lampara:

```
87      Model modelLamp3;//-----------------> Agregando el tercer modelo de farol
```

Agregando las posiciones y las orientaciones del tercer modelo:

```
167     //----------------------> Añadiendo la posición y la orientación de lamp3
168     std::vector<glm::vec3> lamp3Position = { glm::vec3(-38.8671,0,-38.2812)};
169     std::vector<float> lamp3Orientation = { 16.7 };
```

Agregando la referencia del modelo para que se pueda cargar:

```
309         //-------------------------------------------->Modelo de Lampara3
310         modelLamp3.loadModel("../models/Mis modelos/farola/farola.obj");
311         modelLamp3.setShader(&shaderMulLighting);
```

Liberando la memoria:

```
712         modelLamp3.destroy();//---------------------------------->liberando memoria
```

Agregando las características de la luz para el tercer modelo:

```
1086    for (int i = 0; i < lamp3Position.size(); i++) { //----------------------------Luz del tercer modelo de lampara.
1087        glm::mat4 matrixAdjustLamp = glm::mat4(1.0f);
1088        matrixAdjustLamp = glm::translate(matrixAdjustLamp, lamp3Position[i]);
1089        matrixAdjustLamp = glm::rotate(matrixAdjustLamp, glm::radians(lamp3Orientation[i]), glm::vec3(0, 1, 0));
1090        matrixAdjustLamp = glm::scale(matrixAdjustLamp, glm::vec3(1.0, 1.0, 1.0));
1091        matrixAdjustLamp = glm::translate(matrixAdjustLamp, glm::vec3(0, 3.0026, 0));
1092        glm::vec3 lamp3Position = glm::vec3(matrixAdjustLamp[3]);
1093        shaderMulLighting.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].light.ambient", glm::value_ptr(glm::vec3(0.2, 0.16, 0.01)));
1094        shaderMulLighting.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].light.diffuse", glm::value_ptr(glm::vec3(0.4, 0.32, 0.02)));
1095        shaderMulLighting.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].light.specular", glm::value_ptr(glm::vec3(0.6, 0.58, 0.03)));
1096        shaderMulLighting.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].position", glm::value_ptr(lamp3Position));
1097        shaderMulLighting.setFloat("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].constant", 1.0);
1098        shaderMulLighting.setFloat("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].linear", 0.09);
1099        shaderMulLighting.setFloat("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].quadratic", 0.01);
1100
1101        shaderTerrain.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].light.ambient", glm::value_ptr(glm::vec3(0.2, 0.16, 0.01)));
1102        shaderTerrain.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].light.diffuse", glm::value_ptr(glm::vec3(0.4, 0.32, 0.02)));
1103        shaderTerrain.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].light.specular", glm::value_ptr(glm::vec3(0.6, 0.58, 0.03)));
1104        shaderTerrain.setVectorFloat3("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].position", glm::value_ptr(lamp3Position));
1105        shaderTerrain.setFloat("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].constant", 1.0);
1106        shaderTerrain.setFloat("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].linear", 0.09);
1107        shaderTerrain.setFloat("pointLights[" + std::to_string(lamp1Position.size() + lamp2Position.size() + i) + "].quadratic", 0.01);
1108    }
```

Renderizando las lámparas del tercer modelo:

```
1222        //------------------------------------------------------------->Render de la lampara3
1223        for (int i = 0; i < lamp3Position.size(); i++) {
1224            lamp3Position[i].y = terrain.getHeightTerrain(lamp3Position[i].x, lamp3Position[i].z);
1225            modelLamp3.setPosition(lamp3Position[i]);
1226            modelLamp3.setScale(glm::vec3(2.0,2.0,2.0));
1227            modelLamp3.setOrientation(glm::vec3(0, lamp3Orientation[i], 0));
1228            modelLamp3.render();
1229        }
```

El tercer modelo de las lámparas se puede observar en la siguiente imagen:

3. Agregar el movimiento al lamborghini que se encuentra para girar y avanzar. **(10 Puntos)**.

Para controlar el Lamborghini y poder moverlo libremente en el escenario se ocuparon las teclas flechas del teclado:

Tecla	Movimiento
↑	Adelante
↓	Atrás
→	Gira hacia la derecha
←	Gira hacia la izquierda

Agregando los controles del modelo del Lamborghini:

```
884     //---------------------------------------->Controles del modelo del Lamborginni
885     if (glfwGetKey(window, GLFW_KEY_LEFT) == GLFW_PRESS) {
886         modelMatrixLambo = glm::rotate(modelMatrixLambo, 0.03f, glm::vec3(0, 1, 0));
887     }
888     else if (glfwGetKey(window, GLFW_KEY_RIGHT) == GLFW_PRESS) {
889         modelMatrixLambo = glm::rotate(modelMatrixLambo, -0.03f, glm::vec3(0, 1, 0));
890     }
891     else if (glfwGetKey(window, GLFW_KEY_UP) == GLFW_PRESS) {
892         modelMatrixLambo = glm::translate(modelMatrixLambo, glm::vec3(0.0, 0.0, 0.1));
893         rotWheelsX += 0.2;
894     }
895     else if (glfwGetKey(window, GLFW_KEY_DOWN) == GLFW_PRESS) {
896         modelMatrixLambo = glm::translate(modelMatrixLambo, glm::vec3(0.0, 0.0, -0.1));
897         rotWheelsX -= 0.2;
898     }
```

Agregando la animación de las llantas para que giren cuando se mueva el automóvil:

```
1164        //---------------------------------------------------------------------------->Lambo car
1165        glDisable(GL_CULL_FACE);
1166        glm::mat4 modelMatrixLamboChasis = glm::mat4(modelMatrixLambo);
1167        modelMatrixLamboChasis[3][1] = terrain.getHeightTerrain(modelMatrixLamboChasis[3][0], modelMatrixLamboChasis[3][2]);
1168        modelMatrixLamboChasis = glm::scale(modelMatrixLamboChasis, glm::vec3(1.3, 1.3, 1.3));
1169        modelLambo.render(modelMatrixLamboChasis);
1170        glActiveTexture(GL_TEXTURE0);
1171        glm::mat4 modelMatrixLamboLeftDor = glm::mat4(modelMatrixLamboChasis);
1172        modelLamboLeftDor.render(modelMatrixLamboLeftDor);
1173        modelLamboRightDor.render(modelMatrixLamboChasis);
1174        glm::mat4 modelMatrixLamboFrontLeftWheel = glm::mat4(modelMatrixLamboChasis);
1175        modelMatrixLamboFrontLeftWheel = glm::translate(modelMatrixLamboFrontLeftWheel, glm::vec3(0.948337, 0.3771, 1.39999));
1176        modelMatrixLamboFrontLeftWheel = glm::rotate(modelMatrixLamboFrontLeftWheel, rotWheelsX, glm::vec3(1, 0, 0));
1177        modelMatrixLamboFrontLeftWheel = glm::translate(modelMatrixLamboFrontLeftWheel, glm::vec3(-0.948337, -0.3771, -1.39999));
1178        modelLamboFrontLeftWheel.render(modelMatrixLamboFrontLeftWheel);
1179
1180        glm::mat4 modelMatrixLamboFrontRightWheel = glm::mat4(modelMatrixLamboChasis);
1181        modelMatrixLamboFrontRightWheel = glm::translate(modelMatrixLamboFrontRightWheel, glm::vec3(-0.948337, 0.3771, 1.39999));
1182        modelMatrixLamboFrontRightWheel = glm::rotate(modelMatrixLamboFrontRightWheel, rotWheelsX, glm::vec3(1, 0, 0));
1183        modelMatrixLamboFrontRightWheel = glm::translate(modelMatrixLamboFrontRightWheel, glm::vec3(0.947495, -0.378069, -1.40386));
1184        modelLamboFrontRightWheel.render(modelMatrixLamboFrontRightWheel);
1185
1186        glm::mat4 modelMatrixLamboRearLeftWheel = glm::mat4(modelMatrixLamboChasis);
1187        modelMatrixLamboRearLeftWheel = glm::translate(modelMatrixLamboRearLeftWheel, glm::vec3(0.948827, 0.398055, -1.60098));
1188        modelMatrixLamboRearLeftWheel = glm::rotate(modelMatrixLamboRearLeftWheel, rotWheelsX, glm::vec3(1, 0, 0));
1189        modelMatrixLamboRearLeftWheel = glm::translate(modelMatrixLamboRearLeftWheel, glm::vec3(-0.948827, -0.398055, 1.60098));
1190        modelLamboRearLeftWheel.render(modelMatrixLamboRearLeftWheel);
1191
1192        glm::mat4 modelMatrixLamboRearRightWheel = glm::mat4(modelMatrixLamboChasis);
1193        modelMatrixLamboRearRightWheel = glm::translate(modelMatrixLamboRearRightWheel, glm::vec3(-0.948827, 0.398055, -1.60098));
1194        modelMatrixLamboRearRightWheel = glm::rotate(modelMatrixLamboRearRightWheel, rotWheelsX, glm::vec3(1, 0, 0));
1195        modelMatrixLamboRearRightWheel = glm::translate(modelMatrixLamboRearRightWheel, glm::vec3(0.948827, -0.398055, 1.60098));
1196        modelLamboRearRightWheel.render(modelMatrixLamboRearRightWheel);
```

4. Agregar los faros de tipo SpotLight de tal forma que al moverse en la escena, éstos también realicen el mismo movimiento **(30 Puntos)**.

El siguiente código es el necesario para agregar los dos faros del automóvil, tanto para el derecho, como para el izquierdo:

```
985     //------------------------------------------------------------------------>Propiedades, Luz SpotLight del faro derecho
986     shaderMulLighting.setInt("spotLightCount", 2);
987     shaderTerrain.setInt("spotLightCount", 2);
988     glm::vec3 spotLightPosition = glm::vec3(modelMatrixLambo * glm::vec4(0.779388, 0.645692, 2.28855, 1.0));
989     shaderMulLighting.setVectorFloat3("spotLights[0].light.ambient", glm::value_ptr(glm::vec3(0.0, 0.0, 0.0)));
990     shaderMulLighting.setVectorFloat3("spotLights[0].light.diffuse", glm::value_ptr(glm::vec3(0.2, 0.35, 0.2)));
991     shaderMulLighting.setVectorFloat3("spotLights[0].light.specular", glm::value_ptr(glm::vec3(0.3, 0.3, 0.3)));
992     shaderMulLighting.setVectorFloat3("spotLights[0].position", glm::value_ptr(spotLightPosition));
993     shaderMulLighting.setVectorFloat3("spotLights[0].direction", glm::value_ptr(glm::vec3(0.0, 0.0, 1.0)));
994     shaderMulLighting.setFloat("spotLights[0].constant", 1.0);
995     shaderMulLighting.setFloat("spotLights[0].linear", 0.074);
996     shaderMulLighting.setFloat("spotLights[0].quadratic", 0.03);
997     shaderMulLighting.setFloat("spotLights[0].cutOff", cos(glm::radians(12.5f)));
998     shaderMulLighting.setFloat("spotLights[0].outerCutOff", cos(glm::radians(12.5f)));
999
1000    shaderTerrain.setVectorFloat3("spotLights[0].light.ambient", glm::value_ptr(glm::vec3(0.0, 0.0, 0.0)));
1001    shaderTerrain.setVectorFloat3("spotLights[0].light.diffuse", glm::value_ptr(glm::vec3(0.2, 0.35, 0.2)));
1002    shaderTerrain.setVectorFloat3("spotLights[0].light.specular", glm::value_ptr(glm::vec3(0.3, 0.3, 0.3)));
1003    shaderTerrain.setVectorFloat3("spotLights[0].position", glm::value_ptr(spotLightPosition));
1004    shaderTerrain.setVectorFloat3("spotLights[0].direction", glm::value_ptr(glm::vec3(0.0, 0.0, 1.0)));
1005    shaderTerrain.setFloat("spotLights[0].constant", 1.0);
1006    shaderTerrain.setFloat("spotLights[0].linear", 0.074);
1007    shaderTerrain.setFloat("spotLights[0].quadratic", 0.03);
1008    shaderTerrain.setFloat("spotLights[0].cutOff", cos(glm::radians(12.5f)));
1009    shaderTerrain.setFloat("spotLights[0].outerCutOff", cos(glm::radians(12.5f)));
```

```
1011    //------------------------------------------------------------------------>Propiedades, Luz SpotLight del faro izquierdo
1012    glm::vec3 spotLightPositionLambo = glm::vec3(modelMatrixLambo * glm::vec4(-0.779388, 0.645692, 2.28855, 1.0));
1013    shaderMulLighting.setVectorFloat3("spotLights[1].light.ambient", glm::value_ptr(glm::vec3(0.0, 0.0, 0.0)));
1014    shaderMulLighting.setVectorFloat3("spotLights[1].light.diffuse", glm::value_ptr(glm::vec3(0.2, 0.35, 0.2)));
1015    shaderMulLighting.setVectorFloat3("spotLights[1].light.specular", glm::value_ptr(glm::vec3(0.3, 0.3, 0.3)));
1016    shaderMulLighting.setVectorFloat3("spotLights[1].position", glm::value_ptr(spotLightPositionLambo));
1017    shaderMulLighting.setVectorFloat3("spotLights[1].direction", glm::value_ptr(glm::vec3(0.0, 0.0, 1.0)));
1018    shaderMulLighting.setFloat("spotLights[1].constant", 1.0);
1019    shaderMulLighting.setFloat("spotLights[1].linear", 0.074);
1020    shaderMulLighting.setFloat("spotLights[1].quadratic", 0.03);
1021    shaderMulLighting.setFloat("spotLights[1].cutOff", cos(glm::radians(12.5f)));
1022    shaderMulLighting.setFloat("spotLights[1].outerCutOff", cos(glm::radians(12.5f)));
1023
1024    shaderTerrain.setVectorFloat3("spotLights[1].light.ambient", glm::value_ptr(glm::vec3(0.0, 0.0, 0.0)));
1025    shaderTerrain.setVectorFloat3("spotLights[1].light.diffuse", glm::value_ptr(glm::vec3(0.2, 0.35, 0.2)));
1026    shaderTerrain.setVectorFloat3("spotLights[1].light.specular", glm::value_ptr(glm::vec3(0.3, 0.3, 0.3)));
1027    shaderTerrain.setVectorFloat3("spotLights[1].position", glm::value_ptr(spotLightPositionLambo));
1028    shaderTerrain.setVectorFloat3("spotLights[1].direction", glm::value_ptr(glm::vec3(0.0, 0.0, 1.0)));
1029    shaderTerrain.setFloat("spotLights[1].constant", 1.0);
1030    shaderTerrain.setFloat("spotLights[1].linear", 0.074);
1031    shaderTerrain.setFloat("spotLights[1].quadratic", 0.03);
1032    shaderTerrain.setFloat("spotLights[1].cutOff", cos(glm::radians(12.5f)));
1033    shaderTerrain.setFloat("spotLights[1].outerCutOff", cos(glm::radians(12.5f)));
```

Conclusión.

El manejo de las luces y sus características es importante para poder darle un mayor realismo a los escenarios y los personajes que conformen las animaciones o videojuegos que se piensen desarrollar. Aún así es necesario cuidar el número de luces que existen en el escenario ya que pueden representar una carga (en algunas ocasiones innecesaria) para la tarjeta gráfica.

Enlace a repositorio

https://github.com/HillaryGil97/ComputacionGraficaAvanzada