En esta clase veremos como hacer que nuestra raqueta pueda cambiar de color, esto nos ayudara a obtener más herramientas para la sincronización de nuestro juego.

La solución será la siguiente:

1. Vamos a guardar el sprite renderer de la raqueta en una variable llamada sr.
2. Cada número corresponderá a un color.
3. Si el usuario presiona un número, el sr cambiará el color.
4. Esos cambios se propagan al servidor y luego al resto de clientes.

Todos nuestros cambios los vamos a hacer en el archivo Scripts/Player.cs. Como de costumbre vamos a hacerlo primero offline y luego online, lo primero es guardar el sprite renderer en la variable:

...

// Añadimos esta nueva variable

**public** SpriteRenderer sr;

...

**void** **Start**()

{

// Y la inicializamos en el Start

sr = GetComponent<SpriteRenderer>();

}

Ahora vamos a asignarle a cada número un color, podemos hacerlo así:

**void** **Update**()

{

**if** (Input.GetKeyDown(KeyCode.Alpha1))

{

sr.color = Color.white;

}

**if** (Input.GetKeyDown(KeyCode.Alpha2))

{

sr.color = Color.red;

}

...

}

Está bien, funciona, pero podemos mejorarlo un poco si usamos diccionarios:

Dictionary<KeyCode, Color> colors = **new** Dictionary<KeyCode, Color>(){

{KeyCode.Alpha1, Color.white},

{KeyCode.Alpha2, Color.red},

{KeyCode.Alpha3, Color.green},

{KeyCode.Alpha4, Color.gray},

{KeyCode.Alpha5, Color.blue},

{KeyCode.Alpha6, Color.yellow}

};

**void** **Update**()

{

**foreach** (**var** keycode **in** colors.Keys)

{

**if** (Input.GetKeyDown(keycode))

{

sr.color = colors[keycode];

}

}

}

Falta un pequeño detalle, tenemos que cambiarle el color únicamente a la raqueta que estamos controlando:

**void** **Update**()

{

**if** (!isLocalPlayer)

{

**return**;

}

....

}

Vamos progresando, ahora si vamos a nuestro juego, nuestra raqueta debería cambiar de color.

Claro que todo es local, si iniciamos otra instancia tendremos un error. El color no se va a actualizar en los demás clientes.

Para solucionarlo vamos a utilizar un concepto muy interesante, las variables sincronizadas, estas son variables que cuando cambian desde el servidor se actualizan en todos los clientes y además llama una función para avisar del cambio, vamos a verlo:

// Puede enviarse cualquier variable sincronizada y hasta 60 por script

// El hook es el nombre de la función que se llama cuando la variable

// cambia, si no lo necesitamos, podemos llamar simplemente

// [SyncVar]

[SyncVar(hook = nameof(SetColor))]

Color playerColor = Color.white;

**void** **SetColor**(Color oldColor, Color newColor)

{

sr.color = newColor;

}

**void** **Update**()

{

**foreach** (**var** keycode **in** colors.Keys)

{

**if** (Input.GetKeyDown(keycode))

{

playerColor = colors[keycode];

}

}

}

Ahora, este código tiene un problema gravísimo, recordemos algo clave sobre syncvar **estas son variables que cuando cambian desde el servidor se actualizan en todos los clientes**. Este código todavía se está llamando desde el cliente, de esta manera solo hay cambios si llamamos la variable desde el host (recordemos que es cliente + servidor) o desde el servidor propiamente dicho.

Para solucionar esto, necesitamos enviar la información al servidor y que el servidor haga el cambio, eso lo podemos hacer con un Command:

**void** **Update**()

{

**foreach** (**var** keycode **in** colors.Keys)

{

**if** (Input.GetKeyDown(keycode))

{

CmdChangeColor(colors[keycode]);

}

}

}

// Los comandos son funciones que se llaman en el

// Servidor desde los clientes

// Siempre empiezan por Cmd como los RPCs

[Command]

**public** **void** **CmdChangeColor**(Color newColor)

{

playerColor = newColor;

}

Lo sé, puede parecer redundante, pero es que las dos funciones se llaman desde lugares totalmente distintos, el update se llama desde el cliente, y el CmdChangeColor se llama desde el servidor.

Ahora sí podemos cambiar de colores nuestras raquetas. Este es un ejemplo un poco básico, no obstante se puede llevar a otras mecánicas, como la vida en un shooter, el estado en un juego de cartas, un sistema de animaciones, etc.

Te dejo con una imagen de cómo se ve nuestro juego:

