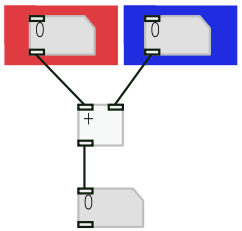
Vocabulary

Our vocabulary of objects to start with:

ctl-1	Object, still empty	Objekt, leer
	Object addition	Objekt zur Addition
ctl-2	Message Box	Nachrichtenfeld
ctl-3	Number Box	Zahlenfeld

Objects receive messages to act upon.



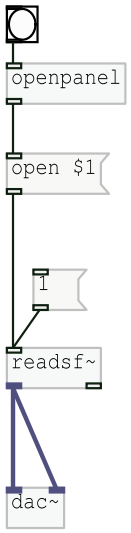
dry basics to start with:

1 Pd has edit-mode and run-mode

2 objects have one warm inlet, the leftmost.

3 objects have one or more cold inlets

let's_play



BANG! create an object-box and write "bng"

openpanel makes the file selector popup on a bang message. It spits out the path to the selected file.

$1 in a message box is a local variable and gets replaced by the file path from openpanel

play!

readsf~ means read soundfile. The ~ (Tilde) indicates we are dealing with audio signals.

The thicker patchcords as well.

Digital Analog Converter a.k.a the Loudspeakers.

Read the helpfile for readsf~ (right-click on the object)

Use .wav or .aiff soundfiles, not .mp3 or .aac

No Sound? Is Audio turned on? In Pure data as well?

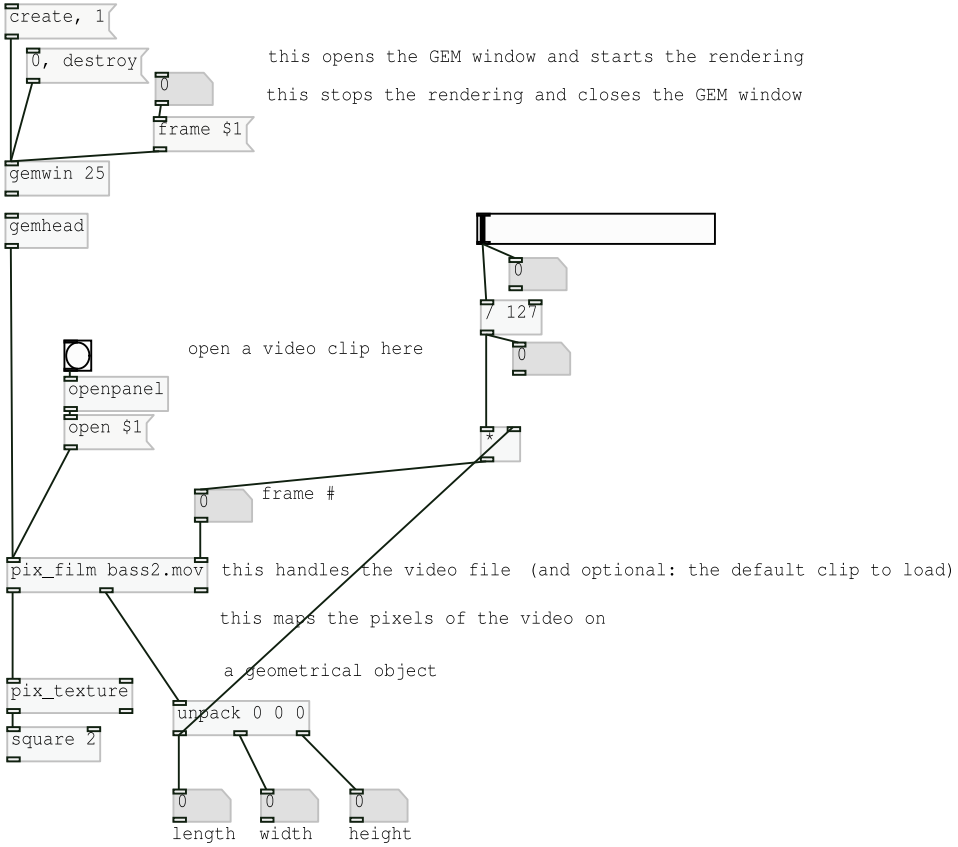
Max Neupert, GMU - Bauhaus-University Weimar 2009

let's_play_video

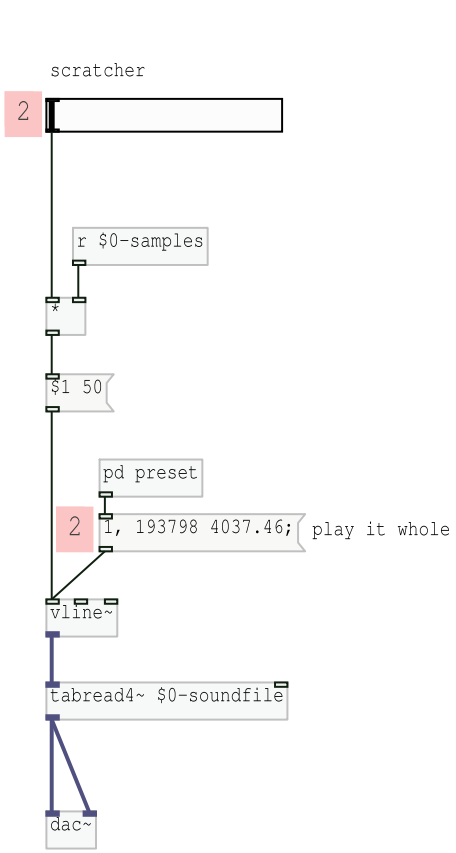
We'll use an extension (in Pd called external) to Pure data, GEM to play video

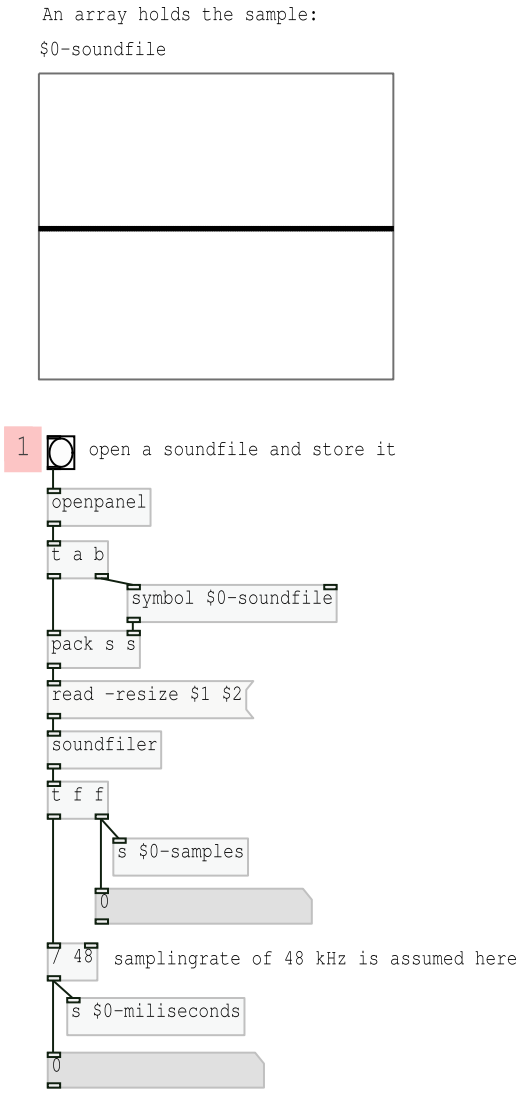
You load extensions either by adding it generally to the startup flags like this: -lib Gem

Or by declaring it to this patch (but only to this one) `declare -lib Gem`

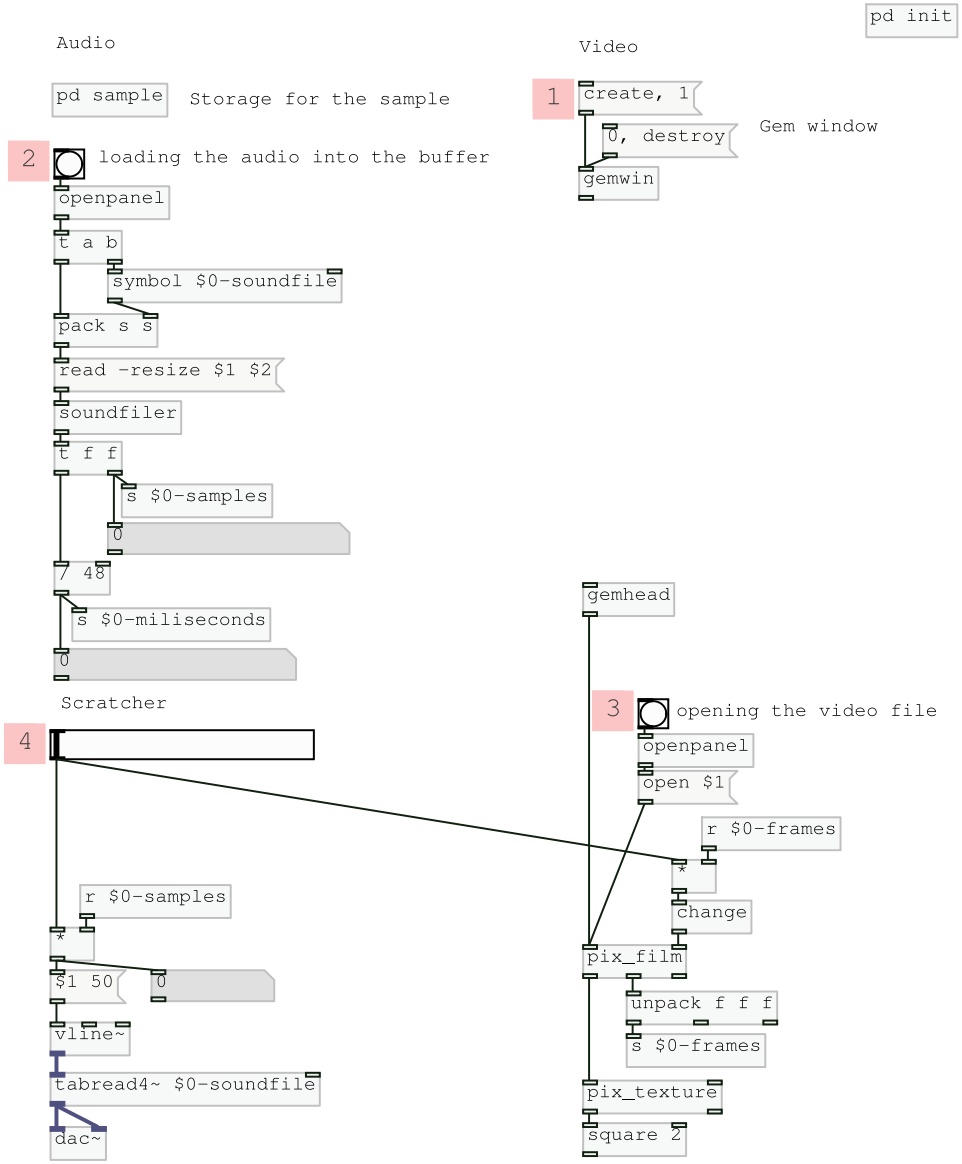


Soundfile_from_memory



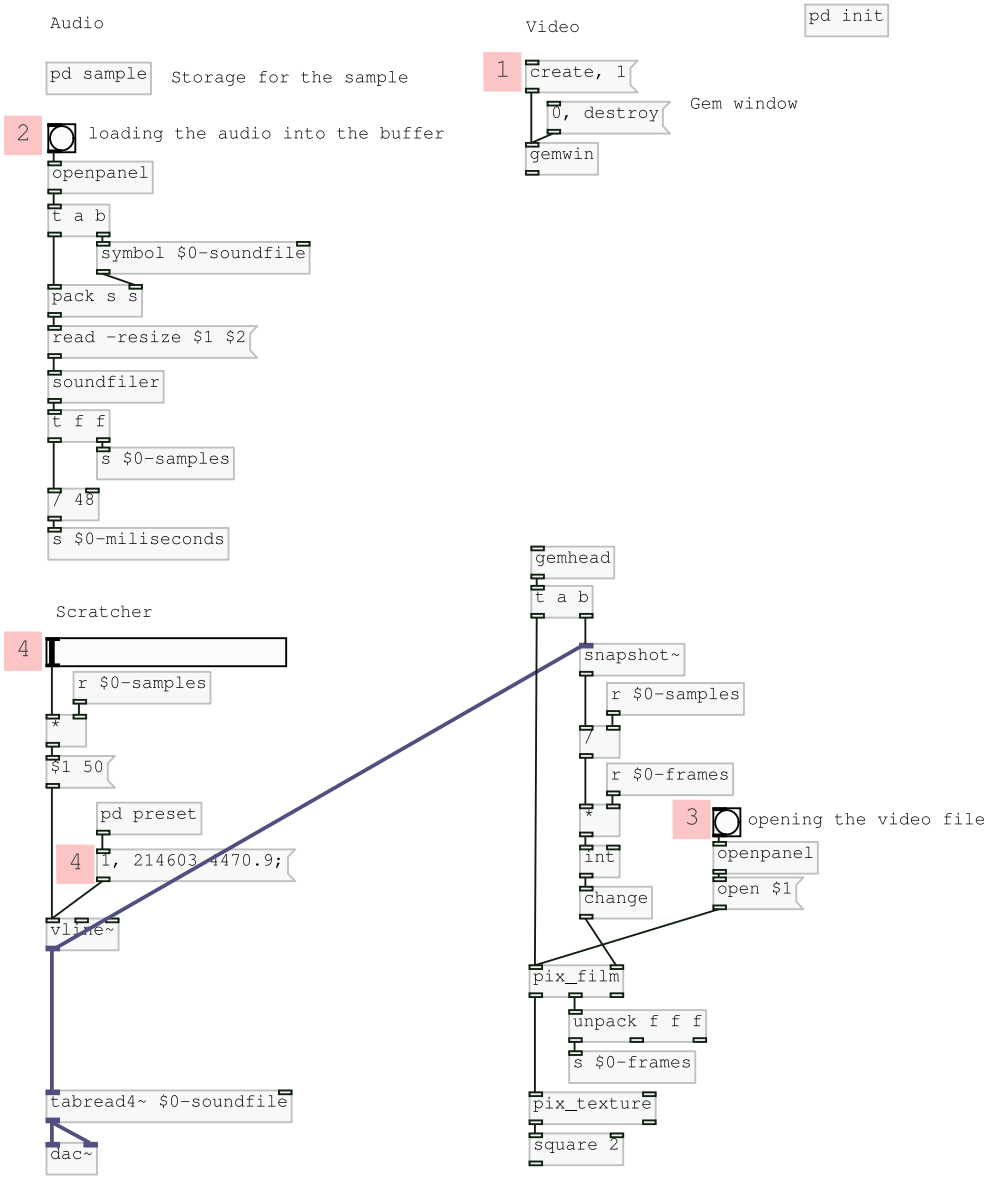


A/V_Combination



Max Neupert, GMU - Bauhaus-University Weimar 2009

A/V_Combination_better



Max Neupert, GMU - Bauhaus-University Weimar 2009

A/V_Combination_using_a_video_buffer

pd init

Audio

pd sample Storage for the sample

2 load sample into the array

openpanel

t a b

symbol $0-soundfile

pack s s

read -resize $1 $2

soundfiler

t f f

s $0-samples

/ 48

s $0-miliseconds

Video

pix_buffer $0-speicher 800 Storage for the video maximum 800 frames in this case

1 create, 1 create Gem window

0, destroy

gemwin

4

r $0-samples

*

pd laden

$1 50

4 1, 214603 4470.9; play

vline~

tabread4~ $0-soundfile

dac~

gemhead

t a b

snapshot~

r $0-samples

/

r $0-frames

*

int

change

pix_buffer_read $0-speicher

pix_texture

rectangle 3 2

3 load video into the buffer

openpanel

open $1

pix_buffer_filmopen $0-speicher

unpack f f f

s $0-frames

/

t b f

3

*

this adjusts the rectangle dimensions to fit to the aspect ratio of the video

A/V_Combination_using_a_video_buffer

pd init

2

openpanel

splitfilename .

this is part of the iemlib external

s $0-filename

Audio

pd sample Storage for the sample

Video

pix_buffer $0-speicher 800 Storage for the video maximum 800 frames in this case

Gem window

1 create, 1

0, destroy

gemwin

3

r $0-samples

*

pd laden

$1 50

3 1, 0 0;

vline~

tabread4~ $0-soundfile

dac~

r $0-filename

makefilename %s.wav

t a b

symbol $0-soundfile

pack s s

read -resize $1 $2

soundfiler

t f f

s $0-samples

/ 48

s $0-miliseconds

gemhead

t a b

snapshot~

r $0-samples

/

r $0-frames

*

int

change

pix_buffer_read $0-speicher

pix_texture

rectangle 3 2

loading the video into the buffer

r $0-filename

makefilename %s.mov

open $1

pix_buffer_filmopen $0-speicher

unpack f f f

s $0-frames

/

t b f

3

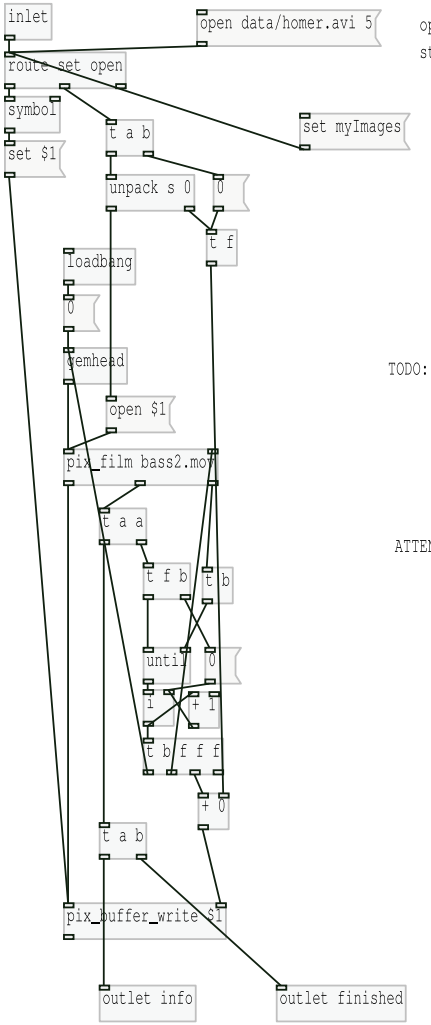
*

this adjusts the rectangle dimensions to fit to the aspect ratio of the video

Max Neupert, GMU - Bauhaus-University Weimar 2009

read a film into a buffer

declare -lib Gem



open film "data/homer.avi" and write it into buffer
starting at bufferposition "5"

set the buffer to write to

TODO: set upper limit to write to
guess write-limit from pix_buffer size

ATTENTION: this object will only work with rendering turned on