UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

Facultad de Ingeniería de Producción y Servicios

Ciencia de la Computación

Laboratorio 2

ALUMNOS:
Pfuturi Huisa, Oscar David
Quispe Menor, Hermogenes
Quiñonez Lopez, Efrain German
Fernandez Mamani, Brayan Gino
Santos Apaza, Yordy Williams

DOCENTE:
MSc. Vicente Machaca Arceda

CURSO:
Computación Gráfica

27 de abril de 2021

Índice

1. Github

- https://github.com/oscar-pfuturi-h/Comp-Grafica/tree/main/practica_02

2. Funciones

Para la implementación de este trabajo de animación 3D, hemos creado clases para definir los componentes que conforman el escenario y el objeto principal, una esfera que estará sujeta a ciertas condiciones.

La clase MySphere define la bola de billar que utilizaremos para este caso, tomando su posición y su radio como parámetros.
La clase MyFloor define la parte del escenario que se usará como base que soporten la esfera.
La clase MyPared define los límites del escenario, las cuales no puede sobrepasar la esfera durante la ejecución del programa.

2.1. Creación de clases

```
class MySphere:
    def __init__(self, pos, radius):
        self.pos = pos
        self.radius = radius
        self.velocity = [0, 10, 0]
        self.last_velocity = [0, -10, 0]
        self.source1 = vtk.vtkSphereSource()
        self.sphere_mapper = vtk.vtkPolyDataMapper()
        self.sphere_actor = vtk.vtkActor()

    def createSphere(self):
        self.source1.SetThetaResolution(50)
        self.source1.SetRadius(self.radius)
        self.source1.Update()

    def createMapper(self):
        self.sphere_mapper.SetInputData(self.source1.GetOutput())

    def createActor(self):
        reader = vtk.vtkJPEGReader()
        reader.SetFileName("padoru.jpg")
        # Create texture object
        texture = vtk.vtkTexture()
        texture.SetInputConnection(reader.GetOutputPort())
        #Map texture coordinates
        map_to_plane = vtk.vtkTextureMapToPlane()
        map_to_plane.SetInputConnection(self.source2.GetOutputPort())
        self.floor_actor.SetMapper(self.floor_mapper)
        self.floor_actor.SetMapper(mapper)
        self.floor_actor.SetTexture(texture)
```

```
        self.floor_actor.GetProperty().SetOpacity(0.8)
        self.floor_actor.SetPosition(self.pos[0], self.pos[1], self.
   pos[2])
    def getActor(self):
        return self.sphere_actor

class MyFloor:
    def __init__(self, pos, height):
        self.pos = pos
        self.height = height
        self.velocity = np.array([0, 0, 0])
        self.actor = None

        self.source2 = vtk.vtkCubeSource()
        self.floor_mapper = vtk.vtkPolyDataMapper()
        self.floor_actor = vtk.vtkActor()
    ...

class MyPared:
    def __init__(self, pos, height):
        self.pos = pos
        self.height = height
        self.velocity = np.array([0, 0, 0])
        self.actor = None
    ...
```

2.2. Source, Mapper, Actor

Se definen los objetos que se crean a partir de las clases que proporciona la librería VTK, como mappers y actors.

```
# source
sphere.createSphere()
floor.createFloor()
paredes.createCubes()

# mapper
sphere.createMapper()
floor.createMapper()
paredes.createMapper()

# actor
paredes.createActor()
floor.createActor()
sphere.createActor()

# camera
camera = vtk.vtkCamera()
camera.SetFocalPoint(0,0,0)
```

```
camera.SetPosition(50,50,50)

# renderer
#Se agregan los actores al renderer
renderer = vtk.vtkRenderer()
renderer.SetBackground(0.0, 0.0, 0.0)
renderer.AddActor(sphere.getActor())
renderer.AddActor(floor.getActor())
addActores(renderer,paredes.getActor())
renderer.SetActiveCamera(camera)
```

2.3. Función Callback

Esta función se ejecutará recursivamente para la renderización de las formas que hemos instanciado con anterioridad.
El proceso es el siguiente:

- Utiliza variables globales, las cuales usamos para definir la velocidad y la rotación de la esfera.
- Modificamos la posición de la esfera por su velocidad en cada instante
- Al mismo tiempo, rotamos la esfera según la dirección respecto a los ejes (X, Z)
- Reducimos la velocidad una milésima parte de su valor.
- Definimos las condiciones para que la esfera no sobrepase los límites definidos por los objetos de la clase MyPared.

```
def callback_func(caller, timer_event):
    global vx,vz,ax,az,rotacion
    sphere.pos[0] += vx
    sphere.pos[2] += vz
    sphere.sphere_actor.SetPosition(sphere.pos)
    if rotacion > 0.001 or rotacion < -0.001:
        rotacion -= rotacion * 0.004
        if (vx > 0):
            sphere.sphere_actor.RotateZ(rotacion)
        else:
            sphere.sphere_actor.RotateZ(-rotacion)
        if vz > 0:
            sphere.sphere_actor.RotateX(rotacion)
        else:
            sphere.sphere_actor.RotateX(-rotacion)
    vx -= vx*0.004
    vz -= vz*0.004
    render_window.Render()
    x,y,z=sphere.sphere_actor.GetPosition()
    if (x<-largo/2 + 3 or x>largo/2 - 3):
        vx*=-1
    if (z<-ancho/2 + 3 or z>ancho/2 - 3):
        vz*=-1
```

3. Resultados

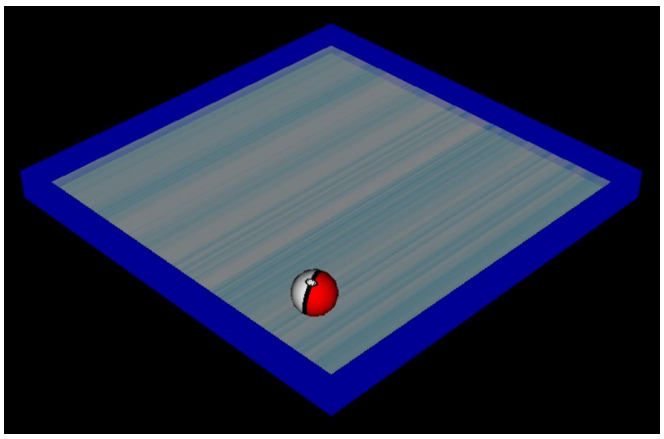

Figura 1: Movimiento 1

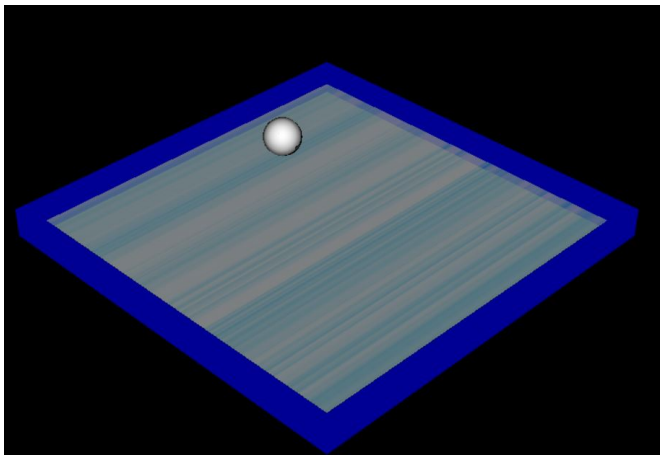

Figura 2: Movimiento 2

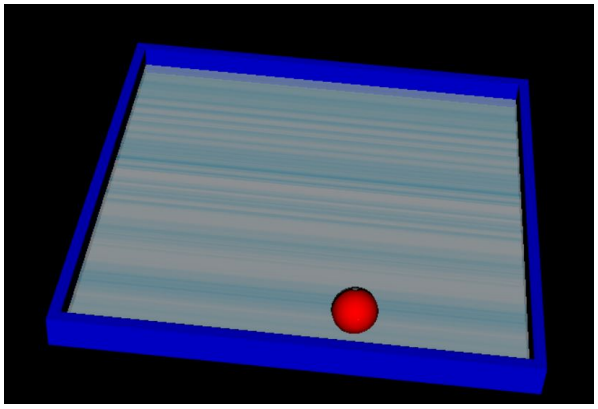

Figura 3: Movimiento 3