algorithm\_functions (funtkionen werden deklariert)

algorithm\_main (Funktionen werden aufgerufen, in richtiger reihenfolge, iteration)

Algorithmus Ablauf:

GUI gibt gui\_dict={sp#: Winkel, Abstand, Dateipfad}

Standardparameter wave sp#: 44100Hz, 1 Kanal, 16 bit

Aufruf nach GUI Aufruf:

get\_wave\_param {sp#: [2]}

return wave\_param\_dict={ sp#: [#\_of\_samples, samplerate, bit-coding]}

Aufruf nach GUI Aufruf:

wave\_param\_common=[samplerate=441000, bit=16]

Aufruf nach GUI Aufruf:

define output\_fps

Aufruf algorithmus\_block\_iteration:

set\_fft\_param ( output\_fps, gui\_dict{ sp#: [1 ]})

return fft\_blocksize=fixed=1024

return iterationcounter

return wave\_blockbeginend\_dict ={ sp#: [wave\_blockbegin+ wave\_blocksize, wave\_blockend+ wave\_blocksize]}

Aufruf algorithmus\_block\_iteration:

get\_wave\_block(gui\_dict{sp#: [2}, wave\_blockbeginend\_dict{sp#: [wave\_blockbegin, wave\_blockend]})

return wave\_block\_dict{sp#, [samplelist]} -> #sp: unterschiedliche blocksize

Entscheidend von sample a bis sample b speaker # als array einlesen

get\_fft\_block (wave\_param\_common, fft\_blocksize , wave\_block\_dict{sp#, [samplelist]})

return fft\_block\_dict{sp#, [samplelist]} -> #sp: fft\_blocksize gleich, bit gleich, framerate

Entscheidend interpolieren auf wave\_param\_common, blocksize muss am ende konstant sein

2 Funktionen:

get\_param\_wave{GUI-dict}:

...

return

def get\_hrtf({list\_of\_angles})

get correct hrtf interpolated

return {speaker#:hrtf\_list\_to\_speaker}

def get\_wave(frame\_begin, frame\_end, {path})

return {sp# : framelist}{sp# : #\_of\_samples : samplerate : samples : bit-coding

get