

extOSC

Open Sound Control Protocol by V. Sigalkin



extOSC

About 3

Fast Start 4

Examples 4

URLs 9

About

extOSC is a tool dedicated to simplify creation of applications with OSC protocol usage.

Features:

OSC Without Coding

With extOSC components you can easily implement basic program logic in your application without coding.

OSC Console

New separated console for tracking sending and receiving OSC packages.

OSC Debug

New easy to use tool for debugging sending and receiving OSC packages.

Map OSC packages

OSCMapping allows you to map float, int, bool values.

UI

Four new components for easy creation of remote control apps with OSC-protocols.

Addresses with masks support

You can bind addresses with mask (for instance: «/lights/*/value»).

Auto bundle

extOSC will bundle your packages for optimisation purposes

And also:

- Every data type support (except Array and MIDI)
- Compatible with OS X, Windows, iOS & Android
- Tested with TouchOSC, VOpenSource, OpenFrameworks and others
- Examples

And much more

Fast Start

Fast Start Video Tutorial: <https://www.youtube.com/watch?v=HYYBKOsI5KE>

Examples

Folder: extOSC/Examples

There you can find all what you need to work with extOSC. If the above examples do not help you, please do not worry, and send you question on me E-mail or Forum thread.

Getting Started

This example demonstrates usage of two scripts: "SimpleMessageReceiver.cs" and "SimpleMessageTransmitter.cs". These scripts include the basic logic for sending and receiving messages.

You can see how these scripts work in "OSC Console":

> Window > extOSC > Console Window.

Events

This example shows how to use Event-components. Event-components allow you to bind components with OSC.

Event-components can be found here:

> Add Component > extOSC > Components > Receiver.

Informers

This example shows how to use Informers-components. Informers-components allow you to bind components with OSC.

Notice that "Use Bundle" in OSCTransmitter is enabled. This allows OSCTransmitter to send only one bundle (instead of three) per frame.

Informers-components can be found here:

> Add Component > extOSC > Components > Transmitter.

Mapping

This example shows how to map values in messages. File "Mapping Example.asset" is connected to Receiver and includes config for three different messages. It is also possible to connect map-files to Transmitter.

Scripting

This example shows how to easily implement OSC-message sending (File: "ScriptingExample.cs").

ScriptingExample.cs:

```
// Creating a transmitter.
_transmitter = gameObject.AddComponent<OSCTransmitter>();

// Set remote host address.
_transmitter.RemoteHost = "127.0.0.1";

// Set remote port;
_transmitter.RemotePort = 7001;

// Creating a receiver.
_receiver = gameObject.AddComponent<OSCReceiver>();

// Set local port.
_receiver.LocalPort = 7001;

// Bind "MessageReceived" method to special address.
_receiver.Bind(_oscAddress, MessageReceived);

// MessageReceived implementation
protected void MessageReceived(OSCMessage message)
{
    Debug.Log(message);
}

// Create message
var message = new OSCMessage("Hello, world!");
message.AddValue(OSCValue.String("Hello, world!"));
message.AddValue(OSCValue.Float(Random.Range(0f, 1f)));

// Send message
_transmitter.Send(message);
```

Address Masks

This example shows how to use masks for OSC-packages receive.

Mask example:

*/example/9/**

*/example/9/*value*

If you use "*" address, binded method will be called regardless of OSC-package address.

Serialization

This example shows how to serialize object in OSC-message. (**Warning:** Not work with Windows Store!) (File: "SerializationExample.cs")

```
// Class with serialization
public class ExampleClass
{
    [OSCSerialize]
    public string StringValue;

    [OSCSerialize]
    public float FloatValue;

    [OSCSerialize]
    public bool BoolValue;

    [OSCSerialize]
    public Vector2 VectorValue;
}

// Using in code
// Get message from class:
var message = OSCSerializer.Serialize(_address, exampleClass);

// Get class from message:
var exampleClass = OSCSerializer.Deserialize<ExampleClass>(message);
```

Ping

This example shows how does work OSCTransmitterPing. This can be useful to maintain to monitor that app is still running.

You can see how these scripts work in "OSC Console":

> Window > extOSC > Console Window

OSCTransmitterPing can be found here:

> Add Component > extOSC > Components > Misc > Ping

Array

This example shows how to easily use Array type in OSC
(File: "ArrayExample.cs").

```
// Create message
var message = OSCMessage.Create(_address);

// Create array
var array = OSCValue.Array();
array.AddValue(OSCValue.Int(1));
array.AddValue(OSCValue.Float(2.5f));
array.AddValue(OSCValue.Color(Color.red))
// You can use AddValue(OSCValue) method only with OSCValue what stored
// Array type.

// You can store another array inside array.
// Warning! OSCValue with "Array" type cannot store itself.
// It can do infinite loop.
var secondArray = OSCValue.Array();
secondArray.AddValue(OSCValue.String("This array..."));
secondArray.AddValue(OSCValue.String("...inside another array!"));
array.AddValue(secondArray);

// Add array in message
message.AddValue(array);
```

Match Pattern

This example shows how to use OSCMathPattern class.
(File: "MatchPatternExample.cs")

```
// Create match pattern.
// (For bool values you can use True or False ValueType)
var matchPattern = new OSCMatchPattern(OSCValueType.String,
                                         OSCValueType.Int,
                                         OSCValueType.True,
                                         OSCValueType.False);

// Check match pattern
if (message.IsMatch(matchPattern))
{
    // Correct message
}
else
{
    // Wrong message
}
```

Marshalling

This example shows how to use Marshalling with extOSC.

(File: "MarshallingExample.cs")

```
// Structure
// Marshalling works only with structures.
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]
public struct MarshallingStructure
{
    public int IntValue;

    [MarshalAs(UnmanagedType.LPStr, SizeConst = 20)]
    public string StringValue;

    public float FloatValue;
}

// Send Structure
// Create Message
var message = new OSCMessage(_address);

// Create structure
var structure = new MarshallingStructure();
structure.IntValue = 1337;
structure.StringValue = "Hello, OSC World!";
structure.FloatValue = 13.37f;

// Convert structure to bytes by marshalling!
// Marshalling can sometimes be quicker, than any other form of
// conversion of data in OSC
var bytes = OSCUtilities.StructToByte(structure);

// Add bytes to message
message.AddValue(OSCValue.Blob(bytes));

// Send message
Transmitter.Send(message);

// Receive Structure
// Get bytes from message
if (!message.ToBlob(out bytes))
    return;

// Convert bytes to structure!
var structure = OSCUtilities.ByteToStruct<MarshallingStructure>(bytes);
```


URLs

Asset Store

<https://www.assetstore.unity3d.com/#!/content/72005> or <http://u3d.as/ADA>

Forum Thread

<https://forum.unity.com/threads/436159/>

Videos

OSC Console - <https://www.youtube.com/watch?v=ihVw6v2Meto>

OSC Debug - <https://www.youtube.com/watch?v=PU2oSwbbliE>

OSC Mapping - <https://www.youtube.com/watch?v=73Hjglgx6ss>

OSC UI - https://www.youtube.com/watch?v=phV4Y8Go0_U

Support E-mail

ext@iron-wall.org