Solve Dynamics

Alexander Gomez Almanza

Resumen

Solve Dynamics es un prototipo de software que se encarga de simulaciones en 3D de diferentes escenarios de la Física Dinámica

Abstract

Solve dynamics is a software prototype that makes 3D simulations from a query of sceneries for Dynamics Physics

1. Introduction

La librería VPython es una librería de Python (Visual Python) creada para la fácil realización de simulaciones físicas en 3D, la cual con unas pocas keywords puede crear objetos en un espacio tridimensional y agregarle propiedades como la masa o velocidad.
Permite realizar cualquier simulación que se nos pueda ocurrir con algunas limitaciones propias del mismo software. Con ello podremos lograr simulaciones a tiempo real aproximado de diferentes objetos y sus movimientos.

2. Marco Teórico

La física dinámica es la parte de la física clásica que se encarga de los objetos en movimiento siguiendo las leyes de Newton.

$$F = m*a$$

$$P = m*v$$

Y las ecuaciones de movimiento

$$x = x_0 + v_0*t + \frac{1}{2}*a*t^2$$

Y los métodos computacionales

$$x \mathrel{+}= v * dt$$

$$v \mathrel{+}= a*t$$

$$a = k$$

Que salen de las ecuaciones

$$v = \frac{d}{dt}(x)$$

$$a = \frac{d}{dt}(v)$$

Con estas formulas se pueden modelar muchos de los problemas y escenarios de la Física Dinámica. La Física clásica en general fue muy influenciada por el trabajo de Isaac Newton, Galileo Galilei y los descubrimientos matemáticos del Calculo y el avance matemático y científico del siglo XVI. Es decir desde hace ya más de trescientos años.

3. Desarrollo de Ingeniera

Como se dijo en la introducción se utilizó la librería vpython y se realizó una interfaz en QT

Podemos mirar el ejemplo de la colisión lineal elástica

```
from vpython import *
from readfile import *

import os

# Obtener la ruta del directorio actual
dir_actual = os.path.dirname(os.path.abspath(__file__))

# Construir la ruta al archivo .txt
filename = os.path.join(dir_actual, 'data_Colisiónlinealelástica.txt')

data_array = read_file(filename)
data_array1 = filter_numbers(data_array)
data = float_array(data_array1)
m1=data[0]
m2=data[1]
d=data[2]
v01=data[3]

#creacion de objetos
carroiz = box(pos=vec(-10,0,0),size=vec(2,2,2),color = color.blue)
carrode = box(pos=vec(10+d,0,0),size=vec(2,2,2),color = color.red)
flechaiz = arrow(pos=vec(-10,0,0),axis=vec(3,0,0),color=color.yellow)
piso = box(pos=vec(d,-0.9,0), size=vec(40+d,0.5,10),color=color.green)
#asignar propiedades
carroiz.vel=vec(v01,0,0) #blocks velocity
carrode.vel=vec(0,0,0)
flechaiz.vel=vec(v01,0,0)
```

```
carroiz.mass = m1 #blocks mass
carrode.mass =m2

t = 0
t2 =0
t3 =0
dt = 0.0025

while t<15:
#simulacion antes de la colision
    rate(650)
    t += dt
    carroiz.pos=carroiz.pos + carroiz.vel*dt
    flechaiz.pos=flechaiz.pos + flechaiz.vel*dt
    carrode.pos = carrode.pos+carrode.vel*dt
    if not flechaiz.pos.x<(8+d):
        carrode.vel=((2*carroiz.mass)/(carroiz.mass+carrode.mass))*carroiz.v
el
        flechade =
arrow(pos=vec(10+d,0,0),axis=vec(3,0,0),color=color.yellow)
        flechade.vel=((2*carroiz.mass)/(carroiz.mass+carrode.mass))*carroiz.
vel
        flechade.pos = flechade.pos + flechade.vel*dt
        carroiz.vel=((carroiz.mass-
carrode.mass)/(carroiz.mass+carrode.mass))*carroiz.vel
        flechaiz.vel=((carroiz.mass-
carrode.mass)/(carroiz.mass+carrode.mass))*flechaiz.vel
        if not carroiz.vel.x>=0:
            flechaiz.axis=flechaiz.axis*-1
        break
    #simulacion despues de la colision
while t2<4.5:
    rate(650)
    t2 += dt
    carroiz.pos = carroiz.pos + carroiz.vel*dt
    flechaiz.pos = flechaiz.pos + flechaiz.vel*dt
    carrode.pos = carrode.pos + carrode.vel*dt
    flechade.pos = flechade.pos + flechade.vel*dt
```

Aquí podemos ver que ajustamos el tiempo para que corra en ciclos de 0.0025 s y a una tasa de refresco de 650 FPS

4. Resultados

Los resultados son simulaciones muy sensibles a los datos iniciales, valores extremos terminaran por destruir la simulación y su disfuncionamiento. Por ello hay que tener en cuenta que es un prototipo y los errores se presentaran. Pero en la gran parte de los casos se pueden disfrutar simulaciones físicas en 3D que muestran el comportamiento de los objetos tridimensionales en la Física clásica algo que nos puede dar una pequeña vista a los fenómenos reales.

5. Conclusiones

Las conclusiones son que los fenómenos de la dinámica se pueden modelar con cierta precisión y dar una pequeña vista a como se comportan en el mundo Físico. Con la aclaración de que el software esta en aun en etapas muy tempranas de desarrollo y por ello muchos problemas podrán ser encontrados por lo cual este no es un producto de software terminado si no un proyecto que esta en la fase de prototipo.

Referencias

VPython. (n.d.). https://vpython.org/

Fisica Universitaria Sears y Zemansky : Sears y Zemansky : Free Download, Borrow, and Streaming : Internet Archive. (2023, April 1). Internet Archive. https://archive.org/details/fisica-universitaria-sears-y-zemansky