Computación Gráfica

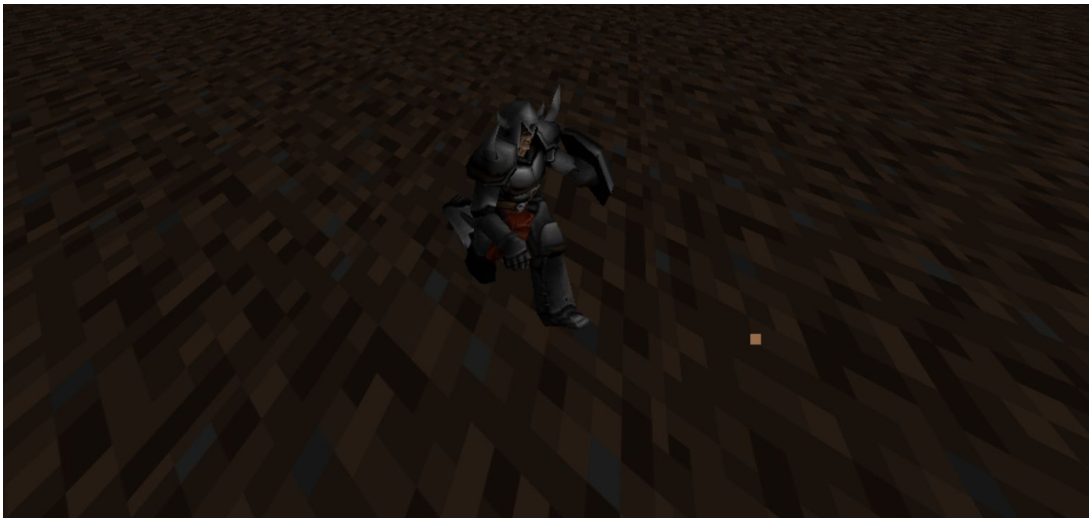
Trabajo Obligatorio

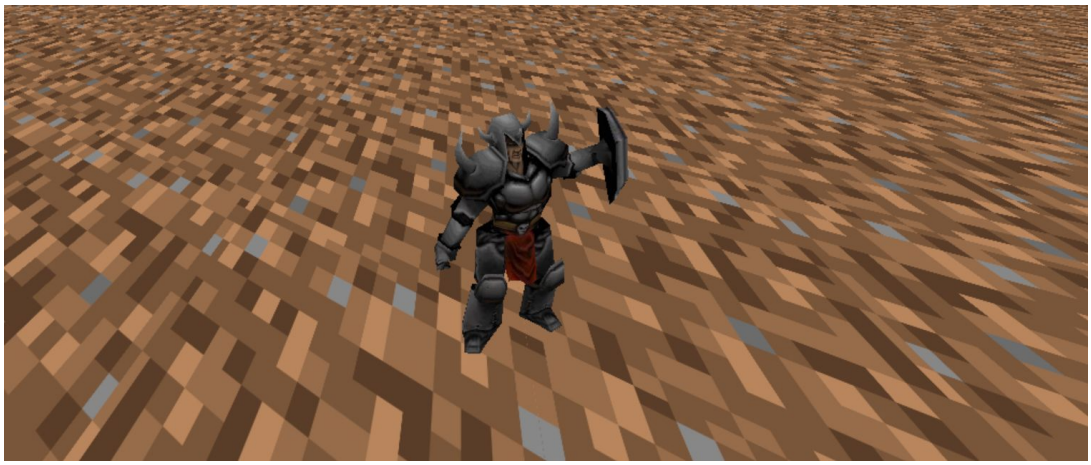
22 / 11 / 20

Ignacio Sica

Introduccion

En el siguiente trabajo se desarrolla una escena 3D interactiva simple. La misma le brinda al usuario numerosas opciones diferentes de visualizar la escena y también de interactuar con la misma haciendo que el personaje corra o cambie de animación. A continuación se presenta la guía del usuario y los fragmentos de código que fueron más relevantes a la hora de llevar a cabo el trabajo obligatorio. A continuación se muestran las versión del Ide utilizado y también las versiones de las librerías utilizadas.





Guia de usuario

- **L:** Cambia de luz general de la escena a una luz que orbita alrededor del modelo
- **T:** Cambia la textura del guerrero
- **R:** Hace que el personaje corra en línea recta sobre el terreno
- **A-S-W-D:** Mueve la cámara para adelante, atras, izquierda y derecha
- **Up-Down:** Mueve la cámara para arriba y para abajo
- **Left-Right:** Órbita la cámara alrededor del modelo
- **1-2-3-4-5-6:** Cambia la animación que está realizando el personaje, desde salto hasta agacharse
- **Z:** Activa o desactiva el z-buffer
- **B:** Activa o desactiva el bfc
- **C:** Activa o desactiva el bfcCW
- **M:** Rellena la figura o la deja simplemente como alambre(Line o Fill)
- **Python 3.8.6**
- **pygame 1.9.6**

Parser

```
class Obj:
    def __init__(self, name):
        self.name = name
        self.v =  [[0.0,0.0,0.0]]
        self.vn = [[0.0,0.0,0.0]]
        self.vt = [[0.0,0.0,0.0]]
        self.faces = []
        self.drawV = []
        self.drawN = []
        self.drawT = []

    def parse(self, path):
        f = open(path, "r")

        for l in f:
            l = l.strip('\n')
            if l.find("v ") >= 0:
                ver = l.split(" ")
                verF = [float(x) for x in ver[1:]]
                self.v.append(verF)
            elif l.find("vn ") >= 0:
                norm = l.split(" ")
                normF = [float(x) for x in norm[1:]]
                self.vn.append(normF)
            elif l.find("vt ") >= 0:
                text = l.split(" ")
                textF = [float(x) for x in text[1:]]
                self.vt.append(textF)
            elif l.find("f ") >= 0:
                face = l.split(" ")
                for vnt in face[1:]:
                    faceI = [int(x) for x in vnt.split('/')]
                    self.faces.append(faceI)

        for f in self.faces:
            self.drawV.append(self.v[f[0]])
            self.drawN.append(self.vn[f[1]])
            self.drawT.append(self.vt[f[2]])
```

Animaciones

A modo de poder tener varias animaciones precargadas tuve que desarrollar la siguiente estructura. Esto le permite al programa cargar todas las animaciones de una vez y luego con las animaciones ya cargadas en memoria funcionar fluidamente. Dicha estructura consiste en una lista de animaciones donde cada animación es un conjunto de modelos obj y también cuentan con un entero que indica la cantidad de cuadros que posee dicha animación ya que todas tienen un largo distinto. Los modelos obj se cargan utilizando el parser y ya poseen toda la información necesaria para dibujar el mismo a través del algoritmo de glDrawArrays.

```
 modelAnimation = []

    animations = []
      animations_title = ["stand", "run", "crouch_stand", "fallback", "jump",
"wave", "point"]
    animations_frames = [40,6,19,17,6,11,12]

    for i in range(7):
        animation = []
        for j in range(animations_frames[i]):
               model_animation = Obj("model " + animations_title[i] + " # " +
str(j))
             path = "./knight_animado/knight_"+animations_title[i]+"_"+str(j)+
".obj"
            model_animation.parse(path)
            animation.append(model_animation)
        animations.append(animation)
        animations.clear

    animation_index = 0
```

glDrawArrays

Tomando la información de las animaciones, esta sección del código es la encargada de dibujar el modelo obj correspondiente. Para ello consulta el índice de animación en el que se encuentra y el número de cuadro por el que va.

```
glEnableClientState(GL_VERTEX_ARRAY)
        glEnableClientState(GL_NORMAL_ARRAY)
        glEnableClientState(GL_TEXTURE_COORD_ARRAY)

                                        glVertexPointer(3,     GL_FLOAT,     0,
animations[animation_index][index].drawV)
        glNormalPointer(GL_FLOAT, 0, animations[animation_index][index].drawN)
                                      glTexCoordPointer(2,     GL_FLOAT,     0,
animations[animation_index][index].drawT)
        glBindTexture(GL_TEXTURE_2D, texts[text_index])
                                             glDrawArrays(GL_TRIANGLES,      0,
len(animations[animation_index][index].faces))
        glBindTexture(GL_TEXTURE_2D, 0)

        glDisableClientState(GL_VERTEX_ARRAY)
        glDisableClientState(GL_NORMAL_ARRAY)
        glDisableClientState(GL_TEXTURE_COORD_ARRAY)
```