Manual de Usuario

En este apartado se va a desarrollar una explicación detallada de las configuraciones necesarias para poder utilizar el proyecto realizado en un proyecto de Unity propio. En primer lugar, hay que realizar las configuraciones adecuadas de creación del proyecto e importación de los archivos necesarios. Para que funcione correctamente lo desarrollado hay que crear un proyecto con la versión de Unity LTS 2021.3.14f1. La mejor opción es utilizar la opción de configuración del proyecto que ofrece Unity Hub de proyecto 3D con la implementación para URP.

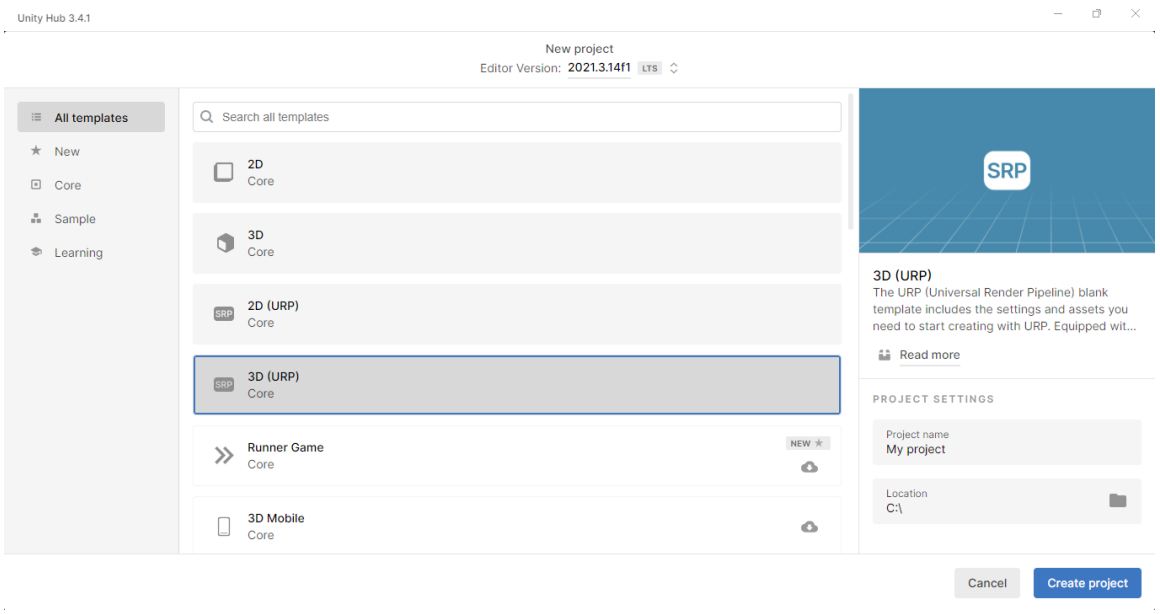


Figura 1. Creación de un proyecto de Unity compatible con URP.

Una vez creado el proyecto hay que importar el *Asset* de la librería de programación DryWetMidi que utiliza el proyecto. Para ello se busca en la página web de la *Asset* Store o en Window→Search→Asset Store y se añade a tus *Assets* como se ve en la Figura 2. Una vez añadido a tus *Assets* en la configuración del proyecto se tiene que ir al Packet Manager e importar el *Asset* dentro del proyecto en la versión 7.0.0.

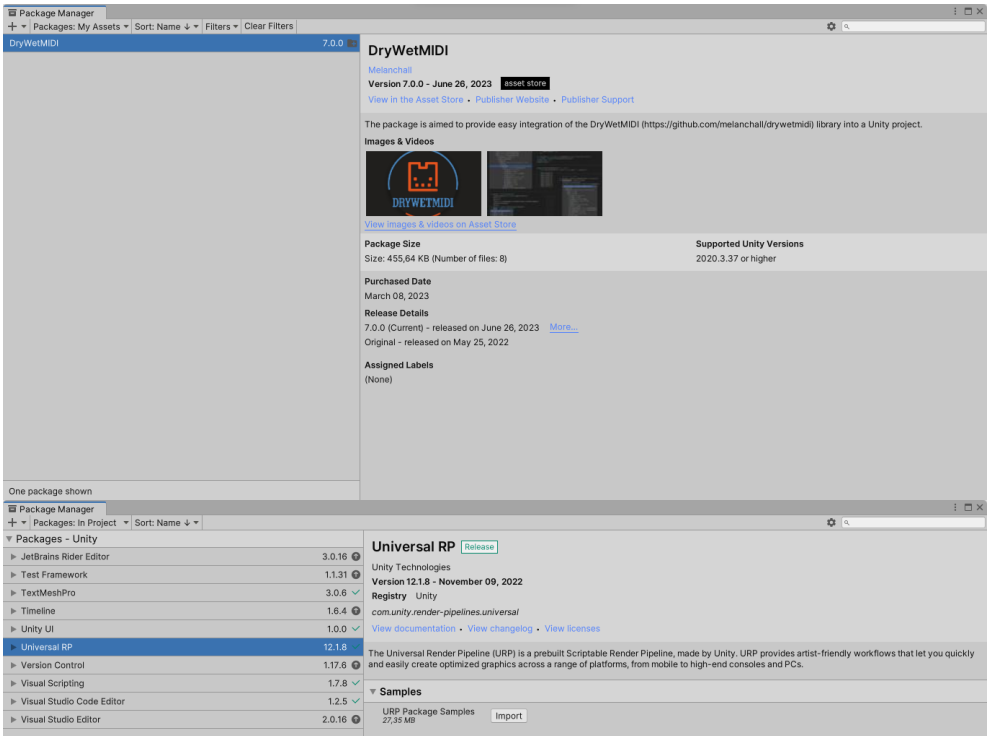


Figura 2. Configuración inicial del Packet Manager del proyecto.

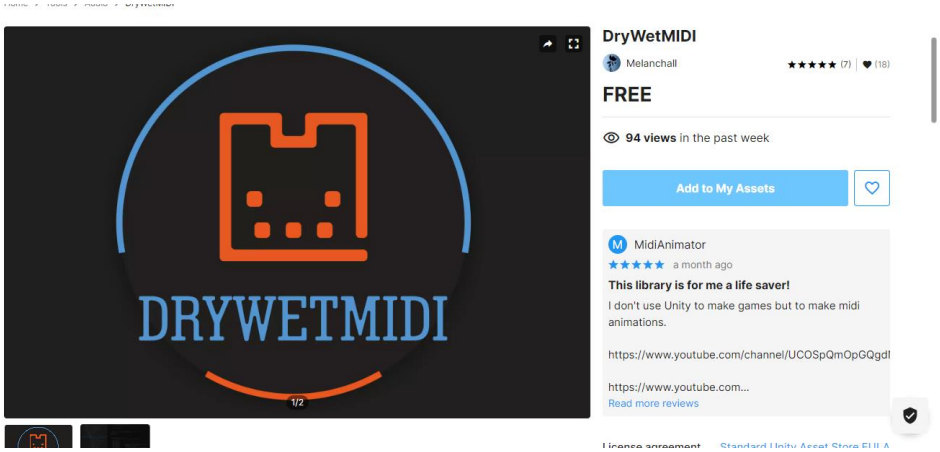
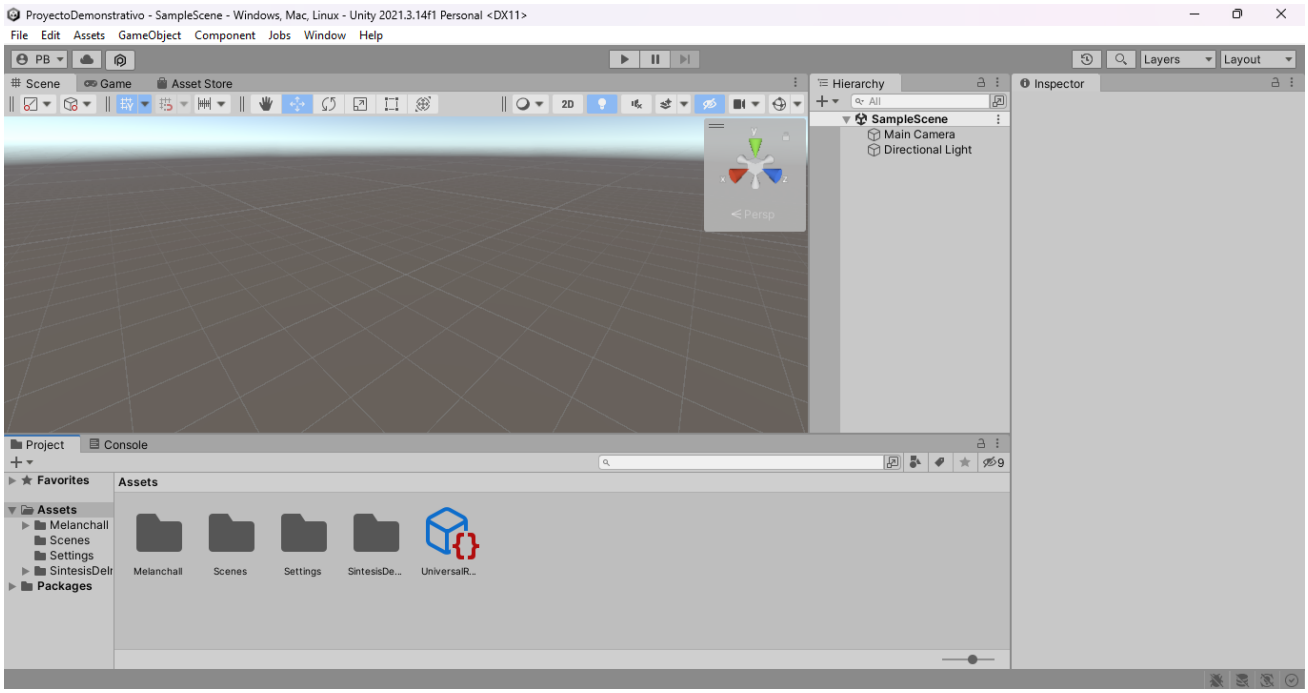


Figura 3. Página de descarga del Asset DryWetMidi.

Una vez configurado esto, se descarga los archivos del directorio del github creado para el proyecto en el enlace: *https://github.com/pedrogzb/SintesisDeImagenesUsandoMIDI.git*. En los archivos que se pueden descargar hay un «.unitypackage» que se puede importar en el directamente en el editor de Unity o copiar la carpeta *"SintesisDeImagenesUsandoMIDI"* dentro del directorio de "*Assets"* del proyecto.



Antes de poder utilizar los recursos que se han descargado se tiene que realizar una configuración de las opciones de URP. Hay que cambiar las configuraciones en los recursos *URP-High Fidelity-Renderer* y *URP-Performant-Renderer* que están en la carpeta *"Assets/Settings"*, como muestra la Figura 4. En la Figura 5 se muestra los valores que tienen que aparecer para el correcto funcionamiento. En concreto hay que cambiar *Depth Priming Mode* a *Disabled* e *Intermediate Texture* a *Always en ambos recursos.*

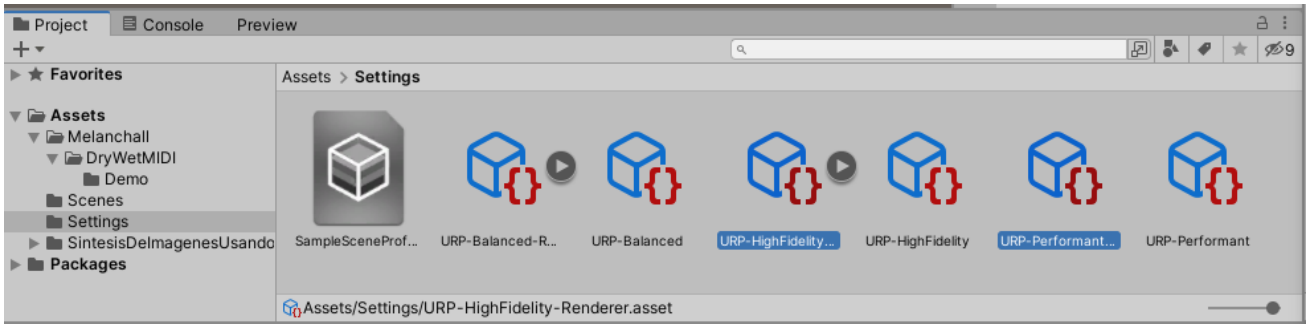


Figura 4. Recursos necesarios de configurar de URP para el correcto funcionamiento.

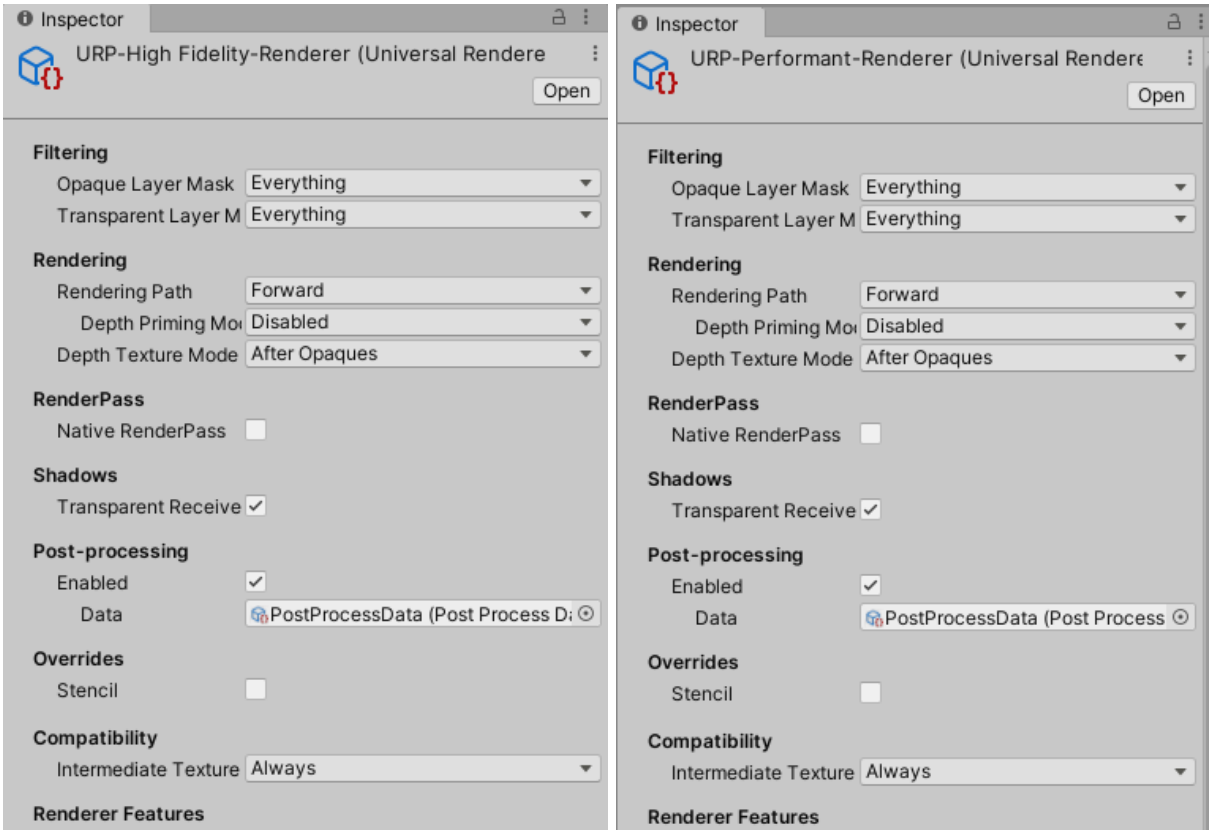


Figura 5. Valores de configuración de URP para un correcto funcionamiento del proyecto.

En este punto ya se puede utilizar todas las funcionalidades del proyecto desarrollado. A continuación, se va a detallar los diferentes pasos a seguir según lo que se quiera conseguir.

Configuración de entrada MIDI

Para configurar una entrada MIDI hay que crear un *GameObject* vacío en la jerarquía de la escena. Este objeto es el que gestionará la comunicación para recibir los eventos MIDI de entrada entre el dispositivo o el archivo MIDI. Si se quiere recibir mensajes de un dispositivo se tiene que adjuntar el componente *InputDevices* y si se quiere recibir mensajes de un archivo se tiene que utilizar el componente *InputMidiFile*. Para seleccionar las diferentes fuentes de MIDI se tiene que crear una instancia de un *ScriptableObject* de tipo *SelectorRecursoEntradaSO*. Para ello hay se pulsa el botón derecho del ratón en el inspector de proyecto y en el menú desplegable se selecciona Create→*ScriptableObject*→ *SelectorRecursoEntradaSO*, Figura 6.

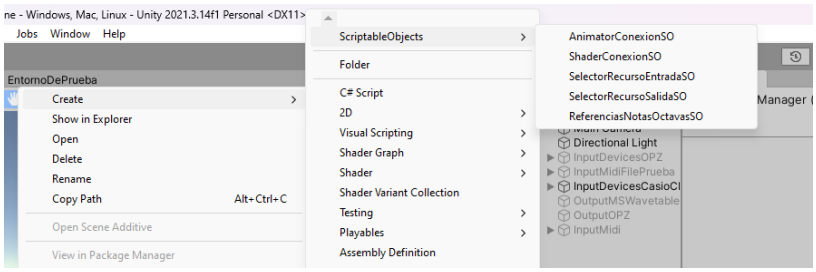


Figura 6. Creación de instancias de ScriptableObject en el inspector de Unity.

Una vez se tiene el *Asset* se le pone un nombre identificativo que se quiera y se pueden ir añadiendo los diferentes dispositivos que se quieran incluir en el apartado *Nombres De Recursos*, Figura 7 derecha. Si no se sabe el nombre del dispositivo una vez se de a ejecutar el programa saltará un error y se mostrará una lista de los nombres de los dispositivos disponibles en el sistema. Para configurar el *InputDevices* se tiene que incluir en un objeto vacío como componente, añadir el *SelectorRecursoEntradaSO* que se ha configurado y seleccionar la posición en la que está el nombre en la lista de *Nombres De Recursos* del *SelectorRecursoEntradaSO;* Figura 7 izquierda*.*

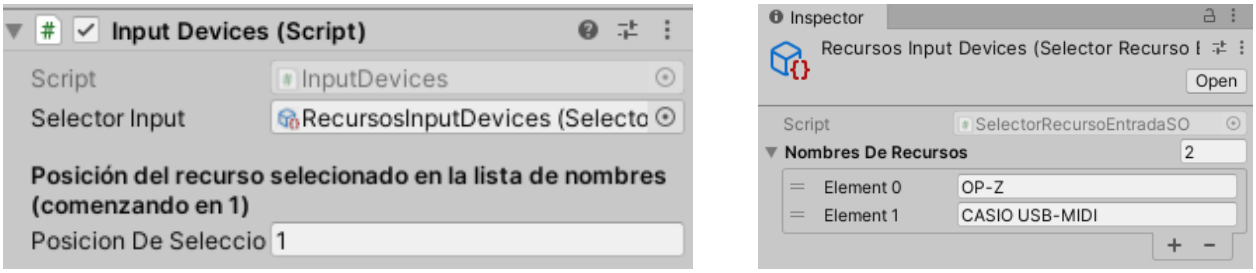


Figura 7. Ejemplo de configuración de dispositivos de entrada. Izq: Configuración InputDevices. Dcha: Configuración SelectorRecursoEntradaSO para dispositivos

Se recomienda que si se quieren poner también recursos de entrada para archivos MIDI se haga en un *Asset* diferente. En este caso el *Asset* tiene que guardar los nombres de los archivos MIDI o la ruta de acceso con el nombre del archivo. Estos archivos MIDI tienen que estar dentro del directorio Asset/SintesisDeImagenesUsandoMIDI/Resources o en un subdirectorio de este como se muestra en la Figura 8.

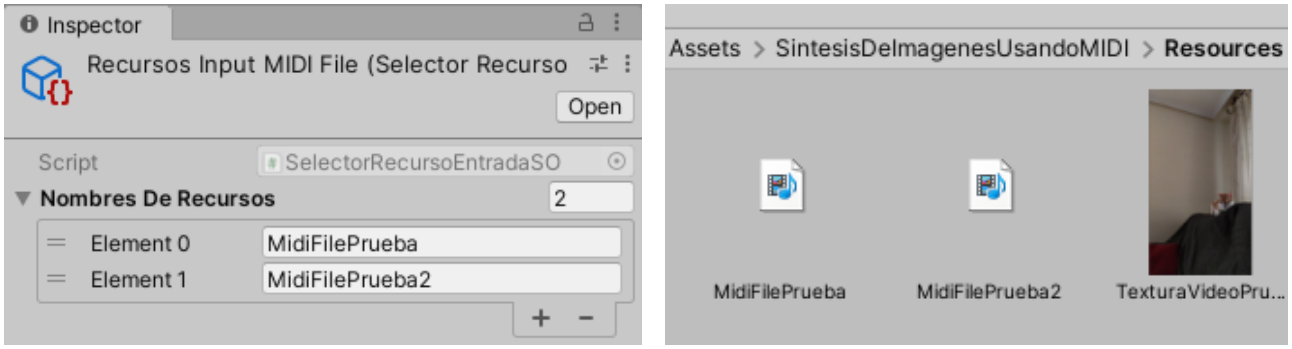


Figura 8. Configuración de referencias de archivos MIDI de entrada. Izq: Configuración SelectorRecursoEntradaSO para archivos. Dcha: Carpeta donde se guardan los recursos MIDI

Una vez se ha configurado esto ya se pueden recibir mensajes MIDI tanto de dispositivos como de archivos. Una vez se tiene esto hay dos opciones para enviar estos mensajes a otros módulos de interés. Se puede mandar directamente a un dispositivo de salida para que se reproduzca el sonido o se puede mandar a otros objetos para que gestionen los mensajes y los procesen para el propósito que se desee.

Para recibir los mensajes MIDI en otros objetos que sean los que gestionen y procesen los datos para poder realizar los efectos que se quieran se tiene que poner como objetos hijos del objeto vacío donde se ha configurado el recurso de entrada. En cada objeto que quiera recibir los mensajes MIDI de este recurso de entrada se deberá introducir el componente de *InputFilter*. Este componente sirve para realizar la conexión de todos los mensajes MIDI recibidos en el recurso de entrada y conectarlos con el sistema de preprocesado de datos que se deberá poner. Este componente también ofrece la posibilidad de filtrar los mensajes

que pasa al sistema pudiendo elegir el rango de canales que se quiere recibir y/o el rango de octavas que se quiere recibir. Si se quieren recibir todos los mensajes simplemente no se activará ninguna de las funciones de filtrado. Este componente también tiene la posibilidad de enviar los mensajes después de filtrarlos a un dispositivo de salida. En la Figura 9 se muestra la interfaz para configurar *InputFilter*, se activa el filtrado pulsando las cajas de selección y se indica el rango a filtrar en las cajas de texto, siguiendo el rango que se puede introducir indicado en la interfaz y siendo el rango inferior el que se pone en la X y el rango superior el de la Y inclusives.

Figura 9. Configuración de InputFilter.

Configuración de los dispositivos de Salida

Como ya se ha mencionado hay una posibilidad de poder escuchar los mensajes que se envían a través de diferentes dispositivos. Para ello se tiene que realizar configuración análoga a lo que se ha realizado para un dispositivo de entrada. Se tiene que crear un objeto vacío en el que se le añade el componente *OutputDevices* y se tiene que añadir a este una instancia de un ScriptableObject del tipo *SelectorRecursoSalidaSO*. Para cada dispositivo de salida hay que crear una instancia de este. En esa instancia se tiene que configurar el nombre del dispositivo y luego configurar el número de programa que se quiere registrar para cada canal en el dispositivo de salida. El número de programa es un número de 0 a 127 y tiene que haber 16 para que se pueda configurar adecuadamente. Este indica el sonido que predefinido que tiene el dispositivo de salida, si este sigue el estándar General MIDI la correspondencia sonido y número de programa es la que se muestra en la Tabla 1. Si no se ha configurado de la forma adecuada se tiene por defecto el número de programa 0.De la misma forma que para los dispositivos de entrada si no se sabe el nombre concreto del dispositivo se puede iniciar la ejecución del programa con el objeto del dispositivo de salida activo e indicará que no ha podido conectar con este dispositivo y mostrara una lista de dispositivos disponibles. Para que este objeto reciba los mensajes hay que referenciarlo en los componentes de *InputMidiFile, InputDevices* o *InputFilter* que se quiera como se ve en la Figura 10 y habilitar la salida. En la figura se puede ver también que se configura para el dispositivo de salida *Microsoft GS Wavetable Synth* que es un sintetizador de Windows.

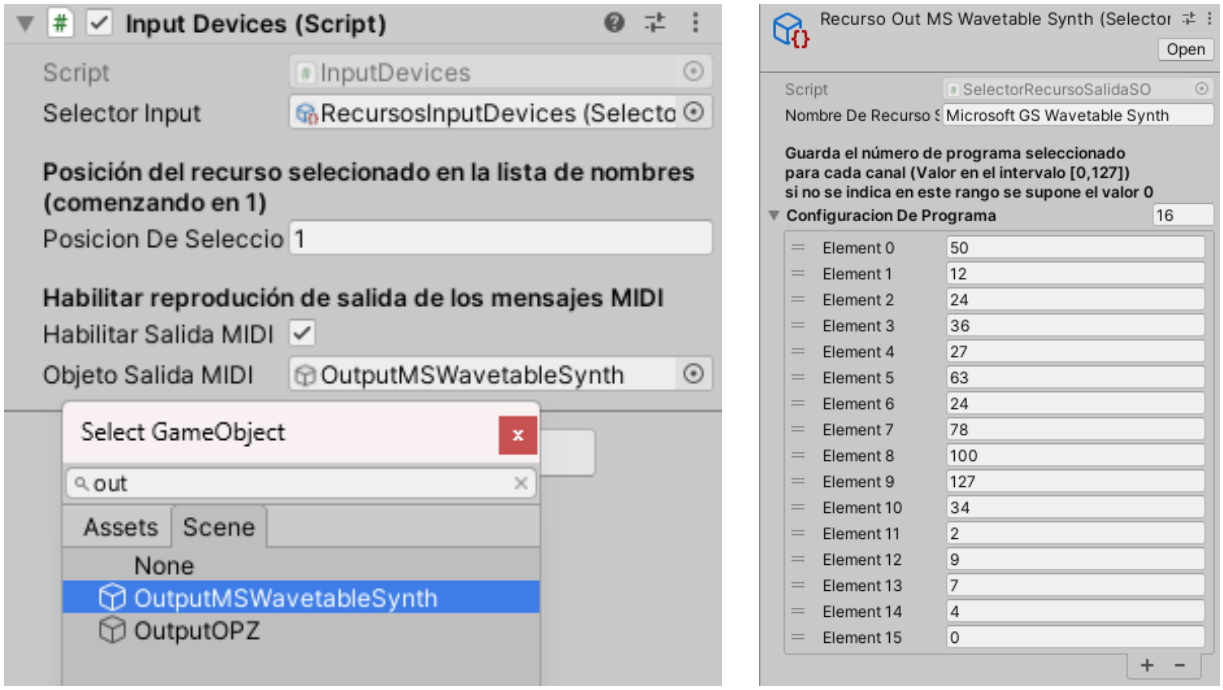


Figura 10. Configuración de dispositivos de salida. Izq: Salida MIDI. Dcha: Configuración SelectorRecursoSalidaSO

Tabla 1. Correspondencia de número de programa con el sonido en General MIDI

Sonidos	Rango de número de programa	Sonidos	Rango de número de programa
Piano	[0 , 8]	Instrumentos de viento de lengüeta	[64 , 73]
Percusión Cromática	[9 , 17]	Aerófonos de tubo	[72 , 81]
Órgano	[16 , 25]	Sintetizador solista	[80 , 89]
Guitarra	[24 , 33]	Sintetizador de acompañamiento	[88 , 97]
Bajo	[32 , 41]	Efectos especiales	[96 , 105]
Cuerdas	[40 , 49]	Sonidos étnicos	[104 , 113]
Conjuntos de Cuerda	[48 , 57]	Percusión	[112 , 127]
Instrumentos de viento metales	[56 , 65]	-	-

Configuración del sistema de preprocesado

Para la realización de esta configuración hay cuatro componentes que son los que gestionan la conexión entre los mensajes MIDI procedentes de los componentes *InputFilter* y la tecnología que se utilice para hacer los efectos visuales, pudiendo ser o Shaders o las animaciones controladas por el *Animator*. Luego, hay tres *ScriptableObject* que guardan la información de cómo comunicarse con las tecnologías. Estos componentes son:

- Bypass
- FadeOut
- ShaderPreprocesado
- AnimatorPreprocesado

En una primera aproximación si se necesita de alguna forma información directa de las notas que se han pulsado o dejado de pulsar y la correspondiente octava a la que pertenece se pueden utilizar los componentes de *Bypass* o *FadeOut*. Éstas ofrecen información de cada octava y cada nota que se está pulsando en cada momento. La diferencia que hay entre los dos componentes es que los valores que toman las variables de salida de *Bypass* son siempre o 1 o 0. Los valores de *FadeOut* cuando se pulsa la nota son 1 y cuando se despulsa la nota se va disminuyendo el valor hasta llegar a 0 según el tiempo de transición especificado. La única configuración que hay que realizar es crear una instancia de un *ScriptableObject* del tipo *ReferenciasNotasOctavasSO.* En éste se tienen que introducir los nombres de las 12 variables que controlen las notas y las 11 que controlan las octavas, que cubrirían todas las posibles combinaciones de las 128 notas diferentes que se pueden señalar en una trama MIDI. La configuración de los componentes se hace de añadiendo una instancia de *ReferenciasNotasOctavasSO* como se muestra en la Figura 11. Estas configuraciones solo funcionan con los Shaders.

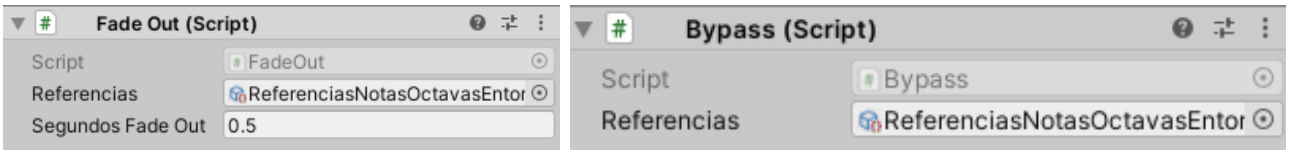


Figura 11. Configuración de los componentes Bypass y FadeOut.

Si se quiere una mayor flexibilidad para que se pueda utilizar el *Animator* o se quiere tener un mayor control de las variables deseadas se puede utilizar los componentes de *ShaderPreprocesado* y *AnimatorPreprocesado*. Para realizar la configuración de esta parte del proyecto se aconseja tener ya realizado, aunque sea de forma general, un boceto de las variables y los valores que pueden tomar estas en el Shader o *Animator* que se quiera realizar. Esto es debido a que hay que realizar una serie de configuraciones que van a ser mucho más fáciles si se tienen esos aspectos planificados. Se va a dividir en dos la explicación, por tanto.

Configuración *AnimatorPreprocesado*

La configuración del componente *AnimatorPreprocesado* es simplemente añadir el Animator que se encuentra en el objeto y adjuntarle una instancia del tipo *AnimatorConexionSO,* Figura 12.

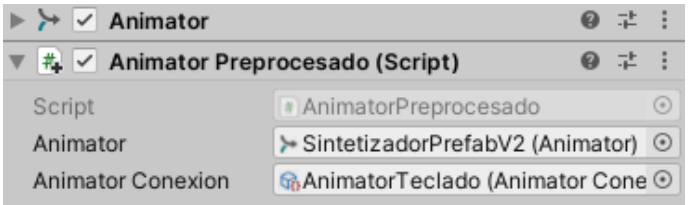


Figura 12. Configuración AnimatorPreprocesado.

Lo complicado es realizar la configuración de la instancia de *AnimatorConexionSO*. En un primer momento la configuración se puede dividir en los campos que se ven en la Figura 13. Los pasos a seguir para la configuración son:

- Elegir si se quieren tener en cuenta las octavas
- Introducir el nombre de la variable.
- Seleccionar el tipo de variable de salida
- Seleccionar la zona de ejecución en la que actualizar los valores

- Seleccionar la opción de cálculo que se quiere realizar
- Seleccionar la opción de remapeo que se quiere realizar
- Introducir los valores adecuados para realizar el remapeo

Hay una primera configuración del campo *Con Octavas*. Hay que elegir si se tiene en cuenta las octavas o solo se quiere obtener información relativa a las notas. Esto quiere decir que las variables de salida tendrán un rango de valores de 0 a 11 (no se tienen en cuenta octavas) o de 0 a127 (Se tienen en cuenta octavas).

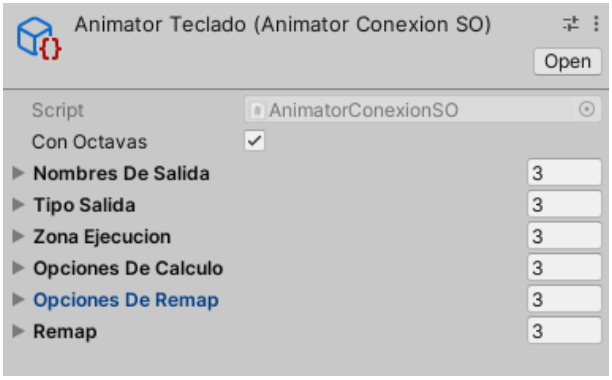


Figura 13. Variables de configuración para la conexión con el Animator

La siguiente configuración es de los *Nombres de Salida*, aquí hay que añadir campos y poner los nombres de las variables que se quieran utilizar para comunicar con el *Animator,* Figura 14. También se tiene que elegir el tipo de salida en el campo *Tipo Salida*, que tendrá que coincidir con lo que se ha introducido como variables en *Animator,* Figura 15.

Figura 14.Configuración de los nombres de entrada. Izq: Parámetros introducidos en el AnimatorConexionSO. Dcha: Configuración de parámetros en el Animator

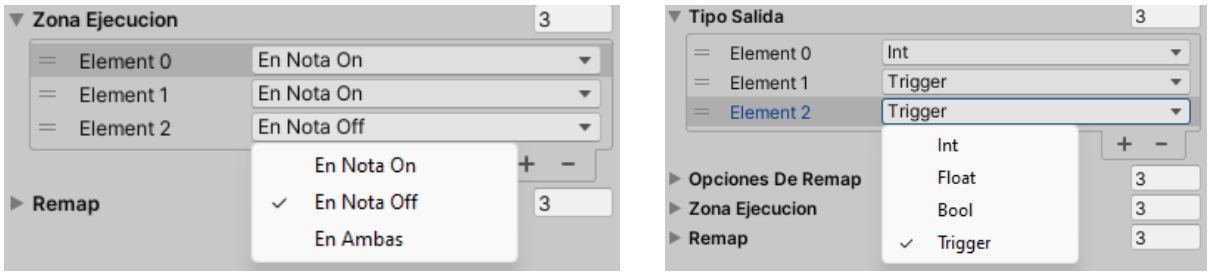


Figura 15. Izq: Configuración de la zona de ejecución. Dcha: Configuración del tipo de variable de salida.

A continuación, se tiene que configurar el campo *Opciones De Calculo,* Figura 16. Este indica que valor calculado se elige para dar valor a la variable que se modifica del *Animator*. Estos valores calculados son:

- Media: Da la media del número de notas pulsados. Si se tiene activado con octavas dará un valor de entre 0 y 127 y si se tiene desactivado dará un valor de entre 0 y 11.
- Ultima Pulsada: Da el número de nota del último mensaje recibido de *NotaOn*. Con las octavas activadas da un número de 0 a 127 y de lo contrario de 0 a 11.
- Ultima DesPulsada: Da el número de nota del último mensaje recibido de *NotaOff*. Con las octavas activadas da un número de 0 a 127 y de lo contrario de 0 a 11.
- Max: Da el número de la nota más pulsada, siempre ofrece números de 0 a 11.
- Min: Da el número de la nota menos pulsada, siempre ofrece números de 0 a 11.

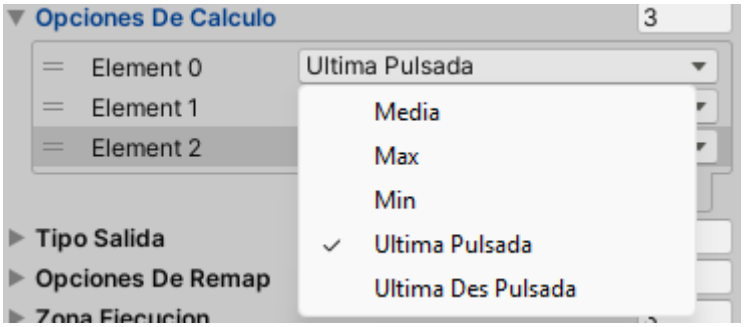


Figura 16. Configuración de las opciones de cálculo.

El rango de valores que requieren las variables del *Animator* no tiene por qué coincidir con los del número de nota de la trama MIDI, por tanto, hay una opción de remapeo de los valores del intervalo dado a otro que se quiera. Estas opciones de remapeo son las que se muestran en la Figura 17. En la Tabla 2 se muestran las operaciones que realizan para cada caso, donde **x** es el valor de la variable que se quiere remapear y **V** es el Vector4 que se configura en el campo *Remap* que se muestra en la Figura 17.

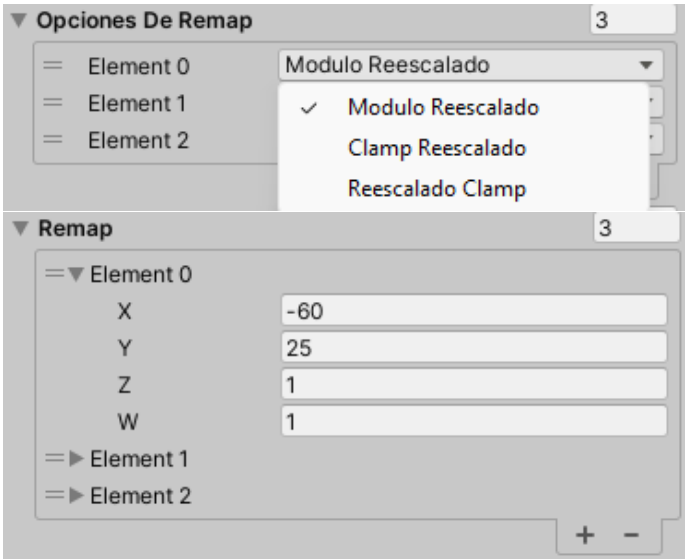


Figura 17. Configuración de parámetros para el remapeo del rango de salida.

Tabla 2. Definición de operaciones de remapeo para el AnimatorConexionSO

Tipo de Transformación	Variables de entrada	Resultado
ModuloReescalado	x, V	$\left(mod\left(x + V_x\ ,\ V_y\right) + V_z\right) \cdot V_w$
ClampReescalado	x, V	$\min\left(\max\left(x, V_x\right), V_y\right) \cdot V_z + V_w$
ReescaladoClamp	x, V	$\min\left(\max\left(x \cdot V_z + V_w, V_x\right), V_y\right)$

Como se puede observar el número elementos de las listas que conforman cada campo es igual. Esto se tiene que cumplir para que funcione correctamente la conexión. Para cada variable que se quiera modificar se tendrán que configurar todos los campos del *AnimatorConexionSO* y estos tienen que estar en la misma posición de la lista, es decir, el primer elemento de la lista de cada campo configura una misma variable.

Configuración *ShaderPreprocesado*

Este componente comparte muchas cosas en común con el de *Animator* pero tiene algunos aspectos característicos solo de este. Para la configuración del componente simplemente hay que pasarle el material y un *ScriptableObject* de tipo *ShaderConexionSO,* Figura 18. Además, hay que seleccionar si se quiere actualizar las variables de las propiedades de los Shaders por fotograma o solo cuando llegue un mensaje. Actualizar las variables por *frame* hace que el cambio de los valores no sea tan brusco y ofrezca un resultado mejor para determinados efectos.

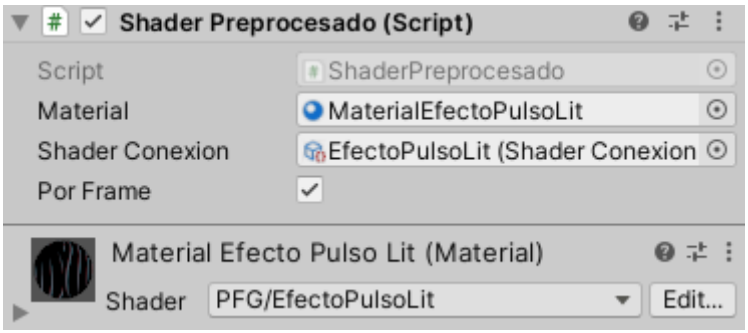


Figura 18.Configuración ShaderPreprocesado.

Luego se tiene que configurar una instancia de *ShaderConexionSO,* Figura 19. Esta tiene una estructura muy similar al *AnimatorConexionSO*. Los pasos a seguir para la configuración son:

- Elegir si se quieren tener en cuenta las octavas
- Introducir el nombre de la variable.
- Seleccionar el tipo de variable de salida
- Seleccionar la zona de ejecución en la que actualizar los valores
- Seleccionar la opción de cálculo que se quiere realizar
- Seleccionar la opción de remapeo que se quiere realizar
- Introducir los valores adecuados para realizar el remapeo

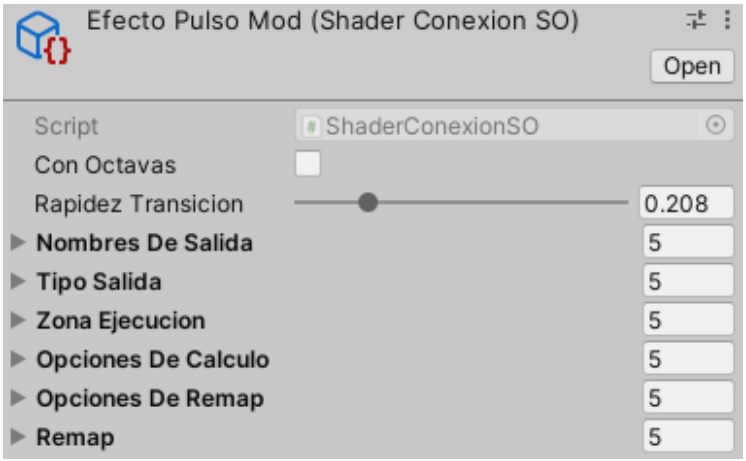


Figura 19. Variables de configuración para la conexión con los Shaders.

El campo *Con Octavas* funciona de la misma forma. La rapidez de transición se utiliza para configurar cuanto de rápido es el cambio en el caso de que este activado el modo por *frame*. Luego se tiene que introducir de la misma forma el nombre, Figura 20, el tipo de variable de salida que se quiere poner y la zona de ejecución donde se cambia el valor de la variable, Figura 21.

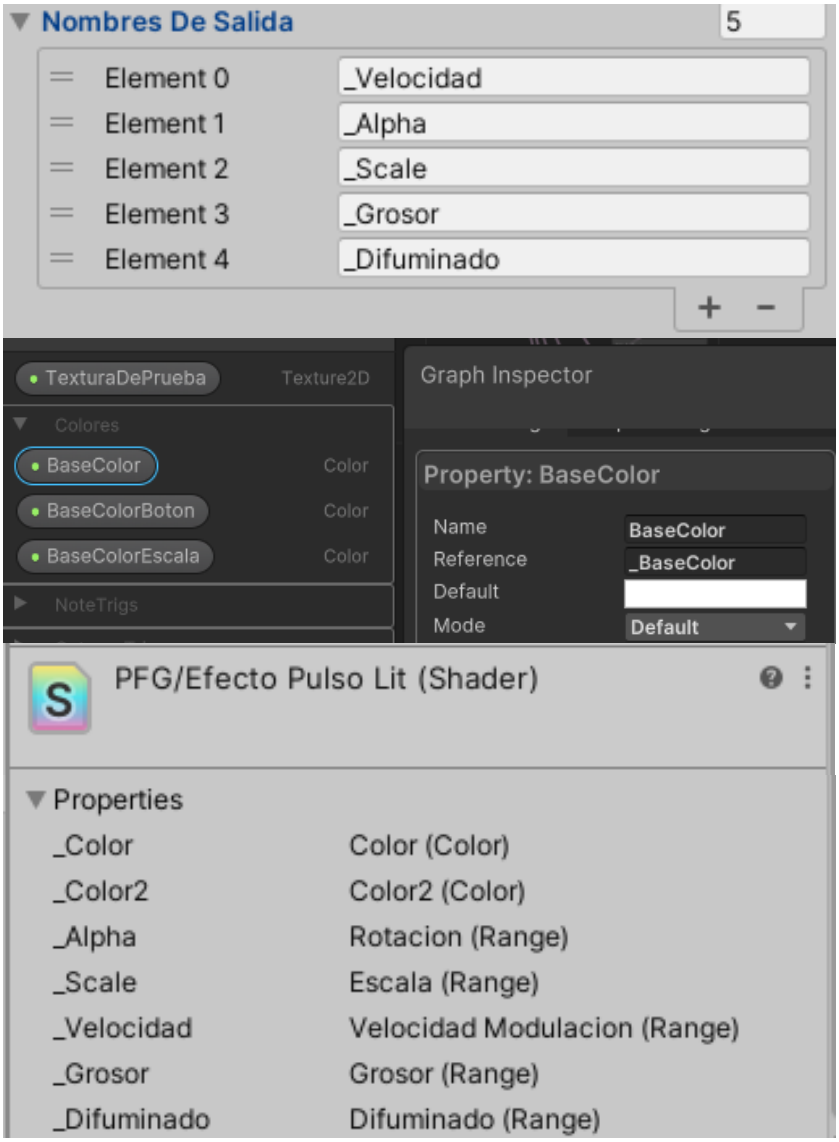


Figura 20. Configuración de los nombres de entrada. Arriba: Parámetros introducidos en el ShaderConexionSO. Centro: Configuración de parámetros en el ShaderGraph. Debajo: Parámetros definidos en el archivo Configuración de parámetros en el archivo «.shader».

Las opciones de cálculo son las mismas que en el caso de *AnimatorConexionSO* anterior y dan el mismo valor con las mismas condiciones.

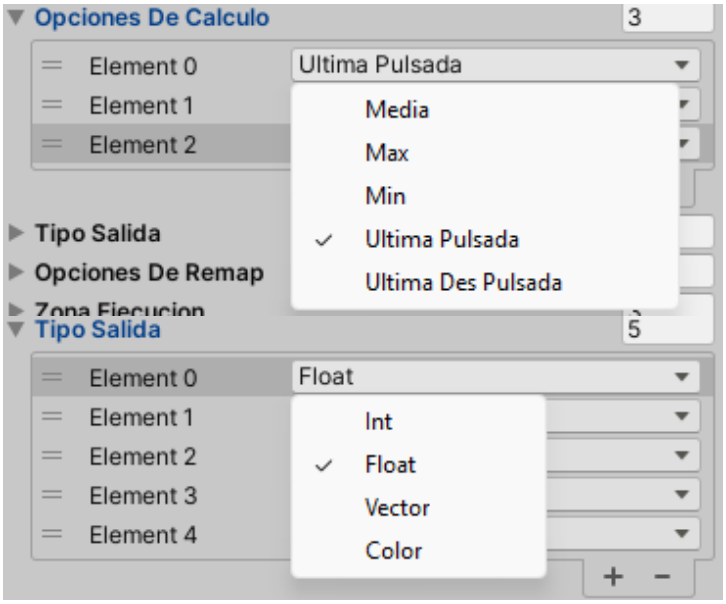


Figura 21.Configuración de Opciones De Calculo y Tipo Salida

En la Tabla 3 se muestran las operaciones que se realizan para cada caso, donde **x** es el valor de la variable que se quiere remapear y **V** es el Vector4 que se configura en el campo *Remap* que se muestra en la Figura 22. Internamente siempre se trabaja con vectores4 pero dependiendo del tipo de salida o de la operación de mapeado solo se modifican las variables que interesan al usuario. Por ejemplo, si se quiere modificar una variable de tipo vector de dos dimensiones se debe utilizar la opción de cálculo Vector2Reescalado.

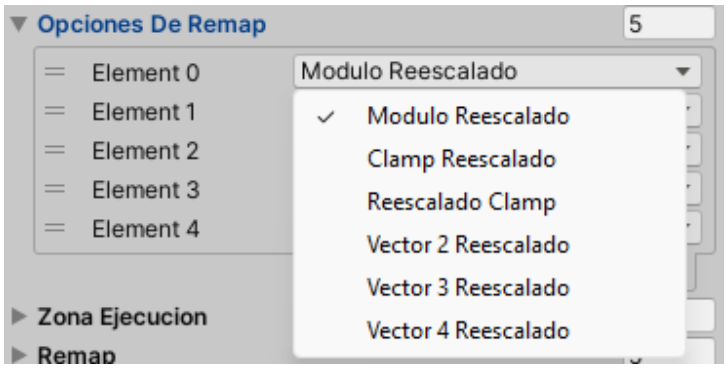


Figura 22. Configuración de Opciones De Remap

Tabla 3. Definición de operaciones de remapeo para el ShaderConexionSO

Tipo de Transformación	Variables de entrada	Resultado
Modulo Reescalado	x, V	$R = \{R_x = \left(mod\left(x + V_x\ ,\ V_y\right) + \ V_z\right) \cdot \ V_w,$ $R_y = 0,$ $R_z = 0,$ $R_w = 0\}$
Clamp Reescalado	x, V	$R = \{R_x = \min\left(\max(x, V_x)\ , V_y\right) \cdot V_z + V_w,$ $R_y = 0,$ $R_z = 0,$ $R_w = 0\}$
Reescalado Clamp	x, V	$R = \{R_x = \min\left(\max(x \cdot V_z + V_w, V_x)\ , V_y\right),$ $R_y = 0,$ $R_z = 0,$ $R_w = 0\}$
Vector2 Reescalado	x, V	$R = \{R_x = x \cdot V_x + V_y,$ $R_y = x \cdot V_z + V_w,$ $R_z = 0,$ $R_w = 0\}$
Vector3 Reescalado	x, V	$R = \{R_x = x \cdot V_x + V_y,$ $R_y = x \cdot V_x + V_z,$ $R_z = x \cdot V_x + V_w,$ $R_w = 0\}$
Vector4 Reescalado	x, V	$R = \{R_x = x \cdot V_x,$ $R_y = x \cdot V_y,$ $R_z = x \cdot V_z,$ $R_w = x \cdot V_w\}$

Como se puede observar el número elementos de las listas que conforman cada campo es igual. Esto se tiene que cumplir para que funcione correctamente la conexión. Para cada variable que se quiera modificar se tendrán que configurar todos los campos del *ShaderConexionSO* y estos tienen que estar en la misma posición de la lista, es decir, el primer elemento de la lista de cada campo configura una misma variable.

Guía de programación

En este punto se quiere explicar y detallar las diferentes características del código desarrollado para que se pueda utilizar e implementar otro código que suplemente, modifique o amplíe el código realizado en este proyecto.
Para desarrollar cualquier código que sea compatible con el código que ya está establecido hay que implementar las interfaces que han definido.

Si se quiere realizar código que realice un filtrado entre los mensajes que se reciben y el módulo de preprocesado de datos de entrada se tiene que implementar la interfaz *IConexionManagerFilter*, mostrada en el Fragmento de código 1. Esta interfaz es la que implementa el script *InputFilter* y tiene dos métodos, una llamada *EventoMidiNoteOn* y otra *EventoMidiNoteOff.* Estos dos métodos tienen un parámetro de entrada que es de tipo Vector3Int en la que la dimensión x representa el número de canal del mensaje de 0-15, la dimensión y representa el número de nota que va de 0-127 y la dimensión z representa la velocidad de pulsación que va de 0-127.

Fragmento de código 1. Definición de interfaz IConexionManagerFilter

```
using UnityEngine;
public interface IConexionManagerFilter
{
    public void EventoMidiNoteOn(Vector3Int NoteOn);
    public void EventoMidiNoteOff(Vector3Int NoteOff);
}
```

Si se quiere realizar código en una etapa posterior a *InputFilter* se tiene que implementar la interfaz *IConexionFilterPreprocesado*, mostrada en el Fragmento de código 2. Algunos ejemplos son los scripts *Bypass*, *FadeOut*, *ShaderPreprocesado* y *AnimatorPreprocesado*. Esta interfaz define dos métodos, una llamada *NotaPulsada* y otra *NotaDesPulsada.* Estos dos métodos tienen un parámetro de entrada que es de tipo Vector3Int en la que la dimensión x representa el número de nota sin contar la octavas siendo do (C) el número 0 y el Si (B) el número 11, la dimensión y representa el número de octava que va de 0 a 10 y la dimensión z representa la velocidad de pulsación que va de 0-127.

Fragmento de código 2. Definición de interfaz IConexionFilterPreprocesado.

```
using UnityEngine;
public interface IConexionFilterPreprocesado
{
    public void NotaPulsada(Vector3Int pulsacion);
    public void NotaDesPulsada(Vector3Int pulsacion);
}
```

Para realizar la una codificación distinta de cómo se transmiten los eventos de salida del MIDI se tiene que utilizar la interfaz *IConexionInputOutputDevice*, definida en el Fragmento de código 3. En este se definen tres tipos de funciones, un es *SendEventoMidiNoteOn, SendEventoMidiNoteOff y SendEventoCambioDePrograma. Los dos primeros* métodos tienen dos implementaciones posibles. La primera es teniendo un parámetro de entrada de tipo *MidiEvent* y la segunda es con un Vector3Int como parámetro de entrada, que funciona de la misma forma que en los métodos de *IConexionFilterPreprocesado.* Por último, el evento para el cambio de programa tiene como parámetros de entrada el número de canal al que está dirigido el mensaje y el número de programa al que se quiere cambiar, este varía de 0-127.

Fragmento de código 3. Definición de interfaz IConexionInputOutpuDevice.

```
using UnityEngine;
using Melanchall.DryWetMidi.Core;
public interface IConexionInputOutputDevice
{
    public void SendEventoMidiNoteOn(MidiEvent e);
    public void SendEventoMidiNoteOff(MidiEvent e);

    public void SendEventoMidiNoteOn(Vector3Int mensaje);
    public void SendEventoMidiNoteOff(Vector3Int mensaje);
    public void SendEventoCambioDePrograma(int numeroDeCanal, int numeroDePrograma);
}
```

Para realizar otros Shaders o el control de animaciones solo hay que realizar la configuración detallada que se explica en el apartado anterior junto con la codificación o configuración de los Shaders.