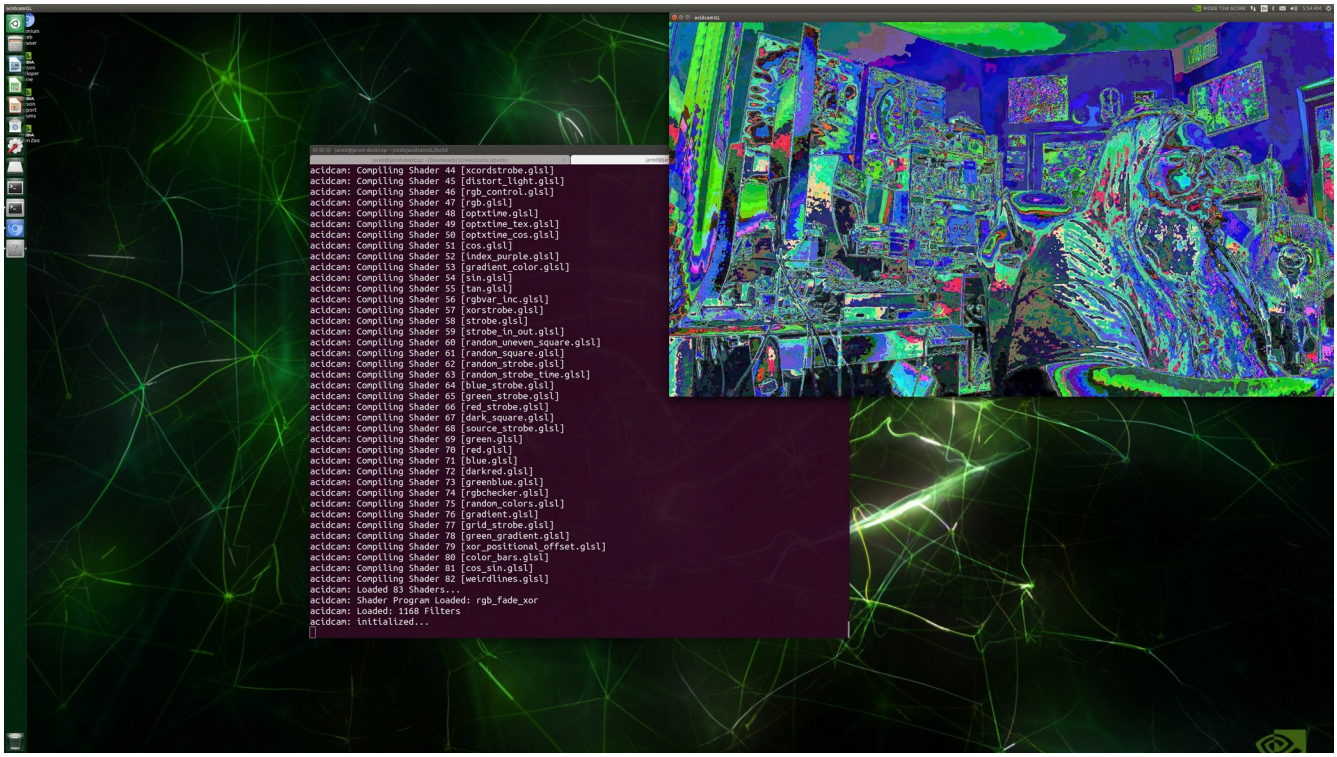
acidcamGL Manual – LostSideDead Software

Acid Cam is a project I started developing in 2011 as an augmented reality hallucination simulator. The project grew and had various iterations over the years including the v1 series, v2 series, Qt app, command-line tool, and OpenGL visualization instrument. Different variations of the project run on Windows, Linux, and macOS. The most popular version of the project is the v2 OSX video editor version with over 2,000 CPU-based filters.

Acid Cam filters are puzzle pieces of a visible language to bend and alter data to create abstract visualizations through mixing them together in different orders. It's a gift.
Available also is a collection of free stock footage I have created on my google drive that you can download for free and use in any of your projects or videos and evolve and expand. No credit or payment is needed. I want to share my love with you all.
The software is free, and the code is open source. You can download it on my GitHub or my website here: https://lostsidedead.biz

How the program works:

The program works when you load it from the command line you select either a camera, video file, or desktop. You choose whether you want to use a shader or a filter or both. You move between the filters and shaders via the arrow keys moving back and forth between them producing different combinations of visuals by pressing the keys in different orders. There are thousands of possible combinations just by pressing the keys in different orders. You toggle the filters on and off by pressing the space bar and shift the shaders up and down with the up and down arrow keys. You move between the filters with the left and right arrow keys. You add your own shaders by adding the name of the shader to the index.txt file in the path you pass to the program in the command you give it to start the program. You can use shaders and filters on your desktop as well as on cameras and video files. You can pass the program a playlist of filters and even shuffle them while the program is running or have the program randomize the playlist at the desired beats per minute. You can create custom filter stacks and add them to playlists.



Option 1: Compile From Source

How to compile on macOS:

This project reqeuires libacidcam,GLFW,GLEW,OpenGL,OpenCV

First install homebrew: https://brew.sh

Then install GLFLW,GLEW,OpenGL,OpenCV, ffmpeg

```
brew install glfw3 glew opencv ffmpeg
```

Dependending on what you have installed this could take quite some time. You can choose to statically compile

or use your package manager on Debian or Linux distro to install GLFW,GLEW, and OpenCV, FFMPEG

Next download libacidcam:

```
git clone https://github.com/lostjared/libacidcam.git
```

Enter the directory

```
cd libacidcam
```

Create a build directory

```
mkdir build && cd build
```

Configure the project:

```
cmake ..
```

Build the project

```
make -j4
```

Install the project:

```
sudo make install
```

Next download acidcamGL:

```
git clone https://github.com/lostjared/acidcamGL.git
```

Enter the directory

```
cd acidcamGL
```

Create a build directory

```
mkdir build && cd build
```

Configure the project:

```
cmake ..
```

Build the project

```
make -j4
```

for macOS x86_64 static build with Syphon enter the macos-static directory and

```
make -j4
```

You will still need to use install_name_tool on an app bundle see copy.sh for how I do it

Option 2: Run The .app Bundle

Download the .app from the Releases section of the GitHub page

https://github.com/lostjared/acidcamGL/releases

Then go to System Settings and select Security & Privacy. Select Anywhere as the location of apps.

Unzip
Copy to /Applications
Its path must be /Applications/acidcamGL
Open terminal use
You might need to adjust your security settings in the Settings by clicking allow after trying to run the program.

$ open /Applications/acidcamGL/acidcamGL.app --args -g -p /Applications/acidcamGL/filters

I have also included a test-server program so you can see the output that is sent to the GUI.
Place it in the working directory you are going to run the program from and use this command.

$ open /Applications/acidcamGL/acidcamGL.app --args -g -p /Applications/acidcamGL/filters -P && ./test-server

You will need to Control+C to exit after it says exited.

This would be the easiest/best method you could skip all that and just invoke it like this:
$ /Applications/acidcamGL/acidcamGL.app/Contents/MacOS/acidcamGL -g -p /Applications/acidcamGL/filters

Run the program:

```
Either use acidcamGL ./acidcamGL or /Applications/acidcamGL/acidcamGL.app/Contents/MacOS/acidcamGL
Depends on which options you have chosen.


./acidcamGL -g -p ../filters
```

The default Resolution is:

Camera: 1280x720 FPS: 24 Window Size: 1280x720

you can change that using arguments passed to the program How to use the program:

Arguments:

```
-X codec
-o output filename
-Z filter name
```

```
-S filter start index
-H shader start index
-C set color map
-T set material texture filename
-N play list slideshow timeout
-k shortcut-key file
-L playlist of filters
-b restore black
-g output debug strings
-u fps
-n print filter name
-e snapshot prefix
-p shader path
-M monitor index
-f fullscreen (resize)
-F fullscreen (windowed mode)
-d capture device
-i input_video.mp4
-R loop input video
-x Stereo Mode
-r resolution 1920x1080
-c Camera resolution 1280x720
-G Screen Capture Mode
-Y Enable Syphon Server
-U Screen Capture Position X,Y
-P Redirect Standard Output to Socket
-W custom filter path
-B enable playback filter mode
-q shuffle playlist
-w beats per minute for shuffle
-l list filters
-t list filters no info
-l list search
-v version
-4 enable ffmpeg x264 support
-5 enable ffmpeg x265 support
-m crf for x265 for video mode
--mux outputted_file source_file [ Mux audio (copy audio) ]
```

Controls:

```
L - enable disable playlist
N - set index to the end
P - index reset to zero
K - jump forward index by 25
J - jump backwar index by 25
Z - take screenshot
H - Shuffle Playlist
M - Enable/Disable Playlist Slideside random timeout
F - process keyboard input for index
S - process keyboard input for shader
C - clear keyboard input
[+] - increase blend percentage
[-] = decrease blend percentage
[SPACE] - Acid Cam filters enabled/disabled
[LEFT] - Filter index move left
[RIGHT] - filter index move right
[UP] - Shader Index move up
[DOWN] - Shader Index move down
[ENTER] - Jump to currently typed index (type index with number keys at anytime)
[SHIFT}+[ENTER} – Release Stored Frames
```

```
[PAGEUP] - Store Index Position
[PAGEDOWN] - Restore Position Index
 Q,Q,E,R,Y,U,I,O - move movement rate
 T - reset color offset
```

Run the program:

Full Screen Monitor 0

```
./acidcamGL -g -p filters -F -M 0
```

or Monitor 1

```
./acidcamGL -g -p filters -F -M 1
```

Windowed mode 1920x1080 Camera/Window

```
./acidcamGL -g -p filters -c 1920x1080 -r 1920x1080
```

Windowed mode 1920x1080 Camera 1280x720

```
./acidcamGL -g -p filters -c 1280x720 -r 1920x1080
```

Default Windowed mode:

```
./acidcamGL -g -p filters
```

Default Fullscreen (Resize) mode:

```
./acidcamGL -g -p filters -f
```

Default FullScreen (Windowed) mode:

```
./acidcamGL -g -p filters -F
```

Use Webcam Device by Index:

```
./acidcamGL -g -p filters -d 1
```

Use different frames per second:

```
./acidcamGL -g -p filters -u 30
```

or (if supported by your USB 3.0 Webcam)

```
./acidcamGL -g -p filters -u 60
```

or even better

```
./acidcamGL -g -p filters -u 60 -c 1920x1080 -r 1920x1080
```

Use Video File as Input:

```
./acidcamGL -g -p filters -i file.mp4
```

Record to MP4 file:

```
./acidcamGL -g -p filters -i file.mp4 -h -o outfile.mp4
```

List all included filters by index:

```
./acidcamGL -l
```

Grab from Screen (macOS only):

./acidcamGL -g -p filters -G -U X,Y

X,Y being location of screen, resize the window to desired width,height

Key shortcut format:

in a text file add each line for the desired keyshort cuts in this format:

key filter_index shader_index

an example file would be

```
F 75 0
Q 25 7
```

If you saved this to keys.key you would use it with -k key like his:

```
./acidcamGL -g -p filters -k keys.key
```

Playlist file:

use: ./acidcamGL -l

Playlist Now uses String names versus ID identifiers: to list the different filters then list them one after the other in a text file line by line like this:

```
StrobeEffect
SelfAlphaBlend
```

save the file then when in the program press the L key to toggle the playlist on and off and use the arrow keys to move through the list. To use the playlist file its he same as the keys just with Use:

```
./acidcamGL -L playist.txt -g -p filters
```

Shell Variables

Use these shell variables if you don't want to set the path variables every time you run the program set the *"/path/to/acidcam" to the location in your file system*

```
export ACIDCAM_PATH="/path/to/acidcam"
export SHADER_PATH="/path/to/acidcam/filters"
export AC_PLUGIN_PATH="/path/to/acidcam/plugin"
export AC_CUSTOM_PATH="/path/to/acidcam/custom"
```

Playing the instrument:

How it works:

You decide whether you want to do a live session on live video or on video file and record and pipe to FFMpeg. If you decide to pipe to FFMpeg you use the -4 or -5 switch for x264 or x265 and -m for the crf compression level. Then as the video processes you can press the keys in different orders and play the instrument to produce different effects. Shift up and down between the shaders and toggle on and off the filters. You can write your own shaders and filters In GLSL/C++. Shaders written in GLSL need to be added to the index.txt file that are passed to the start of the program in the -p path variable path or in SHADER_PATH environment variable. New filters C++ examples are shown in the plugin folder in the source code of the project.

Live streaming with OBS is easy on macOS just use the -Y command line argument after compiling as a static macOS library on x86_64 to output as Syphon server and use Syphon input on OBS. Otherwise you could use Window input on OBS. Use a Cam Link 4K to get video input from HDMI devices such a Playstation 4, NES classic, TG16 Mini, or Sega Genesis Mini.