NEMIISINDO UNITY PLUGINS USAGE GUIDE V1

The Package comes with a set of .dll files for windows or with a set of .bundle files for mac osx and a bunch of c# scripts for UI, Serialisation and Control.

The Below Is a step by step guide useful for the initial set of the plugin pack.

Step 1: From the **Assets** menu access **import Package,** From the file explorer select the .unitypackage file that was provided for testing.

This should open up an import **unity package** dialog box showing you the contents of the .unitypackage that you are trying to import. Do not unselect anything and to ensure everything is selected you can :

Step 2: click on **all** on the bottom left corner. Now click on the **import button** in this dialog box.

This will load up all the necessary files into the current unity project.

Navigate to the audio mixer section and in each mixer channel an audio plugin is loaded and they are routed into the master audio channel.

Contents of the Package:

1) .bundle (MAC os x) or .dll (windows) file: This is the actual Native Audio Plugin.
Note: unity requires the audio plugin to be named starting with audioplugin_ so do not rename the plugin file name

This package also inlcludes Android and iOS builds in the respective folders

2) audioPluginName_UI.cs : This is the C# script for the UI. Do not change anything or it might break the UI. However towards the end of this file there is a properly commented presets section and you can save your own presets in here. You can find instructions for that towards the end of this document.

3) Serialisation file: This is named as audioPlugin_name_Unity.cs. The main purpose of this script is to enable data serialisation between the UI and the dsp core of the plugin via audio parameters. Kindly do not make any changes to the script.

4) Controller file: this is named as pluginName_controller.cs. This is created and provided for conveniently interacting with the plugin and changing its parameters using the unity scripting interface. Each plugin has a set of functions that can be used to control the sliders, triggers and dropdown boxes in the plugin and all the functions can be referred to from below.
Note: The controller scripts can be added as a component to any object. It can be found under a nemisdo submenu under when you click on add component or from the menu bar at [component]->[Nemisindo]

5) The Audio Init Script: this can be found inside the scripts folder. Add this script to at least one audio source attached to every audio mixer to enable sound from the native audio plugin. This can be accessed from the nemisindo submenu under the component menu [component]->[Nemisindo].

API:

Every plugin is provided with a `setFloatValue(int channel, int index, float value)` function and a `getFlowValue(int channel, int index)` function to enable easy communication between the plugin via scripting.

`setFloatValue(int channel, int index, float value)`

Channel is the current plugin instance that can be explicitly mentioned by the user. Each plugin can have a different or the same instance and the parameters in that particular instance can be accessed by mentioning the plugin instance value here.

There are a total of 32 channels specified which means that there can be 32 different instances of any plugin running at once in your project.

Index is index of the plugin parameter, this is marked in the plugin interface right in front of the parameters name.This is used to access that specific parameter in the chosen plugin instance.

Value is used to set the value of the chosen audio parameter.

Eg:

- For example to set the carrier frequency of the 0th instance of an alarm plugin to 1500Hz.
  `setFloatValue(0,1,1500)` can be used

- Similarly to get the carrier frequency of the same alarm plugin
  `getFloatValue(0,1)` can be used

In addition to the setFloatValue function you can also additionally address each of these parameters specifically using the **plugin dedicated functions listed below:**

Footsteps:

```
setPace             (float channel, float value); // Set the speed of the
footsteps
setFirmness         (float channel, float value); // Set the firmness of each step
setSteadiness       (float channel, float value); // Set steadiness of pace
setShoeType         (float channel, float value); // Choose from 5 shoe types
setSurfaceType      (float channel, float value); // Choose from 7 surface types
seTerrainType       (float channel, float value); // Choose between flat or uphill
setVolume           (float channel, float value); // Set overall volume
setStart            (float channel);                    // automate the footstep
movement
setStop             (float channel);                    // stop the footsteps
setTrigger          (float channel);                    // trigger each footstep

//Shoe type 5 is custom shoe where you can shape your shoe using these parameters
below
setHeelGain         (float channel, float value);
setHeelAttack       (float channel, float value);
setHeelDecay        (float channel, float value);
setHeelSustain      (float channel, float value);
setHeelRelease      (float channel, float value);
setBallGain         (float channel, float value);
setBallAttack       (float channel, float value);
setBallDecay        (float channel, float value);
setBallSustain      (float channel, float value);
setBallRelease      (float channel, float value);
setRollGain         (float channel, float value);
```

Adding your own presets:

In every plugin's UI script find the comment which says: *"//Here you can add the name of your new preset"*
Right after that add the name of your new preset inside

```
pluginName_parameter_options = new string[] { }
```

Then inside:

```
if(pluginName_prevPreset != pluginName_Presets) { }
```

Add your own preset code, Best practice is to copy paste code from another preset and adding it towards the end of the list of presets and change the required values to avoid confusion.

Example:

To add a new preset to the Footsteps plugin open the `Footsteps_UI.cs` file in your code editor and find *"//Here you can add the name of your new preset"* in the file.
Now, add new preset names to the parameter_options and add audio parameter values for the new preset inside the condition block.

```
if(Footsteps_prevPreset != Footsteps_Presets).
     GUILayout.BeginHorizontal();
     //Here you can add the name of your new preset
string[] Footsteps_parameter_options = new string[]
            {
            "Use Slider Values","NewPreset"
             };
Footsteps_Presets = EditorGUILayout.Popup("Presets", Footsteps_Presets,
         Footsteps_parameter_options, GUILayout.Width(400));
            if(Footsteps_prevPreset != Footsteps_Presets)
            {
       //Here you can add your own presets to the plugin.

                if (Footsteps_Presets == 0)
                {
                    Footsteps_Firmness = 0.34f;
                    Footsteps_ShoeType = 1;
                    Footsteps_SurfaceType = 1;
                    Footsteps_TerrainType = 0;
                    Footsteps_Pace = 76.0f;
                    Footsteps_Steadiness = 0.2f;
                    Footsteps_Volume = 0.8f;
                }

                    //New Preset Code
                if (Footsteps_Presets == 1)
                {
                    Footsteps_Firmness = 0.24f;
                    Footsteps_ShoeType = 0;
                    Footsteps_SurfaceType = 3;
                    Footsteps_TerrainType = 0;
                    Footsteps_Pace = 96.0f;
                    Footsteps_Steadiness = 0.6f;
                    Footsteps_Volume = 0.8f;
                }
             }
```

Creating your own controller script:

The provided controller script is sufficient to control the plugin from any object but if you want to create your own controller script for any plugin.

In your c# file
Make sure you have
```
using System.Runtime.InteropServices;
```
To enable communication with the plugin

In your monoscript class make sure you've imported the required function from the plugin as follows:
```
#if UNITY_IOS
      [DllImport("__Internal")]
#else
    [DllImport("audioplugin_PluginName")] static extern void setFloatValue(float
channel, int index, float value);
#endif
```
If you want to import multiple function add another similar block right after the previous `#endif` with the required function name.you can import multiple plugins and call their respective functions if required

Example:

```
using UnityEditor;
using UnityEngine;
using System.Runtime.InteropServices;
using UnityEngine.SceneManagement;

public class Footsteps_Controller : MonoBehaviour
{
#if UNITY_IOS
      [DllImport("__Internal")]
#else
    [DllImport("audioplugin_Footsteps")] static extern void setFloatValue(float
channel, int index, float value);
#endif

#if UNITY_IOS
      [DllImport("__Internal")]
#else
    [DllImport("audioplugin_Footsteps")] static extern float getFloatValue(int index);
#endif

#if UNITY_IOS
      [DllImport("__Internal")]
#else
    [DllImport("audioplugin_Footsteps")] static extern float setStart(float channel);
```

Spatializing the Audio Plugin:

To Spatialize the plugin Add an external audio spatializer after the audioPlugin.