Documentación del código

Clase InputDevices

Descripción

Representa un dispositivo MIDI que se puede conectar con los componentes dentro de Unity.

Herencia

Object → Component →Behaviour → MonoBehaviour

Declaración

```
public class InputDevices : MonoBehaviour
```

Propiedades

Declaración	Visible en el editor	Descripción
`private SelectorRecursoEntradaSO SelectorInput;`	Sí	Objeto en el que se guardan los datos para obtener las referencias a los dispositivos
`private int PosicionDeSeleccion;`	Sí	La posición del dispositivo en la lista que tiene el selector de recursos de entrada
`private GameObject objetoSalidaMIDI;`	Sí	Guarda la referencia al objeto que tiene asociada la salida MIDI
`private bool HabilitarSalidaMIDI;`	Sí	Si está a true se manda los mensajes MIDI recibidos al objeto de salida
`private InputDevice _inputDevice;`	No	Es el objeto que maneja la gestión de los eventos MIDI que se quieren recibir del dispositivo. Clase importada de Melanchall.DryWetMidi.Multimedia
`private IConexionManagerFilter[] _referenciasConexion;`	No	Guarda las referencias a los componentes que pueden recibir los mensajes MIDI
`private IConexionInputOutputDevice _referenciaOutputDevice;`	No	Guarda la referencia del componente que maneja la salida de eventos MIDI

Métodos

Start()	
Declaración	`void Start()`
Descripción	Realiza la inicialización del dispositivo de entrada y encuentra las referencias (si las hay) a las conexiones con el dispositivo de salida y los objetos hijos que quieren recibir mensajes MIDI
Sobreescribe	MonoBehaviour.Start()

OnEventReceived(Object, MidiEventReceivedEventArgs)		
Declaración	`private void OnEventReceived(object sender, MidiEventReceivedEventArgs e)`	
Descripción	Función delegada que se pasa su referencia al objeto _inputDevice para indicar los pasos a seguir cuando se reciba un evento MIDI de un dispositivo	
Parámetros		
Tipo	Nombre	Descripción
Object	Sender	Objeto que ha detectado el evento
MidiEventReceivedEventArgs	e	Datos sobre el evento que se ha detectado. Este es una clase importada de Melanchall.DryWetMidi.Multimedia

EnviarEventoNoteOn(Vector3Int)		
Declaración	`public void EnviarEventoNoteOn(Vector3Int EventoNoteOn)`	
Descripción	Función para enviar los datos de datos obtenidos del evento MIDI cuando se ha pulsado una nota. Se envía a los componentes de los GameObjects hijos que implementen la interfaz *IConexionManagerFilter*.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	EventoNoteOn	Datos que se tienen que enviar al objeto hijo sobre el evento cuando se ha pulsado la tecla. En el vector la primera dimensión es el canal MIDI, la segunda dimensión es el número de nota y la tercera dimensión es la intensidad de la pulsación.

EnviarEventoNoteOff(Vector3Int)		
Declaración	`public void EnviarEventoNoteOff(Vector3Int EventoNoteOff)`	
Descripción	Función para enviar los datos de datos obtenidos del evento MIDI cuando se ha despulsado una nota. Se envía a los componentes de los GameObjects hijos que implementen la interfaz IConexionManagerFilter.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	EventoNoteOff	Datos que se tienen que enviar al objeto hijo sobre el evento cuando se ha despulsado la tecla. En el vector la primera dimensión es el canal MIDI, la segunda dimensión es el número de nota y la tercera dimensión es la intensidad de la pulsación.

InicializarInputDevice ()	
Declaración	`private void InicializarInputDevice()`
Descripción	Realiza el proceso de identificación del dispositivo utilizando el selector de recurso de entra SelectorInput y comprueba que hay un dispositivo conectado y accesible que se pueda conectar. Si no se puede conectar se muestra un mensaje en consola de que no se ha podido conectar y se muestra una lista de los nombres de los dispositivos disponibles para conectarse. Si se ha podido conectar se guarda la referencia al dispositivo de entrada y prepara los recursos necesarios para empezar a escuchar los eventos del dispositivo.

OnApplicationQuit ()	
Declaración	`private void OnApplicationQuit()`
Descripción	Si se ha podido realizar la conexión con un dispositivo realiza el proceso para deshabilitar la escucha de eventos y liberar los recursos guardados para la realización de estas operaciones. Este método se ejecuta siempre al terminar la ejecución del programa.
Sobreecribe	MonoBehaviour. OnApplicationQuit()

Clase InputMidiFile

Descripción

Representa un archivo MIDI que se puede conectar con los componentes dentro de Unity.

Herencia

Object →Component → Behaviour → MonoBehaviour

Declaración

```
public class InputMidiFile : MonoBehaviour
```

Propiedades

Declaración	Visible en el editor	Descripción
`private SelectorRecursoEntradaSO SelectorInputFile;`	Sí	Objeto en el que se guardan los datos para la obtener las referencias a los archivos MIDI
`private int PosicionDeSeleccion;`	Sí	La posición del archivo en la lista que tiene el selector de recursos de entrada
`private GameObject objetoSalidaMIDI;`	Sí	Guarda la referencia al objeto que tiene asociada la salida MIDI
`private bool HabilitarSalidaMIDI;`	Sí	Si está a true se manda los mensajes MIDI recibidos al objeto de salida
`private Playback _playback;`	No	Es el objeto que maneja la gestión de los eventos MIDI que se quieren recibir de la lectura del archivo. Es una clase importada de Melanchall.DryWetMidi.Multimedia
`private bool ReproducirEnBucle;`	Sí	Habilita la posibilidad de reproducir en bucle el archivo
`private IConexionManagerFilter[] _referenciasConexion;`	No	Guarda las referencias a los componentes que pueden recibir los mensajes MIDI
`private IConexionInputOutputDevice _referenciaOutputDevice;`	No	Guarda la referencia del componente que maneja la salida de eventos MIDI

Métodos

Awake()	
Declaración	`void Awake()`
Descripción	Realiza la inicialización del archivo de entrada utilizando el método InicializarMidiFile()
Sobrescribe	MonoBehaviour. Awake()

Start()	
Declaración	`void Start()`
Descripción	Realiza la función de encontrar las referencias (si las hay) a las conexiones con el dispositivo de salida y los objetos hijos que quieren recibir mensajes MIDI. Si no hay necesidad de sincronizarse el archivo con otros se inicia la reproducción el archivo.
Sobrescribe	MonoBehaviour.Start()

OnMidiEventPlayed(Object MidiEventPlayedEventArgs)		
Declaración	`private void OnMidiEventPlayed(object sender, MidiEventPlayedEventArgs e)`	
Descripción	Función delegada que se pasa su referencia al objeto _playback para indicar los pasos a seguir cuando se reciba un evento MIDI de un archivo	
Parámetros		
Tipo	Nombre	Descripción
Object	Sender	Objeto que ha detectado el evento
MidiEventPlayedEventArgs	e	Datos sobre el evento que se ha detectado. Esta es una clase importada de Melanchall.DryWetMidi.Multimedia

EnviarEventoNoteOn(Vector3Int)		
Declaración	`public void EnviarEventoNoteOn(Vector3Int EventoNoteOn)`	
Descripción	Función para enviar los datos obtenidos del evento MIDI cuando se ha pulsado una nota. Se envía a los componentes de los GameObjects hijos que implementen la interfaz IConexionManagerFilter.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	EventoNoteOn	Datos que se tienen que enviar al objeto hijo sobre el evento cuando se ha pulsado la tecla. En el vector la primera dimensión es el canal MIDI, la segunda dimensión es el número de nota y la tercera dimensión es la intensidad de la pulsación.

EnviarEventoNoteOff(Vector3Int)		
Declaración	`public void EnviarEventoNoteOff(Vector3Int EventoNoteOff)`	
Descripción	Función para enviar los datos obtenidos del evento MIDI cuando se ha despulsado una nota. Se envía a los componentes de los GameObjects hijos que implementen la interfaz IConexionManagerFilter.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	EventoNoteOff	Datos que se tienen que enviar al objeto hijo sobre el evento cuando se ha despulsado la tecla. En el vector la primera dimensión es el canal MIDI, la segunda dimensión es el número de nota y la tercera dimensión es la intensidad de la pulsación.

InicializarMidiFile()	
Declaración	`private void InicializarMidiFile()`
Descripción	Realiza el proceso de identificación del dispositivo utilizando el selector de recurso de entra SelectorInput y comprueba que hay un archivo valido y accesible que se pueda leer. Si no se puede conectar se muestra un mensaje en consola que no se ha podido conectar y se muestra cómo se tiene que indicar el nombre y dónde se tiene que ubicar el archivo. Si se ha podido conectar se guarda la referencia al archivo y se preparan los recursos necesarios para empezar a escuchar los eventos del archivo.

EmpezarReproduccion ()	
Declaración	`private void EmpezarReproduccion ()`
Descripción	Cuando se ha inicializado la propiedad _playback se puede utilizar esta función para comenzar la reproducción. Se utiliza también para que el script que sincroniza varios archivos pueda iniciar la reproducción del archivo.

OnApplicationQuit ()	
Declaración	`private void OnApplicationQuit()`
Descripción	Si se ha podido realizar la conexión para la lectura de un archivo realiza el proceso para deshabilitar la escucha de eventos y liberar los recursos guardados para la realización de estas operaciones. Este método se ejecuta siempre al terminar la ejecución del programa.
Sobrescribe	MonoBehaviour.OnApplicationQuit()

Clase SelectorRecursoEntradaSO

Descripción

Representa un archivo recipiente donde se pueden guardar las referencias de los nombres de los diferentes dispositivos de entrada o de los archivos que se quieran utilizar en el proyecto.

Herencia

Object→ScriptableObject

Declaración

```
public class SelectorRecursoEntradaSO : ScriptableObject
```

Propiedades

Declaración	Visible en el editor	Descripción
`private List<string> NombresDeRecursos`	Sí	Guarda los nombres de los recursos de entrada. Sirve tanto para guardar los nombres de dispositivos como para guardar las direcciones de los archivos.

Métodos

obtenerNombre(Int)		
Declaración	`public string obtenerNombre(int PosicionDeSeleccion)`	
Descripción	Devuelve el nombre que se ha guardado en la lista de recursos a partir de la posición que se especifique. Como se pide la posición de en la lista, el elemento 0 de la lista será denotado por la posición 1.	
Parámetros		
Tipo	Nombre	Descripción
int	PosicionDeSeleccion	Indica la posición de la lista empezando por la posición 1.
Returns		
Tipo	Descripción	
string	Devuelve el nombre de la posición indicado si el parámetro de entrada es válido. Si no es válido se devuelve la cadena de caracteres “Nombre no Registrado”	

Interfaz IConexionInputOutputDevice

Descripción

Representa una estructura para conectar un objeto que controla los mensajes MIDI de entrada y un objeto que controla la salida del MIDI a dispositivos.

Declaración

```
public interface IConexionInputOutputDevice
```

Métodos

SendEventoMidiNoteOn(MidiEvent)		
Declaración	`public void SendEventoMidiNoteOn(MidiEvent e);`	
Descripción	Envía los datos de un evento de NotaOn que se han producido en un objeto que recibe eventos MIDI y envía el en formato evento MIDI. Este en principio está diseñado para que se envíe el mismo evento que se ha recibido.	
Parámetros		
Tipo	Nombre	Descripción
MidiEvent	e	Objeto que representa el evento MIDI que se quiere transmitir, este tiene toda la información del evento. Esta es una clase importada de Melanchall.DryWetMidi.Core

SendEventoMidiNoteOn(Vector3Int)		
Declaración	`public void SendEventoMidiNoteOn(Vector3Int mensaje);`	
Descripción	Envía los datos de un evento de NotaOn que se han producido en un objeto que recibe eventos MIDI y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene el número de canal, en la segunda se tiene el número de nota y en la última se tiene la velocidad de pulsación. Este en principio está diseñado para cuando solo se tiene la información de canal, número de nota y velocidad y se requiere construir la trama del mensaje MIDI.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	mensaje	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.

SendEventoMidiNoteOff(MidiEvent)		
Declaración	`public void SendEventoMidiNoteOff(MidiEvent e);`	
Descripción	Envía los datos de un evento de NotaOff que se han producido en un objeto que recibe eventos MIDI y envía el en formato evento MIDI. Este en principio está diseñado para que se envíe el mismo evento que se ha recibido	
Parámetros		
Tipo	Nombre	Descripción
MidiEvent	e	Objeto que representa el evento MIDI que se quiere transmitir, este tiene toda la información del evento. Esta es una clase importada de Melanchall.DryWetMidi.Core

SendEventoMidiNoteOff(Vector3Int)		
Declaración	`public void SendEventoMidiNoteOff(Vector3Int mensaje);`	
Descripción	Envía los datos de un evento de NotaOff que se han producido en un objeto que recibe eventos MIDI y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene el número de canal, en la segunda se tiene el número de nota y en la última se tiene la velocidad de pulsación. Este en principio está diseñado para cuando solo se tiene la información de canal, número de nota y velocidad y se requiere construir la trama del mensaje MIDI.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	mensaje	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.

SendEventoCambioDePrograma(int, int)		
Declaración	`public void SendEventoCambioDePrograma(int numeroDeCanal, int numeroDePrograma);`	
Descripción	Envía los datos de un evento de Cambio de programa para cambiar el sonido predefinido que se tiene en el canal. Utiliza el canal al que se quiere enviar el mensaje y el número de programa nuevo que se quiere poner	
Parámetros		
Tipo	Nombre	Descripción
int	numeroDeCanal	Representa el número de canal al que se quiere cambiar el número de programa
int	numeroDePrograma	Representa el nuevo número de programa que se quiere establecer en el canal

Clase SelectorRecursoSalidaSO

Descripción

Representa un recipiente de los nombres y parámetros configurables de un dispositivo de salida que se quieren utilizar en el proyecto

Herencia

Object→ScriptableObject

Declaración

```
public class SelectorRecursoSalidaSO : ScriptableObject
```

Propiedades

Declaración	Visible en el editor	Descripción
`public string NombreDeRecursoSalida;`	Sí	Guarda el nombre del dispositivo de salida que representa este objeto
`private List<string> ConfiguracionDePrograma`	Sí	Guarda el número de programa seleccionado para cada canal (Valor en el intervalo [0,127]) si no se indica en este rango se supone el valor 0

Clase OutputDevices

Descripción

Clase que representa la conexión con un dispositivo de salida.

Herencia

Object → Component → Behaviour → MonoBehaviour

Implementation

IConexionInputOutputDevice

Declaración

```
public class OutputDevices : MonoBehaviour, IConexionInputOutputDevice
```

Propiedades

Declaración	Visible en el editor	Descripción
`private OutputDevice _outputDevice;`	No	Clase importada de la librería Melanchall.DryWetMidi.Multimedia que contiene los métodos para enviar eventos MIDI a un dispositivo MIDI.
`private SelectorRecursoSalidaSO SelectorOutput;`	Sí	Objeto que guarda toda la información necesaria para poder conectarse con un dispositivo de salida
`private bool UsarValorDeVelocidad;`	Sí	Variable que controla si se utiliza el valor de velocidad que viene en los mensajes MIDI o se utiliza el máximo.
`private bool ActualizarNumerosDePrograma;`	Sí	Variable que se utiliza para poder mandar otra vez el mensaje de cambio de programa al dispositivo para que se actualice cuando el programa se está ejecutando
`private BytesToMidiEventConverter bytesToMidi;`	No	Objeto que se importa de la librería Melanchall.DryWetMidi.Core utilizado para convertir los bits de una trama de datos a la estructura de datos MidiEvent.

Métodos

Start()	
Declaración	`void Start()`
Descripción	Realiza el proceso de crear un objeto nuevo de la clase BytesToMidiEventConverter y realiza la gestión de la conexión con el dispositivo de salida. Este método se ejecuta una vez se hayan ejecutado todos los métodos Awake() de los objetos pertenecientes a la escena
Sobrescribe	MonoBehaviour.Start()

Update()	
Declaración	`void Update()`
Descripción	Esta función se ejecuta una vez cada fotograma y sirve para que cuando el parámetro *ActualizarNumerosDePrograma* sea true se envíe los mensajes de cambio de programa a todos los canales.
Sobrescribe	MonoBehaviour.Update()

InicializarOutputDevice ()	
Declaración	`private void InicializarOutputDevice ()`
Descripción	Realiza el proceso de identificación del dispositivo de salida utilizando el selector de recurso de entra *SelectorOutput* y comprueba que hay un dispositivo valido y accesible para conectar. Si no se puede conectar se muestra un mensaje en consola que no se ha podido conectar y se muestra una lista de los nombres de los dispositivos disponibles. Si se ha podido conectar se guarda la referencia al dispositivo de salida y prepara los recursos necesarios para poder enviar los eventos. También manda los mensajes de número de programa que están guardados en el *SelectorOutput*

SendEventoMidiNoteOn(MidiEvent)		
Declaración	`public void SendEventoMidiNoteOn(MidiEvent e);`	
Descripción	Envía los datos de un evento de NotaOn que se han producido en un objeto que recibe eventos MIDI y envía el en formato evento MIDI.	
Parámetros		
Tipo	Nombre	Descripción
MidiEvent	e	Objeto que representa el evento MIDI que se quiere transmitir, este tiene toda la información del evento. Esta es una clase importada de Melanchall.DryWetMidi.Core.
Sobrescribe	IConexionInputOutputDevice.SendEventoMidiNoteOn(MidiEvent)	

SendEventoMidiNoteOn(Vector3Int)		
Declaración	`public void SendEventoMidiNoteOn(Vector3Int mensaje);`	
Descripción	Envía los datos de un evento de NotaOn que se han producido en un objeto que recibe eventos MIDI y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene el número de canal, en la segunda se tiene el número de nota y en la última se tiene la velocidad de pulsación. Se generan los bytes de la trama de datos a partir de la entrada del Vector3Int y se convierten a un evento MIDI con el objeto bytesToMidi	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	mensaje	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionInputOutputDevice.SendEventoMidiNoteOn(Vector3Int)	

SendEventoMidiNoteOff(MidiEvent)		
Declaración	`public void SendEventoMidiNoteOff(MidiEvent e);`	
Descripción	Envía los datos de un evento de NotaOff que se han producido en un objeto que recibe eventos MIDI y envía el en formato evento MIDI.	
Parámetros		
Tipo	Nombre	Descripción
MidiEvent	e	Objeto que representa el evento MIDI que se quiere transmitir, este tiene toda la información del evento. Esta es una clase importada de Melanchall.DryWetMidi.Core
Sobrescribe	IConexionInputOutputDevice.SendEventoMidiNoteOff(MidiEvent)	

SendEventoMidiNoteOff(Vector3Int)		
Declaración	`public void SendEventoMidiNoteOff(Vector3Int mensaje);`	
Descripción	Envía los datos de un evento de NotaOff que se han producido en un objeto que recibe eventos MIDI y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene el número de canal, en la segunda se tiene el número de nota y en la última se tiene la velocidad de pulsación. Se generan los bytes de la trama de datos a partir de la entrada del Vector3Int y se convierten a un evento MIDI con el objeto bytesToMidi	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	Mensaje	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionInputOutputDevice.SendEventoMidiNoteOff(Vector3Int)	

SendEventoCambioDePrograma(int, int)		
Declaración	`public void SendEventoCambioDePrograma(int numeroDeCanal, int numeroDePrograma);`	
Descripción	Envía los datos de un evento de Cambio de programa para cambiar el sonido predefinido que se tiene en el canal. Utiliza el canal al que se quiere enviar el mensaje y el número de programa nuevo que se quiere poner. Se generan los bytes de la trama de datos MIDI y se utiliza el objeto bytesToMidi para convertirlos.	
Parámetros		
Tipo	Nombre	Descripción
int	numeroDeCanal	Representa el número de canal al que se quiere cambiar el número de programa
int	numeroDePrograma	Representa el nuevo número de programa que se quiere establecer en el canal
Sobrescribe	IConexionInputOutputDevice.SendEventoCambioDePrograma(int, int)	

ActualizarNumeroDePrograma ()	
Declaración	`public void ActualizarNumeroDePrograma()`
Descripción	Realiza el proceso de mandar un mensaje de programa a todos los canales de la trama MIDI

OnApplicationQuit ()	
Declaración	`private void OnApplicationQuit()`
Descripción	Si se ha podido realizar la conexión para la lectura de un archivo realiza el proceso para deshabilitar la escucha de eventos y liberar los recursos guardados para la realización de estas operaciones. Este método se ejecuta siempre al terminar la ejecución del programa.
Sobrescribe	MonoBehaviour. OnApplicationQuit()

Interfaz IConexionManagerFilter

Descripción

Representa una estructura para conectar un objeto que controla los mensajes MIDI de entrada de un dispositivo o archivo MIDI a un módulo de filtrado de eventos.

Declaración

```
public interface IConexionManagerFilter
```

Métodos

EventoMidiNoteOn(`Vector3Int`)		
Declaración	`public void EventoMidiNoteOn(Vector3Int NoteOn);`	
Descripción	Envía los datos de un evento de NotaOn que se han producido en un objeto que recibe eventos MIDI y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene el número de canal, en la segunda se tiene el número de nota y en la última se tiene la velocidad de pulsación.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	NoteOn	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.

EventoMidiNoteOff(Vector3Int)		
Declaración	`public void EventoMidiNoteOff(Vector3Int NoteOff);`	
Descripción	Envía los datos de un evento de NotaOff que se han producido en un objeto que recibe eventos MIDI y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene el número de canal, en la segunda se tiene el número de nota y en la última se tiene la velocidad de pulsación.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	NoteOff	Objeto que representa el evento MIDI para transmitir, con información del canal, número de nota e intensidad de pulsación.

Clase InputFilter

Descripción

Clase que representa el módulo de filtrado de los mensajes de eventos MIDI.

Herencia

Object → Component → Behaviour → MonoBehaviour

Implementation

IConexionManagerFilter

Declaración

```
public class InputFilter : MonoBehaviour, IConexionManagerFilter
```

Propiedades

Declaración	Visible en el editor	Descripción
`private bool FiltrarPorCanal;`	Sí	Variable que habilita la posibilidad de filtrar por canal
`private Vector2Int _RangoCanalAFiltrar;`	Sí	Variable que representa el valor mínimo y máximo en el que se filtra el canal
`private bool FiltrarPorOctava;`	Sí	Variable que habilita la posibilidad de filtrar por octava
`private Vector2Int _RangoOctavaAFiltrar;`	Sí	Variable que representa el valor mínimo y máximo en el que se filtra por octava
`private bool HabilitarSalidaMIDI;`	Sí	Variable que habilita el envío de los mensajes MIDI a la salida
`private GameObject objetoSalidaMIDI;`	No	Objeto que tiene los recursos de salida para enviar mensaje en dispositivos de salida
`private IConexionInputOutputDevice _referenciaOutput;`	No	Objeto que tiene el componente de la salida de los ventos MIDI
`private IConexionFilterPreprocesado _referencia;`	No	Referencia para mandar mensajes al módulo de preprocesado
`private Queue<Vector3Int> _NotasOn;`	No	Variable que guarda los mensajes que se reciben de Nota On y los mantienen hasta que se mandan
`private Queue<Vector3Int> _NotasOff;`	No	Variable que guarda los mensajes que se reciben de Nota Off y los mantienen hasta que se mandan

Métodos

Start()	
Declaración	`void Start()`
Descripción	Realiza la gestión de obtener las referencias a los objetos y crea los objetos que van a servir como colas. También se inicia el tratamiento del desencolado de los mensajes usando corrutinas. Este método se ejecuta una vez se hayan ejecutado todos los métodos Awake() de los objetos pertenecientes a la escena
Sobrescribe	MonoBehaviour.Start()

EventoMidiNoteOn(Vector3Int)		
Declaración	`public void EventoMidiNoteOn(Vector3Int NoteOn);`	
Descripción	Envía los datos de un evento de Nota On que se han producido en un objeto que recibe eventos MIDI y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene el número de canal, en la segunda se tiene el número de nota y en la última se tiene la velocidad de pulsación.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	NoteOn	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionManagerFilter.EventoMidiNoteOn(Vector3Int)	

EventoMidiNoteOff(Vector3Int)		
Declaración	`public void EventoMidiNoteOff(Vector3Int NoteOff);`	
Descripción	Envía los datos de un evento de Nota Off que se han producido en un objeto que recibe eventos MIDI y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene el número de canal, en la segunda se tiene el número de nota y en la última se tiene la velocidad de pulsación.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	NoteOff	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionManagerFilter.EventoMidiNoteOff(Vector3Int)	

StartManejoDeColas ()	
Declaración	`private void StartManejoDeColas()`
Descripción	Realiza el proceso de empezar las corrutinas para desencolar los mensajes de las colas.

DesEncoladoDeNotaOn()	
Declaración	`private IEnumerator DesEncoladoDeNotaOn()`
Descripción	Función que sirve para desencolar de forma rutinaria siempre que haya algún mensaje que mandar de notaOn
Returns	
Tipo	Descripción
IEnumerator	Devuelve variable que sirve para manejar las corrutinas

DesEncoladoDeNotaOff()	
Declaración	`private IEnumerator DesEncoladoDeNotaOff()`
Descripción	Función que sirve para desencolar de forma rutinaria siempre que haya algún mensaje que mandar de notaOff
Returns	
Tipo	Descripción
IEnumerator	Devuelve variable que sirve para manejar las corrutinas

OnApplicationQuit ()	
Declaración	`private void OnApplicationQuit()`
Descripción	Para la gestión de corrutinas para liberar el espacio de los recursos que están consumiendo
Sobrescribe	MonoBehaviour.OnApplicationQuit()

Interfaz IConexionFilterPreprocesado

Descripción

Representa una estructura para conectar un objeto que filtra los mensajes MIDI a un módulo que procesa los datos.

Declaración

```
public interface IConexionFilterPreprocesado
```

Métodos

NotaPulsada(Vector3Int)		
Declaración	`public void NotaPulsada(Vector3Int pulsacion);`	
Descripción	Envía los datos de un evento de NotaOn que se han filtrado y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene la nota pulsada, en la segunda se tiene la octava a la que pertenece y en la última se tiene la velocidad de pulsación.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	pulsacion	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.

NotaDesPulsada (Vector3Int)		
Declaración	`public void NotaDesPulsada(Vector3Int pulsacion);`	
Descripción	Envía los datos de un evento de NotaOff que se han filtrado y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene la nota pulsada, en la segunda se tiene la octava a la que pertenece y en la última se tiene la velocidad de pulsación.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	pulsacion	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.

Clase ReferenciasNotasOctavasSO

Descripción

Representa un recipiente de los nombres de las variables de los Shaders que se comunican con los scripts Bypass y FadeOut.

Herencia

Object→ScriptableObject

Declaración

```
public class ReferenciasNotasOctavasSO : ScriptableObject
```

Propiedades

Declaración	Visible en el editor	Descripción
`private List<string> _referenciasNotas`	Sí	Guarda los nombres de las variables de notas de los Shaders que se quieren controlar
`private List<string> _referenciasOctavas`	Sí	Guarda los nombres de las variables de octavas de los Shaders que se quieren controlar

Métodos

Awake()	
Declaración	`void Awake()`
Descripción	Comprueba que se hayan inicializado el suficiente número de variables para que se pueda utilizar correctamente. Tienen que haber 12 referencias a notas y 11 referencias a octavas.
Sobrescribe	ScriptableObject.Awake()

getReferenciaNota(int)		
Declaración	`public string getReferenciaNota(int Nota)`	
Descripción	Devuelve el nombre de la variable de la nota que esta guardada según se especifique. 0 es la referencia a la nota Do (C) y 11 es la referencia a la nota Si(B)	
Parámetros		
Tipo	Nombre	Descripción
int	Nota	Número de nota al que se le asocia el nombre de la variable
Returns		
Tipo	Descripción	
string	Devuelve el nombre de las variables de los Shaders que se quiere modificar	

getReferenciaOctava(int)		
Declaración	`public string getReferenciaOctava(int Octava)`	
Descripción	Devuelve el nombre de la variable de la nota que esta guardada según especifica. -1 es la referencia a la octava más baja y 9 es la referencia a la octava más alta	
Parámetros		
Tipo	Nombre	Descripción
int	Octava	Número de octava al que se le asocia el nombre de la variable
Returns		
Tipo	Descripción	
string	Devuelve el nombre de las variables de los Shaders que se quiere modificar	

Clase Bypass

Descripción

Clase para gestionar los valores de las variables de los Shaders transformando los valores que se reciben de los mensajes MIDI después de ser filtrados. Solo ofrecen valores de 1 o 0 si esta pulsada una nota o no.

Herencia

Object →Component → Behaviour → MonoBehaviour

Implementación

IConexionFilterPreprocesado

Declaración

```
public class Bypass : MonoBehaviour, IConexionFilterPreprocesado
```

Propiedades

Declaración	Visible en el editor	Descripción
`private ReferenciasNotasOctavasSO _referencias;`	Sí	Contenedor donde están todas las referencias de las variables.
`private MaterialPropertyBlock _propertyBlock;`	No	Sirve para poder tener un material diferente en cada usando el mismo Shader objeto cuando se empieza a ejecutar el programa
`private Renderer _renderer;`	No	Este sirve para gestionar los materiales que pertenecen al GameObject en el que está el componente

Métodos

Awake()	
Declaración	`void Awake()`
Descripción	Obtiene la referencia al renderer del GameObject y crea un nuevo objeto de la clase MaterialPropertyBlock. Este se ejecuta al principio del programa antes del método Start()
Sobrescribe	MonoBehaviour. Awake()

NotaPulsada(Vector3Int)		
Declaración	`public void NotaPulsada(Vector3Int pulsacion)`	
Descripción	Envía los datos de un evento de Nota On que se han filtrado y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene la nota pulsada, en la segunda se tiene la octava a la que pertenece y en la última se tiene la velocidad de pulsación.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	pulsación	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionFilterPreprocesado.NotaPulsada(Vector3Int)	

NotaDesPulsada (Vector3Int)		
Declaración	`public void NotaDesPulsada(Vector3Int pulsacion);`	
Descripción	Envía los datos de un evento de Nota Off que se han filtrado y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene la nota pulsada, en la segunda se tiene la octava a la que pertenece y en la última se tiene la velocidad de pulsación.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	pulsacion	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionFilterPreprocesado.NotaDesPulsada(Vector3Int)	

Clase FadeOut

Descripción

Clase para gestionar los valores de las variables de los Shaders transformando los valores que se reciben de los mensajes MIDI después de ser filtrados. Ofrecen valores entre 0 y 1 si esta pulsada una nota o no.

Herencia

Object → Component → Behaviour → MonoBehaviour

Implementación

IConexionFilterPreprocesado

Declaración

```
public class Bypass : MonoBehaviour, IConexionFilterPreprocesado
```

Propiedades

Declaración	Visible en el editor	Descripción
`private ReferenciasNotasOctavasSO _referencias;`	Sí	Contenedor donde están todas las referencias de las variables.
`private MaterialPropertyBlock _propertyBlock;`	No	Sirve para poder tener un material diferente en cada objeto cuando se empieza a ejecutar el programa
`private Renderer _renderer;`	No	Este sirve para gestionar los materiales que pertenecen al GameObject en el que está el componente
`float _segundosFadeOut;`	Sí	Indica el tiempo que tiene que durar el desvanecimiento del valor de la nota
`private const float NUMERO_DE_PASOS;`	No	Indica el número de pasos de desvanecimiento que es necesario ejecutar. Cuantos más suave parecerá la transición
`private readonly bool[] flagNotas;`	No	Indica si se está pulsando una determinada nota en algún momento
`private readonly bool[] flagOctavas;`	No	Indica si se está pulsando una nota en la determinada octava en algún momento
`private int[] _pulsacionesOctava;`	No	Indica el número de notas que se están pulsando en una misma octava en algún momento

Métodos

Awake()	
Declaración	`void Awake()`
Descripción	Obtiene la referencia al renderer del GameObject y crea un nuevo objeto de la clase MaterialPropertyBlock. Este se ejecuta al principio del programa antes del método Start()
Sobrescribe	MonoBehaviour. Awake()

NotaPulsada(Vector3Int)		
Declaración	`public void NotaPulsada(Vector3Int pulsacion)`	
Descripción	Envía los datos de un evento de Nota On que se han filtrado y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene la nota pulsada, en la segunda se tiene la octava a la que pertenece y en la última se tiene la velocidad de pulsación.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	pulsación	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionFilterPreprocesado.NotaPulsada(Vector3Int)	

NotaDesPulsada (Vector3Int)		
Declaración	`public void NotaDesPulsada(Vector3Int pulsacion);`	
Descripción	Envía los datos de un evento de Nota Off que se han filtrado y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene la nota pulsada, en la segunda se tiene la octava a la que pertenece y en la última se tiene la velocidad de pulsación.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	pulsacion	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionFilterPreprocesado.NotaDesPulsada(Vector3Int)	

FadeNota(int)		
Declaración	`private IEnumerator FadeNota(int _indiceNota)`	
Descripción	Función que sirve para reducir el valor de la variable de la nota poco a poco en el tiempo especificado por _segundosFadeOut y en el NUMERO_DE_PASOS	
Parámetros		
Tipo	Nombre	Descripción
Int	_indiceOctava	Indica la posición del array en las referencias de Octavas
Returns		
Tipo	Descripción	
IEnumerator	Devuelve variable que sirve para manejar las corrutinas	

FadeOctava(int)		
Declaración	`private IEnumerator FadeOctava(int _indiceOctava)`	
Descripción	Función que sirve para desencolar de forma rutinaria siempre que haya algún mensaje que mandar de notaOn	
Parámetros		
Tipo	Nombre	Descripción
Int	_indiceNota	Indica la posición del array en las referencias de notas
Returns		
Tipo	Descripción	
IEnumerator	Devuelve variable que sirve para manejar las corrutinas	

Namespace OpcionesConexionPreprocesado

Descripción

Define los tipos enumerados que representan las opciones de configuración de ShaderConexionSO y AnimatorConexionSO.

```
namespace OpcionesConexionPreprocesado
{
    public enum OpcionesDeCalculo
    {
        Media,
        Max,
        Min,
        UltimaPulsada,
        UltimaDesPulsada
    }
    public enum OpcionesDeRemapAnimator
    {
        ModuloReescalado,
        ClampReescalado,
        ReescaladoClamp
    }
    public enum OpcionesDeRemapShader
    {
        ModuloReescalado,
        ClampReescalado,
        ReescaladoClamp,
        Vector2Reescalado,
        Vector3Reescalado,
        Vector4Reescalado
    }
    public enum TipoSalidaAnimator
    {
        Int,
        Float,
        Bool,
        Trigger
    }
    public enum TipoSalidaShader
    {
        Int,
        Float,
        Vector,
        Color
    }
    public enum ZonaEjecucion
    {
        EnNotaOn,
        EnNotaOff,
        EnAmbas
    }
}
```

Clase AnimatorConexionSO

Descripción

Clase que representa un contenedor de información con todos los parámetros de configuración de conexión entre el *AnimatorPreprocesado* y el componente *Animator*.

Herencia

Object → ScriptableObject

Declaración

```
public class AnimatorConexionSO : ScriptableObject
```

Propiedades

Declaración	Visible en el editor	Descripción
`private bool ConOctavas;`	Sí	Variable que controla si los cálculos se hacen teniendo en cuenta el número de nota [0,127] y por tanto distinguiendo las notas según su octava o solo la nota [0,11]
`public List<string> nombresDeSalida;`	Sí	Nombres de la variable que se quiere modificar en el Animator
`public List<TipoSalidaAnimator> tipoSalida;`	Sí	Configuración del tipo de variable que se quiere modificar en el Animator
`public List<ZonaEjecucion> zonaEjecucion;`	Sí	Configuración del lugar de ejecución en la que se quiere actualizar el valor que se quiere modificar en el Animator
`public List<OpcionesDeCalculo> opcionesDeCalculo;`	Sí	Configuración del tipo de descriptor que se quiere extraer para realizar el cálculo que se quiere modificar en el Animator
`public List<OpcionesDeRemapAnimator> opcionesDeRemap;`	Sí	Configuración del tipo operación que se quiere realizar para modificar los valores en el Animator
`public List<Vector4> remap;`	Sí	Valores que se van a utilizar para el cálculo del remapeo
`public int Count;`	No	Número de elementos de las listas. Si se ha inicializado correctamente este es igual a el tamaño que tienen todas las listas de las variables anteriores
`private int[] auxMaxMin;`	No	Array que guarda el número de veces que se ha pulsado una nota
`private float Max;`	No	Valor de la nota más pulsada
`private int indiceMax;`	No	Índice del array auxMaxMin donde se encuentra la nota más pulsada
`private float Min;`	No	Valor de la nota menos pulsada
`private int indiceMin;`	No	Índice del array auxMaxMin donde se encuentra la nota menos pulsada
`private float Media;`	No	Valor de la media de los valores de notas que se han pulsado
`private float UltimaPulsada;`	No	Valor de la última nota pulsada
`private float UltimaDesPulsada;`	No	Valor de la última nota despulsada

Métodos

Inicializar()	
Declaración	`public bool Inicializar()`
Descripción	Realiza la inicialización de las variables de cálculo que sirven para dar los valores que se envía al Animator
Returns	
Tipo	Descripción
bool	Indica si se ha podido realizar la inicialización de las variables de cálculo de forma correcta

ActualizarEnNotaOn(Vector3Int)		
Declaración	`public void ActualizarEnNotaOn(Vector3Int notaOn)`	
Descripción	Actualiza las variables de cálculo cuando se recibe un mensaje de Nota On	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	notaOn	Mensaje de Nota On recibido para que sea procesado por este objeto

ActualizarEnNotaOff(Vector3Int)		
Declaración	`public void ActualizarEnNotaOff(Vector3Int notaOff)`	
Descripción	Actualiza las variables de cálculo cuando se recibe un mensaje de Nota Off	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	notaOff	Mensaje de Nota Off recibido para que sea procesado por este objeto

getNombre(int)		
Declaración	`public string getNombre(int num)`	
Descripción	Función para obtener el nombre de la variable que se quieren modificar del Animator	
Parámetros		
Tipo	Nombre	Descripción
Int	num	Entero que indica el índice de la lista del nombre de variable que se quiere conseguir
Returns		
Tipo	Descripción	
string	Devuelve el nombre de la variable que se quiere conseguir	

getValor(int)		
Declaración	`public float getValor(int num)`	
Descripción	Obtiene el valor de la variable que quiera según la configuración que se haya realizado	
Parámetros		
Tipo	Nombre	Descripción
int	num	Entero que indica el índice de la lista del nombre de variable que se quiere conseguir el valor
Returns		
Tipo	Descripción	
Float	Devuelve el valor obtenido del cálculo	

variableCalculo(int)		
Declaración	`private float variableCalculo(int num)`	
Descripción	Función que sirve para obtener la variable de cálculo que se ha elegido según configuración para el índice de la lista indicado	
Parámetros		
Tipo	Nombre	Descripción
int	num	Entero que indica el índice de la lista del nombre de variable que se quiere conseguir el valor
Returns		
Tipo	Descripción	
Float	Devuelve la variable del cálculo correspondiente	

remapear(float, int)		
Declaración	`private float remapear(float numero, int num)`	
Descripción	Función que sirve para obtener el remapeo que se ha elegido según la configuración para el índice de la lista indicado	
Parámetros		
Tipo	Nombre	Descripción
float	numero	Es el número al que se quiere realizar la operación de remapeo
int	num	Entero que indica el índice de la lista del nombre de variable que se quiere conseguir el valor
Returns		
Tipo	Descripción	
Float	Devuelve el resultado después del cálculo correspondiente al remapeo	

gestionarMaxMin(int)		
Declaración	`private void gestionarMaxMin(int valor)`	
Descripción	Función auxiliar que realiza el proceso de cálculo de la nota más pulsada y la menos pulsada	
Parámetros		
Tipo	Nombre	Descripción
int	valor	Valor necesario para actualizar los valores de nota más pulsada y menos pulsada.

Clase ShaderConexionSO

Descripción

Clase que representa un contenedor de información con todos los parámetros de configuración de conexión entre el *ShaderPreprocesado* y los Shaders.

Herencia

Object → ScriptableObject

Declaración

```
public class ShaderConexionSO : ScriptableObject
```

Propiedades

Declaración	Visible en el editor	Descripción
`private bool ConOctavas;`	Sí	Variable que controla si los cálculos se hacen teniendo en cuenta el número de nota [0,127] y por tanto distinguiendo las notas según su octava o solo la nota [0,11]
`private float RapidezTransicion;`	Sí	Valor de [0,1] que controla cuanto de rápido se realiza la transición de un valor al siguiente cuando se utiliza los valores por frame
`public List<string> nombresDeSalida;`	Sí	Nombres de la variable que se quiere modificar en el Shader
`public List<TipoSalidaShader> tipoSalida;`	Sí	Configuración del tipo de variable que se quiere modificar en el Shader
`public List<ZonaEjecucion> zonaEjecucion;`	Sí	Configuración del lugar de ejecución en la que se quiere actualizar el valor que se quiere modificar en el Shader
`public List<OpcionesDeCalculo> opcionesDeCalculo;`	Sí	Configuración del tipo de descriptor que se quiere extraer para realizar el cálculo que se quiere modificar en el Shader
`public List<OpcionesDeRemapShader> opcionesDeRemap;`	Sí	Configuración del tipo operación que se quiere realizar para modificar los valores en el Shader
`public List<Vector4> remap;`	Sí	Valores que se van a utilizar para el cálculo del remapeo
`public int Count;`	No	Número de elementos de las listas. Si se ha inicializado correctamente este es igual a el tamaño que tienen todas las listas de las variables anteriores
`private int[] auxMaxMin;`	No	Array que guarda el número de veces que se ha pulsado una nota
`private float Max;`	No	Valor de la nota más pulsada
`private float MaxPorFrame;`		Valor de la nota más pulsada. Este valor se va actualizando en cada frame hasta que su valor es igual al de la variable Max.
`private int indiceMax;`	No	Índice del array auxMaxMin donde se encuentra la nota más pulsada
`private float Min;`	No	Valor de la nota menos pulsada
`private float MinPorFrame;`	No	Valor de la nota menos pulsada. Este valor se va actualizando en cada frame hasta que su valor es igual al de la variable Min.

Declaración	Visible en el editor	Descripción
`private int indiceMin;`	No	Índice del array auxMaxMin donde se encuentra la nota menos pulsada
`private float Media;`	No	Valor de la media de los valores de notas que se han pulsado
`private float MediaPorFrame;`	No	Valor de la media de los valores de notas que se han pulsado. Este valor se va actualizando en cada frame hasta que su valor es igual al de la variable Media.
`private float UltimaPulsada;`	No	Valor de la última nota pulsada
`private float UltimaPulsadaPorFrame;`	No	Valor de la última nota pulsada. Este valor se va actualizando en cada frame hasta que su valor es igual al de la variable UltimaPulsada.
`private float UltimaDesPulsada;`	No	Valor de la última nota despulsada
`private float UltimaDesPulsadaPorFrame;`	No	Valor de la última nota despulsada. Este valor se va actualizando en cada frame hasta que su valor es igual al de la variable UltimaDesPulsada.

Métodos

Inicializar()	
Declaración	`public bool Inicializar()`
Descripción	Realiza la inicialización de las variables de cálculo que sirven para dar los valores que se envía al Shader
Returns	
Tipo	Descripción
Bool	Indica si se ha podido realizar la inicialización de las variables de cálculo de forma correcta

ActualizarEnNotaOn(Vector3Int)		
Declaración	`public void ActualizarEnNotaOn(Vector3Int notaOn)`	
Descripción	Actualiza las variables de cálculo cuando se recibe un mensaje de Nota On	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	notaOn	Mensaje de Nota On recibido para que sea procesado por este objeto

ActualizarEnNotaOff(Vector3Int)		
Declaración	`public void ActualizarEnNotaOff(Vector3Int notaOff)`	
Descripción	Actualiza las variables de cálculo cuando se recibe un mensaje de Nota Off	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	notaOff	Mensaje de Nota Off recibido para que sea procesado por este objeto

getNombre(int)		
Declaración	`public string getNombre(int num)`	
Descripción	Función para obtener el nombre de la variable que se quieren modificar del Animator	
Parámetros		
Tipo	Nombre	Descripción
Int	num	Entero que indica el índice de la lista del nombre de variable que se quiere conseguir
Returns		
Tipo	Descripción	
string	Devuelve el nombre de la variable que se quiere conseguir	

getValor(int, bool)		
Declaración	`public Vector4 getValor(int num, bool PorFrame)`	
Descripción	Obtiene el valor de la variable que quiera según la configuración que se haya realizado	
Parámetros		
Tipo	Nombre	Descripción
int	num	Entero que indica el índice de la lista del nombre de variable que se quiere conseguir el valor
bool	PorFrame	Indica si se tienen que devolver las variables que se utilizan por frame o no.
Returns		
Tipo	Descripción	
Vector4	Devuelve el valor obtenido del cálculo	

variableCalculo(int)		
Declaración	`private float variableCalculo(int num)`	
Descripción	Función que sirve para obtener la variable de cálculo que se ha elegido según configuración para el índice de la lista indicado	
Parámetros		
Tipo	Nombre	Descripción
int	num	Entero que indica el índice de la lista del nombre de variable que se quiere conseguir el valor
Returns		
Tipo	Descripción	
float	Devuelve la variable del cálculo correspondiente	

variableCalculoPorFrame(int)		
Declaración	`private float variableCalculoPorFrame(int num)`	
Descripción	Función que sirve para obtener la variable de cálculo que se ha elegido según configuración para el índice de la lista indicado cuando el cálculo se hace en cada fotograma	
Parámetros		
Tipo	Nombre	Descripción
int	num	Entero que indica el índice de la lista del nombre de variable que se quiere conseguir el valor
Returns		
Tipo	Descripción	
float	Devuelve la variable del cálculo correspondiente	

remapear(float, int)		
Declaración	`private Vector4 remapear(float numero, int num)`	
Descripción	Función que sirve para obtener el remapeo que se ha elegido según la configuración para el índice de la lista indicado	
Parámetros		
Tipo	Nombre	Descripción
float	numero	Es el número al que se quiere realizar la operación de remapeo
int	num	Entero que indica el índice de la lista del nombre de variable que se quiere conseguir el valor
Returns		
Tipo	Descripción	
Vector4	Devuelve el resultado después del cálculo correspondiente al remapeo	

gestionarMaxMin(int)		
Declaración	`private void gestionarMaxMin(int valor)`	
Descripción	Función auxiliar que realiza el proceso de cálculo de la nota más pulsada y la menos pulsada	
Parámetros		
Tipo	Nombre	Descripción
int	valor	Valor necesario para actualizar los valores de nota más pulsada y menos pulsada.

Clase AnimatorPreprocesado

Descripción

Realiza la gestión del cálculo del módulo de procesado y correcta comunicación con el componente del *Animator* teniendo en cuenta las configuraciones realizadas un objeto de tipo *AnimatorConexionSO*.

Herencia

Object → Component → Behaviour → MonoBehaviour

Implementación

IConexionFilterPreprocesado

Declaración

```
public class AnimatorPreprocesado : MonoBehaviour, IConexionFilterPreprocesado
```

Propiedades

Declaración	Visible en el editor	Descripción
`private Animator _animator;`	Sí	Variable que guarda la referencia al componente Animator
`private AnimatorConexionSO _animatorConexion;`	Sí	Referencia al objeto que guarda las opciones de configuración y realiza el cálculo de las variables.

Métodos

Start()	
Declaración	`void Start()`
Descripción	Inicializa los parámetros de cálculo del _animatorConexión y comprueba que se haya podido realizar correctamente. Si no se ha podido se manda un mensaje de error para informar de que no funcionara de forma correcta. Este método se ejecuta una vez se hayan ejecutado todos los métodos Awake() de los objetos pertenecientes a la escena
Sobrescribe	MonoBehaviour.Start()

NotaPulsada(Vector3Int)		
Declaración	`public void NotaPulsada(Vector3Int pulsacion)`	
Descripción	Envía los datos de un evento de Nota On que se han filtrado y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene la nota pulsada, en la segunda se tiene la octava a la que pertenece y en la última se tiene la velocidad de pulsación. Actualiza el valor de las variables y luego si se ha configurado de esa forma se envía una actualización a la variable del Animator.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	pulsación	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionFilterPreprocesado.NotaPulsada(Vector3Int)	

NotaDesPulsada (Vector3Int)		
Declaración	`public void NotaDesPulsada(Vector3Int pulsacion);`	
Descripción	Envía los datos de un evento de Nota Off que se han filtrado y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene la nota pulsada, en la segunda se tiene la octava a la que pertenece y en la última se tiene la velocidad de pulsación. Actualiza el valor de las variables y luego si se ha configurado de esa forma se envía una actualización a la variable del Animator.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	pulsacion	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionFilterPreprocesado.NotaDesPulsada(Vector3Int)	

selector(int)		
Declaración	`private void selector(int num)`	
Descripción	Función que realiza el proceso de comunicación entre este script y el animator y manda el mensaje de actualización de la variable denominada por el índice de la lista que se ha dicho.	
Parámetros		
Tipo	Nombre	Descripción
int	num	Índice de la lista que indica la variable que se quiere modificar en el Animator

Clase ShaderPreprocesado

Descripción

Realiza la gestión del cálculo del módulo de procesado y correcta comunicación con el componente del *Shader* teniendo en cuenta las configuraciones realizadas un objeto de tipo *ShaderConexionSO*.

Herencia

Object → Component → Behaviour → MonoBehaviour

Implementación

IConexionFilterPreprocesado

Declaración

```
public class ShaderPreprocesado : MonoBehaviour, IConexionFilterPreprocesado
```

Propiedades

Declaración	Visible en el editor	Descripción
`private Material _material;`	Sí	Variable que guarda la referencia al componente que se comunica con el Shader del GameObject
`private ShaderConexionSO _shaderConexion;`	Sí	Referencia al objeto que guarda las opciones de configuración y realiza el cálculo de las variables.
`private bool PorFrame;`	Sí	Variable que controla si se actualiza los valores de las variables de los Shaders en cada frame

Métodos

Start()	
Declaración	`void Start()`
Descripción	Inicializa los parámetros de cálculo del _shaderConexión y comprueba que se haya podido realizar correctamente. Si no se ha podido se manda un mensaje de error para informar de que no funcionara de forma correcta. Este método se ejecuta una vez se hayan ejecutado todos los métodos Awake() de los objetos pertenecientes a la escena
Sobrescribe	MonoBehaviour.Start()

Update()	
Declaración	`void Update()`
Descripción	Función llamada una vez cada frame. Si se tiene seleccionada la opción de por frame se realiza el proceso de comunicación con el Shader utilizando las variables por frame. Si no está seleccionada la opción no se hace nada.
Sobrescribe	MonoBehaviour.Update()

NotaPulsada(Vector3Int)		
Declaración	`public void NotaPulsada(Vector3Int pulsacion)`	
Descripción	Envía los datos de un evento de Nota On que se han filtrado y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene la nota pulsada, en la segunda se tiene la octava a la que pertenece y en la última se tiene la velocidad de pulsación. Actualiza el valor de las variables y luego si se ha configurado de esa forma se envía una actualización a la variable del Shader.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	pulsación	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionFilterPreprocesado.NotaPulsada(Vector3Int)	

NotaDesPulsada (Vector3Int)		
Declaración	`public void NotaDesPulsada(Vector3Int pulsacion);`	
Descripción	Envía los datos de un evento de Nota Off que se han filtrado y envía el en formato de un vector de tres dimensiones. En la primera dimensión se tiene la nota pulsada, en la segunda se tiene la octava a la que pertenece y en la última se tiene la velocidad de pulsación. Actualiza el valor de las variables y luego si se ha configurado de esa forma se envía una actualización a la variable del Shader.	
Parámetros		
Tipo	Nombre	Descripción
Vector3Int	pulsacion	Objeto que representa el evento MIDI que se quiere transmitir, este solo tiene información del canal, número de nota e intensidad de pulsación del evento.
Sobrescribe	IConexionFilterPreprocesado.NotaDesPulsada(Vector3Int)	

selector(int, bool)		
Declaración	`private void selector(int num, bool Frame)`	
Descripción	Función que realiza el proceso de comunicación entre este script y el Shader y manda el mensaje de actualización de la variable denominada por el índice de la lista que se ha dicho.	
Parámetros		
Tipo	Nombre	Descripción
int	num	Índice de la lista que indica la variable que se quiere modificar en el Shader
Bool	Frame	Indica si la función se ha llamado en el proceso de frame a frame o normal para condicionar su funcionamiento