TazMan-Audio

Fabric API

FABRIC

Table of contents

Introduction

Welcome to the API reference document for Fabric.

API

Fabric comes with several classes that allow the game to control and manipulate the audio behaviours that are designed in the editor. Though it is possible to implement and trigger audio from the editor there will be cases that it is necessary to trigger an audio from code.

Fabric Manager

Fabric manager is the parent object of all audio components and it is responsible mostly for managing the life time of components and their communication with the custom user interfaces.

Instance

Function

Instance() : FabricManager

Description

Function provides access to the instance of the fabric manager. Use this to gain access to the manager from any script.

Example:

```
FabricManager.Instance().Stop();
```

IsInitialised

Function

IsInitialised() : bool

Description

Returns true if the event manager is initialised, false if it's not.

Stop

Function

Stop() : void

Description

Stops all components from playing.

Pause

Function

Pause(bool pause) : void

Description

Pauses or un-pauses all components depending on the pause flag passed.

LoadAsset

Function

LoadAsset(string name) : void

Description

Loads a prefab by name, the prefab **must** contain a hierarchy of Fabric components.

UnloadAsset

Function

UnloadAsset(string name) : void

Description

Unloads all components that have been previously loaded with the LoadAsset function.

Event Manager

The event manager is responsible for managing events send by the game and the listeners that listen for them in order to perform an action.

Instance

Function

Instance() : EventManager

Description

Function provides access to the instance of the event manager. Use this to gain access to the manager from any script.

Example:

```
EventManager.Instance().IsInitialised();
```

IsInitialised

Function

IsInitialised() : bool

Description

Returns true if the event manager is initialised, false if it's not.

RegisterListener

Function

RegisterListener(IEventListener listener, string eventName) : bool

Description

Registers a listener object to the manager listening for a specific event name.

listener parameter is an object that inherits from the IEventListener interface.

eventName parameter defines the name in which this listener will be waiting for.

UnregisterListener

Function

UnregisterListener(IEventListener listener, string eventName) : bool

Description

Unregisters a listener from the event manager.

listener parameter is an object that inherits from the IEventListener interface.

eventName parameter defines the name in which this listener will be waiting for.

PostEvent

Function

PostEvent(string eventName, GameObject parentGameObject=null) : bool

Description

Posts an event by name.

The eventName parameter is the name to be queued.

The parentGameObject parameter is optional and it is mainly used to attach 3D sounds into game objects.

Returns true or false if the even has been posted successfully.

Example:

An example of triggering a component by code that has an event listener with the name “Simple”.

```
EventManager.Instance.PostEvent("Simple", gameObject);
```

PostEvent

Function

PostEvent(string eventName, EventAction eventAction, object parameter, GameObject parentGameObject=null) : bool

Description

Queues an event by name.

The eventAction parameter defines the type of the event (i.e. Play, Stop, SetVolume, SetPitch etc.)

The parameter allows (optional) parameter data (float, string) to be passed with the event that can be processed by the components.

The parentGameObject parameter is optional and it is mainly used to attach 3D sounds into game objects.

Example:

The following line tells a switch component with the event name “Switch” to switch to a child component with the name “Gravel”

```
EventManager.Instance.PostEvent("Switch",EventAction.SetSwitch, "Gravel");
```

SetParameter

Function

SetParameter(string eventName, string parameterName, float value, GameObject parentGameObject) : void

Description

Helpers function for components that allow receiving custom parameters in order to modify their behaviour.

The eventName parameter is the event to queue.

The parameterName parameter is the name of the parameter

The value parameter is the value in which the parameter is to be set.

Example:

This is an example of how to set a parameter to a component

```
rpm = 3000.0f;

EventManager.Instance.SetParameter("engine", "rpm", rpm, gameObject);
```

Debug log

This is a class that prints out debug messages on the console.

Instance

Function

Instance() : DebugLog

Description

Function provides access to the instance of the debug log. Use this to gain access to the debug log from any script.

Print

Function

Print(string Msg, DebugLevel debugLevel) : void

Description

Prints a message of a specific type into the debug log.

The Msg parameter is the message to display on the debug log.

The debugLevel parameter indicates the debug level of the message.

The DebugLevel enum defines three main level of messages:

- Info
- Warning
- Error