# Unity Data Type

## Enumerations

Es un tipo de dato. Almacena varios valores en el para que sea mas ordenado y mas fácil el acceso a ellos.

Declaracion: enum name : type{value = value, value = value}

Ej: enum Direction: int{Norte = 1, Este = 2, Sur=3, West=4}

Se accede a los datos de esta manera:

Name variablename = name.value

Ej: Direction myDir = Direction.Norte

## Switch

Alternativa a los if else. Se compara una variable y segun su valor se ejecuta un codigo.

Declaracion:

switch(variable a evaluar)

{

case constante:

ejecución

break;

default:

ejecución

break;

}

Ejemplo:

switch(dia)

{

case 1:

Debug.Log(“Buenas Noches”)

break;

case 2:

Debug.Log(“Buenos días”)

break;

}

## Diccionarios y Listas

Parecidos a los arrays pero con unas pocas diferencias.

Las listas son arrays dinámicos, no necesitas saber cuantos elementos vas a tener.

Para incluir las listas se tiene que agregar la colección System.Collections.Generic.

using System.Collections.Generic;

Para declararla se usa List<type> name = new List<type>()

List<Item> items = new List<Item>()

Para añadir elementos a la lista se usa el metodo Add y Insert.

items.Add(item): Añade un elemento en la ultima posición.

items.Insert(index, item): Añade el elemento en la posición indicada.

Se puede acceder igual que el array usando el index.

items[0]

Para averiguar el tamaño se usa el atributo count.

items.count

Para eliminar un elemento se usa:

items.RemoveAll: (elimina todos los elementos de la lista)

items.RemoveAt(index): elimina el elemento por índice.

items.Remove(item): elimina el objecto indicado.

Para ordenar los elementos se usa Sort siempre y cuando el objeto tenga implementada la interfaz IComparable<Class>. Está interfaz se hereda del System.

using System;

public BadGuy : IComparable<BadGuy>{

public int CompareTo(BadGuy other){

“hacer la lógica el programador. Si el parametro es mayor al objeto, devuelve negativo. Si el parámetro es menor al objeto devuelve positivo. Si son iguales se devuelve zero.”

return power- other.power;

}

}

Ahora en la otra clase se usa el Short.

items.Short();

Para resetear la lista se usa el método Clear.

items.Clear();

Los Diccionarios son parecidos a las listas. La diferencia es que tiene dos valores, clave y valor.

Para declararlo se hace de la siguiente manera. Dictionary<key\_type, type> name = new Dictionary <key\_type, type>()

Dictionary<string, Item> items = new Dictionary <string, Item>()

Para añadir elementos se usa Add(clave, valor)

items.Add(“Espada”, item)

Para acceder a ellos se usa la clave.

items[“Espada”]

Si la llave no existe saltará una excepción. Se puede usar la siguiente comprobación.

if (items.TryGetValue(“Espada”, out temp)){

//existe

}else{

//no existe

}