Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Informática

Interfaces Inteligentes
Introducción

Práctica 2

Isabel Sánchez Berriel

4º curso del Grado en Ingeniería Informática. Itinerario de Computación

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Informática

Contenidos

- Introducción a C#
- Scripts
- Físicas

C#

- C# lenguaje de programación orientado a objetos
- Se ejecuta en una máquina virtual llamada .Net Framework
- Sintaxis similar a la de Java.
- *Se compila a un lenguaje intermedio (IL)*
- *Si se cumplen las directivas de seguridad CLR (Common Language Runtime) lo convierte a instrucciones nativas de la máquina.*

Universidad de La Laguna
Escuela Técnica Superior de Ingeniería Informática

Tipos básicos

C#	Tipo en System	Características
byte	System.Byte	entero, 1 byte sin signo
short	System.Short	entero, 2 bytes con signo
ushort	System.UShort	entero, 2 bytes sin signo
int	System.Int32	entero, 4 bytes con signo
long	System.Int64	entero, 8 bytes con signo
ulong	System.ULong64	entero, 8 bytes sin signo
float	System.Single	real, IEEE 754, 32 bits
double	System.Double	real, IEEE 754, 64 bits
decimal	System.Decimal	real, 128 bits (28 dígitos significativos)
bool	System.Boolean	(Verdad/Falso) 1 byte
char	System.Char	Carácter Unicode, 2 bytes
string	System.String	Cadenas de caracteres Unicode
object	System.Object	Cualquier objeto (ningún tipo concreto)

Universidad de La Laguna
Escuela Técnica Superior de Ingeniería Informática

Arrays

- tipo[] nombre = new tipo[TAM]
- tipo[,,] nombre = new tipo[TAM1, TAM2, TAM3]
- tipo[][][] nombre = new tipo[TAM][][]

```
int[] numeros = new int[8] {0, 1, 2, 3, 4, 5, 6, 7};
int[,] array2x2 = new int[2,2] {{11, 12}, {21, 22}};
int[][] array2xn = new int[2][] {new int[] {11, 12},
                                 new int[] {21, 22, 23}};
```

Universidad de La Laguna
Escuela Técnica Superior de Ingeniería Informática

Sentencias de control

```
if (condición)
{
   instrucciones
}
else
{
   instrucciones
}
```

```
switch (expresion)
{
  case constante 1:
   instruciones

  case constante 2:
    instrucciones

  default;
   instrucciones
  break;
}
```

```
while(condición)
{
   instrucciones
}
```

```
do{
   instrucciones
} while(condición)
```

```
for(inicio;condición;siguiente valor)
{
   instrucciones
}
```

```
foreach(tipo nombre in coleccion)
{
   instrucciones
}
```

Universidad de La Laguna
Escuela Técnica Superior de Ingeniería Informática

Varios

- Por defecto los parámetros se pasan por valor.
- Si se necesita pasarlo por referencia hay que especificarlo usando la palabra reservada **`ref`**

```
static void Prueba(ref int i) {….}
Prueba(ref x);
```

- En C# todo son clases.
- Se organizan mediante espacios de nombres
- Se incluye un espacio de nombres mediante la palabra reservada **`using`**
- Punto de entrada: *`public static void Main()`*

Universidad de La Laguna
Escuela Técnica Superior de Ingeniería Informática

Espacios de nombres básicos

- **System**
- **System.Collections**
- **System.Collections.Generic**
- **System.IO**
- **System.Text**
- **System.Threading**

- https://msdn.microsoft.com/es-es/library/system%28v=vs.110%29.aspx

Universidad de La Laguna
Escuela Técnica Superior de Ingeniería Informática

Script

- Mediante los scripts podemos crear o modificar Components:
  - Activar o desactivar eventos
  - Modificar propiedades a lo largo del tiempo
  - Responder a entradas del usuario.
- Crear script: *Assets→ Create → C# Script*
  - Se debe asignar el nombre del fichero desde el primer momento ya que dará el nombre a la clase.
- Hereda de *MonoBehaviour*

Script

- Los constructores se manejan desde el editor y no al inicio del juego.
- Un script no tiene efecto a menos que se haya asociado a un GameObject:
  - Arrastrarlo sobre el GameObject
  - Seleccionar en el menú Component
- Las propiedades `public` se pueden cambiar desde el inspector.

Scripts

```
using UnityEngine;
using System.Collections;

public class SampleScript : MonoBehaviour {
// Use this for initialization
void Start () {
   // Se ejecuta si la instancia del script está habilitada, una sola
vez
    Instantiate(player);
}
// Update is called once per frame
void Update () {
   //Se ejecuta en cada iteración del bucle de juego
}
}
```

Scripts

- Acceso a otros componentes:
  - *Los componentes son instancias de clases: acceso a la referencia*
  - *GetComponent<Tipo>()*
  - *Si el GameObject no dispone de ese componente GetComponent devuelve null.*
  - *Asignar objetos a variables: Arrastrarlos a la variable*
  - *Asignar un GameObject con componente de ese Tipo a la variable*
- Existen casos en que se aconseja manejar grupos de objetos haciéndolos hijos de un objeto padre.

Acceso a Components

Obtener una referencia al component:

```
Transform tf = GetComponent<Transform>();
Rigidbody rb = GetComponent<Rigidbody>();
Renderer rd = GetComponent<Renderer>();
```

Manipular el component:

```
rb.AddForce(Vector3.forward * moveForce)
tf.Translate(Vector3.forward * Time.deltaTime);
rd.material.color = Color.green;
```

Acceso a Components

Añadir un objeto al component al que queremos conectar.

```
public GameObject player;
```

Arrastrar un objeto del tipo del que necesitamos a esa propiedad.

```
transform.position = player.transform.position - Vector3.forward * 10f;
```

Scripts

- Buscar objetos:
  - *Find("Name")*
  - *FindWithTag("Tag")*
  - *FindGameObjectsWithTag("Tag")*
- Instanciar Objetos:
  - *Instatiate*
- Eliminar Objetos:
  - *Destroy()*
- Vectores:
  - *Vector3:*
    - Numerosas utilidades: `Distance, Lerp, Max, Angle,...`

Universidad de La Laguna
Escuela Técnica Superior de Ingeniería Informática

Scripts

- Eventos:
- *Start(): Ocurre justo antes del primer frame del objeto*
- *Update(): Ocurre antes de ser renderizado cada frame*
- *FixedUpdate(): Ocurre antes de cada actualización de la física*
- *Eventos de la interfaz: OnGui(), OnMouseOver(), OnMouseDown()*
- *Eventos de física: OnCollisionEnter(), OnCollisionStay(), OnCollisionExit()*
- *Colisión, pero no hay reacción física: OnTriggerEnter(), OnTriggerStay(), OnTriggerExit()*
- http://docs.unity3d.com/es/current/Manual/ExecutionOrder.html

Scripts

- Input: Objeto que da acceso a la entrada
- En Project settings accedemos al Input Manager para configurar la entrada
  - *Teclas de disparo, salto, etc.*
  - *Input.GetButton*
  - *Input.GetAxis*
  - *Input.GetMouseButton*
  - *Input.GetKey*

Scripts

- Ejes virtuales:
  - *Ejes imaginarios asociados a controladores de entrada.*
  - *Ejemplo: “Horizontal”, “Vertical”, …*
  - *El usuario puede definir ejes virtuales*
  - `Input.GetAxis ("Nombre_eje"):` *Devuelve un valor [-1, 1] cuando se acciona un control mapeado a ese eje.*

```
float horizontalInput = Input.GetAxis("Horizontal");
new Vector3(horizontalInput*velocity, 0, 0);
```

Scripts

- El tiempo entre frames no es constante, afecta a la animación:
  - *Time.deltaTime para escalar los movimientos según el tiempo transcurrido*
  - *Time.fixedTime El motor de física funciona con un paso de tiempo fijo para garantizar la coherencia de la simulación.*
- Corrutinas:
  - *Fragmentos de código cuya ejecución se puede interrumpir y retomar en el punto que se dejó posteriormente*

Física

- Fuerzas rotacionales y posicionales que afectan al sistema de referencia del objeto:
  - *Gravedad, Fricción, Momento, Colisiones con otros objetos.*
- Motor NVIDIA Physx: Mecánica newtoniana
- Accesible a través de la API de Unity

Física

- El comportamiento físico se logra con el Component *Rigidbody*
- *Rigidbody* hace que el objeto responda a la gravedad.
- Si agregamos *Rigidbody* **no se debe usar Transform**, los desplazamientos se realizan por el motor de física.
  - *Si queremos que un Rigidbody no sea movido por el motor de física activamos IsKinematic*

Física

- *Collider* Forma del objeto que se considera de cara a las colisiones
- Por lo general simplificación de la forma exacta por eficiencia
- Tipos
  - *Primitivos: Cubo, esfera, cápsula*
  - *Compound: Combinaciones primitivas*
  - *Mesh collider:* Ajuste a la malla (No se debe usar, en todo caso para la geometría de escena*, Convex)*
  -

Física

- Component *Collider* responde a las colisiones entrantes.
  - *BoxCollider, SphereCollider, CapsuleCollider.*
  - *BoxCollider2D, CircleCollider2D*
  - *Usar static colliders para elementos sin movimiento.*
- Se puede seleccionar el material del Collider mediante el valor que se asigne a la fricción:
  - *Hielo resbala (0), caucho tiene una fricción alta, etc.*

Física

- *Static Collider* Para objetos que nunca se mueven.
  - *No llevan Rigidbody*
  - *Colisionan con objetos Rigidbody, pero no los mueven*
  - *No rotar, no escalar*
  - *Pisos, paredes, etc.*

- *Dynamic Collider*:
  - *Rigidbody Collider*
  - *Kinematic Rigidbody Collider* (Activar isKinematic)

Física

- Colisiones:
  - *No-trigger: al menos uno de los objetos debe ser no Kinematic*
  - *Si ambos son Kinematic OnCollisionEnter no se llama*
  - *En colisiones trigger la restricción anterior no se aplica.*
  - *Static: Los objetos Rigidbody colisionan pero el static collider no se mueve.*
    - No se debe alterar en tiempo de ejecución la propiedad
  - *Rigidbody Collider: Se mueve por física y reaccionan a colisiones y fuerzas.*
  - *Kinematic Rigidbody: Se mueve con Transform liberándolo del motor de física.*

  Matriz de colisiones

Física

- *Joints*: Uniones de objetos físicos o de objetos físicos a un punto fijo.
  - *Hinge Joint*: Articulaciones
  - *Spring Joint*: Elasticidad

- Se pueden romper cuando se supera un umbral de fuerza.

Física

- Controladores de Personaje: Se simula la física de un personaje pero sin tener en cuenta el momento
  - *Collider de cápsula vertical*
  - *No se requiere Rigidbody*
  - *Dispone de métodos especiales para la velocidad y dirección del objeto*
  - *No puede caminar atravesar suelos ni paredes.*
  - *Puede empujar objetos Rigidbody por scripting, pero no es acelerado por colisiones.*

```
OnControlerColliderHint()
```