Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Computación Grafica e

Interacción Humano Computadora

Semestre 2021-1

Grupo 08

Espino Rojas Héctor Daniel

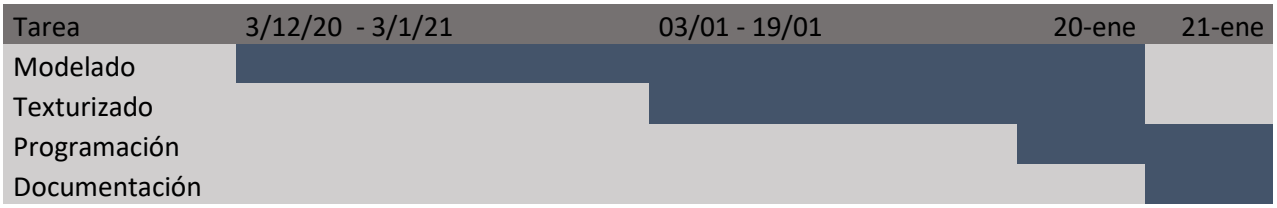
Manual Técnico

Manual de Técnico Rick & Morty.

Objetivo:

El alumno deberá aplicar y demostrar los conocimientos adquiridos durante todo el curso. Recreando un espacio 3D en OpenGL.

Diagrama de Gant:



Alcance del proyecto:

El alumno deberá seleccionar una fachada y un espacio que pueden ser reales o ficticios y presentar imágenes de referencia de dichos espacios para su recreación 3D en OpenGL.

En la imagen de referencia se debe visualizar 7 objetos que el alumno va a recrear virtualmente y donde dichos objetos deben ser lo más parecido a su imagen de referencia.

Es necesario realizar modelado de cada uno de los elementos. Para esto se hará uso del software CAD Maya y dentro del mismo será texturizados los elementos. El trabajo de tanto modelado como código será mediante el uso de una liga de github:
[https://github.com/hectorhmx/ProyectoGrafica]

Después de esto, se programará la interacción y del usuario, posicionamiento de algunos objetos y sus animaciones. Se requerirá hacer uso de todos los conocimientos adquiridos en el curso.

Código:

Se hizo uso de las siguientes variables globales:

```
bool circuito = false;
bool recorrido1 = true;
bool recorrido2 = false;
bool recorrido3 = false;
bool recorrido4 = false;
float rotOjo = 0.0f;
float sentidoOjo = true;
float posCajaZ = 0.0f;
float posCajaX = 0.0f;
//reloj
float rotReloj = 0.0f;
```

```
float rotRelojInv = 0.0f;
float scaReloj = 0.0f;
float transRelojZ = 0.0f;
float transRelojY = 0.0f;
float deltasReloj = 0.2f;
bool aspas = false;
bool recRel0 = true;
bool recRel1 = false;
bool recRel2 = false;
int contVueltas = 0;
//lavadora
float rotPuerta = 0.0f;
bool puerta = false;
bool edo1 = true;
bool edo2 = false;
```

Animaciones:

Puerta:

Para el caso de la puerta se hizo uso de una rotación de -90 a 0 grados. Permitiendo que al activarse la variable booleana puerta se haga un cambio de estado y desactive la variable puerta. Esto hará que cada vez que sea activada puerta, se cierra u abra la misma. Los estados existentes son abierto y cerrado.

```
void animarPuerta() {//0 a -90
    if (puerta) {
        if (edo1)
        {
            printf("Entramos edo1");
            rotPuerta -= 5;
            if (rotPuerta < -90.0)
            {
                edo1 = false;
                edo2= true;
                puerta = false;
                rotPuerta = -90;
            }

            printf("Avanzamos: X:%f Z: %f\n", posCajaX, posCajaZ);

        }
        if (edo2)
        {
            rotPuerta += 5;
```

```
            if (rotPuerta > 0.0)
            {
                edo2 = false;
                edo1 = true;
                puerta = false;
                rotPuerta = 0;
            }

            //printf("Avanzamos: %f", posCajaZ);
            printf("Avanzamos: X:%f Z: %f\n", posCajaX, posCajaZ);
        }
    }

}
```

Se animó el movimiento de la caja y el ojo con la siguiente función. Rotando el ojo de -90 a 90. De este modo podrá el ojo inspeccionar sus alrededores siempre y cuando este activa la variable circuito.

Para la trayectoria del movimiento se utilizó una trayectoria triangular de pendiente igual a 0.4852. La caja se moverá en diagonal a la otra esquina del estante, descenderá por el eje Z y una vez llegado a z0, retornará por el eje x. Fue calculado tanto la pendiente como el ángulo de la trayectoria y los deltas de cambio.

```
void animacionCaja()
{

    //Movimiento de la caja
    if (circuito)//X  de 0 a 1.4799 Z de 0 a 3.05
    {
        //M = (1.4799 - 0)/ (3.05 - 0) = 0.4852
        //angulo = tan = x/z = 0.4852 = 25.8826
        //Si deltaX = 0.01
        //Y M = 0.4852 = deltaX/deltaZ
        // deltaZ = 0.01/M = 0.0206

        if (recorrido1)
        {
            posCajaZ += 0.0206;
            posCajaX += 0.01;
            if (posCajaZ>3.05)
            {
                recorrido1 = false;
                recorrido2 = true;
            }
```

```
            printf("Avanzamos: X:%f Z: %f\n", posCajaX,posCajaZ);

        }
        if (recorrido2)
        {
            posCajaX -= 0.01;
            if (posCajaX < 0)
            {
                recorrido2 = false;
                recorrido3 = true;
            }

            //printf("Avanzamos: %f", posCajaZ);
            printf("Avanzamos: X:%f Z: %f\n", posCajaX, posCajaZ);
        }
        if (recorrido3)
        {
            posCajaZ -= 0.01;
            if (posCajaZ < 0)
            {
                recorrido3 = false;
                recorrido1 = true;
            }
            //printf("Avanzamos: %f", posCajaZ);
            printf("Avanzamos: X:%f Z: %f\n", posCajaX, posCajaZ);
        }
        if (recorrido4)
        {
            posCajaX -= 0.01;
            if (posCajaX < 0)
            {

                recorrido4 = false;
                recorrido1 = true;
            }

            //printf("Avanzamos: %f", posCajaZ);
            printf("Avanzamos: X:%f Z: %f\n", posCajaX, posCajaZ);
        }

        if (sentidoOjo){
            rotOjo += 6;
            if (rotOjo > 90) {
                sentidoOjo = false;
            }
```

```
        }
        else {
            rotOjo -= 6;
            if (rotOjo < -90) {
                sentidoOjo = true;
            }
        }
    }
}
```

Reloj:

La animación del Reloj contó con mayor complejidad en cuanto al orden de sus transformaciones, de modo que fuera posible que las aspas rotaran con respecto a un centro, les afectara una traslación paulatina y tanto las aspas como el centro crecieran proporcionalmente. También fue utilizado modelado jerárquico para esto.

Esto fue logrado mediante la siguiente disposición:

```
        //centroReloj
        view = camera.GetViewMatrix();
        model = glm::mat4(1);//seteamos la matriz
        tmp1 = model = glm::translate(model, glm::vec3(-
1.6844f, 11.5547f + transRelojY, -9.3084f + transRelojZ));//-1.6844 11.5547  -
9.3084
        tmp2 = model;
        model = glm::scale(model, glm::vec3(1.0f + (scaReloj * 0.1f)));
        //tmp = model = glm::translate(model, glm::vec3(0.0f,0.0f,0.0f));
        //model = glm::rotate(model, glm::radians(rot), glm::vec3(1.0f,0.0f,0.0f
));
        glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
        centroReloj.Draw(lightingShader);

        //relojMin
        model = tmp2;
        model = glm::rotate(model, glm::radians(rotReloj), glm::vec3(0.0f, 0.0f,
 1.0f));
        model = glm::scale(model, glm::vec3(1.0f + (scaReloj * 0.1f)));
        model = glm::translate(model, glm::vec3(0.0f, 0.4027f, +0.0371f));//-
1.6844  11.9574  -9.3455
        glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
        relojMin.Draw(lightingShader);

        //relojHor
        model = tmp1;
        model = glm::rotate(model, glm::radians(rotRelojInv), glm::vec3(0.0f, 0.
0f, 1.0f));
```

```
        model = glm::scale(model, glm::vec3(1.0f + (scaReloj * 0.1f)));
        model = glm::translate(model, glm::vec3(0.163f, -0.2941f, -0.0371f));//-
1.5214   11.2606   -9.3455
        glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
        relojHor.Draw(lightingShader);
```

Código de animación del reloj:

Al activar aspas se pasará por un primer estado donde las aspas girarán en sentidos opuestos (La diferencia de velocidades es para que sea más claro este movimiento). Posteriormente se pasará al segundo estado.

En el segundo estado se hará un recorrido lineal de 30 unidades, donde las aspas girarán, y se trasladarán en el eje Z además de crecer con el avance. Donde habrá un crecimiento cuadrático en el eje Y. De este modo generará una curva donde su delta de cambio permite un cambio cuadrático con respecto al de la Y. Además de seguir creciendo hasta haber terminado la animación y regresando a su posición original.

```
void animacionReloj()
{
    //Movimiento del reloj
    if (aspas)//X  de 0 a 1.4799
    {
        if (recRel0) {
            contVueltas += 1;
            rotReloj += 6+ (contVueltas*4/270);
            rotRelojInv -=5.5 + (contVueltas*4 / 270);
            if (contVueltas > 270) {
                recRel1 = true;
                recRel0 = false;
            }
        }
        if (recRel1)
        {
            transRelojZ += deltasReloj;
            scaReloj += 0.005f;
            rotReloj += 10;
            rotRelojInv -= 9.5;
            if (transRelojZ > 30)
            {
                recRel1 = false;
                recRel2 = true;
            }

            printf("Avanzamos:Z: %f\n",transRelojZ);
```

```
        }
        if (recRel2)
        {
            transRelojZ += deltasReloj*2;
            transRelojY += deltasReloj*2.001;
            scaReloj += 0.05f * 0.05f;
            rotReloj += 15;
            rotRelojInv -= 14.5;
            if (transRelojZ > 50)
            {
                recRel2 = false;
                recRel0 = true;
                aspas = false;
                transRelojZ = 0;
                transRelojY = 0;
                rotReloj = 0;
                rotRelojInv = 0;
                scaReloj = 0;
                contVueltas = 0;
            }

        }
    }
}
```

Se utilizó el siguiente código para implementar antibounce en las teclas:

```
    if (keys[GLFW_KEY_C])
    {
        circuito = !circuito;
        printf("AnimacionCaja");
    }

    if (keys[GLFW_KEY_R])
    {
        aspas = true;
        printf("AnimacionAspas");
    }
    if (keys[GLFW_KEY_F])
    {
        puerta =  !puerta;
        printf("InicioPuerta");
    }
```

Referencias:

Hughes, J. (2013). Computer Graphics: Principles and Practice (3rd Revised ed.). Harlow, Reino Unido: Pearson Education.