SCRIPTING EN UNITY
VARIOS

Ramón Mollá
rmolla at dsic.upv.es - ext. 73549
Grupo de Informática Gráfica
Departamento de Sistemas Informáticos y Computación

Índice

C#
struct, vector, cadena,...
Técnicas

C#
Structs

Son similares a las clases

Son tipos de dato por valor

Pueden tener métodos

Unity3D los usa para vectores, atributos sencillos, …

```
struct Vector3 {
float x, y, z;
}
....
Vector3 v;
v.x = 3.0f; v.y = 1.0f; v.z = 0.0f;
```

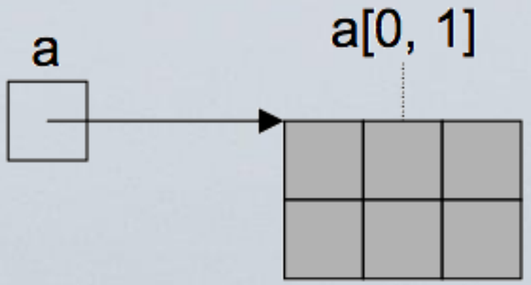
C#
Vectores

Rectangular (more compact, more efficient access)

fil x col

```
int[,] a = new int[2, 3];

int x = a[0, 1];
int len = a.Length;       // 6
len = a.GetLength(0); // 2
len = a.GetLength(1); // 3
```



C#
Cadenas

Una variable *char* puede almacenar cualquier carácter Unicode

Tiene muchos métodos: *char.IsDigit(…); char.IsLetter(…); char.Is Punctuation(…); char.ToUpper(…); char.ToLower(…), char.ToString(…);...*

Una variable de cadena es

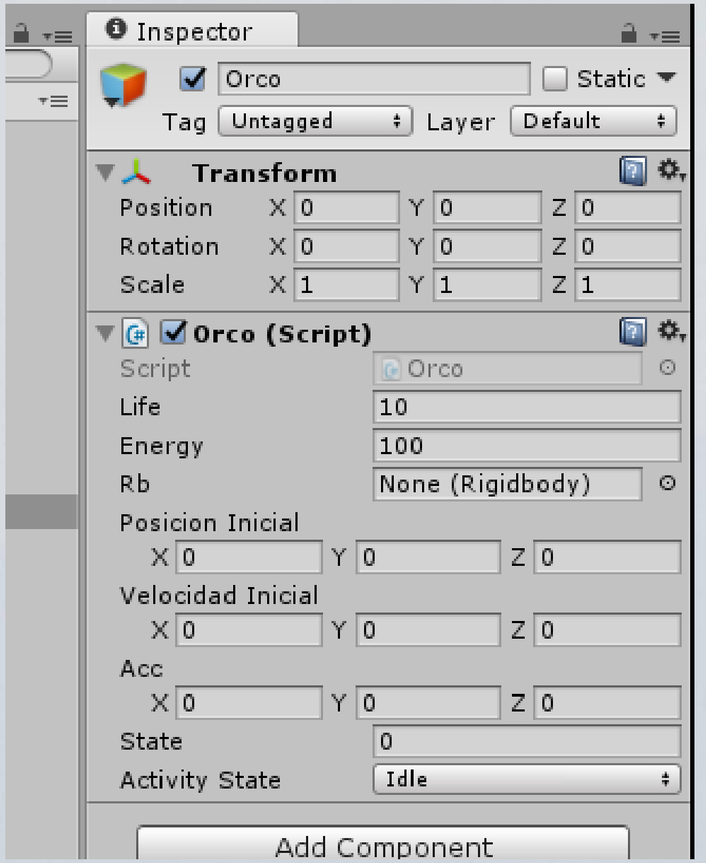
- Una colección de *chars*
- Es una referencia a dónde comienza la cadena
- Inmutable

Concatenación con operador “+” o *string.Concat()*

Operadores **==** y != están sobrecargados a *string.Equals()*

C#
Ejemplo vectores y cadenas

Sacar por pantalla el estado de la clase Orco




```
using UnityEngine;
using System.Collections;

public class Orco : BasicChar{
   public enum State { Idle,
   Running, Eating, Attacking,
   Sleeping, Hurt, Dead, MaxState}

    string[] stateName = { "Idle",
"Running", "Eating", "Attacking",
"Sleeping", "Hurt", "Dead" };

    public State activityState =
State.Idle;

    float time;

    void Start() {
        init();
        time = Time.time;
    }
    void Update(){
        if (Time.time - time > 1.0f){
            time = Time.time;
            activityState++;
            if (activityState >= State.MaxState)
                activityState = State.Idle;
        }
    }
    void OnGUI() {
        GUI.Label(new Rect(20, 50, 100, 20),
      stateName[(int)activityState]);
    }
}
```

C#
Parámetros

Value Parameters (input values)

```
void Inc(int x) {x = x + 1;}
void f() {
    int val = 3;
    Inc(val); // val == 3
}
```

- "call by value"
- formal parameter is a copy of the actual parameter
- actual parameter is an expression

ref Parameters (transition values)

```
void Inc(ref int x) { x = x + 1; }
void f() {
    int val = 3;
    Inc(ref val); // val == 4
}
```

- "call by reference"
- formal parameter is an alias for the actual parameter
  (address of actual parameter is passed)
- actual parameter must be a variable

out Parameters (output values)

```
void Read (out int first, out int next) {
    first = Console.Read(); next = Console.Read();
}
void f() {
    int first, next;
    Read(out first, out next);
}
```

- similar to ref parameters
  but no value is passed by the caller.
- must not be used in the method before it got a value.

C#
Boxing/Unboxing

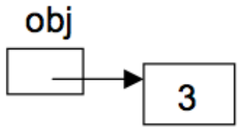
Value types (int, struct, enum) are also compatible with *object*!

Boxing

The assignment

```
object obj = 3;
```

wraps up the value 3 into a heap object



Unboxing

The assignment

```
int x = (int) obj;
```

unwraps the value again

C#
Otras (I)

Tipado dinámico
Tipo de variables se define al vuelo en tiempo de ejecución

Var

```
var i = 100;  // i es del tipo int
var menu = new []{“Open”, “Close”, “Windows”};
```

Nullables

```
int? i = null;
```

C#
Otras (II)

Parámetros opcionales

```
void optMethod(int first, double second = 0.0, string third = “hello”)
{ ... }

// Possibilities
optMethod(99, 123.45, “World);
optMethod(99);
optMethod(first:99, third:”World”);
```

C#
Otras (III)

Sobrecarga de operadores aritméticos +, -, *, /

El polimorfismo no es por defecto (virtual, override, new)

Propiedades -> getter/setters

```
private string name; // private
public string Name   // public
{
  get {return name;}
  set {name = value;}
}
```

```
public class Date
{
    private int month = 7; // Backing store
    public int Month
    {
        get { return month; }
        set { if ((value > 0) && (value < 13))
                { month = value; }
            }
    }
}
```

```
Date d;
d.Month = 12; //the set accessor is invoked here

//the get accessor is invoked here
System.Console.Write(person.Name);
```

C#
Ayuda

- Se encuentra en la opción de menú Help
- Desde *MonoDevelop* también se puede tener acceso a la ayuda seleccionando la clase o método y pulsando la combinación de teclas Ctrl (Cmd en Mac) + ‘
  - En la sección *Reference* o *Scripting* probar la opción *Search*
  - Explorar la clase *MonoBehaviour*
  - Familiarizarse con qué métodos y atributos se pueden heredar de él

Bibliografía

Learning C# by Developing Games with Unity 3D Beginner's Guide. Terry Norton. Packt Publishing. ISBN 978-1-84969-658-6

Manuales en línea de Unity 3D

creative commons

COMMONS DEED

Documentación generada por
Dr. Ramón Mollá Vayá
Sección de Informática Gráfica
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia