Descripción del proyecto.

El proyecto consiste en el manejo de una grúa de construcción a manos del usuario. Los movimientos disponibles son los siguientes:

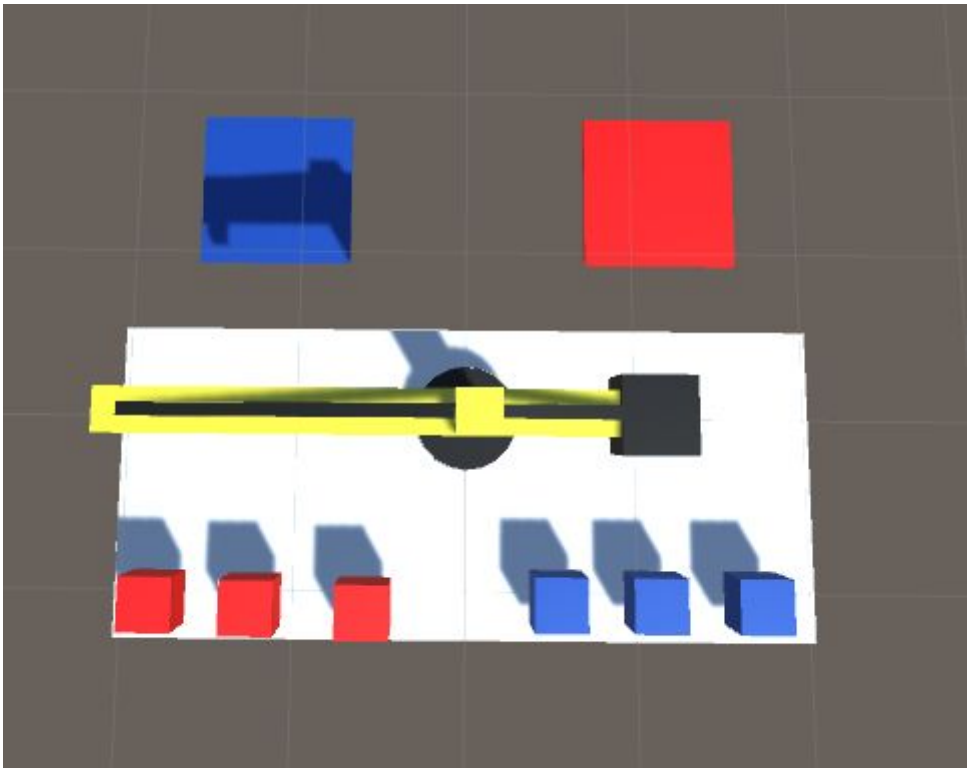
Para el control de la grúa se han utilizado inputs:

- **W+A+S+D** → Movimiento en ejes cardinales → Horizontal + Vertical
- **Q+E** → Rotación alrededor del tronco vertical de la grúa → Rotation
- **J+L** → Movimiento de la pluma sobre el tronco horizontal de la grúa → PlumaHorizontal

Para el control de la cadena se usan eventos de ratón:

- **Click izquierdo** → Bajar la cadena.
- **Click derecho** → Subir la cadena.
- **Click central** → Soltar objeto enganchado.

Teniendo esto claro, hay que conseguir depositar los cubos de cada color en el lugar adecuado. Para ello, la cadena de la grúa dispone de un enganche(último eslabón) que cuando entra en contacto con el cubo, lo agarra de tal manera que sigue el movimiento de la propia cadena. Una vez enganchado, hay que transportarlo y soltarlo encima del pozo de su color.



Objetivos de la práctica.

El objetivo principal de la práctica es crear una jerarquía de objetos adecuada para mover, crear y eliminar elementos padres e hijos sin deformarlos ni romper la jerarquía.

Descripción técnica:

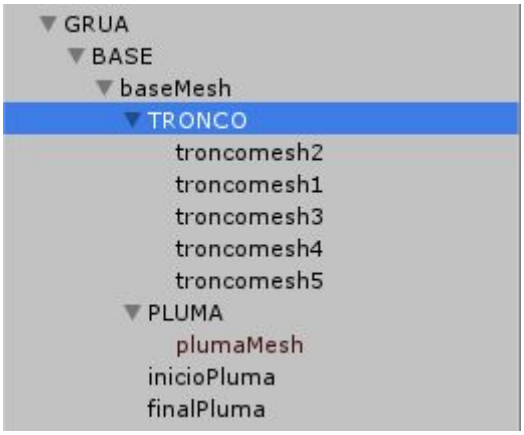
- **moveCrane** → Gestiona todos los eventos de teclado y ratón encargados del movimiento:
    - Movimiento y rotación de la grúa.
    - Movimiento y rotación de la pluma.
    - Subir y bajar cadena.
    - Soltar objeto.

- **cubeScript** → Gestiona los eventos de colisión entre el enganche y los cubos, además de entre los cubos y los pozos de colores, donde deben depositarse.

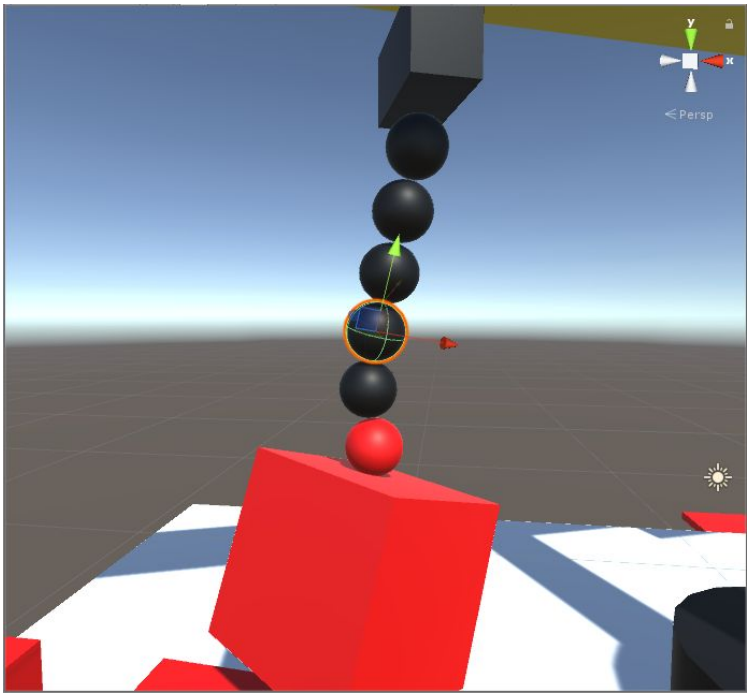
Dentro del código, todas las clases y métodos están comentados, indicando cual es su función principal e información extra en caso de ser necesaria para su claro entendimiento

Análisis funcional.

Como se mencionó previamente, el funcionamiento principal del proyecto es crear una grúa con su cadena, y cuyo movimiento se basa en el sistema de físicas del motor. Para que esto funcione correctamente, lo primero es crear una jerarquía adecuada con todos los elementos de la grúa, de tal manera que mientras se mueva no se deforme. Para ello se ha creado un gameobject vacío por cada elemento funcional, con una escala de unitaria(Vector3(1,1,1)).



Los nombre en mayúscula son estos elementos vacíos, mientras que el resto son las meshes con sus propios valores. Ordenando la jerarquía de esta manera, se consigue el efecto deseado.
Por otro lado, para crear la cadena se usan esferas primitivas como eslabones e “Hinge Joints” para sus uniones. De esta forma, al realizar el movimiento por físicas de la grúa, cada eslabón de la cadena y el peso cargado, tendrá su propio movimiento e inercia.



Consideraciones técnicas adicionales

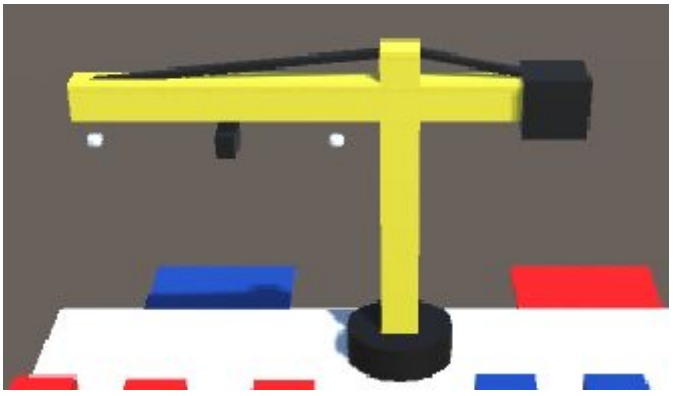
De manera adicional, hay que destacar el funcionamiento de dos elementos:

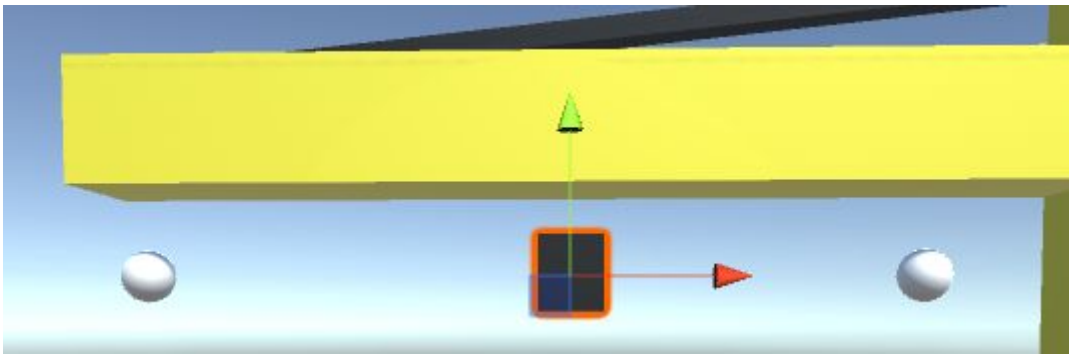
La pluma:

Este elemento tiene la cadena unida a él y se mueve solamente en su eje X, usando como inputs la “J” y la “L”. Siempre se desplaza entre los dos puntos marcados (origen y final del trayecto) y sigue la rotación del tronco del tronco vertical gracias a la forma de la jerarquía.

```
//Rotacion pluma
rb_pluma.MoveRotation(rb_pluma.transform.rotation * deltaRotation);

//Movimiento pluma --> J + L
if (Input.GetAxis("PlumaHorizontal") > 0){
    rb_pluma.position += (finalPluma.transform.position - rb_pluma.transform.position) * 0.01f;
    rb_pluma.transform.parent.position = rb_pluma.position;
}
else if (Input.GetAxis("PlumaHorizontal") < 0){
    rb_pluma.position += (inicioPluma.transform.position - rb_pluma.transform.position) * 0.01f;
    rb_pluma.transform.parent.position = rb_pluma.position;
}
else
    rb_pluma.MovePosition(rb_pluma.transform.parent.position + movement);
```





La cadena:

Está formada por un máximo de 10 eslabones y un mínimo de 1, siempre siguen el movimiento de la pluma y además para simular que la cadena sube y baja, se crean y destruyen eslabones. Además hay que destacar que el último eslabón creado (el último empezando desde abajo) siempre funciona como enganche y es capaz de agarrar objetos.

- **Para bajar** → Se crea un eslabón al final de la cadena, con la misma rotación que el último creado pero con una "y" menor(más bajo). Se crea un Hinge Joint entre el último eslabón y el recientemente creado, para simular el movimiento de una cadena real. En caso de tener un peso ya agarrado, se actualiza su joint con el nuevo "eslabón enganche" para que siga de nuevo en movimiento.

```
//Bajar cadena --> El ultimo eslabon creado funciona como enganche y es de color rojo.
void crearEslabon(){
    if(eslabones.Count <= 10) { //LImite de 10 eslabones

        //El ultimo eslabon creado, se pinta de negro y te taggea como eslabon
        eslabones[eslabones.Count-1].GetComponent<Renderer>().material.color = new Color(0f, 0f, 0f);
        eslabones[eslabones.Count - 1].tag = "eslabon";

        //Se crea uno nuevo, se coloca en el sitio deseado (bajo el ultimo eslabon de la cadena y con su misma rotacion)
        GameObject eslabon = Instantiate(eslabonPrefab);
        eslabon.transform.rotation = eslabones[eslabones.Count - 1].transform.rotation;
        eslabon.transform.position = new Vector3 (eslabones[eslabones.Count - 1].transform.position.x,
                                    eslabones[eslabones.Count - 1].transform.position.y,
                                    eslabones[eslabones.Count - 1].transform.position.z);
        eslabon.transform.Translate(new Vector3(0, -1f, 0),  Space.Self);

        //Se crea un joint con el eslabon anterior
        eslabon.GetComponent<HingeJoint>().connectedBody = eslabones[eslabones.Count - 1].GetComponent<Rigidbody>();

        //Se añade a la cadena y al ser el ultimo se taggea como enganche y se pinta de rojo
        eslabones.Add(eslabon);
        eslabones[eslabones.Count - 1].GetComponent<Renderer>().material.color = new Color(1f, 0, 0);
        eslabones[eslabones.Count - 1].tag = "enganche";

        //Si la cadena ya llevaba un peso enganchado, se renueva el body del joint con el nuevo enganche
        if(peso != null){
            peso.GetComponent<HingeJoint>().connectedBody = eslabones[eslabones.Count - 1].GetComponent<Rigidbody>();
            peso.GetComponent<HingeJoint>().connectedAnchor = new Vector3(0, -0.5f, 0);
        }
    }
}
```

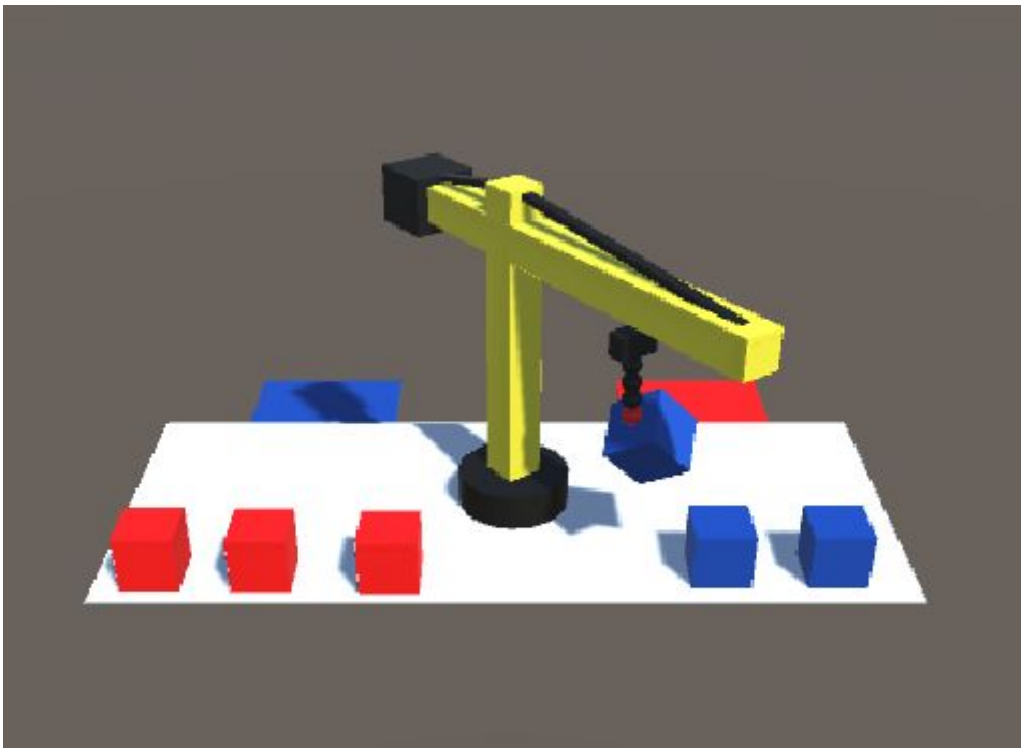
- **Para subir** → Su funcionamiento es igual que para bajar, pero en este caso, se borra el último eslabón y en caso de tener un peso agarrado, se actualiza su joint con el eslabón anterior.

```
//Subir cadena --> Siempre se borra el ultimo eslabon, que funcionaba como enganche
void borrarEslabon(){
    if (eslabones.Count > 1){//Siempre tiene que haber 1 eslabon como mínimo

        //Se borra el elemento y se actualiza la lista
        Destroy(eslabones[eslabones.Count - 1]);
        eslabones.Remove(eslabones[eslabones.Count - 1]);

        //Se pinta el anterior de rojo y se taggea como enganche
        eslabones[eslabones.Count - 1].GetComponent<Renderer>().material.color = new Color(1f, 0, 0);
        eslabones[eslabones.Count - 1].tag = "enganche";

        //Si la cadena llevaba un peso, se actualiza el body del joint conectado con el nuevo enganche
        if (peso != null){
            peso.GetComponent<HingeJoint>().connectedBody = eslabones[eslabones.Count - 1].GetComponent<Rigidbody>();
            peso.GetComponent<HingeJoint>().connectedAnchor = new Vector3(0, -0.5f, 0);
        }
    }
}
```



Los eslabones normales (incapaces de agarrar) son de color negro, mientras que el enganche es de color rojo.