# Tarea : Introducción a la programacion de gráficos 3D.


- Autores : Alien Embarec Riadi, Antonio Chávez López


1. Qué funciones se pueden usar en los scripts de Unity para llevar a cabo traslaciones, rotaciones y escalados.

Podemos hacer cualquiera de estas operaciones invocándolas sobre el objeto *Transform*. Podemos trasladar(mover de lugar) un objeto en cualquiera de sus tres ejes contamos con la propiedad *Translate*, por ejemplo:

```
Vector3 movement = new Vector3(3,0.0f,2);

tf.Translate(movement * Time.deltaTime * speed);
```

En *movement* almacenamos la nueva posición, *deltaTime* calcula el tiempo necesario para alcanzar la nueva posición, y *speed* es la velocidad a la que nos movemos.

Para rotar un objeto:

```
float speed = 50.0f;
void Start (){
}

void Update(){
    transform.Rotate(Vector3.up * speed * Time.deltaTime);
}
```

Y por último, para los escalados, se usa el atributo *localScale*, que sirve para modificar las dimensiones de un objeto.

```
using UnityEngine;

public class ExampleClass : MonoBehaviour
{
    public float x = 0.1f;
    public float y = 0.1f;
    public float z = 0.1f;
    void Update()
    {
        // Widen the object by x, y, and z values
        transform.localScale += new Vector3(x, y, z);
    }
}
```

2. ¿Cómo duplicarías el tamaño de un objeto en un script? Para duplicar el tamaño de un objeto contamos con la propiedad *localScale*, mencionada en la pregunta anterior. Transform ti = GetComponent(); transform.localScale += transform.localScale;

3. ¿Cómo situarías un objeto en la posición (3,5,1)?

Crearía un vector de tipo *Vector3*, a este vector le indicaría la posición en la que quiero situar el objeto, y posteriormente haría uso de la función *Translate* de la clase *Transform* para moverlo, de la siguiente manera:

```
Transform ti = GetComponent<Transform>();

ti.localPosition = new Vector3(3,5,1);
```

4. ¿Cómo trasladarías 3 metros en cada uno de los ejes y luego lo rotas 30º alrededor del eje Y?

ti.Transform(new Vector3(3,3,3)); ti.Rotate(0, 30, 0);

Para trasladar, tomando cada paso de un metro, aumentaría en tres unidades cada uno de los ejes.La velocidad a la que se mueve el objeto es constante, y puede ser un atributo privado predefinido o público indicado por el usuario.

5. Como rotarías un objeto sobre el eje (1,1,1).

```
ti.Rotate(new Vector3(30,30,30));
```

6. Rota un objeto alrededor del eje Y 30ª y desplázalo 3 metros en cada uno de los ejes. ¿Obtendrías el mismo resultado que en 4?

No, no se obtendría el mismo resultado, el desplazamiento no sería el mismo dependiendo de si se cambia de posición antes o después de rotar.

```
ti.Rotate(0, 30, 0);
ti.Transform(new Vector3(3,3,3));
```

7. ¿Cómo puedes obtener la distancia al plano cerca del volumen de vista?

```
Camera cam = GetComponent<Camera>();
cam.nearClipPlane;
```

8. Cómo puedes realizar la proyección al espacio 2D.

```
camera.WorldToViewportPoint(point);
```

9. Investiga qué son los *quaternion*.

Los cuaterniones son una extensión de los números reales, similar a la de los números complejos. Mientras que los números complejos son una extensión de los reales por la adición de la unidad imaginaria i, tal que $i^2$=-1 = -1, los cuaterniones son una extensión generada de manera análoga añadiendo las unidades imaginarias i, j y k a los números reales tal que:

```
i<sup>2</sup> = j<sup>2</sup> = k<sup>2</sup> = ijk =-1
```

10. Analiza la documentación de la cámara en Unity 3D e identifica los conceptos explicados respecto a la cámara.

Los conceptos importantes extraidos son:

Una cámara es un objeto que define una vista en una escena.

Una cámara en el mundo real, o incluso el ojo humano, ven el mundo de una forma que hace que los objetos se vean más pequeños cuanto más alejados están del punto de vista.

Una cámara que no disminuye el tamaño de los objetos con la distancia es conocida como ortográfica y las cámaras de Unity también pueden ser de este tipo. Los modos de ver una escena tanto en perspectiva como ortográfica son conocidos como proyecciones de la cámara.

El volumen de visualización de una cámara ortográfica está definido por una caja rectangular que se extiende entre los dos planos de delimitación.

El volumen de visión de una cámara de perspectiva no es una caja sino una forma piramidal con el vértice en la posición de la cámara y la base en el plano de delimitación lejano.

Un skybox se comporta como una “caja” llena con imágenes de un cielo. La cámara se coloca en el centro de esta caja y puede ver el cielo en todas las direcciones. La cámara ve un área diferente del cielo a medida que gira pero nunca se mueve del centro (por lo que la cámara no puede “acercarse” al cielo).

11. ¿Cómo puedes averiguar la matriz de proyección que se ha usado para proyectar la escena al último frame renderizado?.

```
cam = GetComponent<Camera>();
originalProjection = cam.projectionMatrix;
```

12. ¿Cómo puedes obtener la matriz de transformación entre el sistema de coordenadas local y el mundial?.

```
Matrix4x4 mtr = tr.worldToLocalMatrix;
```

13. ¿Como puedes calcular las coordenadas del sistema de referencia de un objeto con las siguientes propiedades del Transform:?: Position (3, 1, 1), Rotation (45, 0, 45)

```
Transform.position = new Vector3(3, 1, 1);
Transform.Rotate(45, 0, 45);
```