Trabajo Extra: Interpolación lineal, cuadrática y cúbica mediante scripts en Maya.

Autor: Elías Méndez García

1. Introducción a scripting con Maya.

Lo primero que debemos saber sobre crear scripts en Maya es que existen dos lenguajes que se pueden usar. El primero se llama MEL script y es un lenguaje propio de Maya. El segundo es Python, Maya permite usar Python como lenguaje de scripting incluyendo un paquete que permite acceso a todos los comandos que puede usar MEL.

Para acceder al editor de scripts podemos acceder mediante la interfaz de Maya en la esquina inferior derecha.

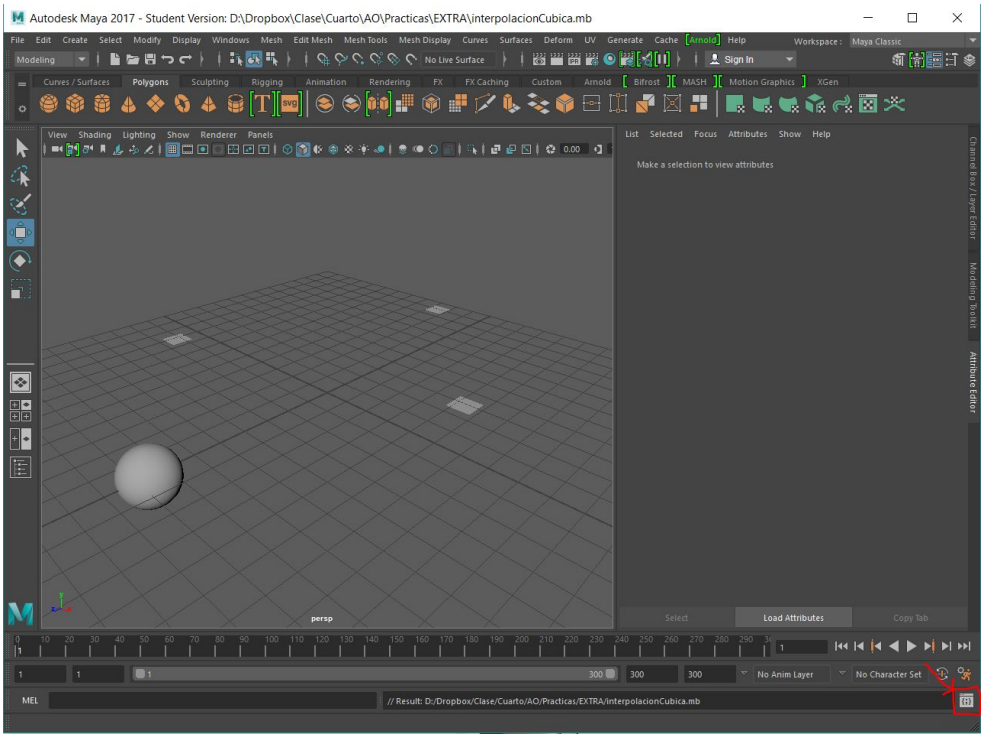

Figura 1: Localización del editor de scripts en la Interfaz de Maya

Una vez le demos al botón se nos abrirá el editor de scripts donde en la parte superior vemos las acciones en MEL de todas la acciones que realizamos con la interfaz de Maya y en la zona inferior vemos el editor del script.

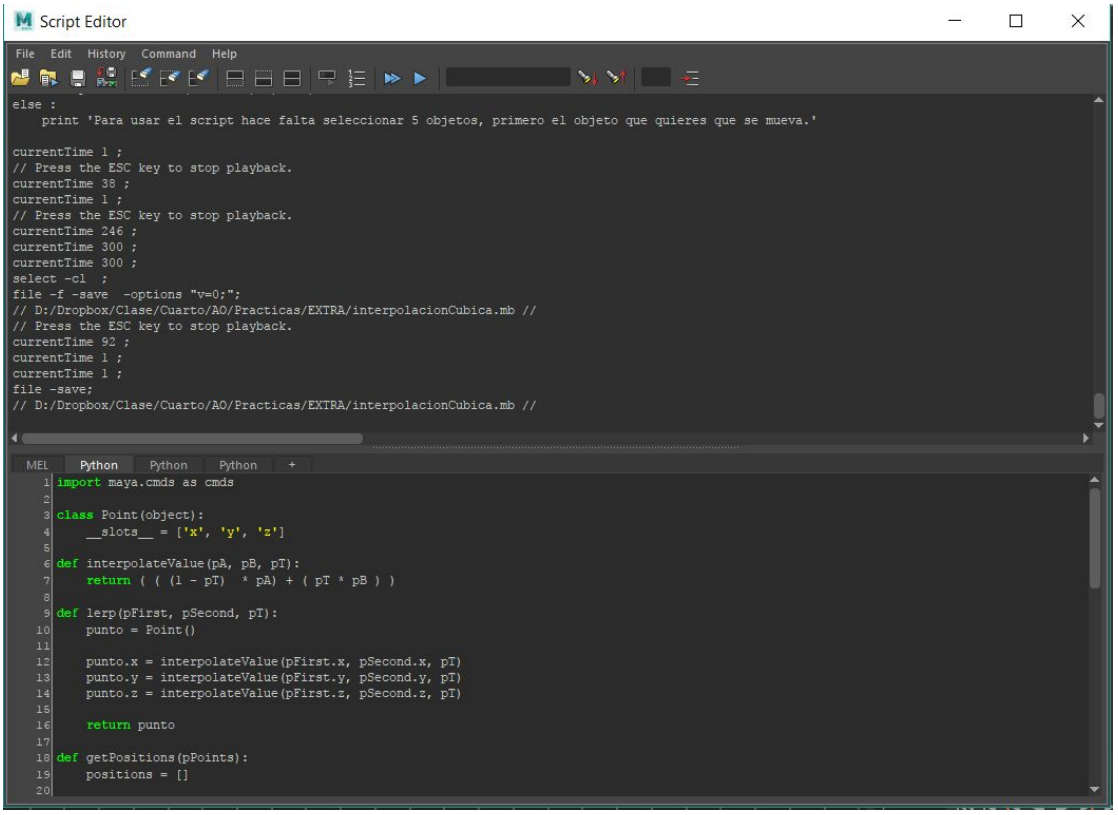

Figura 2: Editor de scripts de Maya.

El editor tiene las acciones comunes de cualquier editor como abrir archivos de scripts o guardarlos. Para ejecutar el contenido del editor debemos seleccionar el código que queremos ejecutar y darle al botón de play azul.

2. Creación del script de interpolación lineal.

Lo primero que debemos hacer es importar el módulo de Maya para poder usar los comandos de Maya desde Python. Para ello lo primero que debemos hacer es añadir esta linea al principio del script.

```
import maya.cmds as cmds
```

Desde este momento podremos usar los comandos de Maya con cmds.

Para ayudarnos crearemos una clase para almacenar las coordenadas de un punto con la siguiente clase:

```
class Point(object):
    __slots__ = ['x', 'y', 'z']
```

Ahora vamos a definir una serie de funciones para poder realizar la interpolación.

Primero vamos a definir una función que nos calcule y devuelva la interpolación lineal de dos valores, el primer y segundo argumento, en un instante t comprendido entre 0 y 1. La interpolación se realiza de la forma que hemos visto en clase.

```
def interpolateValue(pA, pB, pT):
    return ( ( (1 - pT)  * pA) + ( pT * pB ) )
```

Una vez tenemos esta función crearemos una función que reciba dos puntos y devuelva un nuevo punto después de realizar la interpolación lineal entre los dos puntos en el instante t comprendido entre 0 y 1.

```
def lerp(pFirst, pSecond, pT):
    punto = Point()

    punto.x = interpolateValue(pFirst.x, pSecond.x, pT)
    punto.y = interpolateValue(pFirst.y, pSecond.y, pT)
    punto.z = interpolateValue(pFirst.z, pSecond.z, pT)

    return punto
```

Con estas funciones creadas tenemos todas las funciones básicas para generar la posición en la interpolación. Ahora nos falta obtener las posición de los puntos de control. Para ello usaremos el comando de getAttr que nos permite obtener un atributo del objeto que nombremos. Usaremos un bucle para obtener la posición de todas las posiciones de los puntos de control y lo devolveremos como un array.

```
def getPositions(pPoints):
    positions = []

    for point in pPoints:
        p = Point()
        p.x = cmds.getAttr("%s.translateX" % point)
        p.y = cmds.getAttr("%s.translateY" % point)
        p.z = cmds.getAttr("%s.translateZ" % point)
        positions.append(p)

    return positions
```

Ahora crearemos la animación de movimiento del objeto, para ello nos ayudaremos de los comandos de Maya para crear y borrar keys de un objeto dado su ID, primero borraremos las keys de movimiento que tenga definido el objeto. Una vez hecho esto generamos la posición en la que debería estar el objeto en ese frame con la interpolación lineal que hemos implementado antes. Una vez generada la nueva posición modificamos la posición del objeto en el frame en el que estemos.

```
def interpolacionLineal(pSelectionList, pStart, pEnd):
    objectName = pSelectionList[0]
    pSelectionList.remove(objectName)

    positions = getPositions(pSelectionList)

    nEnd = pEnd - pStart - 1

    cmds.cutKey(objectName, time=(pStart, pEnd), attribute='translateX')
    cmds.cutKey(objectName, time=(pStart, pEnd), attribute='translateY')
    cmds.cutKey(objectName, time=(pStart, pEnd), attribute='translateZ')

    for t in range( 0, nEnd ):
        p = lerp(positions[0], positions[1], t/(nEnd * 1.0))

        nT = pStart + t

         cmds.setKeyframe( objectName, time=(nT), attribute='translateX',
value=p.x, inTangentType='linear', outTangentType='linear' )

         cmds.setKeyframe( objectName, time=(nT), attribute='translateY',
value=p.y, inTangentType='linear', outTangentType='linear' )

         cmds.setKeyframe( objectName, time=(nT), attribute='translateZ',
value=p.z, inTangentType='linear', outTangentType='linear' )
```

Para que el movimiento del objeto sea correcto debemos acordarnos de definir que la tangente de salida y de entrada sea lineal, una vez la animación esté creada podremos modificar las keys si queremos aplicar alguna de los principios de animación.

Para obtener los nombres de los objetos de la escena que queremos usar podemos seleccionar los objetos en la escena, siendo el primero el que queremos que se anime y los siguientes los puntos de control. Para usar este método el orden en el que seleccionemos los objetos es importante, ya que se ha dado por hecho en el script que están ordenados.

```
selected = cmds.ls(orderedSelection=True)

if len(selected) == 5:
    interpolacionCubica(selected, 1, 300)
else :
    print 'Para usar el script hace falta seleccionar 5 objetos, primero
el objeto que quieres que se mueva.'
```

3. Interpolación cuadrática y cúbica.

Para la interpolación cuadrática y cúbica se ha usado el algoritmo de Casteljau que dice que usando la interpolación lineal de forma recursiva podemos obtener la interpolación de grados superiores.

Lo único que necesitamos hacer es usar la interpolación lineal varias veces para obtener el punto final, en el caso de la cuadrática necesitaremos usarla 3 veces. En el caso de la cúbica necesitaremos usarla 6 veces.

```
def interpolacionCuadratica(pSelectionList, pStart, pEnd):
    objectName = pSelectionList[0]
    pSelectionList.remove(objectName)

    positions = getPositions(pSelectionList)

    nEnd = pEnd - pStart - 1

    cmds.cutKey( objectName, time=(pStart, pEnd), attribute='translateX'
)
    cmds.cutKey( objectName, time=(pStart, pEnd), attribute='translateY'
)
    cmds.cutKey( objectName, time=(pStart, pEnd), attribute='translateZ'
)

    for t in range( 0, nEnd ):
        fraccion_t =t/(nEnd * 1.0)

        p1 = lerp(positions[0], positions[1], fraccion_t)
        p2 = lerp(positions[1], positions[2], fraccion_t)

        p = lerp(p1, p2, fraccion_t)

        nT = pStart + t
```

```
        cmds.setKeyframe( objectName, time=(nT), attribute='translateX',
value=p.x, inTangentType='linear', outTangentType='linear' )
        cmds.setKeyframe( objectName, time=(nT), attribute='translateY',
value=p.y, inTangentType='linear', outTangentType='linear' )
        cmds.setKeyframe( objectName, time=(nT), attribute='translateZ',
value=p.z, inTangentType='linear', outTangentType='linear' )
```

```
def interpolacionCubica(pSelectionList, pStart, pEnd):
    objectName = pSelectionList[0]
    pSelectionList.remove(objectName)

    positions = getPositions(pSelectionList)

    nEnd = pEnd - pStart - 1

    cmds.cutKey( objectName, time=(pStart, pEnd), attribute='translateX'
)
    cmds.cutKey( objectName, time=(pStart, pEnd), attribute='translateY'
)
    cmds.cutKey( objectName, time=(pStart, pEnd), attribute='translateZ'
)

    for t in range( 0, nEnd ):
        fraccion_t =t/(nEnd * 1.0)

        p1 = lerp(positions[0], positions[1], fraccion_t)
        p2 = lerp(positions[1], positions[2], fraccion_t)
        p3 = lerp(positions[2], positions[3], fraccion_t)

        q1 = lerp(p1, p2, fraccion_t)
        q2 = lerp(p2, p3, fraccion_t)

        p = lerp(q1, q2, fraccion_t)

        nT = pStart + t

        cmds.setKeyframe( objectName, time=(nT), attribute='translateX',
value=p.x, inTangentType='linear', outTangentType='linear' )
        cmds.setKeyframe( objectName, time=(nT), attribute='translateY',
value=p.y, inTangentType='linear', outTangentType='linear' )
        cmds.setKeyframe( objectName, time=(nT), attribute='translateZ',
value=p.z, inTangentType='linear', outTangentType='linear' )
```

4. ¿Cómo usar el script?

Para poder usar los scripts debemos abrir el script con el editor de scripts de Maya y prepararnos para ejecutarlo seleccionando todo el código.

En el editor de escena debemos seleccionar primero el objeto a animar y después seleccionar los objetos que van a hacer de objetos de control en el orden que queramos que se sigan los puntos. Es necesario que se seleccionen exactamente los puntos de control necesarios.

Después de seleccionar los objetos podemos ejecutar el script y nos creará la animación.