**MaxScript**: permite interactuar con 3dsmax sin llegar a usar C++

Scripting > Listener

Hay

* **Enteros**
* **Strings**: “Juan” a=”juan” a.count =4
* Podemos **escapar** comillas “Juan dijo \”hola\” jeje”
* Y para escribir la barra?
* **Variables** a=2 a=3.4
* Las variables pueden **cambiar** el **tipo**
* Podemos **concatenar** **strings** con el +
* Podemos hacer un **cast** así de int a string con **2 as string**

Imagen que contiene Texto

Descripción generada automáticamente

* **Colores**: c1 = color 128 0 0 para acceder a los componentes c1.r c1.g c1.b
* **Vectores**: p1 = [1,2,10] para acceder al segundo componente p1.y
* Interfaz de usuario gráfica, Aplicación

  Descripción generada automáticamente con confianza mediaPodemos ver el tipo de una variable con **classof c1**.
* OJO con asignar vectores
  + Si hago p1=p2 se usa la misma dirección de memoria

por lo que si cambio una se cambia la otra

* + p2 = copy p1 crea una copia distinta
* Listas: v = #(1, “jonh”, 3, 4) y ojo
  + v[1] es el primero
  + v.count dice cuantos elementos
  + SI tengo b=2 puedo acceder a v[b]
  + Y v[b].count me da la longitud de la cadena jonh
  + Si hago v[10]=5 me crea elementos **undefined** hasta la pos 10.
  + Dan error
    - V[-1]
    - V[“jonh”]
    - V[2] = 2 (pq no se puede cambiar el tipo de jonh)
  + Bit array 🡪 {} (Array de 0s y 1s)
  + v[1] = undefined me lo puedo cargar
  + Arrays dentro de arrays

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

* + clearListener()

# OBJETOS

* + Variable $ es la selección actual. Si no tengo nada seleccionado undefined
  + Puedo guardarme esa selección con a=$
  + **Nombre**: a.name
  + Posición: A.pos o a.position
  + **Color**: a.wirecolor
  + **Material**: a.material
  + **Tipo**: Class of a
  + Como es una box puedo acceder a sus **parámetros**: A.length
    - Tamb se pueden modificar a.length=30

Podemos acceder an objeto con $Nombre $Cylinder001 coge el objeto q se llame así

* **$\*** es todos los objetos de la escena
* **$Cylin\*:** todos los objetos cuyo nombre empieza por cilindro
* **$TheCube/\*:** hijos del objeto que se llama TheCube
* **$.**children: hijos de la selección actual.
* **$.**children[1].pos = [0,0,0]: me llevo el primero al origen
* **$.**children[2].parent: obtiene el padre del segundo hijo

# Creo objetos

* Box(): crea un objeto de esa clase (de tipo caja)
* X = Box() crea una caja y lo almaceno para poder luego x.name=”Caja”
* Select x 🡪 selecciona la caja creada
* Delete x 🡪 elimina la caja
* Si llamamos ahora a x 🡪 <Deleted scene node>

**MacroRecorder**

Esto sirve porq todo lo que haces te dice como se traduce a MaxScript

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Y así puedes saber a lo mejor como se hace un UNDO, o cualquier cosa

Para rotar en el eje X 95 grados

Texto

Descripción generada automáticamente con confianza baja

**Showproperties $**

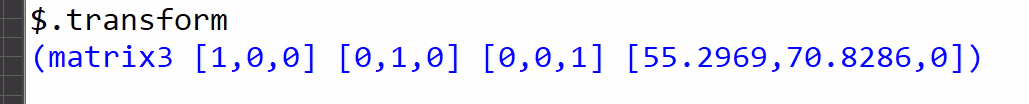
Te dice todas las propiedades que tiene el objeto que se pueden cambiar

**Le da igual las mayúsculas o minúsculas (CaseSensitive)**

Texto

Descripción generada automáticamente

**$.transform** para acceder a su posición, rotación, quaternion



# Scripts

Tools > Evaluate All para compilar y ejecutar

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Si la llamo

Imagen que contiene Texto

Descripción generada automáticamente

Se pone dos veces porq devuelve lo ultimo. Si pongo return “juan” pues ya no

Format

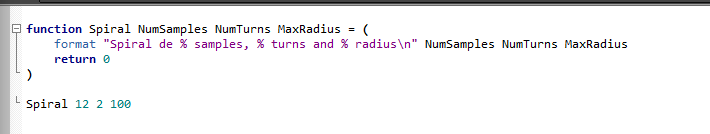


Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Si le pongo local a una variable será local

Ejercicio

Cojo la curva y cojo sus bounds y lo multiplico por un 20% para que se me haga un poco mas grande.

Imagen que contiene hombre, sostener, secadora

Descripción generada automáticamente

Ahora queremos generar arboles. Hay función pa números aleatorios.

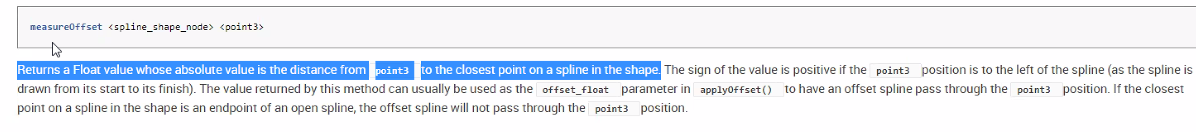
* Si esta cerca de la curva no lo quiero, dame otro punto
* Si un árbol esta cerca de otro árbol tampoco lo quiero
* Podríamos tener unos arboles mas grandes q otros

Como vemos si un punto esta cerca o no de otro?

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

measureOffset <spline> <point3>



Y para los bounds:

X = $.boundingbox

x.min y x.max

random 4 10 (entre el 4 y 10) y además con distintos tipos

Texto

Descripción generada automáticamente

Y lo importante: los vectores

Imagen que contiene Diagrama

Descripción generada automáticamente

Curva, saco bounds y primero rellenamos ese sitio de puntos

Luego la curva para no ponerle los puntos ahi

Como accedo a un modificador

Para hacer los bbox mas grandes

Texto

Descripción generada automáticamente

Pero mas facil

Texto

Descripción generada automáticamente

La clase que viene con UI y seguiremos desde aqui