Motion Controlled Additive Synthesis

Generated by Doxygen 1.8.20

Chapter 1

Motion Controlled Additive Synthesis

With this we are presenting a additive sound synthesis Pure Data External. Alongside we provide a fun demonstration of controlling it by using smartphone accelerometer and gyroscope data from a smartphone

1.1 Where to start?

The repo consist of the external's source code in addition with other needed c files, the pd external binary (MacOS x64) and two sample patches to use it with `PdParty`

1.1.1 Requirements

In order to use the addsi~ external from **External Binary and Help Patch** you either need to have a MacOs running on a 64 bit machine, or you build the external yourself using the provided **External's Source**

1.1.2 Using addsi~ with PdParty

addsi was created to be used with your smartphone! How to?

1. Download `PdParty` (iOS) or `DroidParty` (Android) and see how they work
2. Grab the smartphone patch from **Sample Patches** and move it to your device and start it.
3. Tap the buttons to send OSC messages containing accelerometer and gyroscope data. Make sure to set the host IP correctly in the OSC settings, to match the IP adress of you computer!

1. Start the addsi~patch.pd from **Sample Patches** on your computer
2. Have fun!

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 _atom Struct Reference

Data Fields

- t_atomtype **a_type**
- union word **a_w**

The documentation for this struct was generated from the following file:

- Needed other C Files/m_pd.h

4.2 _gobj Struct Reference

Data Fields

- t_pd **g_pd**
- struct _gobj * **g_next**

The documentation for this struct was generated from the following file:

- Needed other C Files/m_pd.h

4.3 _gpointer Struct Reference

Data Fields

-
  union {
    struct _scalar * **gp_scalar**
    union word * **gp_w**
  } **gp_un**

- int **gp_valid**
- t_gstub * **gp_stub**

The documentation for this struct was generated from the following file:

- Needed other C Files/m_pd.h

4.4 _gstub Struct Reference

Data Fields

- union {
  struct _glist * **gs_glist**
  struct _array * **gs_array**
  } **gs_un**
- int **gs_which**
- int **gs_refcount**

The documentation for this struct was generated from the following file:

- Needed other C Files/m_pd.h

4.5 _resample Struct Reference

Data Fields

- int **method**
- t_int **downsample**
- t_int **upsample**
- t_sample * **s_vec**
- int **s_n**
- t_sample * **coeffs**
- int **coefsize**
- t_sample * **buffer**
- int **bufsize**

The documentation for this struct was generated from the following file:

- Needed other C Files/m_pd.h

4.6 _scalar Struct Reference

Data Fields

- t_gobj **sc_gobj**
- t_symbol * **sc_template**
- t_word **sc_vec** [1]

The documentation for this struct was generated from the following file:

- Needed other C Files/m_pd.h

4.7 _signal Struct Reference

Data Fields

- int **s_n**
- t_sample * **s_vec**
- t_float **s_sr**
- int **s_refcount**
- int **s_isborrowed**
- struct _signal * **s_borrowedfrom**
- struct _signal * **s_nextfree**
- struct _signal * **s_nextused**
- int **s_vecsize**

The documentation for this struct was generated from the following file:

- Needed other C Files/m_pd.h

4.8 _symbol Struct Reference

Data Fields

- char * **s_name**
- struct _class ** **s_thing**
- struct _symbol * **s_next**

The documentation for this struct was generated from the following file:

- Needed other C Files/m_pd.h

4.9 _text Struct Reference

Data Fields

- t_gobj **te_g**
- t_binbuf * **te_binbuf**
- t_outlet * **te_outlet**
- t_inlet * **te_inlet**
- short **te_xpix**
- short **te_ypix**
- short **te_width**
- unsigned int **te_type**:2

The documentation for this struct was generated from the following file:

- Needed other C Files/m_pd.h

4.10 addsi Struct Reference

Internal data structure containing the parameters for the oscillators and buffer data variables

`#include <addsi.h>`

Data Fields

- int tableSize
- float currentIndex
- float basefrequency
- float * lookupTable1
- int numberOfHarmonics
- float harmonicIndex [MAXNUMBEROFHARMONICS]
- float harmonicGain [MAXNUMBEROFHARMONICS]
- float * envelopeTable
- int envelopeIndex
- float * LFO1_Table
- float LFO1frequency
- float LFO1_depth
- float LFO1_currentIndex
- float * LFO2_Table
- float LFO2frequency
- float LFO2_depth
- float LFO2_currentIndex

Related Functions

(Note that these are not member functions.)

- addsi * addsi_new (int sampleRate)

  Sets up new addsi object on first run and creates wave tables
  This function sets up all we need to get started with processing.
- void addsi_free (addsi *x)

  Frees the memory
  Implements mandatory memory management function.
- void addsi_process (addsi *x, float *in, float *out, int vectorSize)

  Main method: Implementing the additive synthesis of ~addsi
  Processes the wave tables.
- void addsi_setbasefrequency (addsi *x, float basefrequency)

  Sets a base frequency for the osc
- void addsi_setLFO1frequency (addsi *x, float LFO1frequency)

  Sets the strength of the first LFO
- void addsi_setLFO2frequency (addsi *x, float LFO2frequency)

  Sets the strength of the second LFO
- void addsi_setnumberOfHarmonics (addsi *x, float numberOfHarmonics)

  Sets the number of partials added to the base frequency

4.10.1 Detailed Description

Internal data structure containing the parameters for the oscillators and buffer data variables

4.10.2 Friends And Related Function Documentation

4.10.2.1 addsi_free()

```
void addsi_free (
            addsi * x )  [related]
```

Frees the memory
Implements mandatory memory management function.

Parameters

*x	pointer to addsi struct

4.10.2.2 addsi_new()

```
addsi * addsi_new (
            int sampleRate )  [related]
```

Sets up new addsi object on first run and creates wave tables
This function sets up all we need to get started with processing.

Parameters

sampleRate	int containing the used sample rate. Note that at this time this is hard coded to 44100 in addsi_pd.c and will not work with differing sample rates

Returns

An addsi struct

4.10.2.3 addsi_process()

```
void addsi_process (
            addsi * x,
```

```
            float * in,
            float * out,
            int vectorSize )  [related]
```

Main method: Implementing the additive synthesis of ~addsi
Processes the wave tables.

Parameters

*x	pointer to an addsi struct
*in	pointer to sound input vector (currently unused)
*out	pointer to sound output vector
vectorSize	size of the sound vectors

Returns

An addsi struct

4.10.2.4 addsi_setbasefrequency()

```
void addsi_setbasefrequency (
            addsi * x,
            float basefrequency )  [related]
```

Sets a base frequency for the osc

Parameters

x	A pointer to the addsi object
basefrequency	float containing the Hz value of the base frequency of the osc

4.10.2.5 addsi_setLFO1frequency()

```
void addsi_setLFO1frequency (
            addsi * x,
            float LFO1frequency )  [related]
```

Sets the strength of the first LFO

Parameters

x	A pointer to an addsi_tilde object
LFO1frequency	float value setting the strength of the first LFO

4.10.2.6 addsi_setLFO2frequency()

```
void addsi_setLFO2frequency (
            addsi * x,
            float LFO2frequency )  [related]
```

Sets the strength of the second LFO

Parameters

x	A pointer to an addsi_tilde object
LFO2frequency	float value setting the strength of the second LFO

4.10.2.7 addsi_setnumberOfHarmonics()

```
void addsi_setnumberOfHarmonics (
            addsi * x,
            float numberOfHarmonics )  [related]
```

Sets the number of partials added to the base frequency

Parameters

x	A pointer to an addsi_tilde object
numberOfHarmonics	float value setting the number of partials

4.10.3 Field Documentation

4.10.3.1 basefrequency

```
float basefrequency
```

Working frequency of the sine oscillator

4.10.3.2 currentIndex

```
float currentIndex
```

Current working index of the sine oscillator

4.10.3.3 envelopeIndex

```
int envelopeIndex
```

Current working index in envelopeTable. Currently this is not used

4.10.3.4 envelopeTable

```
float* envelopeTable
```

Help array for enveloping. Currently this is not used

4.10.3.5 harmonicGain

```
float harmonicGain[MAXNUMBEROFHARMONICS]
```

Gain of Harmonics, size is set by definition of MAXNUMBEROFHARMONICS in addsi.h

4.10.3.6 harmonicIndex

```
float harmonicIndex[MAXNUMBEROFHARMONICS]
```

Index of Harmonics, size is set by definition of MAXNUMBEROFHARMONICS in addsi.h

4.10.3.7 LFO1_currentIndex

```
float LFO1_currentIndex
```

Current working index of lfo1_table

4.10.3.8 LFO1_depth

```
float LFO1_depth
```

Depth of the first LFO

4.10.3.9 LFO1_Table

```
float* LFO1_Table
```

Working array of the first LFO

4.10.3.10 LFO1frequency

```
float LFO1frequency
```

Working frequency of the first LFO

4.10.3.11 LFO2_currentIndex

```
float LFO2_currentIndex
```

Current working index of lfo2_table

4.10.3.12 LFO2_depth

```
float LFO2_depth
```

Depth of the second LFO

4.10.3.13 LFO2_Table

```
float* LFO2_Table
```

Working array of the second LFO

4.10.3.14 LFO2frequency

```
float LFO2frequency
```

Working frequency of the second LFO

4.10.3.15 lookupTable1

```
float* lookupTable1
```

SIne wave table

4.10.3.16 numberOfHarmonics

```
int numberOfHarmonics
```

Number of added harmonics to the baseFrequency

4.10.3.17 tableSize

```
int tableSize
```

Size of waveform tables, based on sample rate. Note that this is hard coded to 44100 in this external

The documentation for this struct was generated from the following files:

- External's Source/addsi.h
- External's Source/addsi.c

4.11 addsi_tilde Struct Reference

The Pure Data struct of the addsi∼ object.

Data Fields

- t_object x_obj
- t_sample f
- addsi ∗ osc
- t_outlet ∗ out

Related Functions

(Note that these are not member functions.)

- t_int ∗ addsi_tilde_perform (t_int ∗w)

  Perform function, mandatory for PureData. Calculates the output vector
  For more information please refer to the Pure Data Docs

- void addsi_tilde_dsp (addsi_tilde ∗x, t_signal ∗∗sp)

  DSP function mandatory for PureData. Adds addsi_tilde_perform to the signal chain.
  For more information please refer to the Pure Data Docs

- void addsi_tilde_free (addsi_tilde ∗x)

  Memory Management function mandatory for Pure Data. Frees our addsi_tilde object.
  For more information please refer to the Pure Data Docs

- void ∗ addsi_tilde_new (t_floatarg f)

  Creates new addsi_tilde object and sets its outlet and sampling rate. note that the externals sampling rate is set at 44100
  For more information please refer to the Pure Data Docs

- void addsi_tilde_setbasefrequency (addsi_tilde ∗x, float basefrequency)

  Wrapper for the base frequency setting of the sine osc

- void addsi_tilde_setLFO1frequency (addsi_tilde ∗x, float LFO1frequency)

  Wrapper for the strength setting of the first LFO

- void addsi_tilde_setLFO2frequency (addsi_tilde *x, float LFO2frequency)

  Wrapper of the strength setting of the second LFO

- void addsi_tilde_setnumberOfHarmonics (addsi_tilde *x, float numberOfHarmonics)

  Wrapper for the number of the partials added to the fundamental frequency

- void addsi_tilde_setup (void)

  Setup function
  This function (or functions called by it) declares the new classes and their properties of the addsi-tilde external. It is only called once, when the external is loaded.
  For more information please refer to the Pure Data Docs

4.11.1 Detailed Description

The Pure Data struct of the addsi~ object.

4.11.2 Friends And Related Function Documentation

4.11.2.1 addsi_tilde_dsp()

```
void addsi_tilde_dsp (
            addsi_tilde * x,
            t_signal ** sp )  [related]
```

DSP function mandatory for PureData. Adds addsi_tilde_perform to the signal chain.
For more information please refer to the Pure Data Docs

Parameters

x	A pointer the addsi_tilde object
sp	A pointer to the input and output vectors

4.11.2.2 addsi_tilde_free()

```
void addsi_tilde_free (
            addsi_tilde * x )  [related]
```

Memory Management function mandatory for Pure Data. Frees our addsi_tilde object.
For more information please refer to the Pure Data Docs

Parameters

x	A pointer to an addsi_tilde object

4.11.2.3 addsi_tilde_perform()

```
t_int * addsi_tilde_perform (
            t_int * w )  [related]
```

Perform function, mandatory for PureData. Calculates the output vector
For more information please refer to the Pure Data Docs

Parameters

w	A pointer to the object, input and output vectors.

Returns

A pointer to the signal chain right behind the addsi_tilde object.

4.11.2.4 addsi_tilde_setbasefrequency()

```
void addsi_tilde_setbasefrequency (
            addsi_tilde * x,
            float basefrequency )  [related]
```

Wrapper for the base frequency setting of the sine osc

Parameters

x	A pointer to an addsi_tilde object
basefrequency	float containing the Hz value of the base frequency of the sine osc

4.11.2.5 addsi_tilde_setLFO1frequency()

```
void addsi_tilde_setLFO1frequency (
```

```
            addsi_tilde * x,
            float LFO1frequency )  [related]
```

Wrapper for the strength setting of the first LFO

Parameters

x	A pointer to an addsi_tilde object
LFO1frequency	float value setting the strength of the first LFO

4.11.2.6 addsi_tilde_setLFO2frequency()

```
void addsi_tilde_setLFO2frequency (
            addsi_tilde * x,
            float LFO2frequency )  [related]
```

Wrapper of the strength setting of the second LFO

Parameters

x	A pointer to an addsi_tilde object
LFO2frequency	float value setting the strength of the second LFO

4.11.2.7 addsi_tilde_setnumberOfHarmonics()

```
void addsi_tilde_setnumberOfHarmonics (
            addsi_tilde * x,
            float numberOfHarmonics )  [related]
```

Wrapper for the number of the partials added to the fundamental frequency

Parameters

x	A pointer to an addsi_tilde object
numberOfHarmonics	float value setting the number of the partials

4.11.3 Field Documentation

4.11.3.1 f

```
t_sample f
```

Also necessary for signal objects, float dummy dataspace for converting a float to signal if no signal is connected (CLASS_MAINSIGNALIN)

4.11.3.2 osc

```
addsi* osc
```

custom addsi type containing all signal generating parameters and wavetables

4.11.3.3 out

```
t_outlet* out
```

needed to store handles to the outlet of the signal

4.11.3.4 x_obj

```
t_object x_obj
```

Necessary for every signal object in Pure Data

The documentation for this struct was generated from the following file:

- External's Source/addsi_pd.c

4.12 word Union Reference

Data Fields

- t_float **w_float**
- t_symbol * **w_symbol**
- t_gpointer * **w_gpointer**
- t_array * **w_array**
- struct _glist * **w_list**
- int **w_index**

The documentation for this union was generated from the following file:

- Needed other C Files/m_pd.h

Chapter 5

File Documentation

5.1 External's Source/addsi.c File Reference

Implementation of the external's working code
Includes all functions needed to create the additive synthesis of ~addsi.

```
#include "addsi.h"
```

5.1.1 Detailed Description

Implementation of the external's working code
Includes all functions needed to create the additive synthesis of ~addsi.

Author

Marius, Richard, Lenni, Kai
Audiocommunication Group, Technical University Berlin

5.2 External's Source/addsi.h File Reference

Header for addsi.c
Includes type definitions and function declarations.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "vas_mem.h"
#include "vas_util.h"
```

Data Structures

- struct addsi

  Internal data structure containing the parameters for the oscillators and buffer data variables

Macros

- #define **M_PI** (3.141592654)
- #define **TWOPI** (2.0 ∗ M_PI)
- #define **MAXNUMBEROFHARMONICS** 32

Typedefs

- typedef struct addsi **addsi**

5.2.1 Detailed Description

Header for addsi.c
Includes type definitions and function declarations.

Author

Marius, Richard, Lenni, Kai
Audiocommunication Group, Technical University Berlin

5.3 External's Source/addsi_pd.c File Reference

Pure data integration of the addsi external
The file includes the basic setup needed in order for the external to work with pure data.

```
#include "m_pd.h"
#include "addsi.h"
```

Data Structures

- struct addsi_tilde

  The Pure Data struct of the addsi∼ object.

Typedefs

- typedef struct addsi_tilde **addsi_tilde**

5.3.1 Detailed Description

Pure data integration of the addsi external
The file includes the basic setup needed in order for the external to work with pure data.

Author

Marius, Richard, Lenni, Kai
Audiocommunication Group, Technical University Berlin

5.4 Needed other C Files/vas_mem.h File Reference

Utilties for dynamic memory allocation

Wrapper for memory allocation Max/MSP SDK suggests using the Max/MSP "sysmem_" - routines instead of malloc/calloc/free So for Max/MSP define the Preprocessor macro "MAXMSPSDK".

```
#include <stdlib.h>
#include <string.h>
```

Functions

- void * **vas_mem_alloc** (long size)
- void * **vas_mem_resize** (void *ptr, long size)
- void **vas_mem_free** (void *ptr)

5.4.1 Detailed Description

Utilties for dynamic memory allocation

Wrapper for memory allocation Max/MSP SDK suggests using the Max/MSP "sysmem_" - routines instead of malloc/calloc/free So for Max/MSP define the Preprocessor macro "MAXMSPSDK".

Author

Thomas Resch
Audiocommunication Group, Technical University Berlin
University of Applied Sciences Nordwestschweiz (FHNW), Music-Academy, Research and Development
Tools for calculating convolution based virtual acoustics (mainly dynamic binaural synthesis)

5.5 Needed other C Files/vas_util.h File Reference

Utilty functions and all #defines for the VAS library

All kinds of utility functions, mostly vector math.

Typedefs

- typedef float **VAS_INPUTBUFFER**
- typedef float **VAS_OUTPUTBUFFER**

5.5.1 Detailed Description

Utilty functions and all #defines for the VAS library

All kinds of utility functions, mostly vector math.

Author

Thomas Resch
Audiocommunication Group, Technische Universität Berlin
University of Applied Sciences Nordwestschweiz (FHNW), Music-Academy, Research and Development