ParaMix

for Unity 3D

Version 1.0

Description:

ParaMix is a plugin for Unity 3D version 5.0 and above designed to allow you to use animating parameters to control various functionality in a game with the initial focus being on sound in both 2D and 3D games. Change sounds and effects for both AudioSource components and the new AudioMixer in Unity 5.0 using triggers, animation, or randomness to make your game sound more awesome. Create, for instance, a complex engine sound that changes based on the speed of your car, wind that uses one small sound loop but sounds like it is constantly changing over time, or rooms that when entered make the music sound more creepy. What it does is really up to you and your creativity!

If you have need any help or just have a question about ParaMix, please send an email to paramixforunity@gmail.com and you will receive a reply as soon as possible.

Release Notes:

This is the initial release of ParaMix. Look here for notes describing the changes for each successive version of the plugin.

Legal:

All included sound effects were downloaded from freesound.org and were licensed under the Creative Commons license. You are free to modify or use them for any purpose, even commercially.

Included Example Scenes

AudioSource Examples

audiosource_example_engine
A car object is set up with an AudioSourceAnimation component on it which mixes 3 different sounds based on the car's current speed. Pressing spacebar will accelerate the car and releasing will decelerate.

audiosource_example_random
A creepy sound is set up with an AudioSourceRandomPitch component to randomly change its pitch over time.

audiosource_example_wind
The sound of ocean waves in a cave is accompanied by the sound of wind which uses a combination of AudioSourceRandomVolume and AudioSourceRandomLowPassFilterCutoff components to constantly change the sound of the wind.

AudioMixer Examples

mixer_example_engine
This is a recreation of the engine sound setup but for an AudioMixer. Slide the bar to hear the engine go from idle to full speed.

mixer_example_hallway
Uses a SetLerpParamValueFromDistAlongTrigger component to fade in a song smoothly as a character moves towards the left of a hallway and back out as they move towards the right.

mixer_example_room
Uses a combination of MixerSmoothLerpParam, SetLerpParamValueOnTriggerEnter and SetLerpParamValueOnTriggerExit components to create a room with a creepy sound that smoothly blends in when the room is entered and blends out when the room is exited.

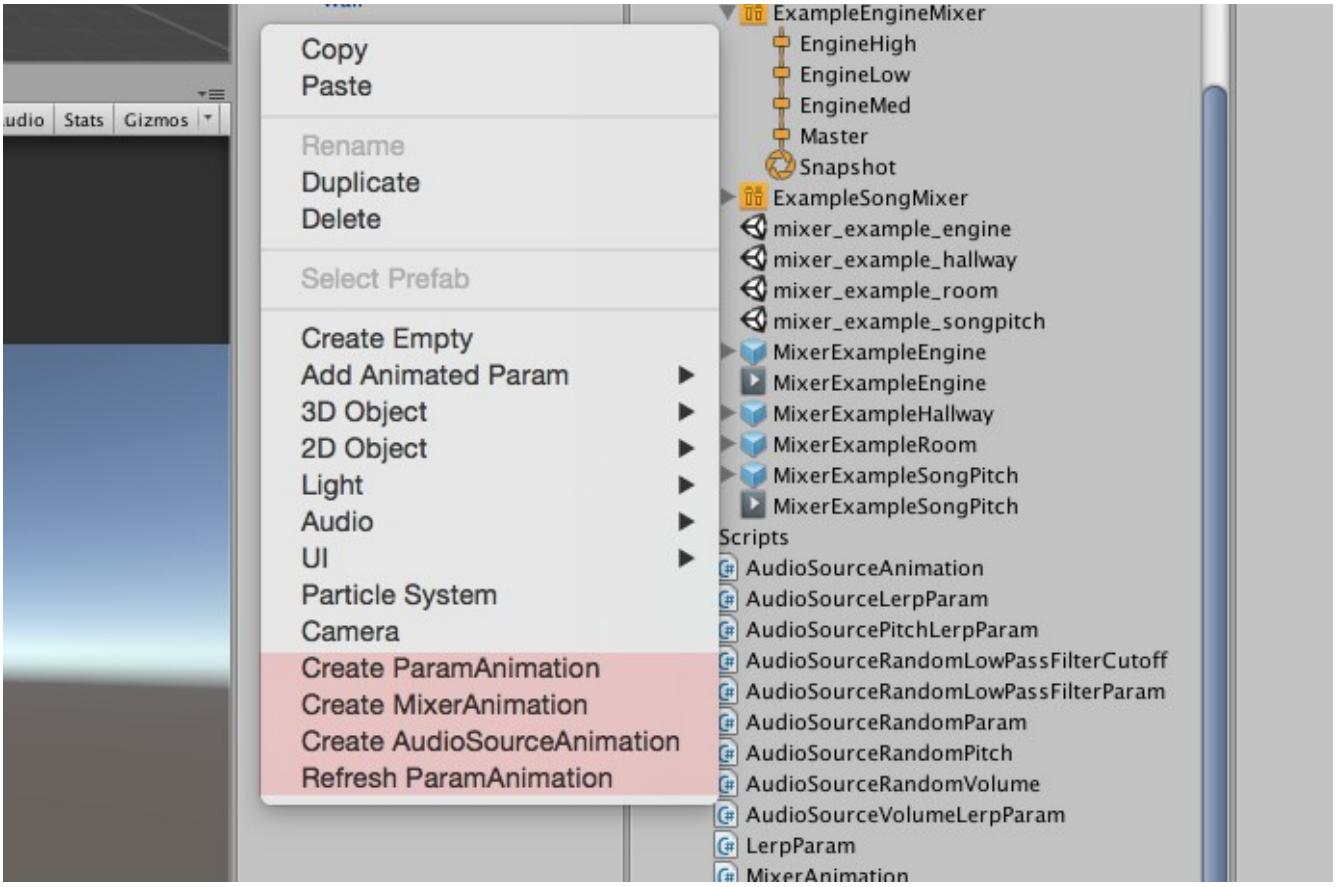
mixer_example_songpitch
Uses a MixerRandomParam component to set the pitch of a song to a random value at random times.

How To Use ParaMix

Prerequisites
To use ParaMix, it is recommended to first become familiar with how AudioSource components and the new AudioMixer in Unity 5.0 work before attempting to use ParaMix. If you would like to jump right in, open one of the many example scenes to see how they work.

Components
Please scroll to the next several pages for the description of each ParaMix component. Simply add one of these components and set the required parameters to use them.

ParamAnimation
ParamAnimation is an advanced feature of ParaMix which allows you to animate on or many parameters as a group to create interesting effects (such as an engine sound). Right-click in the hierarchy window in a blank area to see all of the ParamAnimation options, highlighted in red in the following screenshot.



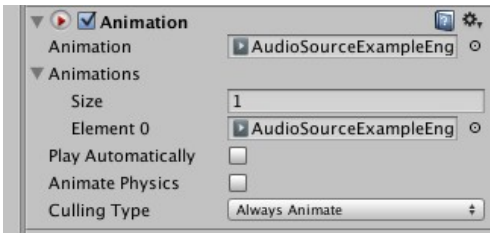
...continued on next page

Click “Create MixerAnimation” or “Create AudioSourceAnimation” to get started.

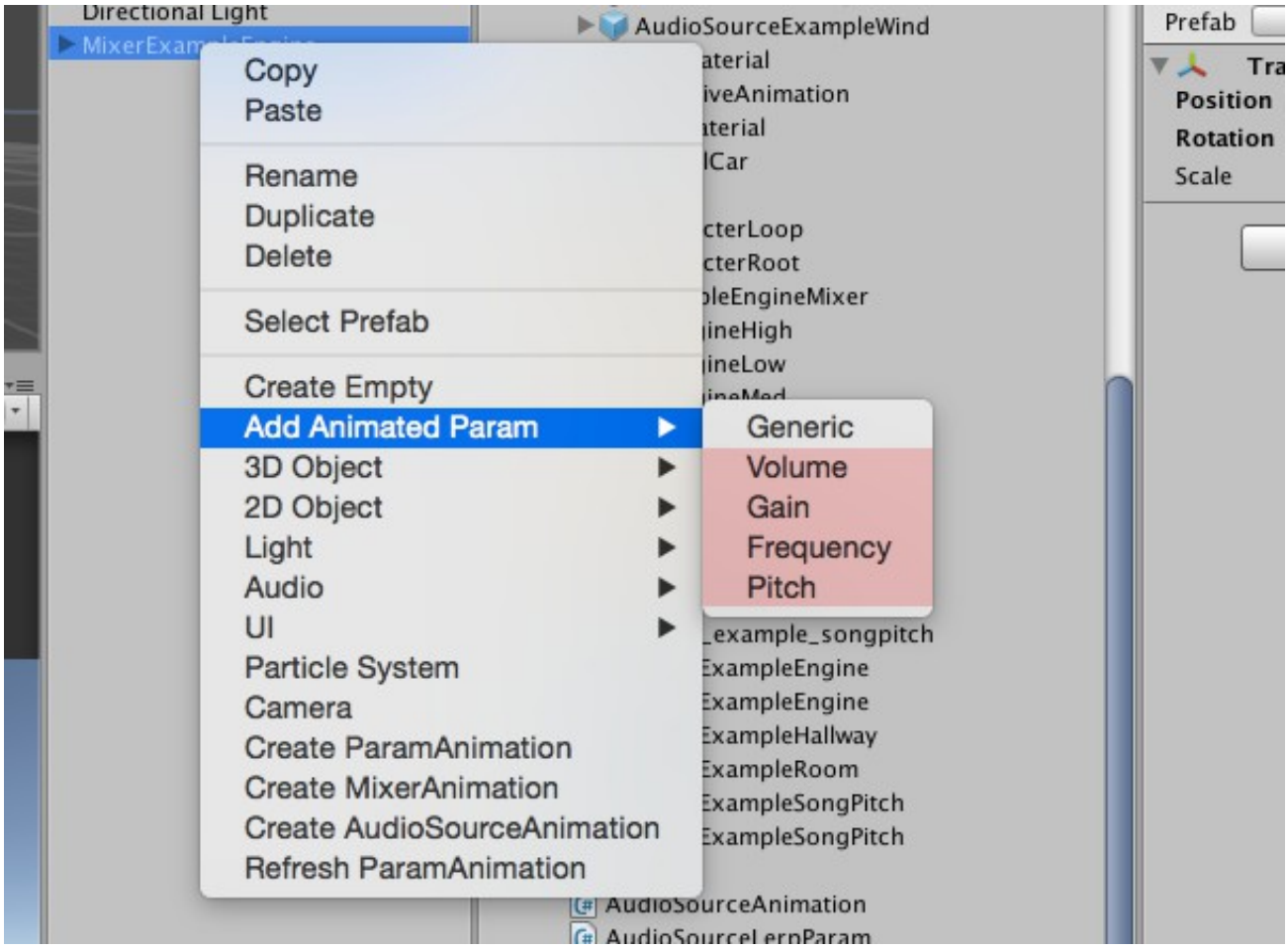
If you chose to create a MixerAnimation, you must link a specific AudioMixer after creating the object.

After creating the ParamAnimation object, you must create an Animation for it to use. To do this, right-click somewhere in the Project window in Unity and select Create, then Animation.

After you create the Animation, you must link it in the Animation component of the ParamAnimation object in the screenshot to the right.



Now you're ready to add parameters. Right-click on the newly created ParamAnimation component and select one of many parameters to add, highlighted in red in the following screenshot.

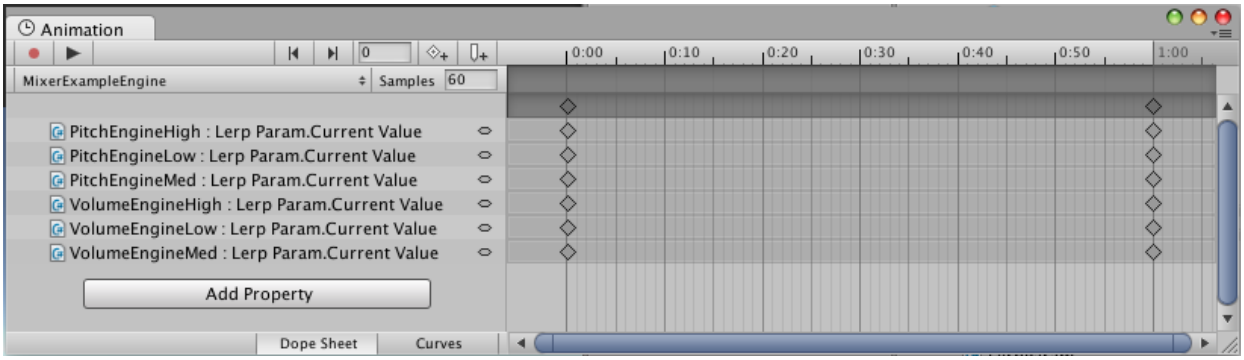


...continued on next page

If the parameter is supported, it will be added as a new object parented under the ParamAnimation object. Note that some parameters, such as those for the AudioSourceAnimation component, may require you to link it to a specific AudioSource component before it will work. Currently, only Volume and Pitch are supported for AudioSourceAnimation parameters. The MixerAnimation component supports any AudioMixer parameter you expose.

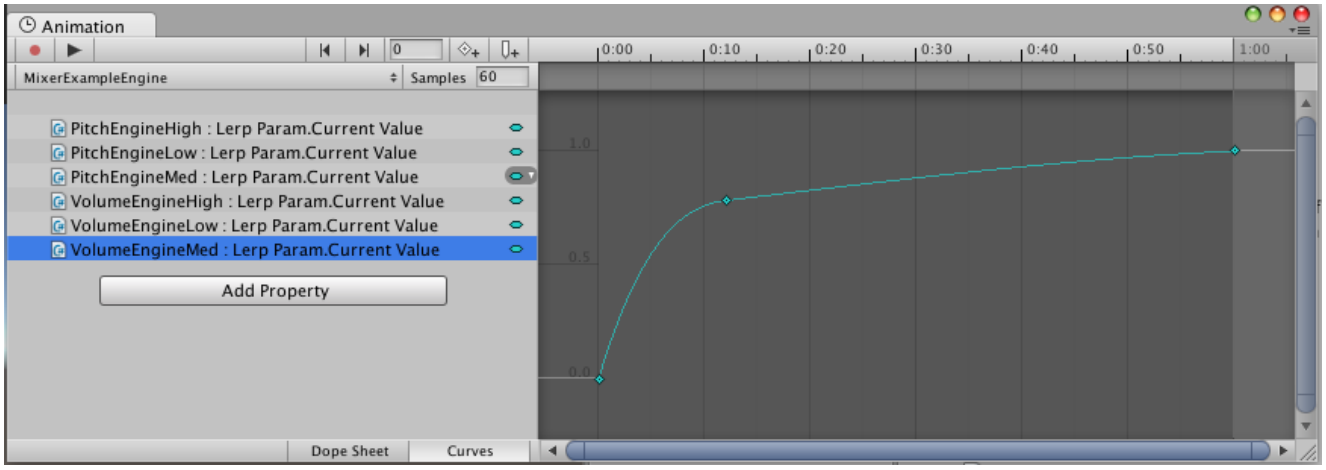
To use an exposed AudioMixer parameter, simply rename the GameObject with the MixerLerpParam on it to the name of the exposed AudioMixer parameter you wish to control. Any affected AudioSource component must also be linked to the same AudioMixer. Please see Unity 3D documentation to learn how to use an AudioMixer and how to expose AudioMixer parameters.

Once you have added all of the parameters you want, or whenever you rename or delete a parameter, you must right-click the ParameterAnimation component and click “Refresh ParamAnimation” from the menu or simply left-click on the ParameterAnimation object and press the shortcut key (ctrl-r on Windows and command-r on Mac OSX). This will add/remove/refresh the animation clips for all of the parameters parented under the ParamAnimation object. Now, with the ParamAnimation object selected, open the Animation window (Go to Window, Animation from Unity's menu bar). You should now see a dope sheet with all of the parameters listed like in the following screenshot.



… continued on next page

Each parameter animates on a range of 0 to 1 second. You should not change this range. Now left-click on each LerpParam and edit the animation curve like in the following screenshot.



The left side of the curve will be the value when the ParamAnimation paramValue is set to 0 and the right side of the curve will be the value when paramValue is set to 1. Changing the paramValue will sample the LerpParam curves anywhere between 0 and 1 seconds.

Once you have created curves for all of the LerpParams, you are finished! See the two engine example scenes for a working example.

Code Documentation

Core Classes

ParamValue
A base class used to store a single floating point parameter (currentValue)

LerpParam
Derives from ParamValue. It uses the ParamValue's 0 to 1 parameter value (currentValue) to interpolate between two other values (paramMin and paramMax)

SmoothLerpParam
Derives from LerpParam. It will smoothly animate towards the set desired value (desiredValue) at the set speed (changeSpeed).

RandomLerpParam
Derives from SmoothLerpParam. It will randomly change the desired value (desiredValue) after a set random time and animate towards the new value at the set speed (changeSpeed).

ParamAnimation
A base class used for parameter animation. Allows you to create many animated parameter curves and sample the curves anywhere from their start at 0 seconds to their end at 1 seconds in the animation by setting the parameter value (paramValue) to any number from 0 to 1.

AudioSource Classes

AudioSourceAnimation
Derives from ParamAnimation. This allows you to control AudioSource parameters affecting one or many AudioSource components as a group using any shape animation curve you can create. Currently supported parameters include pitch and volume.

AudioSourceLerpParam
A base class which derives from LerpParam. It has no inherent functionality on its own. It provides a link to an AudioSource component.

AudioSourcePitchLerpParam
Derives from AudioSourceLerpParam. Used to control pitch on an AudioSource.

AudioSourceVolumeLerpParam
Derives from AudioSourceLerpParam. Used to control volume on an AudioSource.

AudioSourceRandomParam
A base class which derives from RandomLerpParam. It has no inherent functionality on its own. It provides a link to an AudioSource component.

AudioSourceRandomVolume
Derives from AudioSourceRandomParam. It sets the volume of an AudioSource component to a random value between two values at a random time between two values.

AudioSourceRandomPitch
Derives from AudioSourceRandomParam. It sets the pitch of an AudioSource component to a random value between two values at a random time between two values.

AudioSourceRandomLowPassFilterParam
A base class which derives from RandomLerpParam. It has no inherent functionality on its own. It provides a link to an AudioLowPassFilter component.

AudioSourceRandomLowPassFilterCutoff
Derives from AudioSourceRandomLowPassFilterParam. It sets the cutoff value of an AudioLowPassFilter component to a random value between two values at a random time between two values.

Mixer Classes

MixerAnimation
Derives from ParamAnimation. Used for animation of one or more exposed AudioMixer parameters. This allows you to control parameters affecting many different sounds and effects as a group using any shape animation curve you can create.

MixerLerpParam
A base class which derives from LerpParam. Used to control a parameter exposed by an AudioMixer.

MixerRandomParam
Derives from RandomLerpParam. It sets the value of an exposed AudioMixer parameter to a random value between two values at a random time between two values.

MixerSmoothLerpParam
Derives from LerpParam. It will smoothly animate an exposed AudioMixer parameter towards the set desired value (desiredValue) at the set speed (changeSpeed).

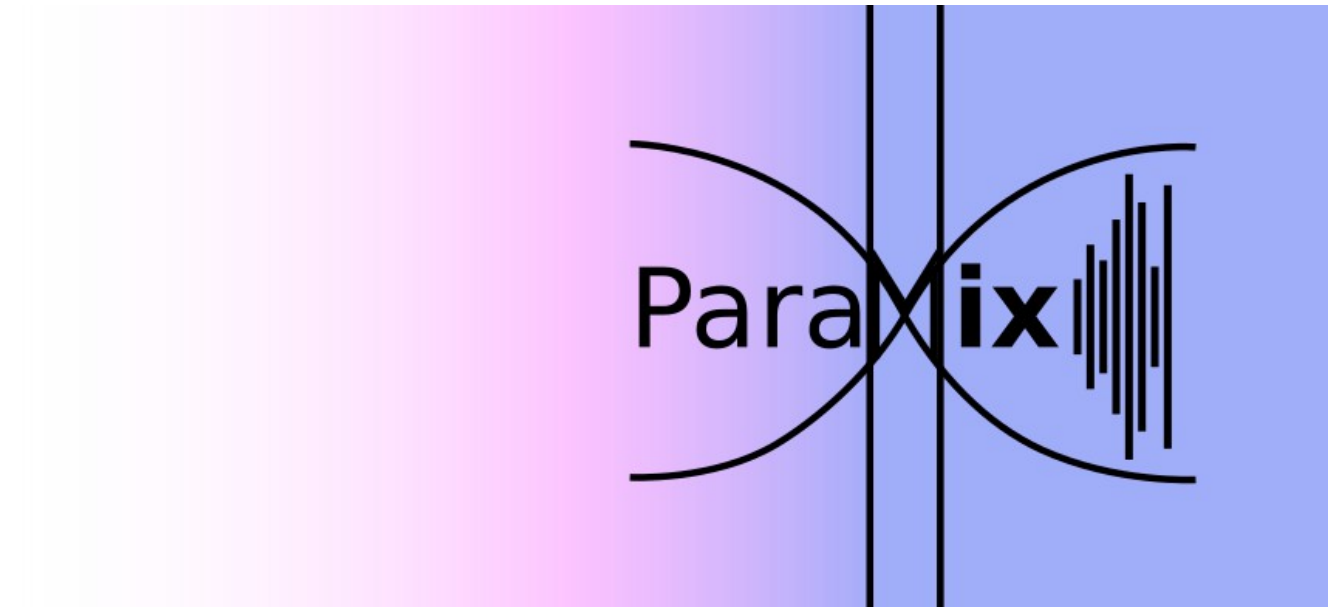
Other Classes

SetLerpParamValueFromDistAlongTrigger
Sets the value of a linked ParamValue component based on the distance between two transforms (endPointA and endPointB) while within the trigger area. Requires that the object this component is on has a collider on it with IsTrigger set to true. The object entering the trigger must have a collider and a RigidBody component. It works for both 2D and 3D physics objects.

SetLerpParamValueOnTriggerEnter
Sets the value of a linked ParamValue component upon entering the trigger area. Requires that the object this component is on has a collider on it with IsTrigger set to true. The object entering the trigger must have a collider and a RigidBody component. It works for both 2D and 3D physics objects.

SetLerpParamValueOnTriggerExit
Sets the value of a linked ParamValue component upon exiting the trigger area. Requires that the object this component is on has a collider on it with IsTrigger set to true. The object entering the trigger must have a collider and a RigidBody component. It works for both 2D and 3D physics objects.

SetAnimSpeedFromParam
Sets the speed of an Animator component based on the current paramValue of a linked ParamValue component.



For questions or support, please email paramixforunity@gmail.com