Dado Doble : Solución a la inserción de un nuevo dado.

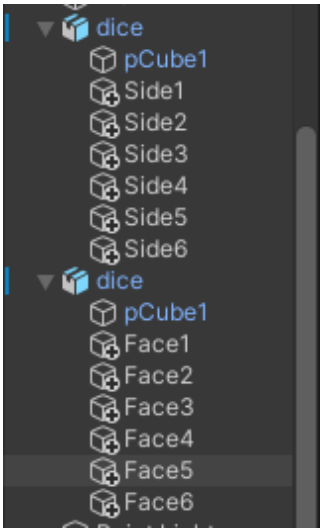
Ing. Luis Felipe Narvaez Gomez. E-mail: luis.narvaez@usantoto.edu.co. Cod: 2312660. Faculta de Ingeniería de Sistemas.

EL PROBLEMA

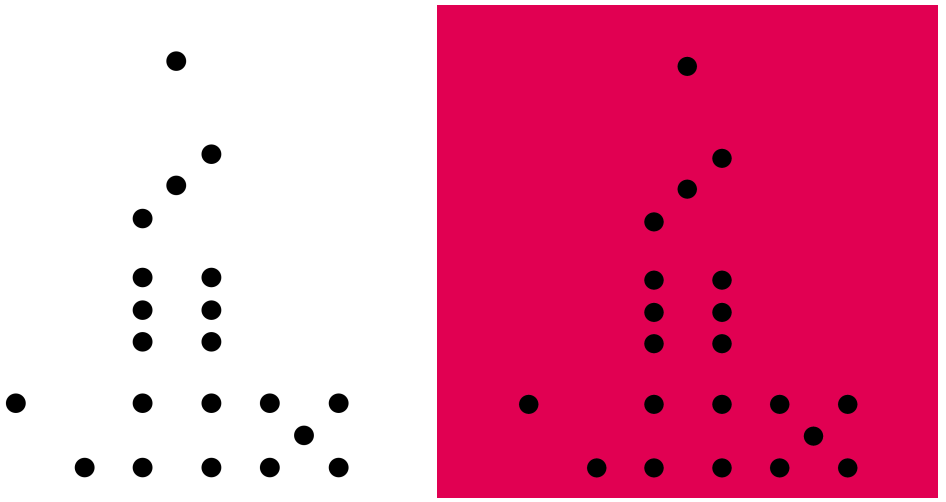
El proyecto que veníamos trabajando ya hemos dejado funcionando en Unity el lanzamiento de un solo dado. Cada que este nuevo dado era lanzado en la caja que diseñamos mostraba por pantalla el valor de la cara superior que teníamos, esto a cuestiones practicas es la sumatoria de la tirada en total.
Ahora bien, debemos e agregar un segundo dado, el cual debe comportarse como el primero pero debe mostrar su resultado independientemente del primer dado, añadido a esto debemos mostrar la sumatoria total de la tirada en pantalla, es decir dado 1 mas el dado 2.
Para esto la solución que propongo es la siguiente:

AGREGAR UN NUEVO DADO

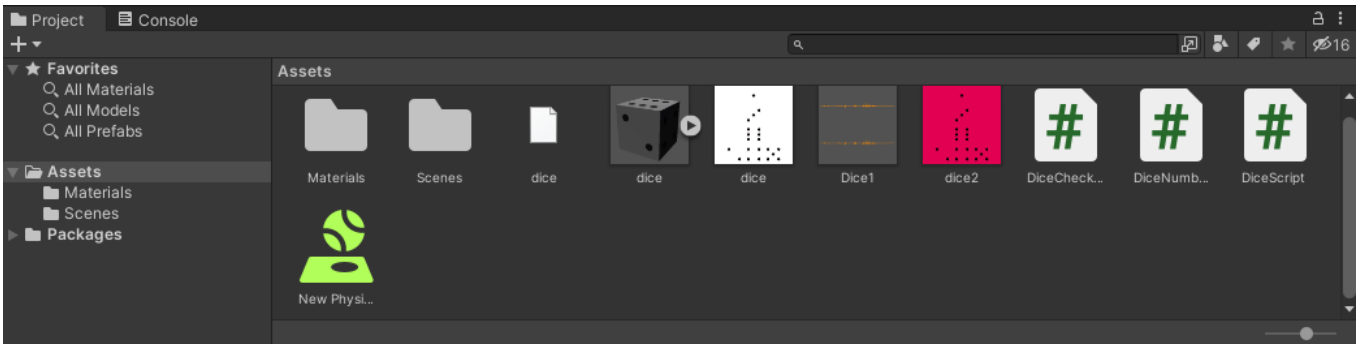
Lo primero es tener una dupla exacta del primer dado. Sin embargo cambiaremos el nombre de las caras del dado a las del dado 2.



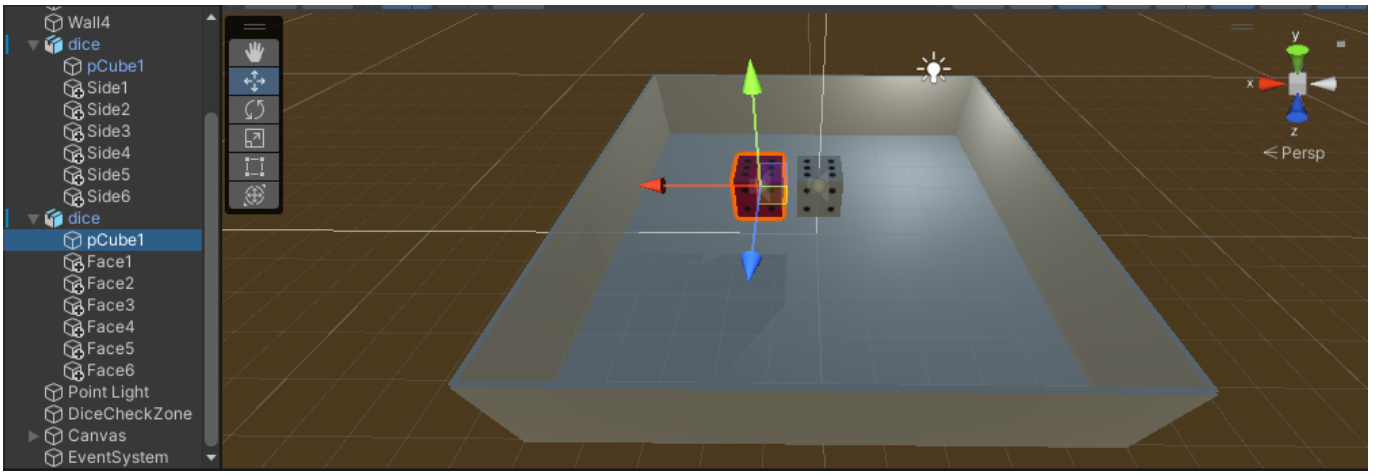
Debemos recordar que este "juego" trabaja en su comportamiento en base a scripts de C#, en ellos las variables con los nombres de las caras "Side" ya están definidas y en caso de tener, por ejemplo, dos "Side1" confundirá al programa, leyendo como resultado a veces el Side1 del dado 1 y en otras ocasiones el Side1 del dado2.
Por este motivo, es mejor tener variables únicas de las caras por ambos dados.
Añadido a esto, para diferenciar los dados entre si, al segundo dado se añade un nuevo color, esto se logra transformando la imagen de "dice.png" en algún programa de edición de imagen.
La imagen original es la de la izquierda , mientras que la imagen transformada para el nuevo dado es en mi caso la de la derecha:



Esta imagen se agregara a la carpeta de "Assets" del menú de "Project".



Una vez allí solo debe arrastrarse al segundo dado dentro de su subcomponente denominado "pCube1".



UNA NUEVA VARIABLE PARA UN NUEVO DADO

Este "Juego" muestra por pantalla el resultado que cae del dado 1 mediante el archivo "DiceNumberTextScript" siendo un parámetro que se envía desde "DiceCheckZoneScript". La lógica es de la siguiente manera:

1. DiceCheckZoneScript detecta que a caído un dado.
2. DiceCheckZoneScript mira entre sus posibilidades que cara del dado a quedado contra el piso.

3. DiceCheckZoneScript reconoce la cara contra al piso como una de sus posibilidades predefinidas, esto mediante el nombre de esta cara "Side#".
4. Dentro de cada posibilidad se examina que, el valor que se debe mostrar es el opuesto al de la cara contra el suelo.
5. DiceCheckZoneScript envía el valor de la cara opuesta del dado, la cara superior. Esto lo logra dando un valor a una variable que corresponde a DiceNumberTextScript.
6. DiceNumberTextScript recibe el valor de esta variable en INT.
7. DiceNumberTextScript actualiza el valor de esta variable y la convierte para visualizarla a un STRING con ayuda del método "ToString()".
8. El resultado se muestra en pantalla.

Siguiendo la lógica ya explicada, primero debemos crear una variable en DiceNumberTextScript que se corresponda a los valores que nos llegaran de DiceCheckZoneScript pero para el dado 2. Abrimos DiceNumberTextScript y escribimos de forma consecutiva a las variables:

```
public static int diceNumberTwo;
```

Ahora necesitamos una variables que se encargue de la sumatoria de ambos dados, es decir la sumatoria de los valores que llegaran a las variables del dado 1 y del dado 2:

```
public static int result;
```

Solo queda mostrar y hacer el calculo en el método de Actualizar, por lo que dentro del método "void Update()" tendremos para calcular el resultado:

```
result = diceNumber + diceNumberTwo;
```

Y para mostrar todo tendremos:

```
text.text = "D1 ["+diceNumber.ToString ()+"]" + "    D2 ["+diceNumberTwo.ToString()+"]" + "    Result:
"+result.ToString();
```

De esta manera tenemos un código al completo de la siguiente forma:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class DiceNumberTextScript : MonoBehaviour {
        Text text;
        public static int diceNumber;
        public static int diceNumberTwo; //new
        public static int result;
        // Use this for initialization
        void Start () {
                text = GetComponent<Text> ();
        }
        // Update is called once per frame
        void Update () {
                result = diceNumber + diceNumberTwo;
                text.text = "D1 ["+diceNumber.ToString ()+"]" + "    D2
["+diceNumberTwo.ToString()+"]" + "    Result: "+result.ToString();
        }
```

```
}
```

NUEVAS CARAS, NUEVAS POSIBILIDADES

Como ya explicamos en el punto anterior, en el código de DiceCheckZoneScript asignamos un valor a la variable del dado 1 "diceNumber" de DiceNumberTextScript según corresponda el caso de la cara contra el suelo. Por ejemplo, para en caso de que la cara 1 "Side1" este contra el suelo, el valor de la variable será 6, la cara opuesta a la cara contra el suelo o la cara superior visible del dado lanzado.

```
case "Side1":
        DiceNumberTextScript.diceNumber = 6;
        break;
```

Ahora bien, para el dado dos tenemos una serie de nuevas caras (Face1, Face2, Face3 , Face4, Face5 y Face6), estas caras deben tener sus propios casos para poder ser leídas tal y como funciona en el dado 1, sin embargo dentro de la decisión de valor, no utilizaremos la variable designada para el dado 1 "diceNumber" si no la nueva variable que creamos en el punto anterior "diceNumberTwo".
De esta forma para el dado uno tendríamos los siguientes casos:

```
case "Side1":
        DiceNumberTextScript.diceNumber = 6;
        break;
case "Side2":
        DiceNumberTextScript.diceNumber = 5;
        break;
case "Side3":
        DiceNumberTextScript.diceNumber = 4;
        break;
case "Side4":
        DiceNumberTextScript.diceNumber = 3;
        break;
case "Side5":
        DiceNumberTextScript.diceNumber = 2;
        break;
case "Side6":
        DiceNumberTextScript.diceNumber = 1;
        break;
```

Y para el segundo dado tendríamos los siguientes casos o posibilidades contempladas:

```
case "Face1":
        DiceNumberTextScript.diceNumberTwo = 6;
        break;
case "Face2":
        DiceNumberTextScript.diceNumberTwo = 5;
        break;
case "Face3":
        DiceNumberTextScript.diceNumberTwo = 4;
```

```
        break;
case "Face4":
        DiceNumberTextScript.diceNumberTwo = 3;
        break;
case "Face5":
        DiceNumberTextScript.diceNumberTwo = 2;
        break;
case "Face6":
        DiceNumberTextScript.diceNumberTwo = 1;
        break;
```

De esta manera tendremos un caso para cada cara del dado 1 y un caso para cada cara del dado 2. El código completo seria entonces:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class DiceCheckZoneScript : MonoBehaviour {
        Vector3 diceVelocity;
        // Update is called once per frame
        void FixedUpdate () {
                diceVelocity = DiceScript.diceVelocity;
        }
        void OnTriggerStay(Collider col)
        {
                if (diceVelocity.x == 0f && diceVelocity.y == 0f && diceVelocity.z == 0f)
                {
                        switch (col.gameObject.name) {
                        case "Side1":
                                DiceNumberTextScript.diceNumber = 6;
                                break;
                        case "Side2":
                                DiceNumberTextScript.diceNumber = 5;
                                break;
                        case "Side3":
                                DiceNumberTextScript.diceNumber = 4;
                                break;
                        case "Side4":
                                DiceNumberTextScript.diceNumber = 3;
                                break;
                        case "Side5":
                                DiceNumberTextScript.diceNumber = 2;
                                break;
                        case "Side6":
                                DiceNumberTextScript.diceNumber = 1;
                                break;
                        // new 6 case
                        case "Face1":
                                DiceNumberTextScript.diceNumberTwo = 6;
                                break;
                        case "Face2":
                                DiceNumberTextScript.diceNumberTwo = 5;
                                break;
                        case "Face3":
                                DiceNumberTextScript.diceNumberTwo = 4;
                                break;
```

```
                        case "Face4":
                                DiceNumberTextScript.diceNumberTwo = 3;
                                break;
                        case "Face5":
                                DiceNumberTextScript.diceNumberTwo = 2;
                                break;
                        case "Face6":
                                DiceNumberTextScript.diceNumberTwo = 1;
                                break;
                        }
                }
        }
}
```

Y listo, una vez salvado los cambios en cada uno de los códigos y compilado automáticamente dentro de Unity, tendriamos el siguiente resultado.