**Vectores y movimiento en línea recta**

Crear un script que mueva el objeto hacia un punto fijo que se marque como objetivo. El objetivo debe ser una variable pública, de esta forma conseguimos manipularla en el inspector y ver el efecto de distintos valores en las coordenadas:

* **Se ha realizado con el script MoveToPoint del cilindro, en el configuramos la coordenada a la que vamos y realiza el movimiento.**

1. Configurar la coordenada Y del Objetivo en 0

* Poner al Objetivo una coordenada Y distinta de cero.
* Cómo modificarías el script para que el objeto despegue del suelo y vuele como un avión
* **Se ha realizado con el script MoveMultiPoint del cilindro. Se han definido 3 puntos, el primero respetando ejes y,z para que haga un movimiento sobre la superficie, después un segundo punto para elevarlo en eje y, y finalmente un tercer punto para que gire totalmente.**

1. Duplicar los valores de X, Y, Z del Objetivo. ¿Es consistente el movimiento?.

* El Objetivo no es un objetivo propiamente dicho, sino una **dirección** en la que queremos movernos.
* La información relevante de un vector es la dirección. Los vectores normalizados, conservan la misma dirección pero su escala no afecta al movimiento. Esto se consigue con el método normalized de la clase Vector3: **this.transform.Translate(goal.normalized);**
* Con el vector normalizado, lo podemos multiplicar por un valor de velocidad para determinar cómo de rápido va el personaje. **public float speed = 0.1f** **this.transform.Translate(goal.normalized\*speed)**
* A pesar de que esas velocidades puedan parecer ahora que son consistentes, no lo son, porque dependen de la velocidad a la que se produzca el **update**.El tiempo entre dos updates no es necesariamente siempre el mismo, con lo que se pueden tener inconsistencias en la velocidad, y a pesar de que en aplicaciones con poca complejidad no lo notemos, se debe usar: **this.transform.Translate(goal.normalized \* speed\*Time.deltaTime);** para suavizar el movimiento ya que Time.deltaTime es el tiempo que ha pasado desde el último frame.

Position. Frame n

Position. Frame n +1

e = v \* t

*t = Time.deltaTime*

*v = speed*

*e =*

*Position Frame n + 1 = Position Frame n + speed \* Time deltaTime*

*Traslación del punto una cantidad equivalente a e*

* **Se ha realizado el script Duplicate con el que dados dos puntos, generamos una dirección en la que movernos, y las indicaciones mostradas anteriormente**

1. En lugar de seguir usando una dirección como objetivo, vamos a movernos ahora hacia una **posición** objetivo.

* Hacemos el objetivo una variable pública **public Transform goal** y añadimos un **public float speed = 1.0f**.
* La dirección en la que nos tenemos que mover viene determinada por la diferencia entre la posición del objetivo y nuestra posición:

**Vector3 direction = goal.position - this.transform.position;**

**this.transform.Translate(direction.normalized \* speed \* Time.deltaTime)**

* **Se ha realizado el script MoveToPosition en el cilindro que tiene como meta (goal) el cubo, y se ha definido la velocidad.**

1. Girar hacia el objetivo. En este caso, se dispone el método LookAt de la clase Transform. Debe girarse primero y luego avanzar.

* **this.transform.LookAt(goal.position)** en el Start para que gire primero y luego se mueva.
* Si lo ejecutamos en este momento, como la orientación del personaje va a cambiar, el translate no va a funcionar correctamente porque los ejes del personaje y el mundo no están alineados. El movimiento se debe hacer de forma relativa al sistema de referencia del mundo.

**this.transform.Translate(direction.normalized \* speed \* Time.deltaTime, Space.World)**.

* **Se ha realizado el script RotateToObjetive en el cubo, para que en el start se gire hacia el cilindro y luego avance hasta su posición.**

1. Añadir **Debug.DrawRay(this.transform.position,direction,Color.red)** para depuración para comprobar que la dirección está correctamente calculada.

* **Añadido al script RotateToObjetive en el cubo, ahora lo muestra (cuando está en pause)**

Si ahora ejecutamos el programa y movemos el cubo, vemos como los personajes lo siguen, recalculando la dirección en la que se tienen que mover continuamente. Sin embargo, ya no miran al objetivo porque está en el Start, hay que cambiarlo al update.

1. Crear un script que haga que el personaje siga al cubo. El cubo debe ser movido usando las teclas de flechas usando controlador de Starter Assets.

* **Añadido el script MoveKeyboard, pero la parte de usar el controlador starter assets no realizado**

1. Sobre la escena que has trabajado ubica un cubo que represente un personaje que vas a mover. Se debe implementar un script que haga de CharacterController. Cuando el jugador pulse las teclas de flecha (o aswd) el jugador se moverá en la dirección que estos ejes indican.
   1. Crear un script para el personaje que lo desplace por la pantalla, sin aplicar simulación física.
   2. Agregar un campo público que permita graduar la velocidad del movimiento desde el inspector de objetos.
   3. Utilizar la tecla de espaciado para incrementar la velocidad del desplazamiento en el tiempo de juego.

* **Añadidos los cambios al script MoveKeyboard**

1. Sobre la escena que has trabajado programa los scripts necesarios para las siguientes acciones:

* Se deben incluir varios cilindros sobre la escena. Cada vez que el objeto jugador colisione con alguno de ellos, se debe mostrar en la consola un mensaje indicando el nombre del cilindro con el que colisiona, cambiar a color rojo y el jugador aumentar la puntuación.