Introduction to Video Editing With

free, open source, cross-platform video editor
v. 18.01.x 2018

https://www.Shotcut.org

This document is an exact PDF version of the **FLOSS Manual** available at :
http://write.flossmanuals.net/introduction-to-video-editing-with-Shotcut.

PDF version 1.3-1 3-apr-2020. Comments welcome.

Table of Contents

About This Guide

Video cameras, computers with web cams and screen-capture software, and video-capable mobile phones now make it possible for a vastly wider group than ever before to capture raw video footage -- of political events, computer software demonstrations, a concert, a trip to Iceland, your cats, or anything else under the sun conceivably worth preserving and presenting in video. 20 minutes spent on YouTube, or any other social media platform, should convince you of this explosion in videography in the present digital era.

However, raw video footage is very often less than useful or effective in its immediately-captured form; it typically needs to be (or could benefit from being) **edited**. Raw segments can be shortened, excerpted or joined together with appropriate transitions; opening titles and captions can be added; often a voiceover narration or a musical track is appropriate for information or atmosphere. When available, shots from different perspectives can be intercut, as in classic film and TV production. Video clips from entirely separate sources can be intercut to help make a point or tell a story. And beyond this there is a galaxy of 'effects' and enhancements that can be applied to video footage, just as we have come to expect in editing digital still photos with tools like Photoshop. Video or motion-picture editing used to be the domain solely of professionals, a highly labor- and skill-intensive craft that typically entailed extensive equipment and training; and in its more sophisticated forms video editing is still a high-skill professional occupation. But the good news is that the basics (and even many advanced aspects) of video editing are now accessible to everyone, thanks to an abundance of digital video editors that are relatively easy to use, and can run on relatively lightweight home computers. Best of all, many of these editing tools are free and a number are open-source.

In this guide, we will learn the basics of video editing using one particular open-source software tool, **Shotcut**, which is available for free download at Shotcut.org. Originally written (like much open-source software) for Linux, **Shotcut** now runs equally well on Windows and Mac OS.

If you have read this far, you may already have decided that **Shotcut** is the best video editor for your own needs, or is at least worth checking out. For our part, we did a fair amount of research into current 'best-free-video-editor' reviews from various established tech-review sites, and concluded that **Shotcut** well worth the effort to document in this way. Among the chief considerations here were:

- We wanted a free editor. An expensive tool may be perfectly appropriate for video and film-editing professionals, but the general public needs access to free editing tools to go along with our access to free video-making tools and easy sharing platforms.
- We wanted an open-source tool. Open-source software is worth supporting for lots of reasons, and some of them are very practical. Many ostensibly 'free' media authoring and editing tools are really just the free or teaser versions of commercial (paid) software: this would include video-editing tools like Lightworks, DaVinci Resolve, Hitfilm Express, Avid and others. In every such case, the 'free' version offers some incentive to upgrade to the paid version: typically this means restriction on features that you would like to use (range of exporting options, for example), or the mandatory inclusion of a brand watermark, or other annoyances. Open-source tools are as full-featured as possible from the start, with no motivation to force an upgrade.
- In addition, paid software tools are almost invariably much heavier in their system-resource requirements than their open-source counterparts, for many reasons including the open-source movement's primordial association with lightweight (and also open-source) operating systems. Very often, even the 'free' versions of commercial software will have essentially the same heavy CPU and memory needs as the paid versions, because they are essentially the same software, just with some features restricted.

Shotcut and other open-source tools are much lighter in their resource requirements, typically quicker to download/install, much quicker to start up for each session, and can generally be run on far more modest (and thus less expensive) hardware than most commercially-based tools.

- All of the above considerations still leave a range of good-to-excellent open-source video editing tools, prominently including (but not limited to) **VSDC**, **OpenShot** and **KDenlive** in addition to **Shotcut**. Each tool has its particular strengths and weaknesses. We feel that **Shotcut** is unusually lightweight as a download/install and a running program, even in comparison with other open-source editors with a comparable feature set. But each tool has its strong proponents; and indeed there is an excellent Floss Manual for Kdenlive (see flossmanuals.net/kdenlive/), which also serves as a general introduction to videography with a social-action focus. It may well be worth checking out several such tools before you settle on one, since it's free to do so.
- Finally, the Floss Manual framework in general is intended to address gaps in existing software documentation, and **Shotcut** is strikingly under-documented even by open-source standards. The **Shotcut** development team is quite small, and has clearly decided to focus their efforts on building and constantly improving their editor, rather than on documention. It therefore seemed appropriate to use the Floss platform to help promote and facilitate further use of this excellent tool.

With those considerations in mind, this guide will be an introduction to **Shotcut** in particular as well as the basics of video editing more generally, and all tutorials here will make use of **Shotcut** software, for the most part running on Windows, in its most recent general release as of this writing (Shotcut 18.01.x, ie the first version released in 2018).

The focus of this guide, and its wider applications

This manual offers a beginner (or perhaps beginner-to-intermediate) guide to using **Shotcut** for video editing. We have neither the space, time nor expertise to exhaustively examine all of the more advanced features of this software (which are multiplied by the incredibly wide range of video and audio file formats this tool is capable of importing and exporting). Our goal is to get you started. Moreover, while this manual borrows at times from the structure of the Floss Kdenlive manual noted above, where appropriate, we do not intend to offer, even to the extent that they do, a general guide to the wider world of video making as such [Beyond the Kdenlive manual, there is also a separate Floss Manual devoted to videography as such, at en.flossmanuals.net/video-production]. Some of our how-to examples will focus on the specific use case of editing video tutorials, such as tutorials for using other software. We believe that video tutorials, which can now be found extensively at many software developers' own sites as well as 'in the wild' on YouTube, are one of the more important and useful practical applications of the new digital videography. Indeed, this manual itself could as easily have been done as a video tutorial series, but we recognize that many users are still more comfortable with a written text-and-screenshot reference, as a jumping-off point in using a new tool like **Shotcut**. This guide will also walk through the features of the **Shotcut** UI in a systematic and referenceable way that is not really possible in video. But the general editing techniques that can make a how-to video more watchable and useful will also apply to most other uses of video, and we will be sure to include Iceland-trip and perhaps cat-video examples as well. We would also add that how-to tutorials themselves need not be restricted to software tools; on YouTube one can find outstanding tutorials on the basics of playing various musical instruments, home and auto repair, making pop-up greeting cards, and really any step-by-step process that can be captured in video and usefully shared with others. And, of course, many of these same basic video-editing techniques would apply as well to the use of video for any *other* kind of 'story-telling' as well, including documentary and purely fictional stories.

You will notice in the Table of Contents that this guide is divided into several main sections. The present Introductory section will continue with a chapter offering a brief history of video editing, as a basis for understanding the editing operations explored later. The next section, "Getting Started with **Shotcut**", begins with a guide to installing **Shotcut** on different operating systems, then offers a systematic tour of the **Shotcut** user interface, with a walk-through of the principal panels or windows of the UI with their various mouse-based or hot-key controls. It is probably useful to get this basic orientation to the UI layout of **Shotcut** to start with, particularly as it differs in some respects from layouts that might be more familar, such as Windows Movie Maker or Adobe authoring products. We then offer a series of 'How-To' chapters focusing on specific editing operations, step by step with instructions and screenshots. Some readers may prefer to dive directly into these latter chapters, as a way of figuring out the UI as they need to use it. This section includes only some fairly basic operations. In the next section, "Taking It Further," we will offer how-to chapters on some more advanced operations in video editing. This overall organization of the manual is designed to allow users to move directly to whatever aspect of **Shotcut** they need or want to learn about.

Video (and Audio) Editing: A Bit of History

Strictly speaking, *digital* video editing, like digital audio editing, has only become feasible in the past few decades, after the technological revolution wrought by the 'digitization' of previously analog modes of audio and video/ image recording, as well as the exponential growth in home computer processing power and data storage. By digitization here we mean the translation into pure digital code, which can be stored in computer files and manipulated in computer memory, of audio and visual information that had previously been stored and consumed in a wide range of physical media: magnet tape, vinyl discs, printed photographs or photographic negatives, celluloid film strips, and so on. Thus the specific technology we are exploring in this manual is still fairly new; and yet the language we use for the procedures involved in digital video and audio editing represents an odd mix of terms familiar from, say, modern word processing -- for example cutting, copying, pasting, deleting -- and far more arcane terms that are now for the most part simply useful metaphors, but once referred to actual physical operations in the long history of editing audio and motion-picture recordings. Simply to understand why we use these terms now, and what they *mean*, it is useful to understand something of that history. Indeed, in a wider sense audio-visual media are only *intelligible* to us, readable as forms of communication, because they make use of a vocabulary that has evolved over the last century or more, and because *we* are *also* naturally tutored in that same vocabulary, starting from nearly as early as we begin learning our own native spoken languages. But just because we are all fluent enough to easily *read* most audio-visual media produced today, it doesn't follow (at all) that we are all naturally fluent enough to produce it ourselves. If the present manual was a guide to the general discipline of videography or film-making, we would spend the following chapters working systematically through the grammar and syntax of that filmic vocabulary, which most of us tend to understand only at a purely intuitive level. Because this guide has much more practical ambitions -- getting you started with video editing using one particular software, **Shotcut**--we will confine the theory-and-history lesson to this single chapter (and invite you to move on to the next, if you're truly not interested). Suffice it to say, though, that digital media editing today has a great deal to do with the history of editing audio and moving pictures.

Audio: from Wax to Bytes

Most people are aware that Thomas Edison invented 'motion-picture' photography, to which we will return shortly. Less well known is his invention, about a decade earlier, of a wax cylinder which was the first commercially-viable sound recording and playback device. Cylinder recordings were mass-produced by the 1880s, and dominated the early sound-recording market until they were displaced by the gramaphone disc record format after 1910 ('gramaphone' because Alexander Graham Bell, of telephone fame, was an early popularizer though not necessarily the inventer of this next-generation audio format). With both cylinders and gramaphone disks, the mechanism of recording and playing was quite mechanical: in recording, the actual physical impressions made by sound waves (as amplified by a microphone) were imprinted into wax on the rotating master cylinders and later wax discs. For playback of the mass-produced consumer item, the wax cylinders and (later) shellac-vinyl discs caused a needle to move with these impressions, which were essentially bumps in a groove, more or less literally reproducing the original sound wave and then amplifying it through a speaker, eventually with the aid of further electrical amplication. It can be seen that the possibilities of audio 'editing' were extremely limited with these technologies. One could decide when to begin and end a recording, which recording 'take' to mass-produce (sometimes), and once vinyl discs became long-playing enough to include multiple audio tracks, you could select which tracks to include on the record. No editing at all, of course, was possible on the consumer side.

This basic situation prevailed in the audio world until the next major technological revolution, the invention and perfection of magnetic tape recording (just before, during and after WWII). The recording process was still in a sense mechanical, but now it was a matter of sound waves re-arranging magnetized iron-oxide dust which clung to a long, narrow acetate (plastic) tape. The mass-produced end product was still usually a vinyl record, with that format dominating the music industry right into the 1980s; but in the recording studio, technicians were now free to physically manipulate the master 'tapes' to a considerable degree: cutting, splicing, copying; and after the perfection of multi-track recording, mixing multiple tracks together. True audio editing was now possible, albeit in a labor-intensive form and only with specialized studio equipment (along with scissors and tape). Beginning in the 1960s, groups like the Beach Boys and Beatles began pushing these editing techniques to sometimes exotic extremes, as part of their more general sound-engineering experimentation. On the consumer side, a small measure of this new freedom to edit became possible once consumer-level tape recorders were available, especially after the compact audio casette format replaced bulkier reel-to-reel machines, though 'editing' here was still largely limited to what could be recorded and overdubbed; it was the hardy soul indeed who attempted to cut-and-splice the actual 4-mm-wide tape of an audio casette. Still, by the 1970s and 80s the homemade compilation (or mix) tape had become an art form of its own among many consumers.

By this time, however, a far more radical revolution was brewing in computer labs and recording studios: the advent of digital sound recording, in which the audio wave itself was 'sampled' at regular and very frequent intervals, and the results coded into a digital sequence that could be de-coded, at playback, into something much like the original sound, with an audio fidelity corresponding to the rate of sampling and the resulting density of information stored. At first this simply meant that studios could record with considerably higher fidelity, and were also free to manupulate recorded sounds far more widely, and sometimes wildly (notable pioneers here were artists like Peter Gabriel and Kate Bush in the early '80s). Subsequently, digitization meant that the vinyl record was finally replaced by the compact digital audio disc, or CD, across the consumer audio market. Technology for *home*-recordable CDs was available, at least in theory, by the late 80s; but the practical limitation was the capacity of home computers to hold and manipulate, in memory, the considerable amounts of digital information involved, and to store and share the resulting digital files. But of course Moore's Law and the internet would take care of that: with computer capacity doubing roughly ever two years, by the late 90s it was quite feasible, on a home computer, to record, edit and then share your own audio in .wav and other file formats. After that it was simpy a matter of more and more mature editing software being developed. In a contemporary sound-editing tool such as Audacity (also free and open-source), we can record or import, cut, splice, cross-fade, amplify, distort, sample and mix down multiple audio tracks--even though every one of these terms actually refer to the simple manipulation of digital information, rather than to the literal, physical, mechanical operations they once meant for sound engineers, before the digital era. It's simply easier to visualize what you're doing if you use the older, mechanical-era terms.

Video Editing: From the cutting-room floor to Shotcut

But this is a manual about digital *video* editing, right? It is; but the history of audio editing is germaine on several counts. First, of course, your video 'tracks' will likely have or require audio 'tracks' as well, which will need their own editing (and while some of this can be done in **Shotcut** or other video editors, more complex sound editing may require the use of a tool like Audacity itself). Second, the history of digital *video* recording and editing runs largely parallel to that of digital audio, with a 3 to 5-year delay at every stage since the technological demands of storing and manipulating digitalized video information are much greater than with audio (just think about the difference in size between a 5-minute mp3 audio file and a 5-minute mp4 video file). Now, working

backward from the digitizing revolution, it happens that for several decades prior to that, video information was *also* often recorded and stored in analog form on magnetic tape; and this video tape could therefore be edited in something like the same physical way as magnetic audio tape. However, analog video tape was never really able to render high-quality, high-resolution motion-picture images, and so was typically reserved for either home video recording (which gave us our first widespread use of the term 'video' itself), or for recording television shows in a period when televised images were typically low-resolution in any case. *High*-quality motion-picture recording still required a technology that was essentially unchanged from its invention way back around 1890, by several inventors simultaneously but most succussfully by an employee of Thomas Edison. This was the motion-picture or 'movie' camera, which worked by exposing a series of still photographs very rapidly in succession (first 12 or 16, then 24, and finally 32 still images or 'frames' per second), as they passed the camera lens on a celluloid strip. In that era, still photography was already fairly advanced, and some photographic cameras had begun to capture their still images on a long strip of negative paper or celluloid (such 'film rolls,' first invented by the Eastman Kodak Company, would remain standard media until the advent of digital cameras). But while these strips of still-photograph negatives were then developed onto separate sheets of paper for the final photographs, the new *movie*-camera film was developed or transfered onto another long, thin celluloid strip; and that strip could then be run past a bright-light projector at the same frame-speed at which the images had been 'shot' or exposed originally. The result, for the viewer, was a sequence of still images succeeding each other so rapidly that the brain processed them as a single, continuous, *moving* image: whatever moving objects had been 'shot' originally appeared to come back to life on the moving-picture screen.

Revolutionary as this was in the general history of media, for our purposes what is important here is that, almost from the very beginning, the nature of motion-picture technology itself meant that it could be, and indeed had to be, *edited*, in a way that early audio media simply could not be. The very first motion pictures shown to audiences, in the late 1890s, were simple novelty pieces consisting of a single, perhaps unedited 'shot' of a few minutes duration. But soon after that, film-makers began trying to tell real *stories* in their films, whether documentary or fictional. And from that point on, however well one had planned and set up one's scenes and cameras, however well-rehearsed the actors might be in a fictional production, it was never possible (then as now) for the raw footage to simply become the end product shown to viewers. For a single scene or even a single "shot"--a few moments of action as shown from a single camera's perspective--one might need any number of 'takes' to get it right, meaning that an hour's worth of exposed film might only yield 10 minutes or less of useable film 'footage' (film strips being so long by this point that it was only practical to measure their length in feet). This useable footage had to be found, cut out from the rest, and spliced together--leaving the remainder "on the cutting-room floor." Once directors began shooting scenes with multiple cameras at once, these different shots needed to be spliced together in appropriate sequences, 'cutting' back and forth for example between one actor's face and another, or between two speeding vehicles, or an actor and a wall-clock. Also, of course, up until the late 1920s all films were silent--there was no technology till then for a synchronized sound track--which meant that all dialogue, as well as introductory and scene titles, and other expository information, had to be presented in the form of static text 'titles' (the same text image repeated in enough frames to appear long enough for the viewer to read), and these too had to be painstakingly 'intercut' at the appropriate times. Of course, even after the advent of sound in film, movies made in one language would need subtitles added in for a different audience. Finally (just in terms of the most basic film editing), there was the matter of how to *transition between scenes*. It was too confusing to the viewer to simply jump instantly from one scene into the next, as you generally would between shots within a single scene. So film-makers invented a whole panoply of ways to effect these transitions, over a few seconds duration: 'fading out' one scene to black (or more rarely white) and then 'fading in' the next; 'dissolving' one scene into another; 'wiping' from one scene to another--by way of a line moving left to right, right to left, top to bottom, etc; or more exotically as an 'iris wipe', in which the new

scene appears as a circle from the center of the previous scene and expands outward, or begins from the outside and moves in til the previous scene disappears. A huge amount of the story-telling burden in motion pictures can be carried by the way they are edited together; indeed one could take the same raw footage of a real or staged event, and edit it differently to tell widely varying stories (which is why the "Director's Cut" of a classic film can be sometimes startlingly different from the familiar version released by the studio).

In addition to editing as such, film often had to be 'treated', altered or enhanced after it was exposed, to give it a more appopriate look for the audience. Sometimes a shot was perfect but slightly too dark, and had to be lightened, or vice versa; or the contrast between light and dark heightened, or lessened; or the film had to be tinted slighty from the original black-and-white. Indeed, in the first decade of film there were occasional efforts to 'colorize' short films (as still photographs had been colorized for decades), which at that time meant painstakingly hand-coloring each individual frame--a practice nearly as labor-intensive as making hand-drawn animated films, and thus not long continued. And once color film-photography itself became feasible, by stages during the 1930s, the resulting coloration not infrequently had to be 'corrected': changing the hue or saturation or contrast level, for example, which could be done either chemically or by projecting the film through a filter of some kind. This practice has continued in film-making up to the present -- indeed if anything the temptation to play with color and light effects is probably greater now, since, like everything else, it can be done with a fraction of the labor in the digital era.

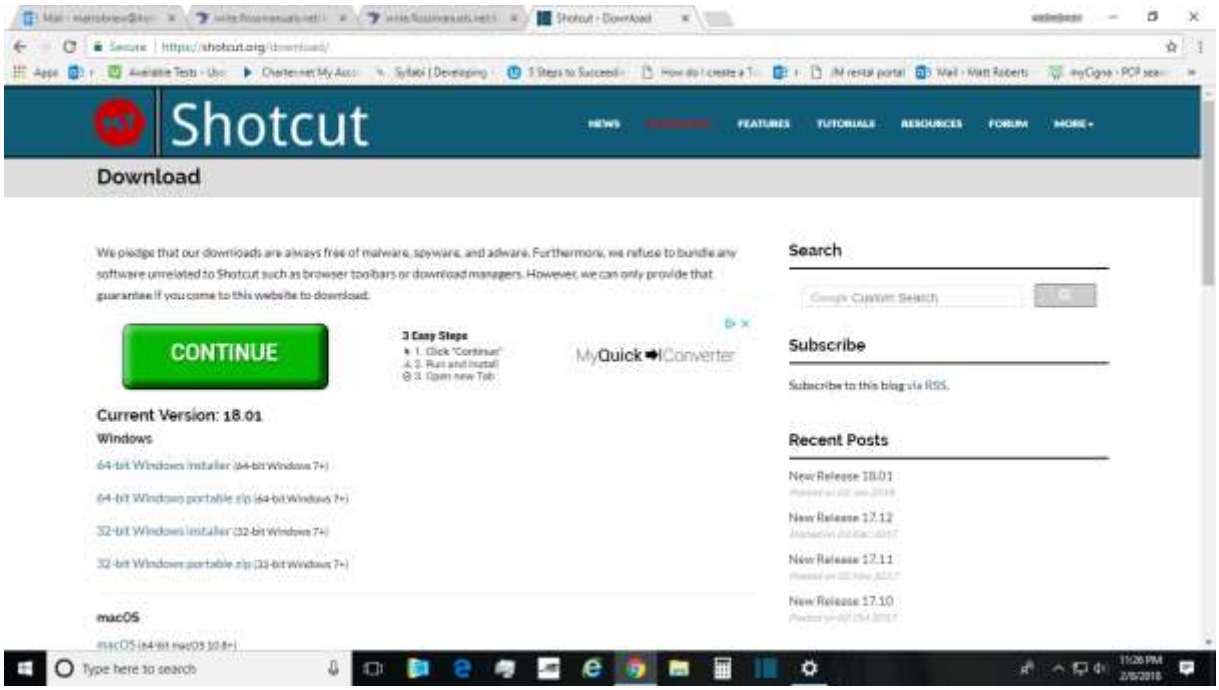
And that really brings us to our major point here: every one of the operations of editing and enhancement discussed above, as applied to historical film-making, can now be accomplished through video-editing software like **Shotcut**. Quite often we *need* these features, because storytelling of any kind through video still depends, to a great extent, on many of the same techniques pioneered by film-makers (and television and music-video producers, following them) over the past 12 decades. And just as we saw in the case of digital audio editing, in digital film editing most of the specific operations (though not all of them) have the same names as the corresponding visual effect in a pre-digital film, even though they are no longer attached to the *physical process* from which the name originated: shots and clips, cutting, splicing, fading in and out, wiping, tinting and other color enhancement, applying filters, bringing in subtitles and intertitles. In the 'how-to' chapters of this manual, we will examine examples of most of these operations, as well as a few basic operations that *wouldn't* have made sense before the era of digital editing: how to open a video file and import further assets; how to add an audio track to your video, or strip one away, etc; and how to export your final product.

With this excursion into theory and history out of the way, it's time to take a look at how we can install the **Shotcut** software, and then dive into how **Shotcut**'s user interface exposes the controls to make these kinds of editing operations possible.

Installing Shotcut

As noted in the previous chapter, one major virtue of **Shotcut** is how lightweight it is, as a download/install and as a running program. One consequence is that the installation process itself should be straightforward and brief.

From the Shotcut.org home page, "Click to download" brings you to the download page, which offers numerous options for Windows, Mac OS and Linux operating sysems:

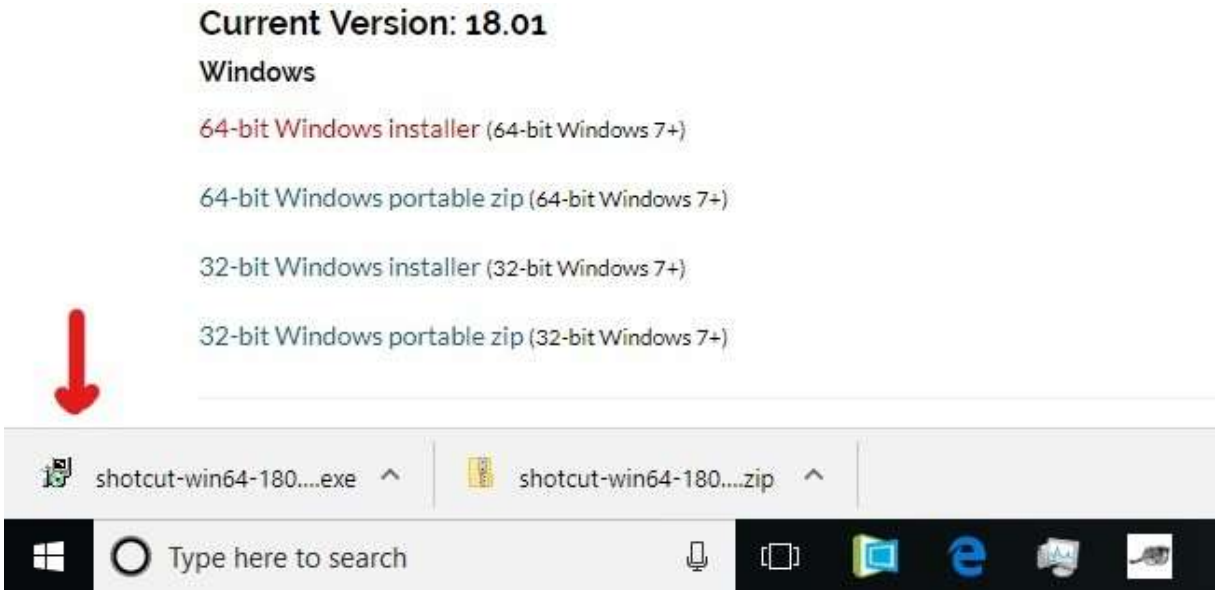


Please ignore the green "Continue" button, which will download a "MyQuickConverter" tool; this is an unrelated Google ad. Find your OS and version: mine is Windows 10, 64-bit. The option here is for Windows 7+, which should cover most Windows users (if you're still running Vista or something, it's probably time to upgrade).

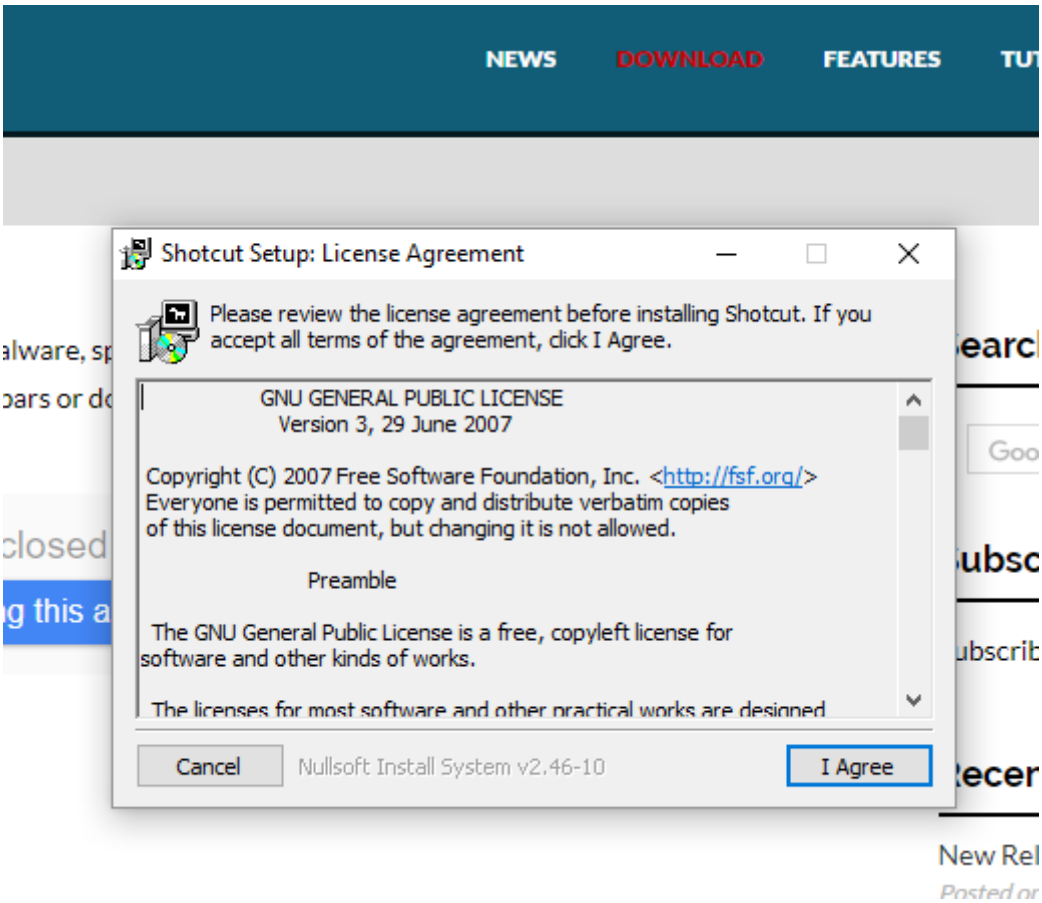
Installing on Windows

Within Windows 64-bit (as in 32-bit) you have the option of either using the Windows Installer version or downloading a portable zip of the install. The latter is a not-unreasonable option if you'd like the app to be portable, because the entire install is relatively small, only 256 Megs for the zip. But I only need it on one computer, so I will choose the Windows Installer option. This will download the install as an executable, which in Chrome will be visible (once the download has finished) at the bottom of the browser, like so:

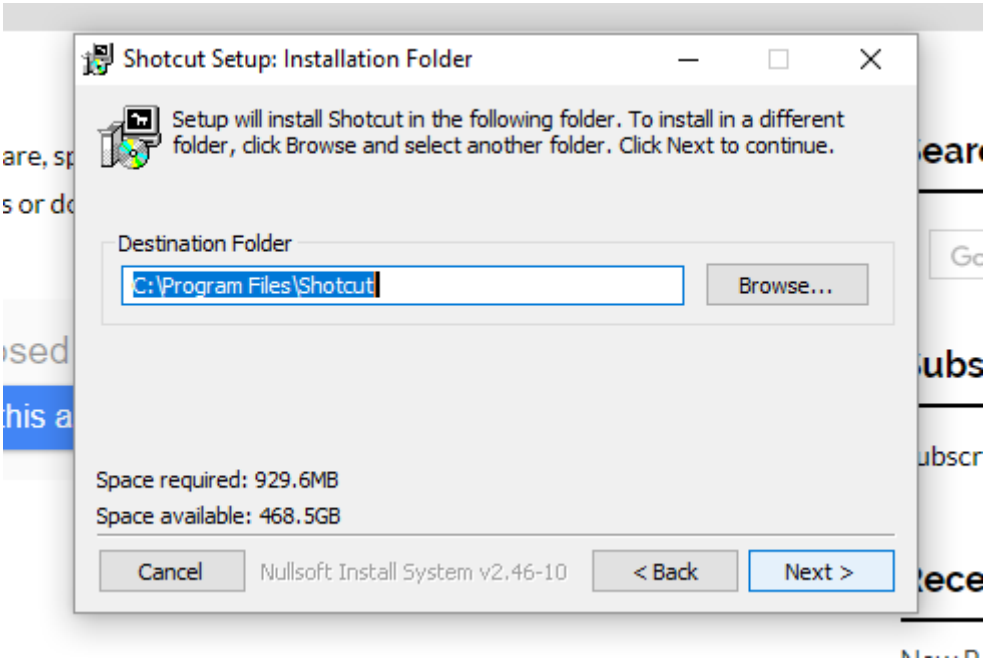
(cont'd next page)



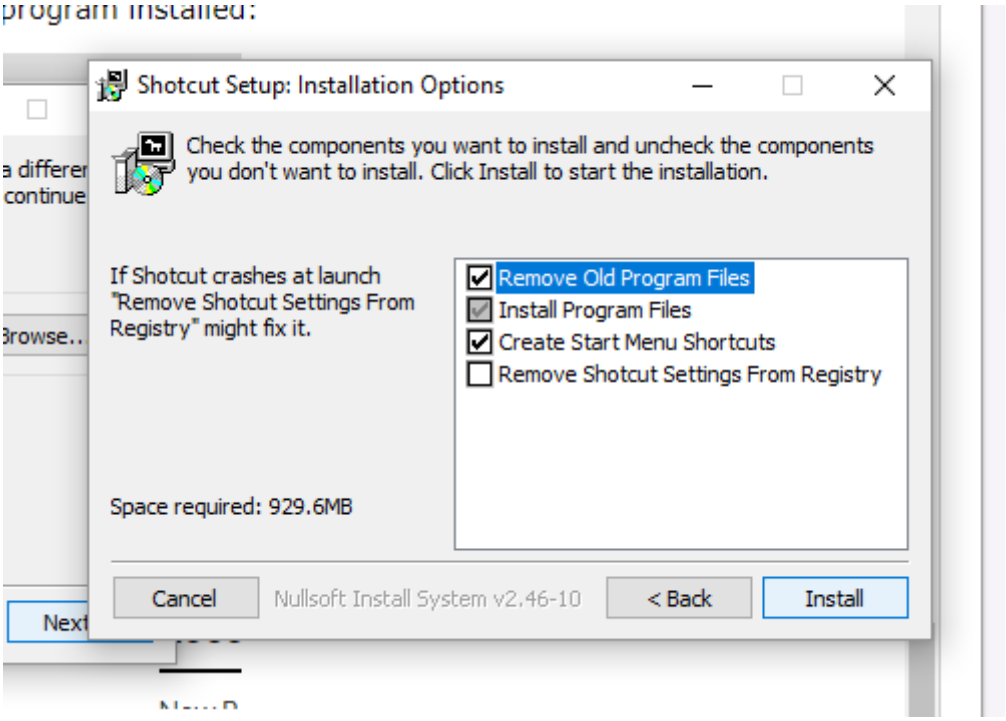
Clicking on this exe will first prompt Windows (at least 8 and 10) to ask if you're willing to make changes to your machine, as it does with any new program install. The answer is yes. This will launch the familiar Windows Installer dialogue:



Read the License Agreement, if you're not familiar with the GNU open-source license, and then agree. Next, as usual, the installer will ask you where you want the program installed:

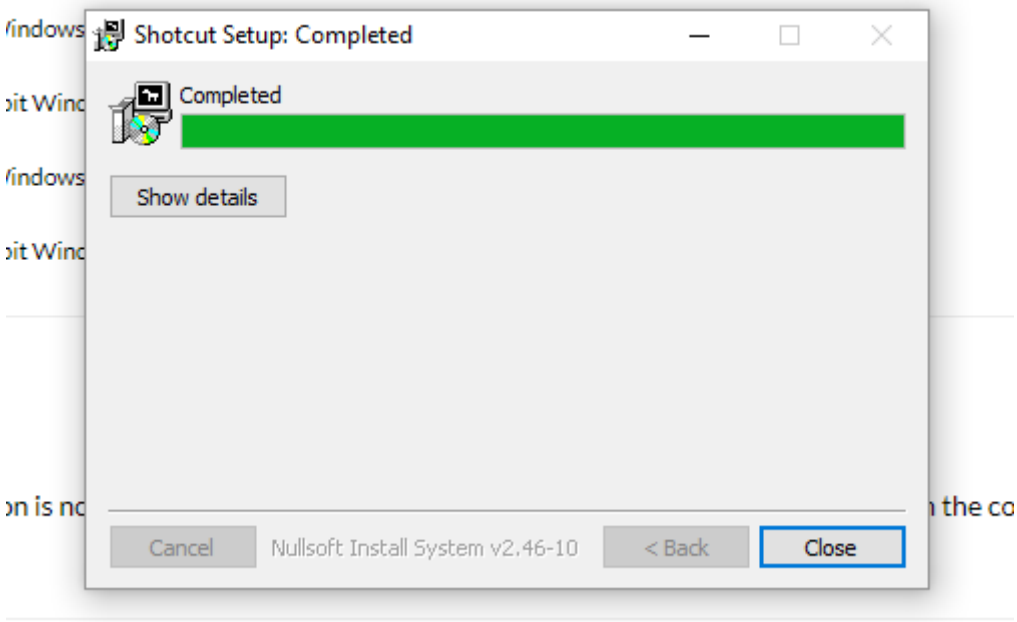


Unless you have good reason to put it elsewhere (like another drive), the default destination of C:\Program Files is generally a good idea since you and your computer will easily know where to find it subsequently. In any case, after you're satisfied with the install location, click Next. This will launch one final dialogue, with some options for the install:



Since this happens to be a re-install in my case, even though I uninstalled the last version already it's a good idea to keep "Remove Old Program Files" selected. I have a cluttered Start Menu so I have unselected "Create Start Menu Shortcuts." When you've made your own selections, click Install to begin the actual installation process. You will now see a progress bar with the installation progress. Again, this is not a particularly heavy install, especially for how full-featured the editor is. On a laptop with fairly modest memory and CPU, my install process took under three minutes. As they've suggested in the previous screen, if your install crashes, you might try again with "Remove Shotcut

Settings from Registry" selected. When the installation is finished successfully, you'll be notified with a final dialogue:

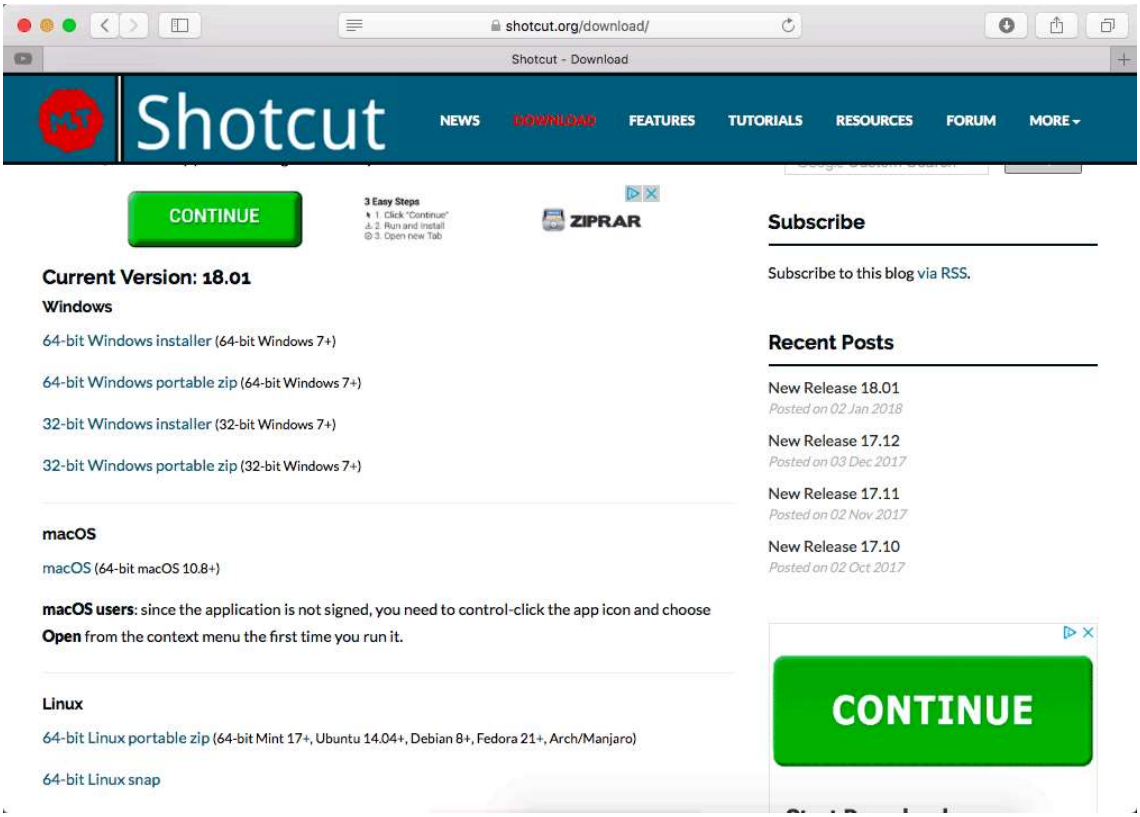


Clicking 'Close' will finalize the install process. You're ready to go! **Shotcut** has elected not to include the "Launch application" option at this point, so if you're ready to start checking out **Shotcut** now, you'll need to find it in your Start Menu, under "Recently Added" or the alphabetical listing of programs.

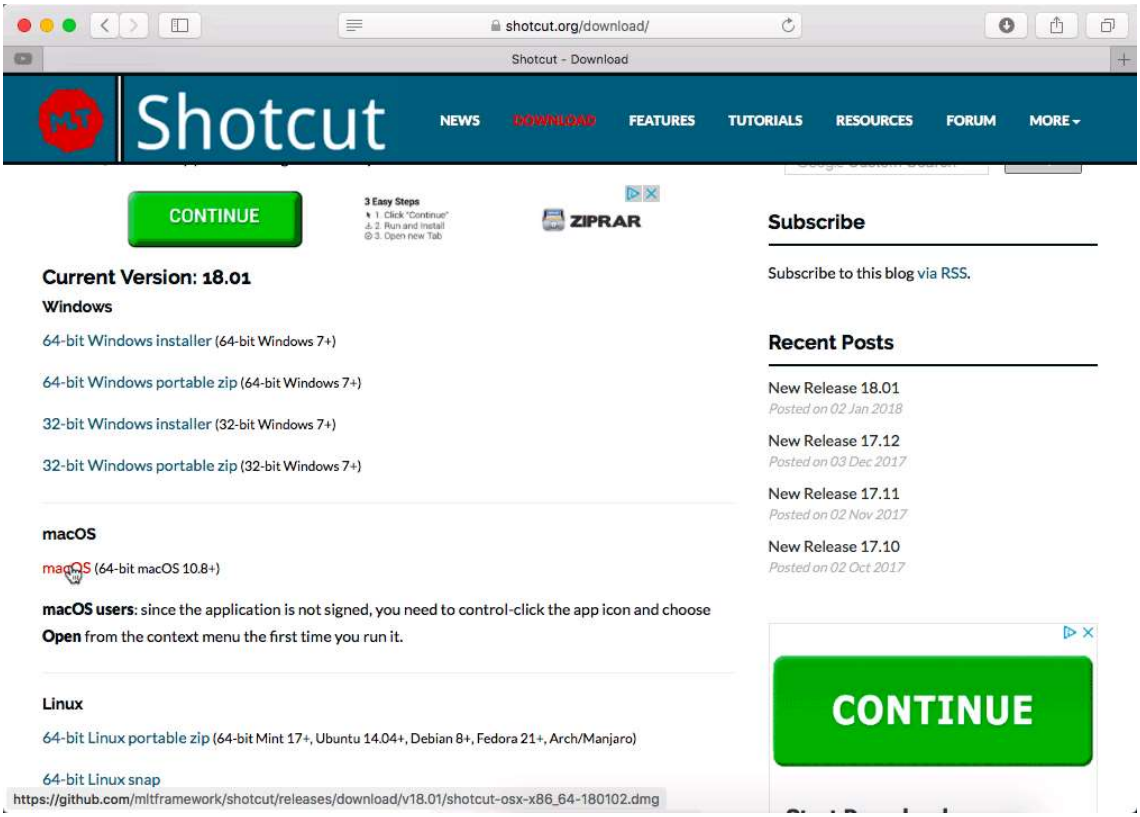
Installing on Mac OS

1. In the download page of Shotcut.org, scroll to where you can download **Shotcut** for the Mac.

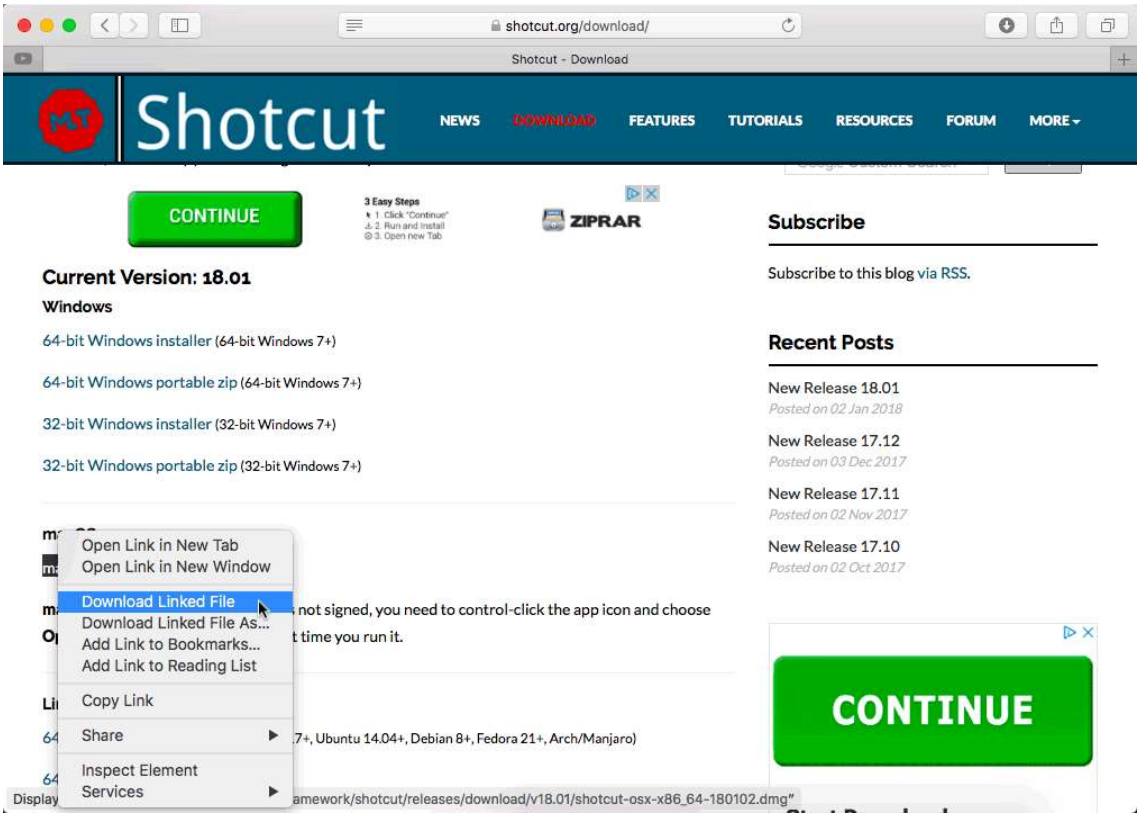
(cont'd next page)



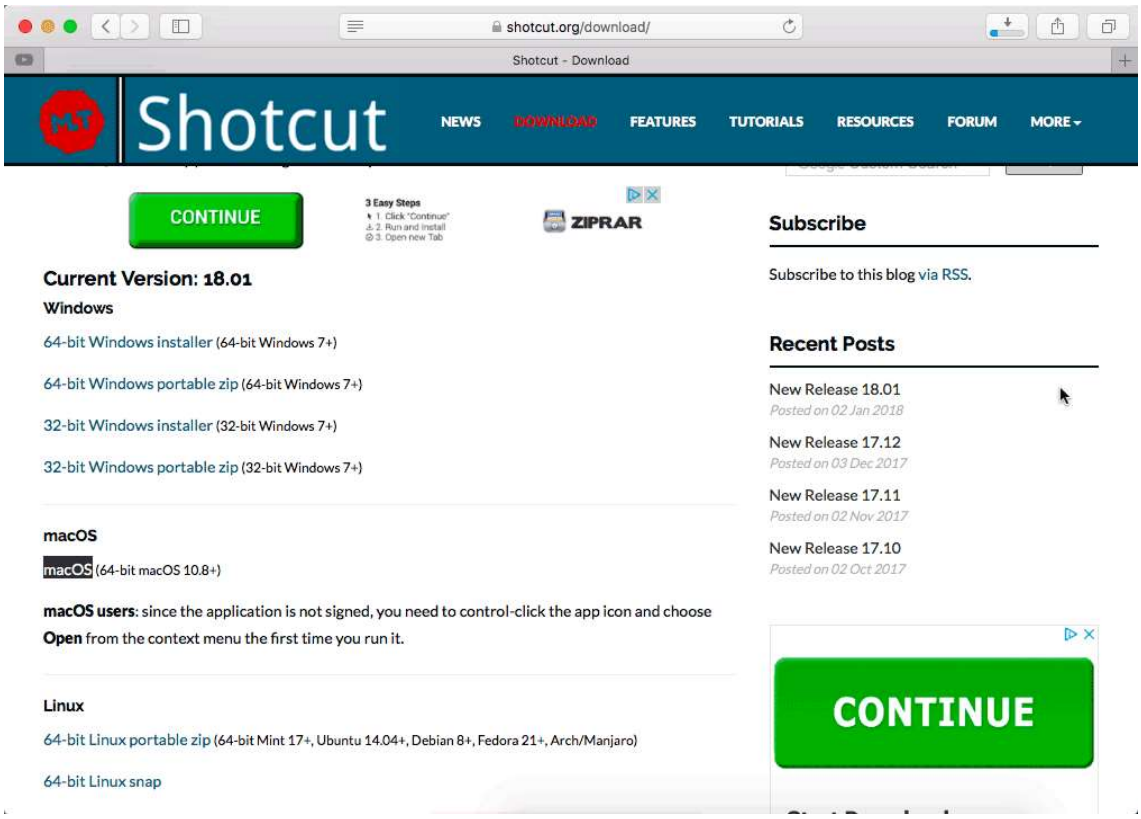
2. Move your mouse so that the cursor is over the link for downloading **Shotcut**.



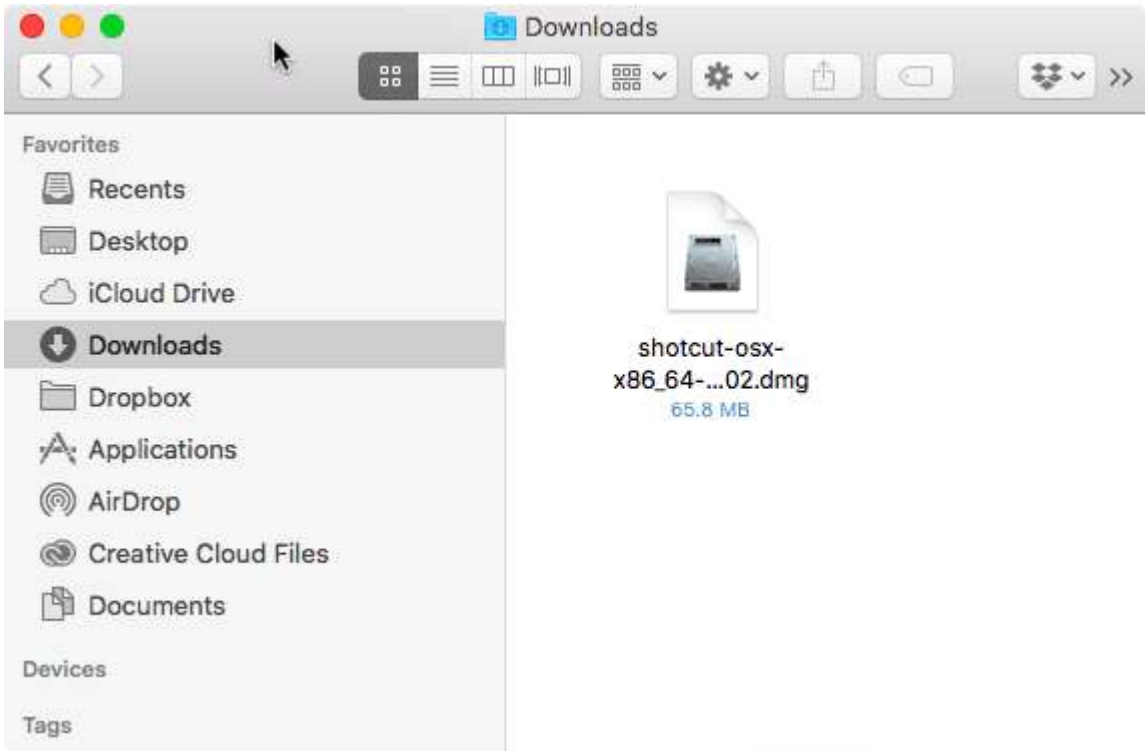
3. Following the instructions for MacOS users, control-click the link for downloading the Mac version of **Shotcut**, and highlight Download Linked File... or Download Linked File As...



4. **Shotcut** will now start to download.

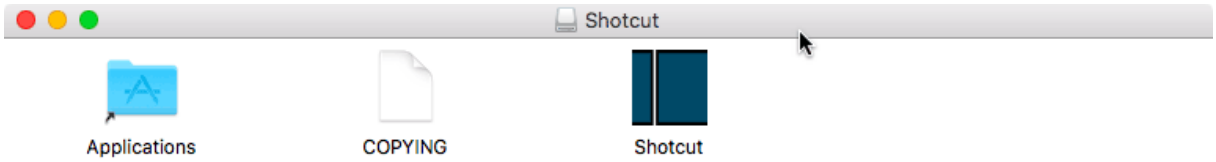


5. Once **Shotcut** is fully downloaded, go to the folder your browser is set for files to be downloaded into — usually it's the Downloads folder (~/Users/nameofuser/Downloads).

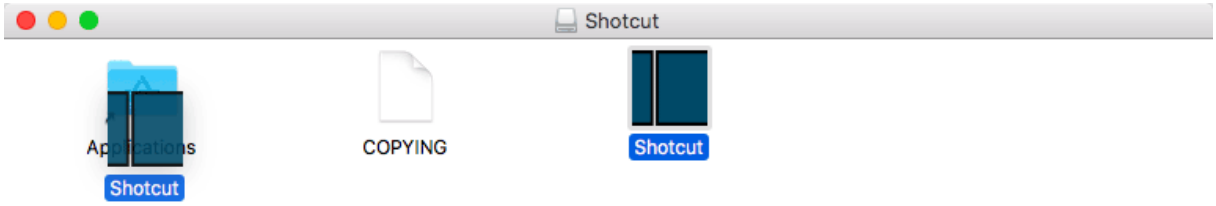


6. Double-click the **Shotcut** .dmg file to open it.

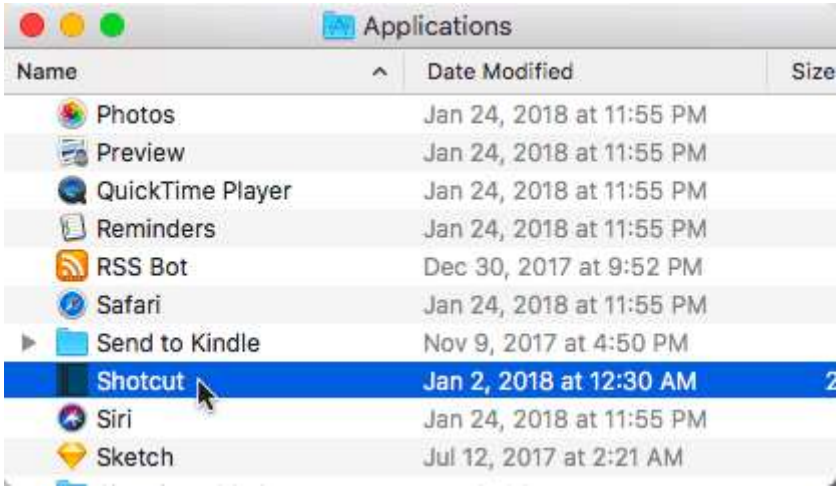
7. A new window with the **Shotcut** application will appear.



8. In this window, click and drag the **Shotcut** application into the Applications folder (~/Users/nameofuser/Applications).



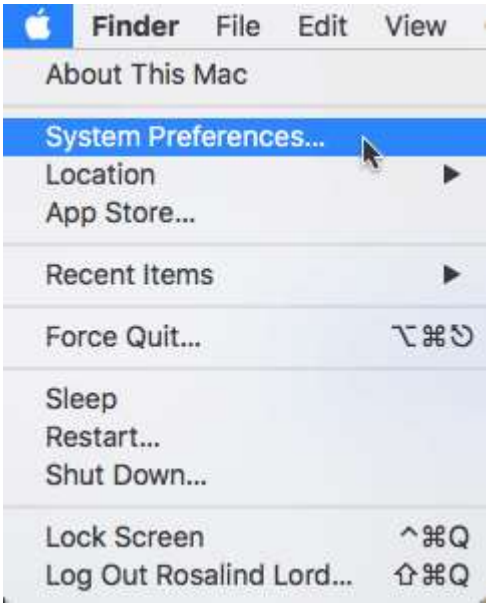
9. Double-click **Shotcut** in the Applications folder (~/Users/nameofuser/Applications) to open it.



10. A new window asking if you're sure you want to open **Shotcut** since it's from an unidentified developer may appear. If it does, click Open, then do the following.

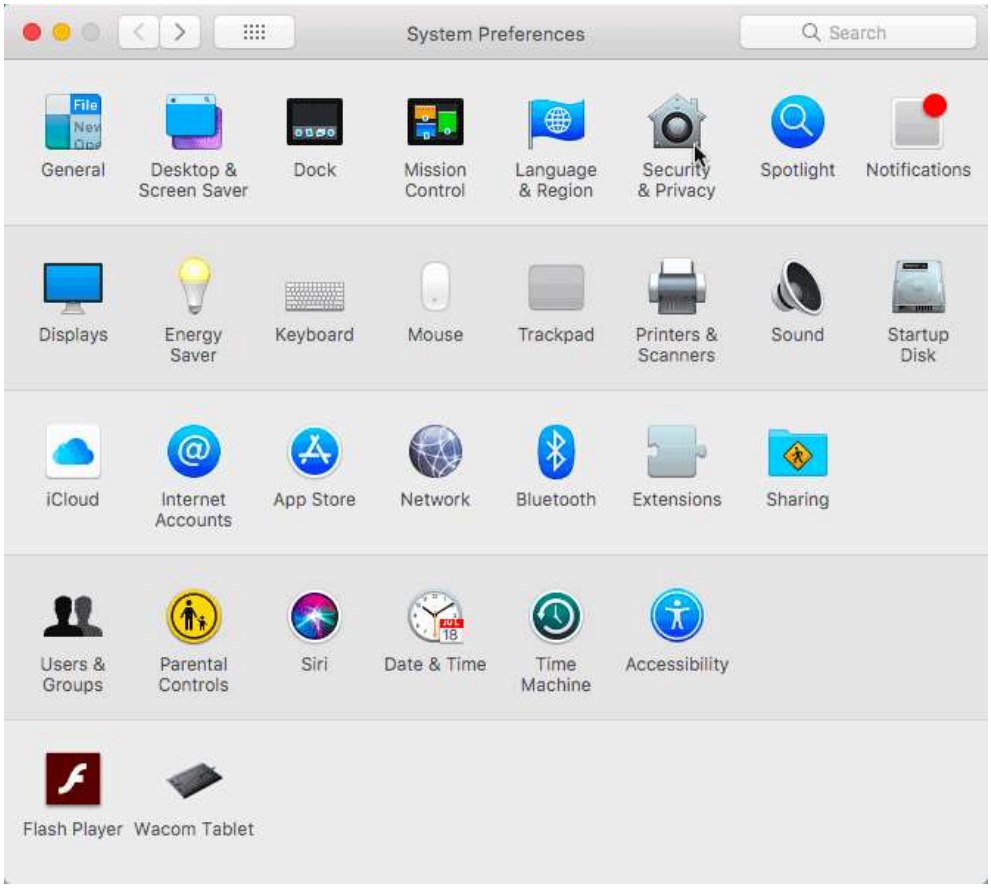
A. Another new window saying that **Shotcut** can't be opened because it is from an unidentified developer will appear. Click OK.

B. Open your System Preferences by clicking the apple in the upper left-hand corner of the screen, then highlighting System Preferences...

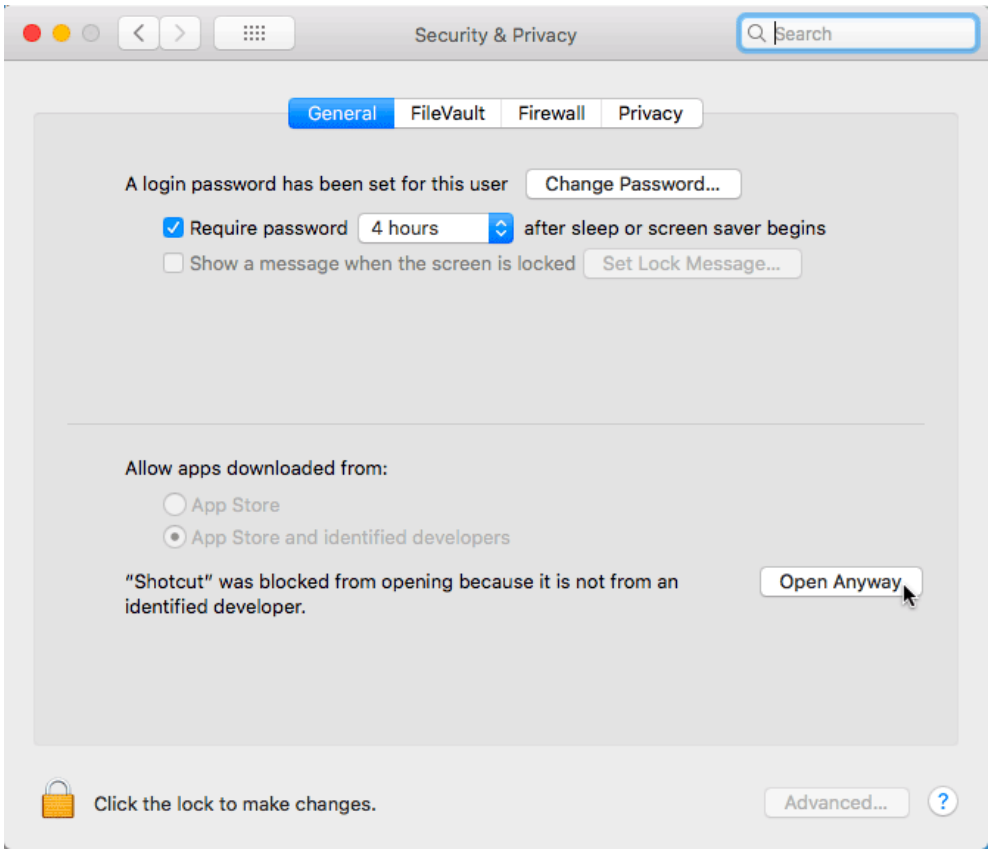


C. The System Preferences window will appear. Click the Security % Privacy icon.

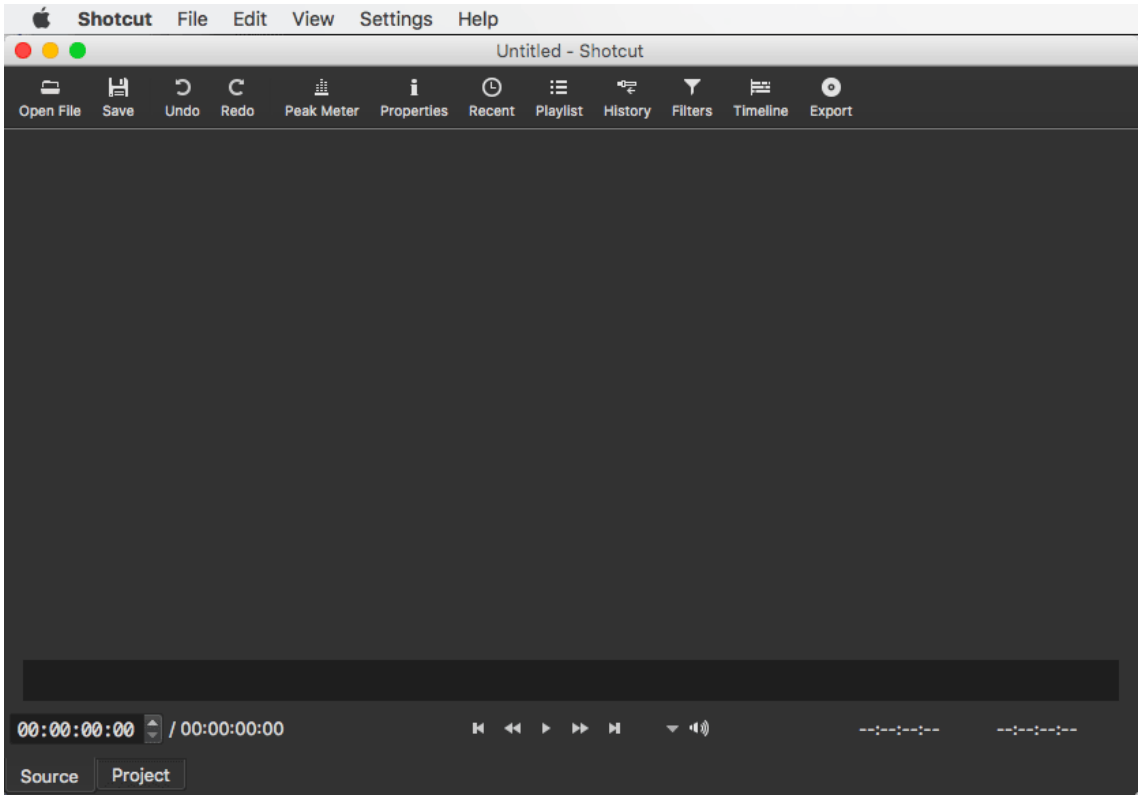
(cont'd next page)



D. The Security % Privacy preferences will appear, saying near the bottom that **Shotcut** was blocked from opening since it was from an unidentified developer. Click Open Anyway.



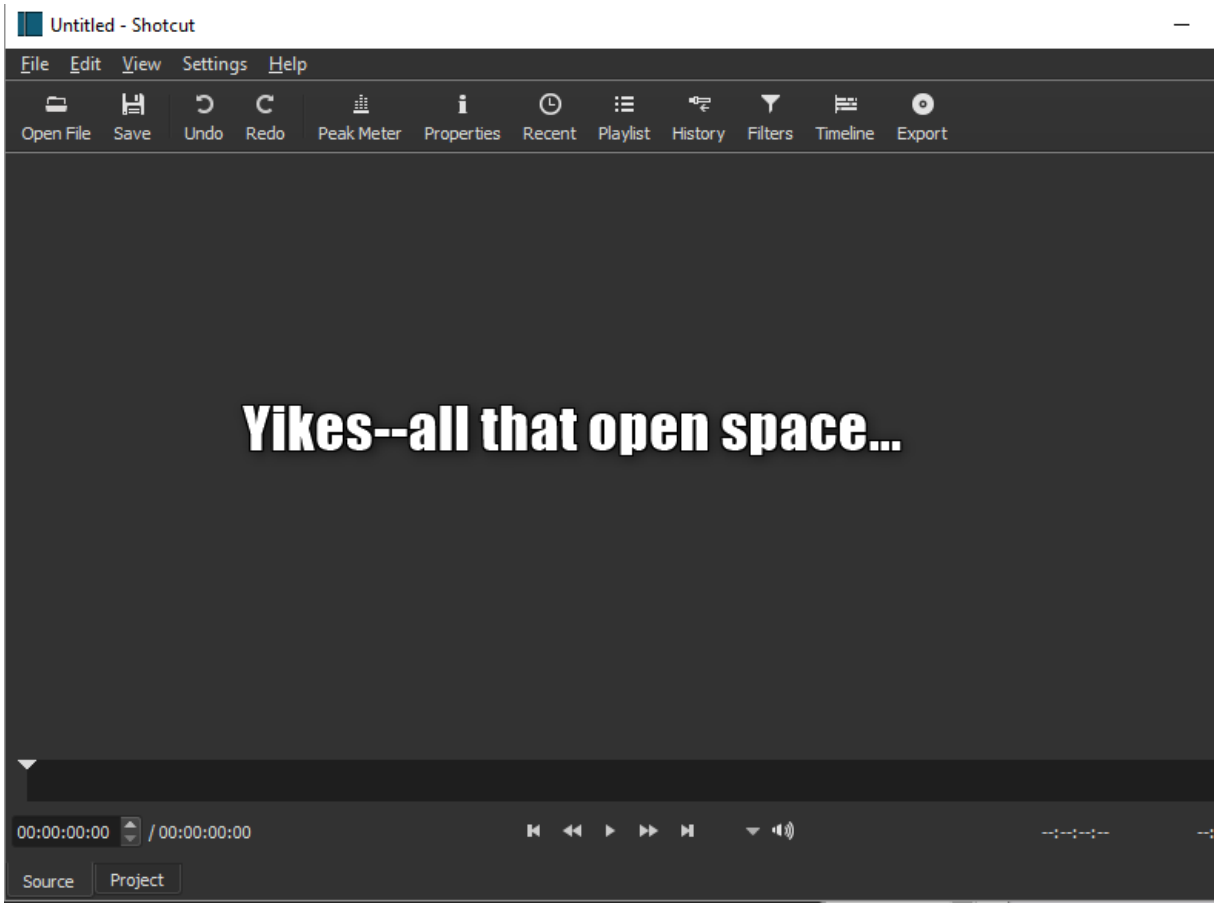
11. **Shotcut** should now open. If it doesn't, trying opening **Shotcut** again. When it does open, its interface will look something like this.



Congratulations. You are now ready to use **Shotcut**.

Getting Oriented: The Shotcut UI and Panels

When you open **Shotcut** for the first time, you may be surprised at how little you actually see by way of interface or controls. Unlike the Microsoft or Adobe UI model, the **Shotcut** team has a preference for beginning with a *very* uncluttered interface, and letting you choose panels from there as you need them. This also means that **Shotcut** opens very quickly, compared to most editing and authoring tools of any complexity, and it consumes a minimum of system resources until it needs them.



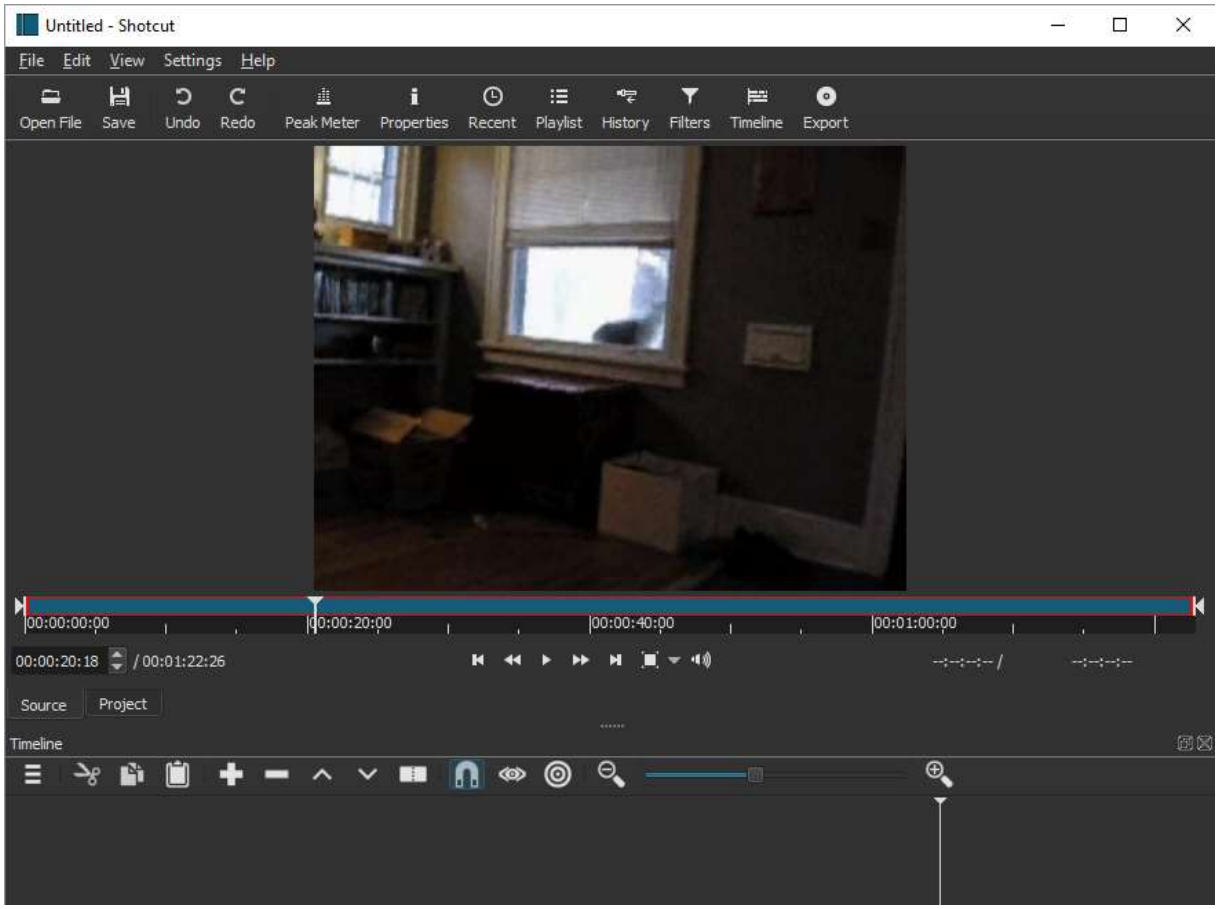
Don't be put off by all the blank space; the Toolbar icons at the top will let you open all of the windows/panels you need, as you need them. Let's take a look at each of these panels in turn, taking them from left to right as their icons appear in the Toolbar above. All of these panels, and how they work together, will be featured again in the 'How to' chapters later in this manual; the following is just for general orientation to the UI and the typical workflow when using **Shotcut**. (On the other hand, as it does walk you step-by-step through a typical workflow, for some readers the present chapter might be all the orientation you need to get started on your own projects. Read it and see. At a minimum, this orientation should allow you to more easily pick and choose among the 'how-to' chapters that follow for whatever most interests you.)

Open File

This control opens a standard dialogue allowing you to browse to and choose a file to open. What happens next depends on what kind of file you open. As with most media editors, **Shotcut** will save a Project in progress in its own proprietary Project format, in this case a .mlt file (which will show up in your file explorer with the **Shotcut** icon, once

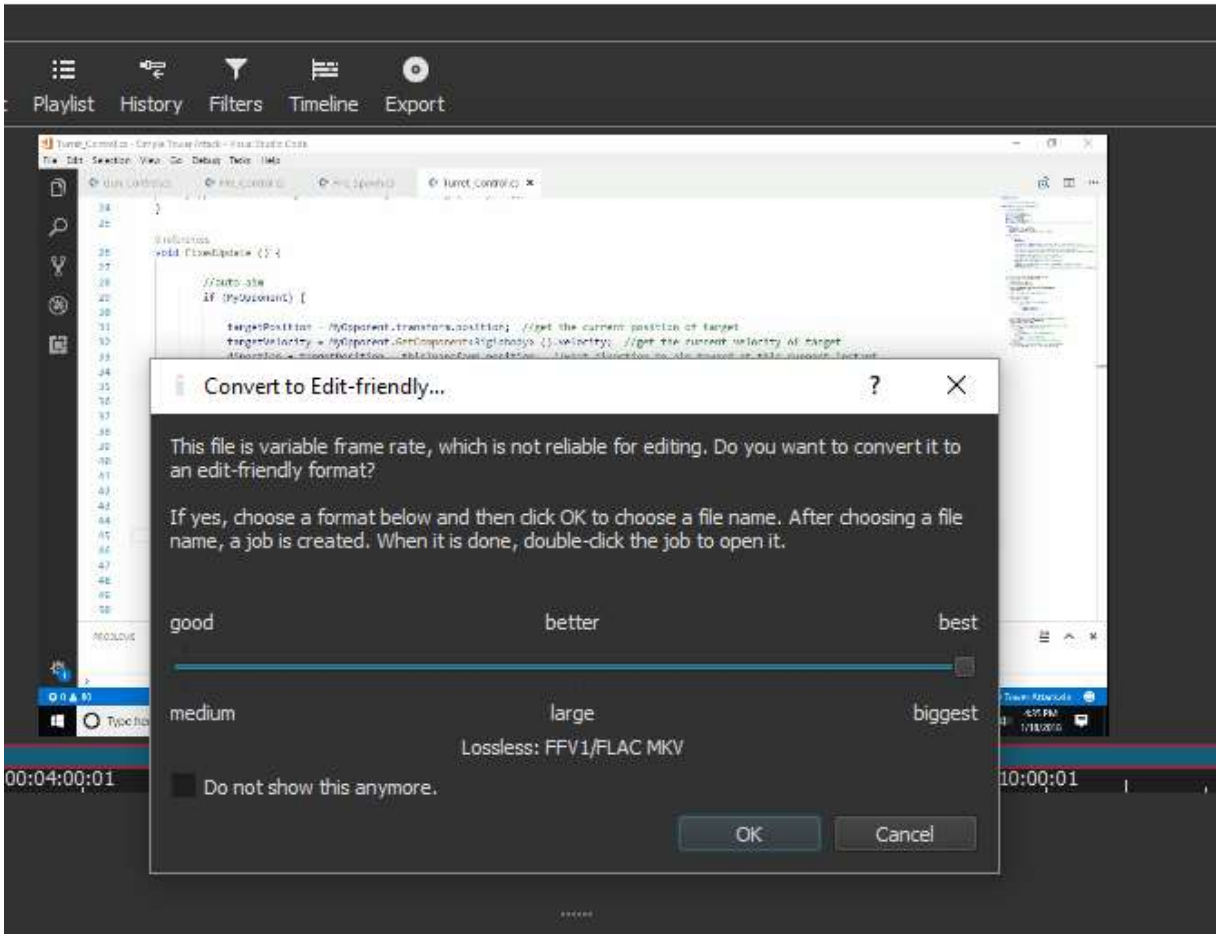
you've saved your first **Shotcut** project). When you open a **Shotcut Project file** of this kind, the **Shotcut** UI will appear just as you last left it for that project, with all imported media assets and all relevant panels showing -- at a minimum probably the Playlist, the Timeline where you've been manipulating the project elements, and the main Source window.

If you choose instead to open a **standard video file**, such as .mov or .wmv or mp4, this file will appear and begin playing in the main Source window. Below I've just opened an old, low-resolution .wmv file featuring my cat outside the livingroom window (did I not promise cats?). This file has automatically loaded in the Source window and begun playing:



Notice that the **Timeline** panel at the bottom is now open, too, and displaying its controls (we'll get to them later), but is not yet populated with anything. *We are not yet ready to do any editing with this video*; at this stage **Shotcut** is just playing it for us to evaluate as a potential 'source'. To edit it, we would need to add it to the Playlist, and from there add it to the Timeline. More on that workflow below.

A final note here: when you select a video file to open, **Shotcut** determines whether that file is in a sufficiently optimal format for editing, and if not, it will open a dialogue offering to convert it to a more useable format. My old .wmv cat video above was already sufficiently edit-friendly. Below, I've just chosen to open a fairly lengthy mp4 screen-capture video, and **Shotcut** opens it in the Source window but also, in front of that, shows a dialog offererring to make this material more edit-friendly:



As you will often find in **Shotcut**, this dialog offers a lot of information that you may understand if you are highly conversant in multimedia formats, but you don't *need* to understand at this point: by default, the software has chosen a format that will allow high-resolution, detailed editing: a particular kind of .mov file. As it states, hitting ok here will prompt you to give that file a name and saving location, then it will send this conversion to a 'Job', at the same time opening the **Jobs** panel so you can see the progess of the conversion (which can be seconds or minutes, depending on the length of the original file). Windows Movie Maker actually does the same thing, for file formats its decides are not edit-friendly enough, but offers you no such options as **Shotcut** does here. When the job has finished, you double click to open it instead in the main Source window. Note that this conversion process will never *overwrite* your original file, even if you gave it the same name and location, since the format will be different.

Save

As with many editing and authoring tools, to Save in **Shotcut** is not the same thing as to Export (which will be at the far right of the Toolbar, and the end of this overview below). Saving here means saving your currently-open **Shotcut project**, in its entirety, with whatever assets are currently included and whatever panels are open. You will be prompted for a location and project name. The resulting .mlt file can only be opened in **Shotcut** again, not in other software. Any sort of complex editing project is probably not going to be completed in a single session, so this is a very useful feature. But even if you think you *are* finished with your edited video, we would always recommend saving (or re-saving) the project itself so that you can come back and make further changes if necessary.

Undo

As with most editors, Undo simply undoes the last thing (or things) you have done, other than opening, closing or exporting a file itself. Bear in mind that you can only 'do things' -- make edits, apply filters or alter properties--with files (or "clips", as we'll see later) that have been added to the Playlist, or from there to the Timeline. But within that scope, you can 'undo' a potentially infinite series of changes you've made, within the current session.

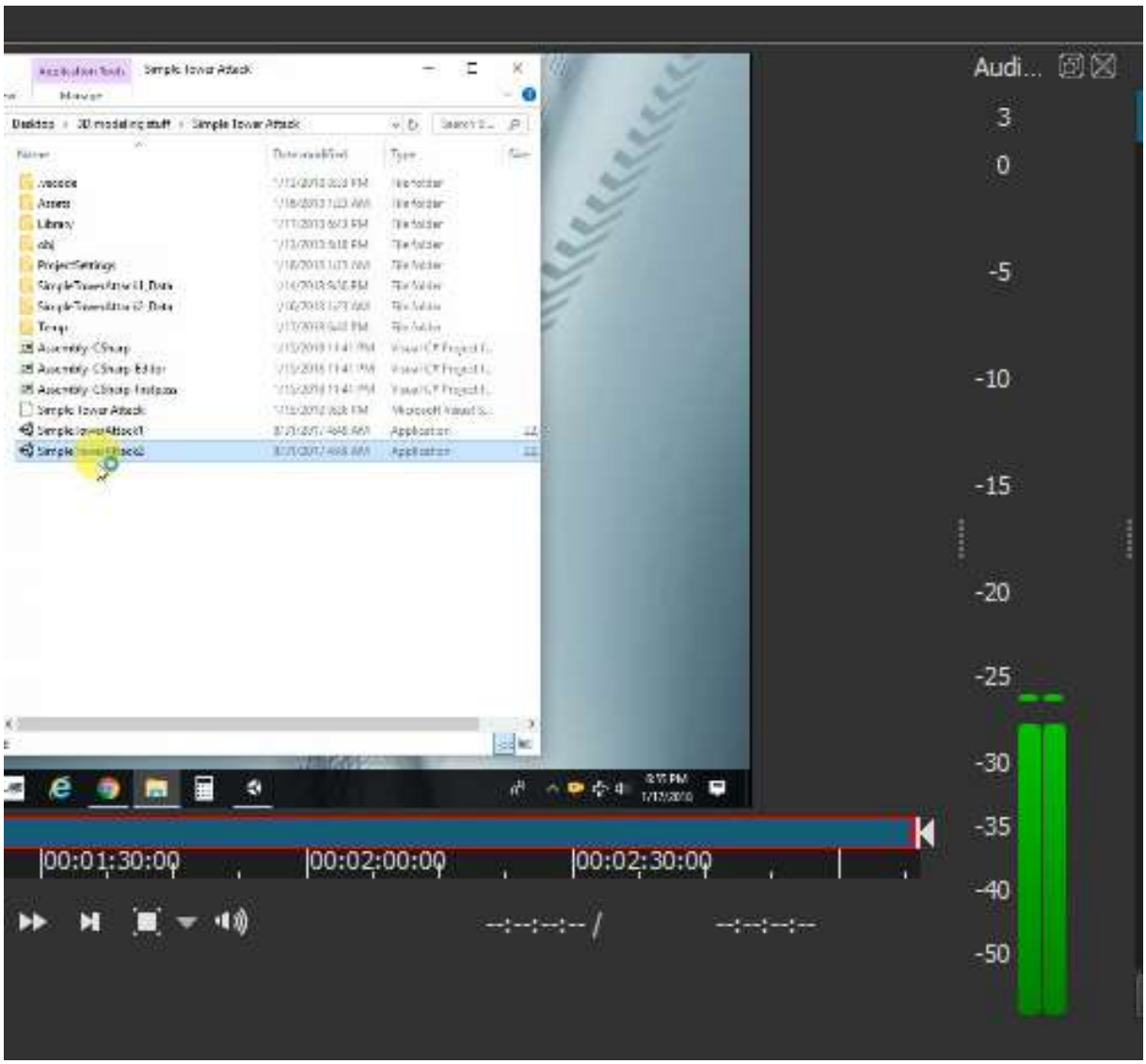
Redo

Again, the standard editor control that lets you redo an action you've just undone, either by accident or after some reconsideration. And again, in **Shotcut** there is a potentially infinite stack of actions you can undo and then redo, within a given session.

Peak Meter

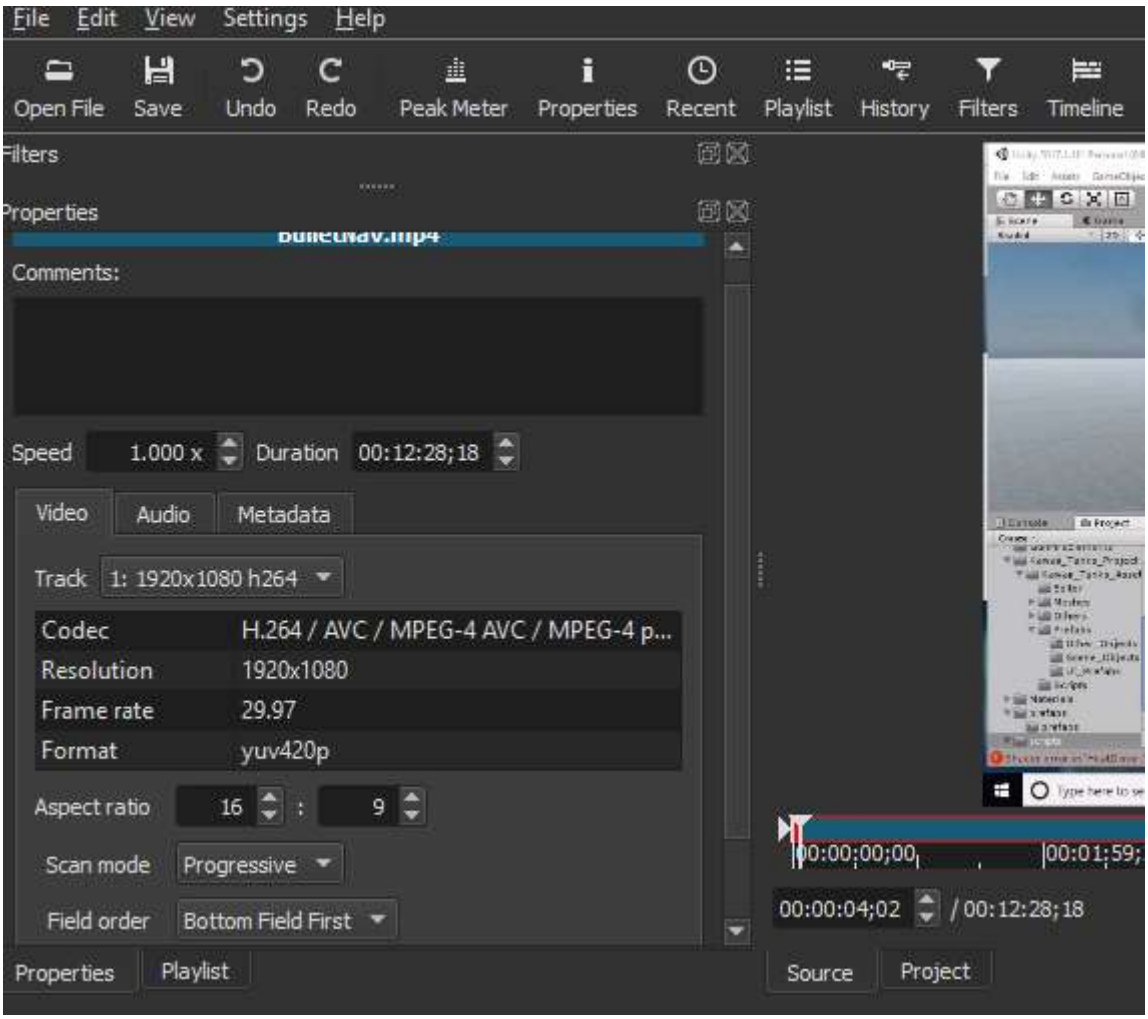
This button opens the Peak Meter (see the vertical green bars and decibel markers on the right side of screenshot below), which indicates the current signal (volume) levels of whatever audio is playing, whether in the main Source panel (as in this screenshot) or the selected track or tracks in the Timeline. This is a separate panel, and like all **Shotcut** panels it can be closed (with the little X button in the upper right), or dragged to be a free-standing window, or redocked elsewhere in the UI. While **Shotcut** is not a full-featured audio editor on the level of Audacity, it does allow volume alteration, as well as multiple audio track mixing, fade-in/out, and the application of reverb and some other effects. One can even record a voiceover narration directly through **Shotcut**, although the method for this is a bit convoluted at present. It is therefore extremely useful to have a peak meter, with L and R stereo channels, to monitor the audio signal intensity of the material you're working on.

(cont’d next page)



Properties

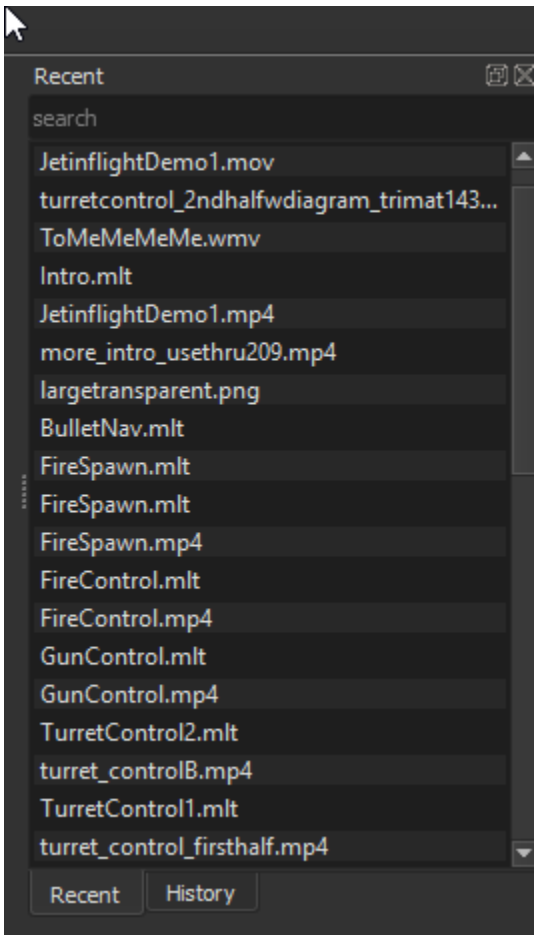
This Toolbar button opens the **Properties panel**, which allows for the inspection, and in some cases the alteration, of a wide range of properties pertaining to the selected asset. For this panel to be populated, of course, you need to have some asset selected: either as an item in the Playlist, as a track in the Timeline, or, as in the case below, as a file that has been opened in the main Source window but not yet added to the Timeline. Here I am displaying the Properties for an mp4 video tutorial segment called "Bullet_Nav.mp4" which I've just opened and am simply displaying in the Source window. I'm showing a bit of the Source window here as well as the whole Properties panel; you can tell where the panel ends by the Close control and the scrollbar:



As you can see here, the Properties panel displays both basic and more in-depth information about this video file: the duration, Codec used for encoding, screen resolution, frame rate, aspect ratio and more (including some Track information because I previously edited this file as a track in **Shotcut**). Note that here I am only displaying the 'Video' tab of the Properties panel; if I switch to the 'Audio' tab I will get information on that file's audio track (if it has one): audio codec, number of channels, sample rate, etc. It's important to stress that, at this point, all of this information is read-only, because I have not yet added this file from the Source to the Playlist, a prerequisite for editing any file or asset in **Shotcut**. Once I do so, a number of these property fields become editable. Some changes you might want to make here, using this panel's controls, will be explored in the "How to" chapters that follow.

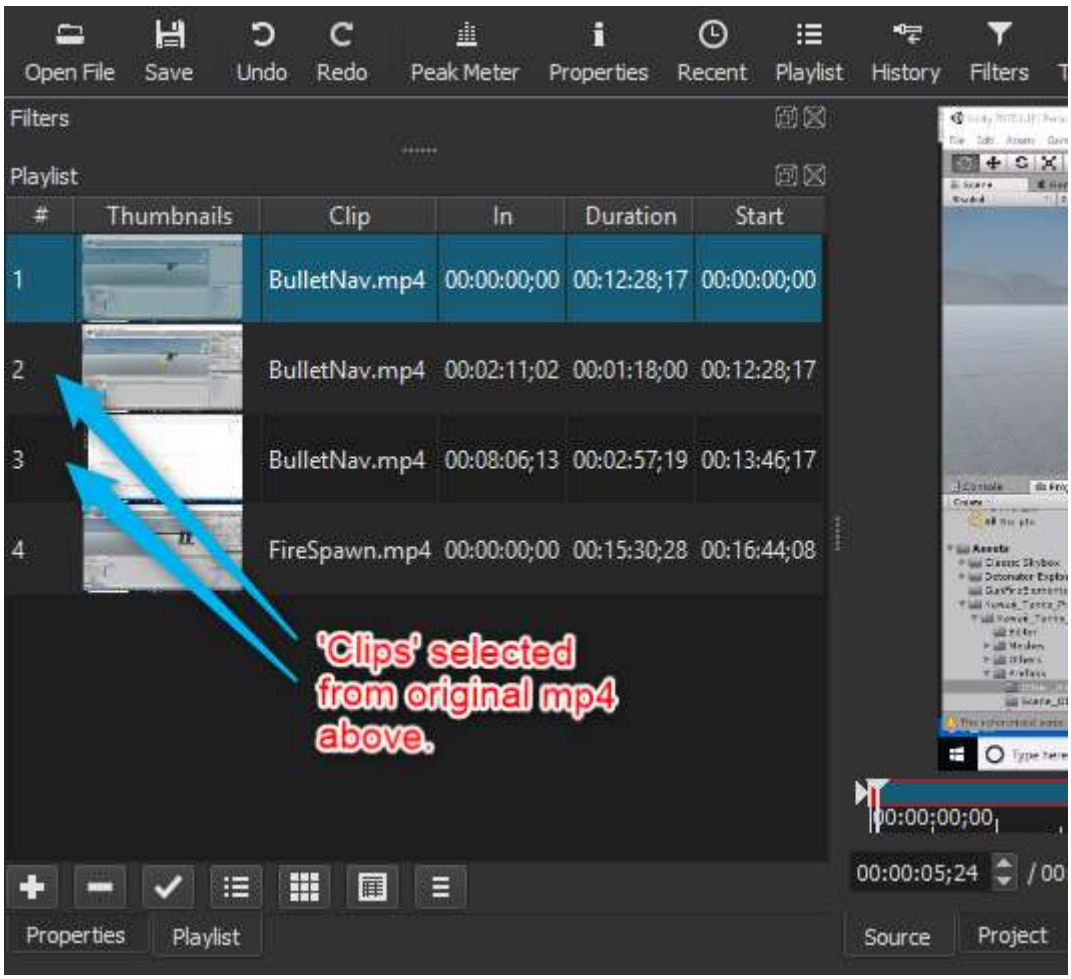
Recent

This toolbar button opens the **Recent Panel**, which is simply a clickable list of all the files (standard format files as well as **Shotcut** project files) you have recently opened and edited in **Shotcut**. My current list (below) has 45 items in total---note the scrollbar--and probably represents all files I have manipulated in this version of **Shotcut** to date. This panel corresponds to the **Open > Recent...** feature found in many File menus:

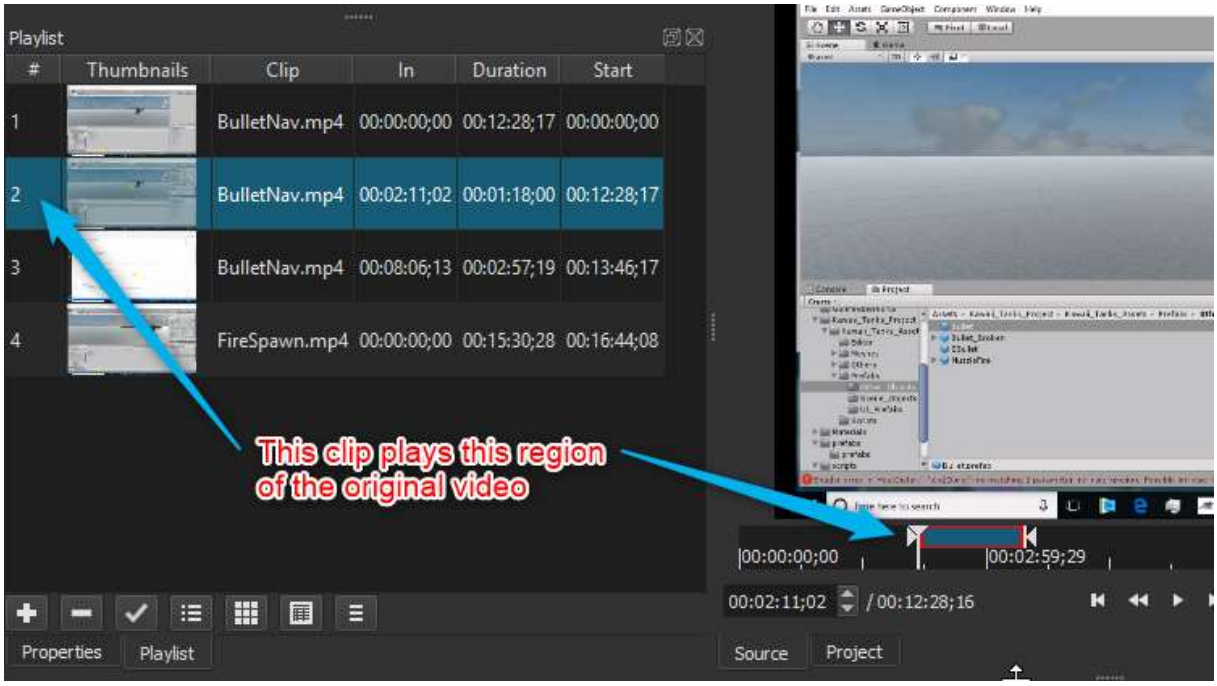


Playlist

One of the most important panels in **Shotcut**, along with the Timeline, **Playlist** is essentially the clickable list of all assets that have been added to the project for manipulation, with this manipulation happening largely on the Timeline panel though also through changing some Properties (see above) and finally, through adding **Filters** (see below) directly to Playlist items. Let's consider this annotated screenshot of a Playlist panel:



There are four items in this Playlist, with the first one selected (and thus highlighted in blue). That item is the mp4 file we saw above, now added to the Playlist by way of the "**+**" button on the lower right, which adds any item displayed in Source to the Playlist. The *fourth* item in the playlist is an additional, newly-added mp4 file, named "FireSpawn". The two *middle* items, as annotated, are "Clips" I have taken from the first mp4: they are specific segments of that video, which I defined by dragging the left start-point (visible here, lower right) and the right end-point (you can see both start and end drag-points in the first video I opened above) to indicate exactly when I want a clip to begin and end. If I were to double-click the first of these clips, I would see the following:



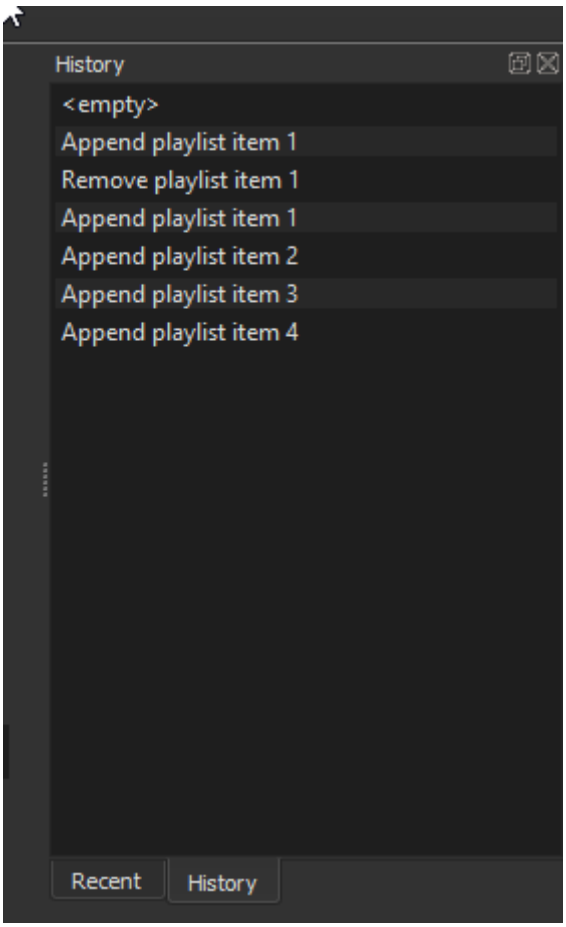
As indicated, this first clip represents only a brief segment or region of the overall BulletNav.mp4 video: the Source now shows me exactly what segment I chose for the clip, and I am able to change the start and end points at this stage as well (or at any later point, as long as that clip has not been deleted from the Playlist).

Here we can see a major reason why video files opened in **Shotcut** are played in the Source window but *not* automatically added to the Playlist, as one might expect: the assumption is that many video files opened here will be 'raw' footage, which we will want to review and then decide exactly what, if any, of that footage we actually want to import, maniplate and ultimately save as some new file. We might want to use most of the raw video but not the first and/or final seconds; we might just want to use one or more short segments and discard the rest (though the original 'raw' file will still be preserved, unless you try very hard to overwrite it). Once we move a clip to the Timeline, we can make further, more fine-grained choices about what we want to leave in and edit out in that work-space, but defining one or more Clips from the Source is our first opportunity to do this, and again can save a lot of CPU by not forcing **Shotcut** to load an entire raw video file into fine-resolution editing memory. Working with Clips will be the subject of a later 'How-To' Chapter.

Note finally in both of the last screenshots that beneath the Playlist panel we can see both 'Properties' and 'Playlist' tabs: these now allow us to toggle back and forth between those two panels in the same UI location. Wherever possible, **Shotcut** tries to maximize efficient use of screen space in this manner, while still preserving easy access to panels we've opened.

History

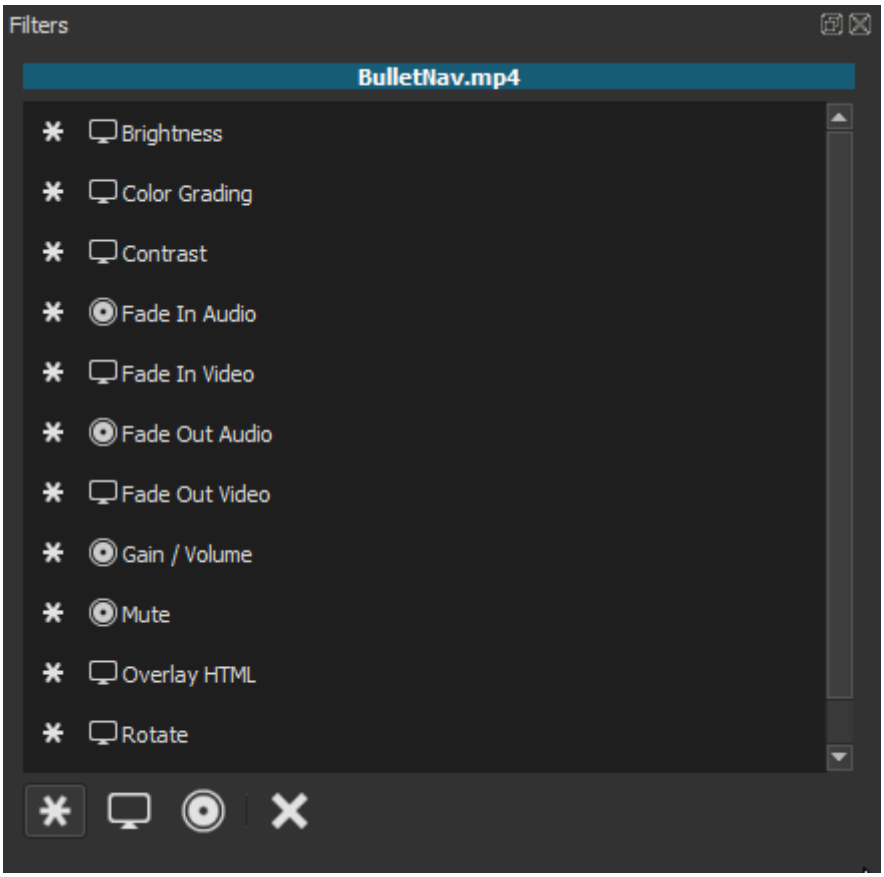
We noted above with **Undo/Redo** that **Shotcut** allows you to undo (and redo) a potentially endless stack of actions within the current session. The History Panel gives you direct access to that stack. In the current session I have simply added certain items to the Playlist, but any Filters added to a Playlist item, or edits made in a track in timeline, would be listed here as well.



Note, again, that in opening this History panel, **Shotcut** has preserved the earlier "Recent" panel (which I did not close) in the same location, so again we can toggle back and forth between them with the tabs at the bottom.

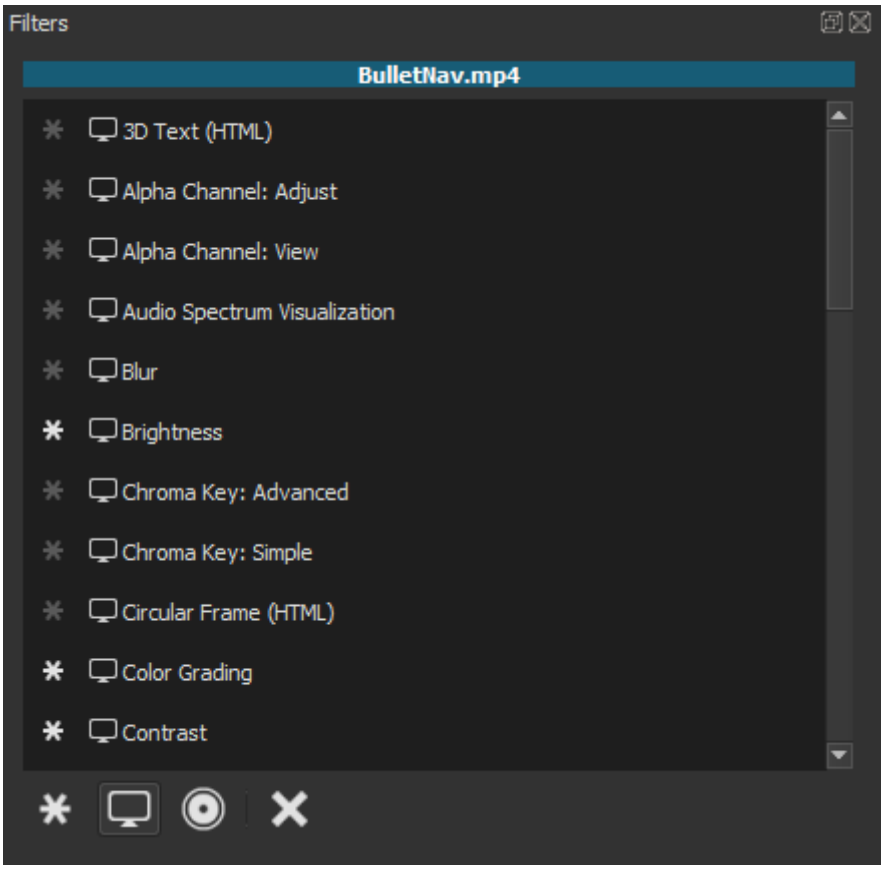
Filters

The Filters Panel is another important control area in **Shotcut**, but one with a complex enough workflow that we will not try to address it comprehensively in this overview. Essentially, Filters represent a very wide collection of effects that we can apply to items we've selected in the Playlist (or that we've selected on the Timeline, as these are automatically part of the Playlist too). Just to give you some sense, here is a list of 'favorite' filters (as defined by the **Shotcut** user community) that we might apply to the current item selected from the Playlist:

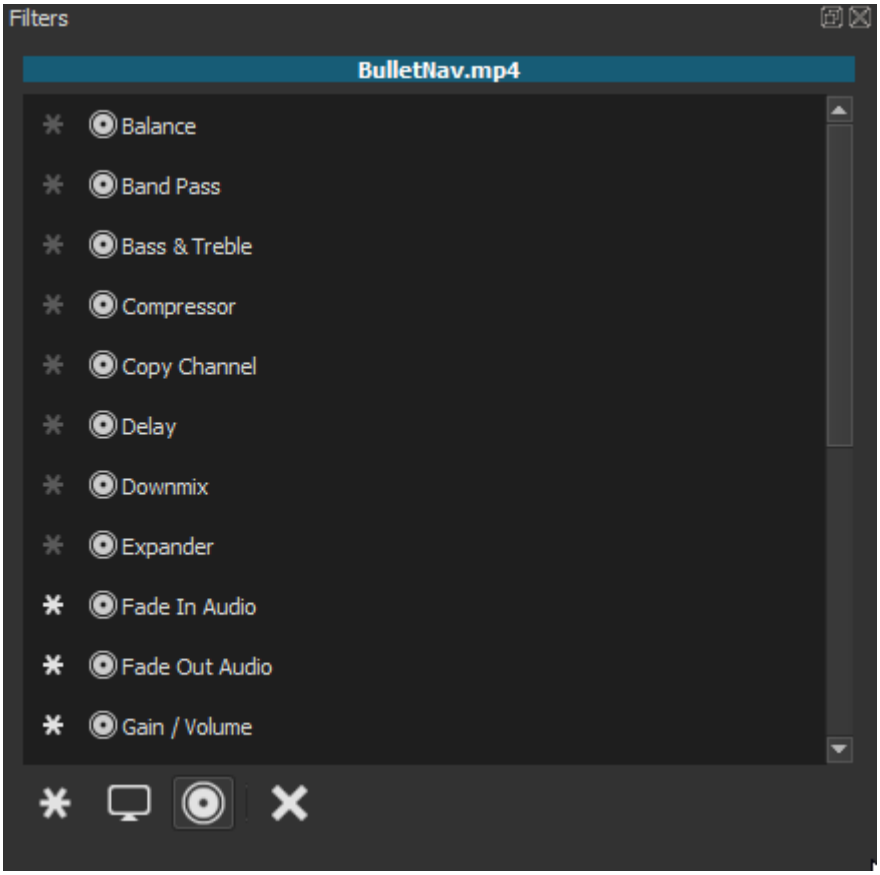


As we can see, this list includes effects relating to enhancement of the video qualities (brightness, contrast, etc), modification of the audio volume, etc, fading in and out of both video and audio segments, along with one example of an extremely useful set of filters: text and html overlays, which you can use to add overlay text captions, static images and even inset videos to your video segment. Among these latter, text overlays are probably the most useful for introductory video-making purposes, and we will cover these in a how-to chapter. The 'favorites' list of filters is opened by default, or by the "*" button at the bottom of the panel here; the next button to its left opens a much longer list of filters for video (or "screen") effects, and the third one over a list of audio effects. Here are screenshots showing just partial listings of each. First are some **visual** filters--just through the C's in an alphabetical list:

(cont'd next page)



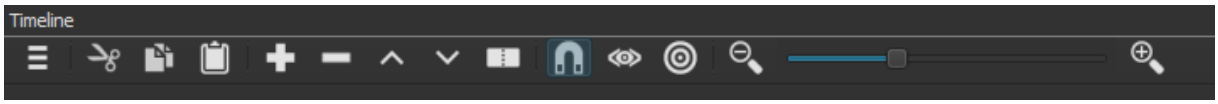
Now some **audio** filters:



As you can see, the list of available filters is considerable, and **Shotcut** being open-source software, this list is constantly growing, and greatly expandable though add-ins. Clicking on any of the filters in these lists will (1) apply it to the selected item (indicated in blue at the top of the panel), and (2) open a dialog with controls specific to that type of filter. One can always chose subsequently to edit or delete this filter from the item. The only real constraint imposed by this way of applying effects to video/audio tracks is that the filter must be applied to an *entire segment or clip*, rather than coming in and out at arbitary points on a continuous track. But as we'll see, the way around this is simply to define separate segments where appropriate, through 'cuts' made in the Timeline video or audio track or through 'Clips', defined from the source file. The general topic of Filters is vast, but hopefully this is at least enough to prompt you to explore the relevant How-To chapters.

Timeline

Arguably the single most important panel in the **Shotcut** UI, and where you will likely spend the most working time, the Timeline panel is placed near the end of the Toolbar button sequence because that is its logical place in the workflow of editing or crafting a video. As we've seen, a video file must first be opened as a (potential) **Source**, then added to the **Playlist**--which might more appropriately be considered the "library" for this project, rather than a 'playlist' in the usual sense, because it is not meant to imply a linear sequencing of its elements. Linear sequencing, as well as synchronization and mixing of different tracks, is instead the job of the Timeline. When opened, the Timeline panel displays its own array of button controls:



Taken from left to right, these button controls represent the following actions:

(cont’d next page)

Menu -- Displays a menu of additional actions, including adding video and audio tracks and adjusting their visual presentation. In the **Shotcut** Timeline, video tracks can contain audio as well, and will display the audio wave-form, but audio tracks can only contain audio wave-forms.

Cut -- cuts the currently-selected video or audio track (or track segment) and 'ripple-deletes' it, meaning that any additional track segment to the right of the cut segment shifts to the left, leaving no gap in the timeline, while the cut segment itself is copied to the clipboard

Copy -- copies the currently-selected track or track segment to the clipboard, but *without* deleting it from its current position (analogous to the difference between Cut and Copy in word processors)

Paste -- inserts the copied material on the clipboard at the point of the cursor, here also called the "playhead", in the currently-selected track (if there is more than one track present), and shifts any following track segments to the right. Copied audio and video material can be pasted either at another point in the track it was cut/copied from, or into a *different* track at any location where the second track is empty.

Append -- Appends the currently-selected Playlist item, or copied track segment, to the *end* of the current track, if there is already existing content in that track. Also how you insert Playlist items into the Timeline (if no track currently exists, this function will create one and insert the Playlist clip starting at point 0).

Ripple-delete -- Deletes the currently-selected track or track segment *without* copying it to the clipboard, and shifts any further segment(s) of this track to the left

Lift --- cuts (ie, removes and copies to clipboard) the currently-selected track segment without shifting the position of any other segments in this track. Unlike cut or ripple-delete, this operation leaves a gap in the current track, but keeps it synchronized with any other tracks in the project.

Overwrite -- pastes the copied material at the point of the cursor in the currently-selected track, overwriting any material that is currently in that location and, like Lift, preserving the overall position of the track in relation to other tracks.

Split at Playhead -- creates a single cut or split at the current point of the cursor (which will also be the Playhead location). The first split divides a track into two segments; additional splits create additional segments. These segments can then be cut, copied, pasted, etc as described above, and as they now have separate properties, can also have **filters** applied to them.

Toggle Snapping -- turns 'snapping' on/off. Snapping automatically creates a tight join between two track segments you are manualy sliding together, rather than potentially leaving a small gap in the track, or overlapping the two segments which creates a cross-fade or dissolve in the video or audio content.

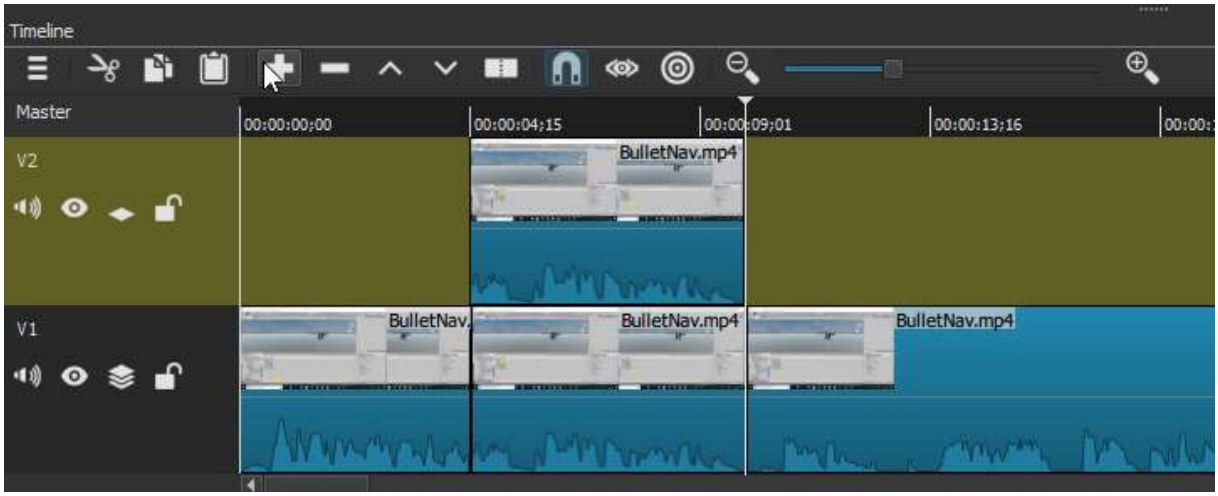
Scrub while dragging -- allows you to drag the play-head across audio & video content in a track and hear/see that content precisely where you are dragging. The playhead will play at normal speed when you hit the play control in the Source view, or hit the spacebar once, but scrubbing allows for more precise selection of points to edit in the video and/or audio. You can also 'step' the playhead forward and backward, in small increments, using the arrow keys on your keyboard.

Ripple Trim and Drop -- The "drop" refers to when you drag from the Playlist or Source player. If Ripple is On, then dropping a clip from either of those locations will insert the clip at the drop point instead of overwriting[1].

Zoom Timeline Out -- horizontally zooms out the Timeline (with all of its tracks), allowing you to see more granular detail, particularly in the audio wave-form or during cross-fades or other visual effects.

Zoom Timeline In -- horizontally zooms the Timeline in, allowing you to see larger segments (or the entirety) of a long Timeline; this is especially useful if you have a complex edit structure with multiple tracks, segments and filters. The slider between these two button provides analog control for zooming in and out.

As should already be clear, the range of possibilities for manipulating material in the Timeline is very wide, and cannot be more than suggested in this overview. Below is a partial view of an example Timeline (the first 20 seconds or so) containing two video tracks, with a few operations performed:



The bottom track labeled "V1" is a video track that was automatically created when I selected my "BulletNav.mp4" item in the Playlist and hit the **Append** button (**+**, which might simply be considered "Add to Timeline" for this purpose). The V1 track includes both the video content of the mp4 clip and also the audio, visible here in its wave-form. V2 is a separate video track I created manually via the drop-down menu ("Add Video Track"). I have used **Split at Playhead** to create two cuts in the first (V1) track, as shown, used my mouse to select the resulting segment between these cuts, **Copied** that segment, and **Pasted** it directly above, in the second video track. This track is brighter to indicate that it is the currently-selected track (otherwise I couldn't have pasted into it). Since these two parallel segments are exactly synchronized, if played at this point the resulting audio-video output would be exactly the same as if I were simply playing the

[1] https://forum.shotcut.org/t/ripple-trim-and-drop-option/291

original V1 track. This would still be true if I were now to **Lift** the original segment in V1, leaving a gap there, since **Shotcut** automatically composites or mixes the two tracks together on playback (as long as both tracks are made visible, with the little eye control button at the left of each track).

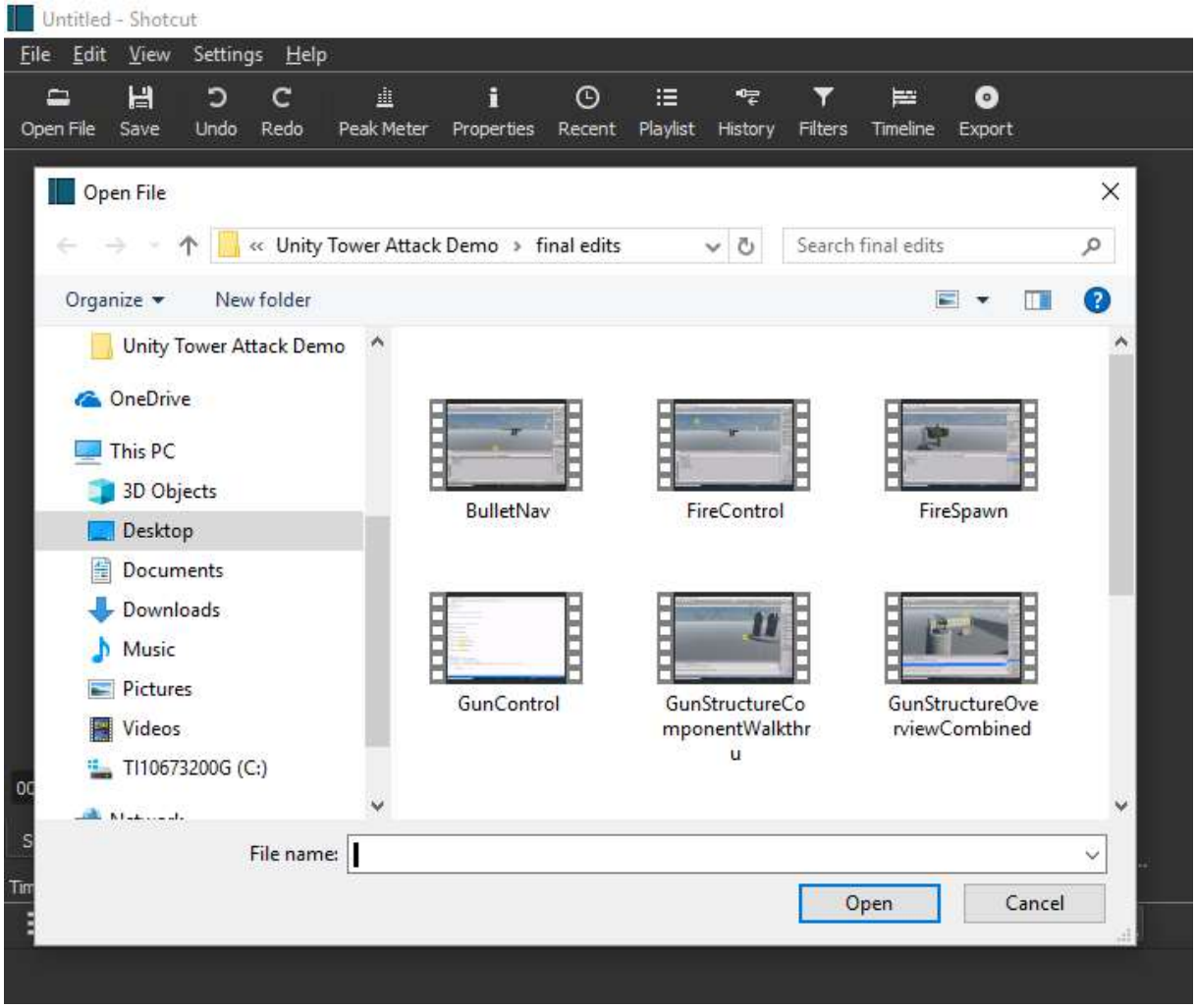
As noted previously, an important application of an operation like this would be to apply one or more **Filters** to the newly-separate V2 track segment, for instance if I wanted to add a text overlay to it, in which case I would probably also want to Fade-In and Fade-Out the upper segment with the text filter applied. But I could alternately have pasted a clip from an entirely different video into the gap in V1, or into the track above it, if I wanted to intercut between those two video contents (in which case I'd probably want to arrange a brief dissolve or cross-fade transition between them, at both transition points). I could even paste in a still-image asset, such as a JPEG file, and stretch it appropriately to the required duration. These and similar operations will be detailed in the How-To chapters that follow.

Export

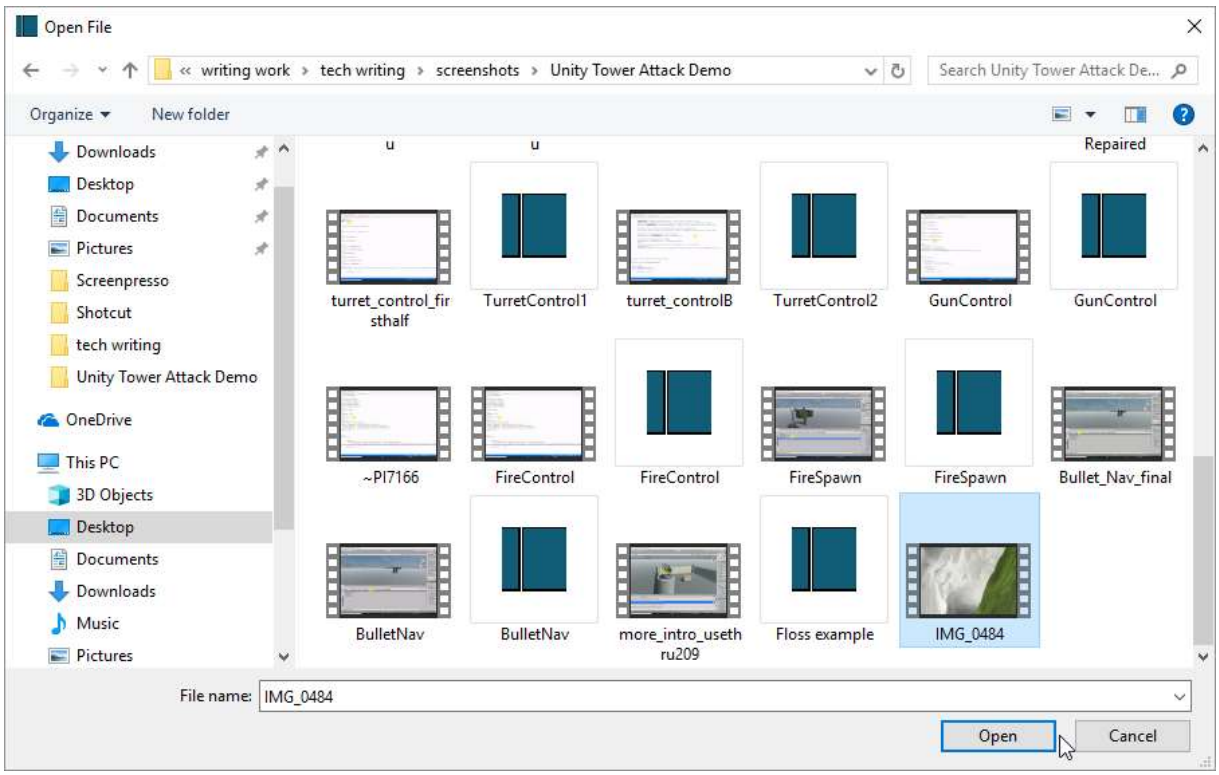
Export is the final control button on the main Toolbar, and the final logical step in the **Shotcut** workflow. As we have seen, **Save** will simply save the entire project, in its current state, in the proprietary **Shotcut** Project File format. If we want (as we certainly will) to render our edited creation in a general-purpose video format playable on other devices, we will need to Export at this point. The Export button opens a panel that allows us to choose among a very wide variety of video file formats, and then to further specify a range of properties or options within that format. Once we've made these choices--or simply accepted **Shotcut**'s defaults--we can then choose a saving location and file name, and the file will then be rendered (a process which can again take seconds or minutes, depending on the length and complexity of the project, and which will be assigned as a job in the Jobs Panel). Since this procedure can involve a great number of options and possibilities, we will defer any more detailed discussion to the chapter devoted to Exporting a File.

Getting Started: Opening a file and adding it to the Playlist and Timeline

In **Shotcut**, we can open a file either by using the **Open File** button on the control bar, or via the top dropdown menu by choosing "File" > "Open File..." Either will invoke your standard file-browser window, allowing you to navigate to and select a file to open:

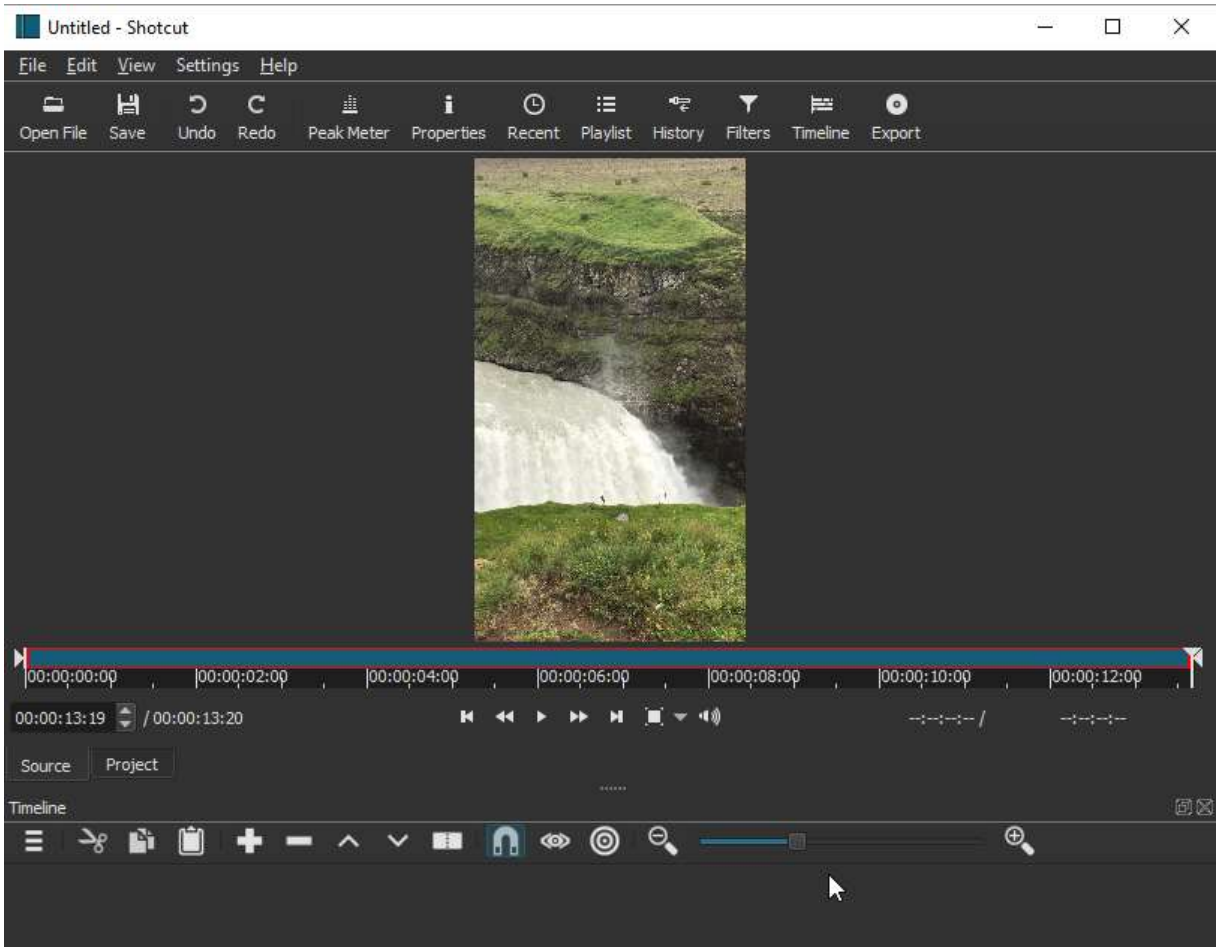


Here, the file browser has taken me by default to a location where I have previously been saving video tutorial videos I've edited in **Shotcut**. However, for this demonstration let's choose something different: a brief clip of 'raw' iPhone footage of a waterfall in Iceland. I've saved it to a folder one level up from the first one:



The video in question is simply named "IMG_0484", and is clearly marked by its icon as a video image--in this case a .mov file, though many of the adjacent files are mp4 video files. Notice, however, that there are also a number of files here marked with the distinctive blue **Shotcut** logo. These are not video files as such but **Shotcut project files** I have previously saved from working on my video-tutorial series, and which I could also choose to open this same way. If I did so, however, they would open an entire project, with the **Shotcut** UI displaying all of the media assets and video and audio tracks I used in that project, exactly as I last left them. We'll save that for another time, and instead open the 'raw' waterfall footage:

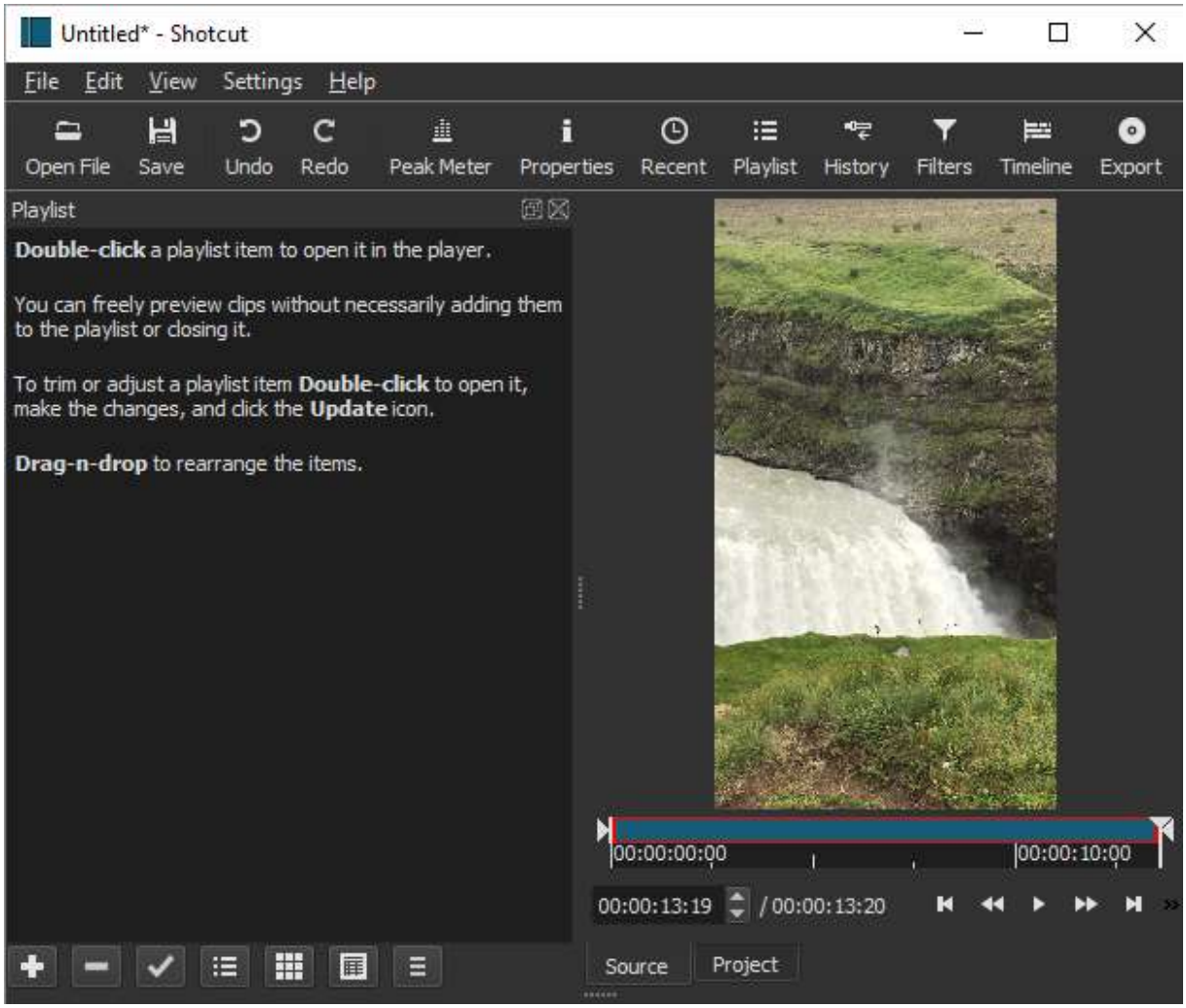
(cont'd next page)



As soon as I open the waterfall video it immediately displays in the central panel of the **Shotcut** UI (called the **Source** panel) and begins playing; in the screen shot it has played to the end, and you can see that the full video is only a bit over 12 seconds duration. At the bottom of the Source window are some familiar video controls, which we can use to rewind the video, play again (or pause), fast forward, let me zoom the view in or out to various percentages between 25% and 200%, or show the Volume Control (which by default will be set to whatever the full volume of the audio is, if audio is present). One thing immediately clear here is the tall, narrow 'aspect ratio' of this video, which was shot with an iPhone in vertical or 'portrait' orientation: if I were to make this part of a longer video sequence I might want to add a **crop filter** to this clip so that it fits in better with other clips that have more of a 'landscape' orientation.

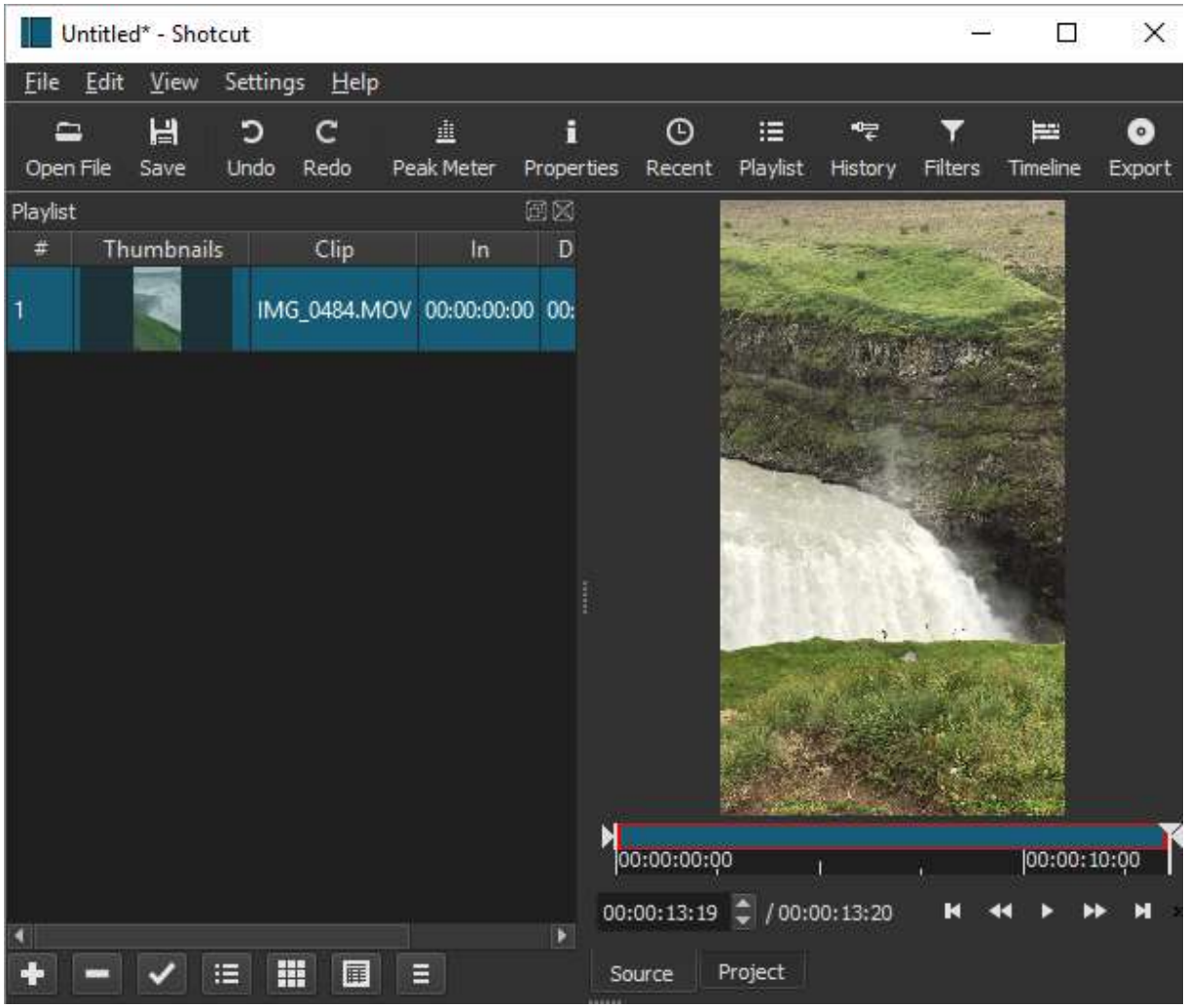
However, *I cannot do any such editing operation yet*; so far, this video is merely a "source" that I can evaluate for inclusion in my project (in whole or in part, as we'll see in the later chapter on "Clips"). To make this file an actual editable asset, I need to add it to my **Playlist**. I will do that by first opening the Playlist panel, by clicking on the Playlist button in the top Toolbar (you can see that button here, 5th from the right in the toolbar). At this point the Playlist will still be empty:

(cont'd next page)



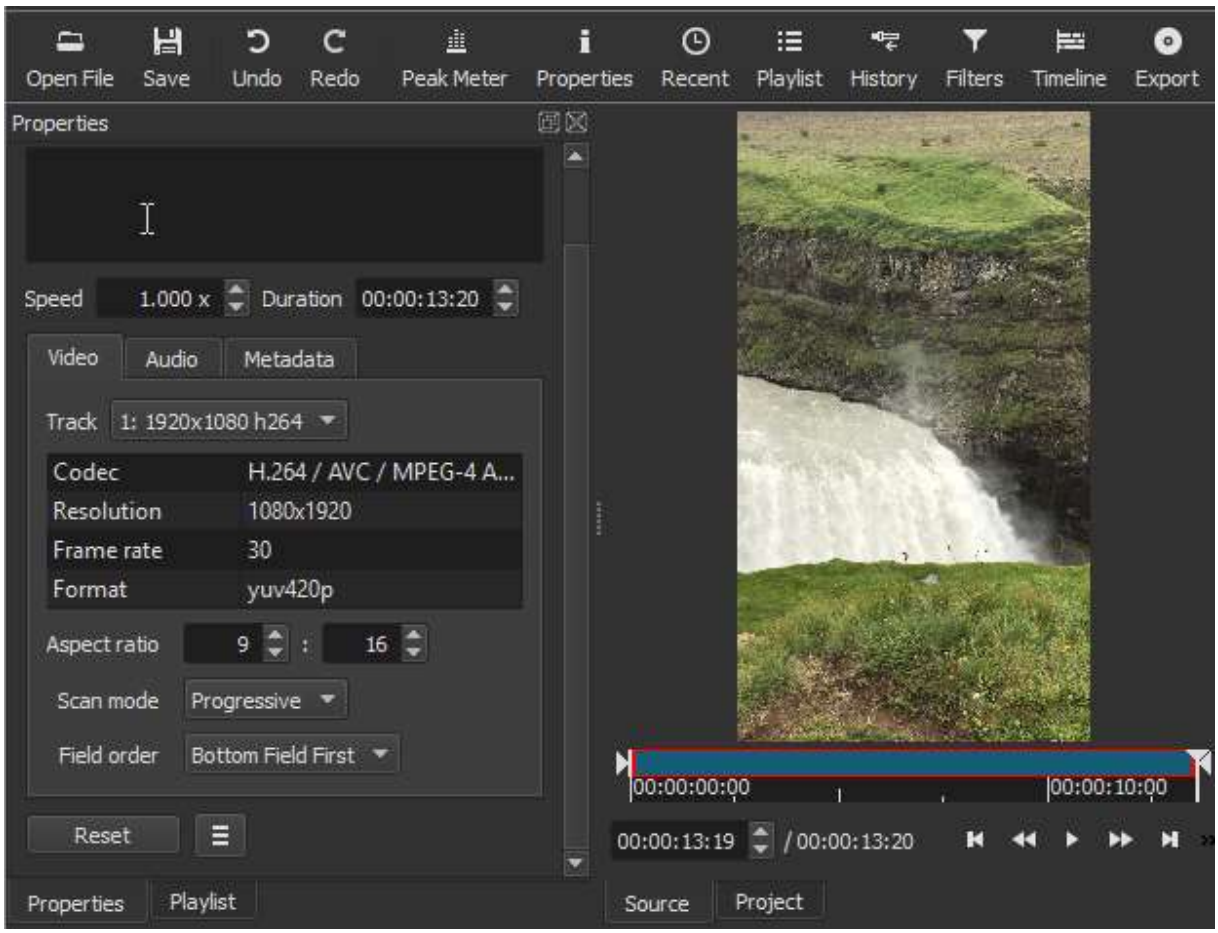
The empty Playlist panel displays a number of useful instructions--but not, interestingly, how to add an item in the first place. We do that by clicking the white **+** button at the bottom left of the panel. This will add whatever is currently displayed in Source into the Playlist, which now displays as follows:

(cont'd next page)



My waterfall .mov clip is now displayed as an item (here the sole item) in the Playlist. **Shotcut** now considers this an asset in the project proper (a project still called "Untitled*", we might notice, because I have not yet saved it as a Project file; we'll probably want to do that before we put too much editing work in). Now that the clip is in the Playlist, we could immediately start modifying it from there in a number of ways. For example, after first clicking on this Playlist item to select it (it will turn blue, as shown, at that point), I can now click the Properties button in the top toolbar to display the **Properties Panel**, loaded with the properties of this waterfall clip:

(cont'd next page)

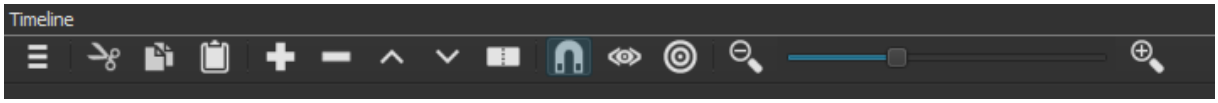


Note that the Properties panel has displaced the Playlist panel itself in this part of the overall UI, but the latter panel is still open, and is accessible by way of the tab now displayed at the bottom of the Properties panel so we can toggle back and forth between the two. Meanwhile, we can see a variety of properties for this video file, some of them fairly familiar--the clip's duration, its resolution in pixels and precise aspect ratio--and others more esoteric: the codec used for encoding this file, the frame rate (30 frames per second, or close to that of cinema-quality film), the format code, the scan mode. And this is only in the "Video" tab of this panel: the "Audio" tab would display various properties of the clip's audio, if it is present (as it is in this file).

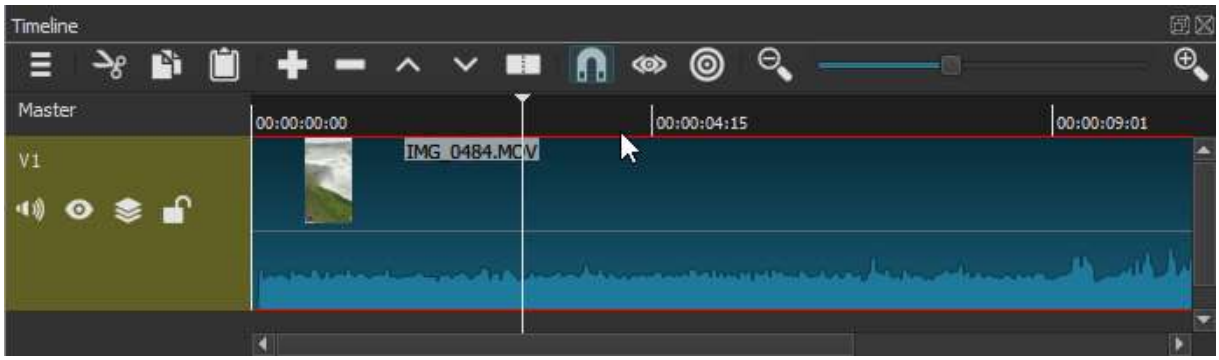
As will be the case at numerous points in your exploration of **Shotcut**, some of the information you're presented with here may mean nothing to you, but *that doesn't matter* as long as you don't fiddle with settings you don't understand: at any point in the workflow where a choice has to be made, **Shotcut** will default to a sensible, middle-of-the-road choice: it is a tool designed for both absolute beginners and users with deep knowledge of video formats, encoding, and other aspects of video editing. Right now, for example, we could already change certain properties of this clip, including the speed of play, the clip's duration and even the aspect ratio itself (though doing it here would mean distorting the actual image -- to achieve the same effect it would probably be wiser to add a crop filter instead). Indeed, with this clip now in the Playlist, we could open the **Filters panel** and begin applying multiple filters to the clip. But that would probably not be wise either, until we're able to step though this clip in much finer detail, and determine exactly where, why and how we might want to apply any of the effects made possible by Filters in **Shotcut**. And to do that, we will need one more step: adding this clip from the Playlist to the **Timeline**.

Typically, the **Timeline panel** will open automatically when you first open a video file, as it did when we opened the waterfall video above. If not, you can open it via the Timeline

button in the top toolbar. The Timeline panel has its own set of controls, which look like this:



These controls are itemized in the UI Orientation chapter preceding this one, in case you skipped it, and we'll discuss them again as we need them for specific operations in subsequent How-to chapters. For now all we need is the big **+** button here (*not* to be confused with the **+** button in the Playlist controls, which added items from Source into Playlist--though if you ever do make a mistake like that, just use the **Undo** button to make it go away). *This* **+** button will add any item selected in the Playlist into the Timeline. If there is already a video track in the timeline, and it's selected, the new video clip will be added to that track (or appended to any content already on it). If there's no track present yet, this control will automatically create one, and add the video clip to it at point 0. That's what happens when adding in our first clip of the project, as here:



What we see here is a video track, "V1", which **Shotcut** has created, with the waterfile clip inserted into it starting at point 0. The audio of this clip is also included in the same track (though we could split it off if we wanted into a separate audio track); we can see it as a wave-form at the bottom of the track. **Shotcut** will now consider this the "Master" track, though we can add as many additional video or audio tracks as we need for the project. Note the scrollbar at the bottom: we are not seeing the entirety of the V1 track in this screenshot, but we're seeing most of it because the full video in this case is only 12 second long. The videos you edit may be much longer, of course, so another very useful set of buttons here are the **zoom out** and **zoom in** controls (far right on the Timeline controls), and the slider between them, which allow you to zoom in horizontally in order to see fine details, or zoom out so as to see more or all of a longer track. But what exactly do we mean by 'seeing' the details? In the case of the audio wave, when we zoom in we can literally see the fine details of the wave-form itself:

These details are not especially varied or interesting in this particular case, since this is simply the roaring sound of the waterfall recorded with the video; but with something like a voiceover or music sound track, one can easily differentiate individual sounds or words in the wave-form shape, and therefore decide exactly where to edit the sound track, if necessary. But what about *visual* details? Here we must understand that this video

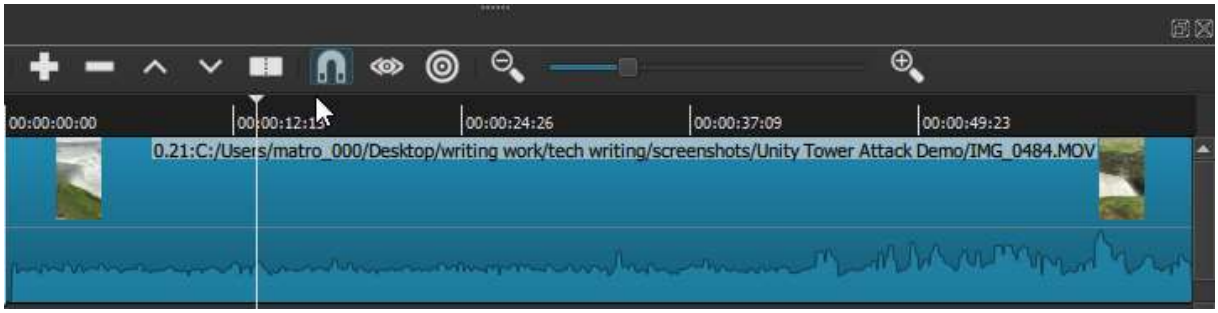
track, once loaded, is now what 'plays' in the main video window. We can use the video play controls that we saw when we first opened the file as a Source; but much more useful in this context is the **Playhead**, the vertical white line visible in the last two screenshots, with its triangular drag-point at the top visible in the first one. With this drag-point, we can use the mouse to drag the playhead to any point, and over any segment, of the track, and the video window will display whatever video content the playhead is over at the moment--right down to individual 'frames', of which we know there are 30 per second in this video, if we wanted to zoom in that far. We can also use the left and right arrow keys on the keyboard to step, in tiny intervals, forward and backward through an area of track. As we do this we will hear the corresponding *audio* for these precise places as well, unless we chose to mute it (the mute button is the furthest-left of the track controls). This dramatically slowed or backward audio this can be a disconcerting experience at first, but it's actually an essential aspect of fine-grained audio-visual editing.

All that remains is to note that the position of the Playhead also determines precisely where many editing actions will take place, such as splitting a track, or pasting the beginning of a copied segment. But now that we have a clip of media content in place for editing, we can consider some basic editing functions in the next chapter.

Basic of single-track editing: Cutting, pasting, appending, transitions

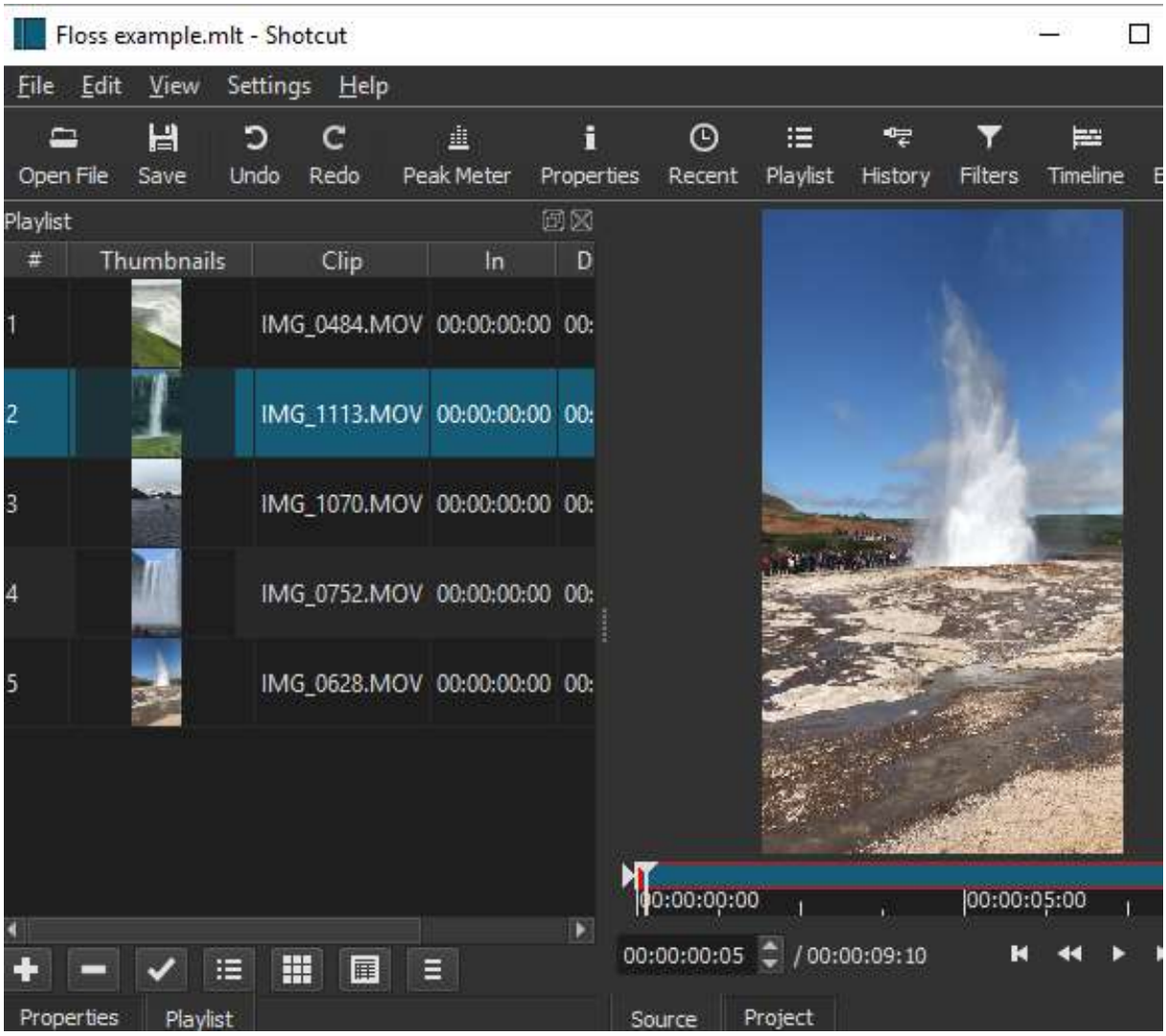
By the end of the previous chapter we had a single video clip, of the waterfall in Iceland, loaded into a video track in **Shotcut**'s **Timeline panel** and ready for editing. We will now continue with that same sample project, as we begin to look at some basic editing functions on a single track. Before we begin any actual editing, however, we need to come up with some basic conception of what we are trying to *achieve* thereby: what is the final, edited video 'product' we want **Shotcut** to render?

One thing we know already is that the waterfall clip itself, while it's a high-resolution capture of a compelling moving image, is extremely brief: only about 12 seconds total duration. That's not really long enough to ask an audience to watch, in most contexts. We want to end up with a longer final video, at least something in the 1 minute range if not longer. But how to do this? One could **Copy** that entire waterfall clip, and **Paste** it end-to-end repeatedly in the V1 track until the latter was a full minute or more in duration; but six iterations of the same short clip would simply look odd (at least for this waterfall; for some captured movements the repetition might be comical or interesting). How about stretching it out, ie slowing down the playing time? We could certainly do that, by going back to the Properties Panel for that clip and changing the **Speed**, which is set by default to real-time or 1.000x, to something a great deal slower, like 0.1x or 10% of real-time. I've actually tried that: the results in the Timeline track now look like this:



The V1 track is now just over 1 minute in duration (I've zoomed it out, but note the second markers along the top which indicate the greatly expanded duration of the clip itself). When we play it, we see the same waterfall, now flowing for a full minute but at a glacially-slow pace. Again, this could conceivably be interesting if the action captured were some kind of wild athletic move, or perhaps a volcanic eruption, that truly benefited from being seen in super-slow-mo. In this case, though, not so much. We will Undo this action, and reconsider.

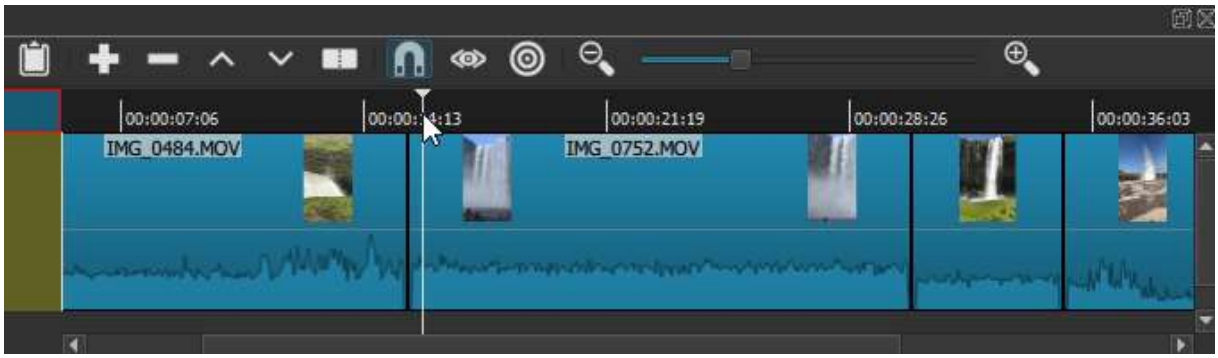
The next obvious question is whether we might compile this clip together with *other*, thematically-related short videos, to achieve a longer-duration end product. Do we have other compelling video clips from the Iceland trip last summer? Indeed we do. We will go back to iCloud and look for more Iceland videos, specifically with water themes. Here is another, extremely tall and pictureque waterfall, but the clip is only 5 seconds long -- will it make a viable contribution? The only way to know, really, is to bring it in and find out through experimentation. Here is a 45-second slow-pan of a volcanic beach-- again, let's bring it into the project and find out if it 'works' in this context. Here is 15 seconds of another tall waterfall; surely that's worth trying. Finally, a very brief clip of a volcanic geyser (it is, in fact, the original Geyser from which all others derive their name). Surely that is worth trying out. I will now open each of these clips in **Shotcut** and add each, in turn, into my Playlist for this project.



Here we see the four additional clips I have just added to the Playlist, along with the original waterfall clip (by the way, as noted in the Orientation chapter the term 'clip' here is a bit ambiguous since it can refer both to full video files, as here, and also to segments, or subsets, of a full video file as defined in the Source view by manipulating the left and right clip ends. Everything shown in blue in the Source window is part of the clip, which by default will be the full video you've opened. Clipping as such will be explored in a later chaper). In this screenshot the last clip/file added to Playlist, the Geyser, still displays in the Source window as well. Here we might observe that "Playlist" may be a somewhat misleading name for this panel, because it is not meant to imply any kind of sequential *ordering* of these clips: I can double-click on any one clip to play it in the main window; I can drag the clips in the Playlist to re-order them as I like; but this order isn't especially meaningful as I can't *play* this sequence of clips, in order, from within the Playlist itself. Sequential or linear ordering is done on the **Timeline**.

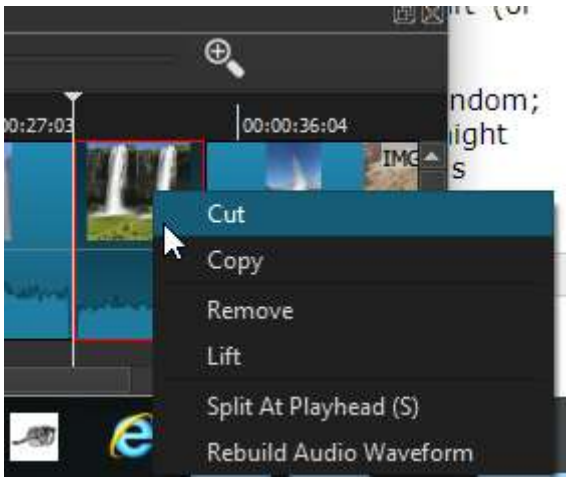
The Playlist is really just a library of assets, imported into the project, for *potential* incorporation into the final video we will define by way of crafting one or more tracks in the Timeline. We could add a dozen, or conceivably a hundred, additional assets into the Timeline -- not just video clips but also still images (for example, from among the hundreds of spectacular jpegs from the Iceland trip), as well as audio files that might be worth considering as as a soundtrack. None of these will become part of the finished video product until and unless we decide to incorporate them into the Timeline, and from there play around until we have something that displays in a way we're satisfied with, and wish to Export as the final product.

For want of a better starting point, then, I will begin by simply selecting each of the new water-themed clips in Playlist, in order, and Appending them to the existing V1 video track in the timeline, by way of the Append (**+**) button in the Timeline controls. For now I'll leave out the beach clip, as being both much longer and thematically distinct from the others. This now yields the following Timeline view, of our single video track V1, when suitably zoomed out to see more or less the whole thing:



As you can see, each of the new clips has been appended, left to right, after our original waterfall clip. The full clip is now some 45 seconds in duration. If we now play this, we have what might be considered a 'first draft' (or first cut) of a compilation video of "Water Wonders of Iceland."

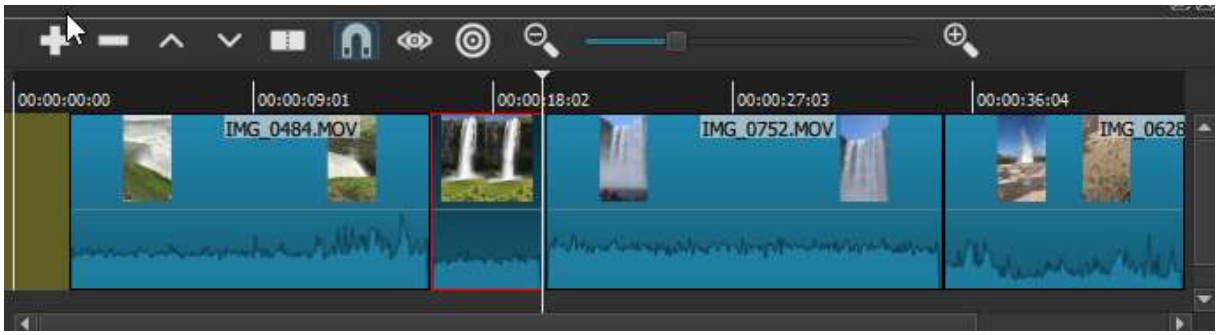
But it's only a first draft, for numerous reasons. First of all, the order of the clips in the timeline is totally random; I have essentially just followed the order in which I first chose the clips and added them to the Playlist. Playing the resulting track, I might decide this works fine as a sequence, or I might decide to rearrange the clips. Perhaps I want the third clip (its name cannot be seen in the above shot but it is IMG_1113) to appear immediately before, rather than after, the current second clip, which is _0752. There are several ways to do this, but for now the safest would be as follows: 1) I select that 1113 clip on the timeline by clicking on it (this clip or segment will now show a red border, as seen below), then 2) right-clicking to reveal a drop-down Edit menu, and selecting Cut:



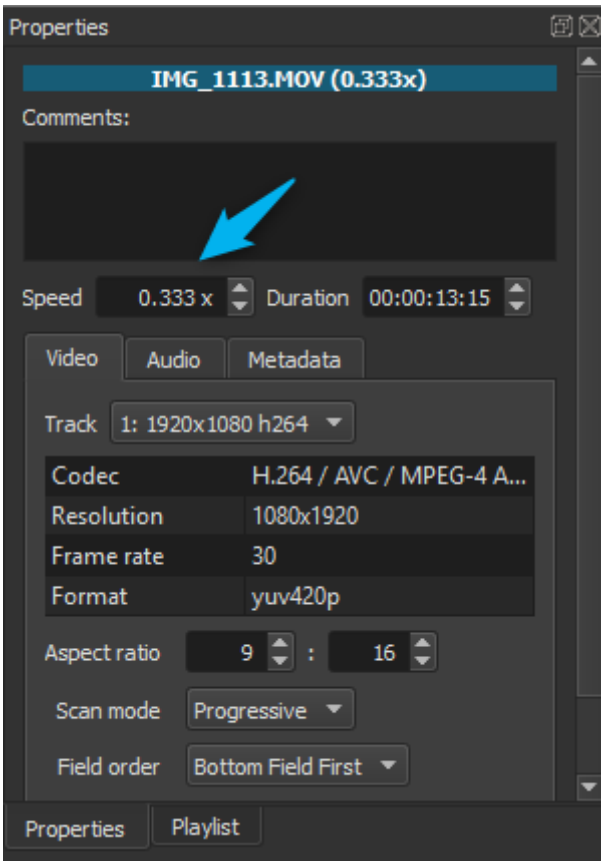
Rather than right-clicking, I could have instead used the **Cut** button from the Playlist toolbar, for the same result. Why Cut in particular? Because this will copy that segment to the clipboard but also ripple-shift the following segments to the left, thus leaving no gap where that 1113 segment was. I might *want* a gap there, but I don't in this case. Having done the Cut, I now 3) move my cursor (which also means moving the **Playhead**) to the point where I want this 1113 clip to appear, which is between the original waterfall clip (0484) and what was the second clip, 0752. Finally 4) I **Paste** the 1113 clip into its new position, using the Paste button on the Timeline controls. This will shift all remaining segments to the right again, preserving them all in their entirety. Had

I instead used the **Overwrite** button, it would it have pasted the 1113 clip in the same position but *not* shifted the others right, instead overwriting a portion of the 0752 clip. Again, you may want to do that in some cases, for example to preserve all other segments' position relative to other tracks in your project, but I don't care about that here.

The resulting edited V1 track now looks like this, with 1113 in its new position:

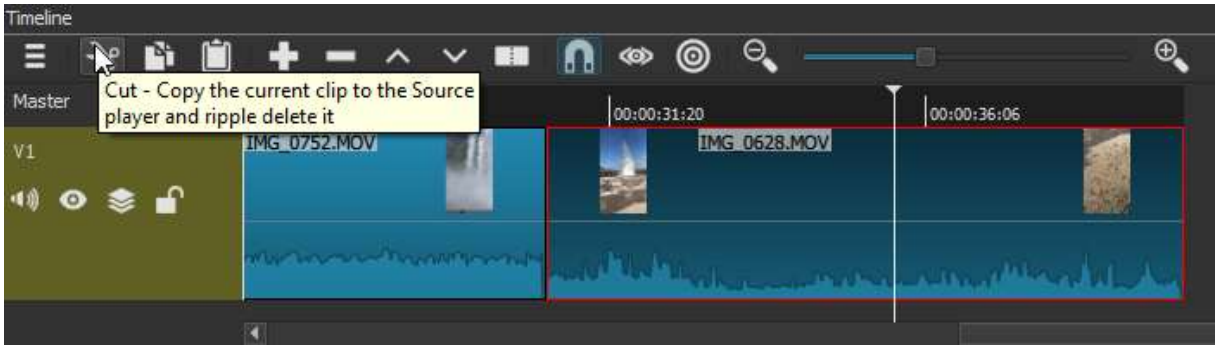


Playing the V1 track now, I am better satisfied with the sequencing of video images. But this track could still be improved. For one thing, we can see here how unequal the video segments/clips are in duration, which a viewer would also percieve and might find annoying. In particular, the 1113 clip we've just repositioned is still much shorter than the others. To some extent we might consider ourselves constrained here by the original durations of the raw videos -- but only to the extent that we need to preserve exactly the same frame-rate for all segments, which is 30/second. Perhaps the 1113 clip, which is fairly spectacular, would actually benefit from being slowed to, say, 1/3 of its original playing speed, which will also have the effect of increasing its duration by 3x? Let's give it a try. I double-click that segment again to select it, open the **Properties** panel, and change the **Speed** from 1.000x to 0.333x:



As soon as I click out of the Speed input box in this panel, I will find that the segment in question has actually trippled its total length, and thus its displayed space in the V1 track, and pushed the remaining segments neatly to the left. (I could also have done this same operation 'manually', by dragging the right edge of the segment itself to make it wider; but to do this I would first have to drag any following segments manually to the right, one at a time, to make space for the wider 1113 segment, so I would suggest doing it via Properties).

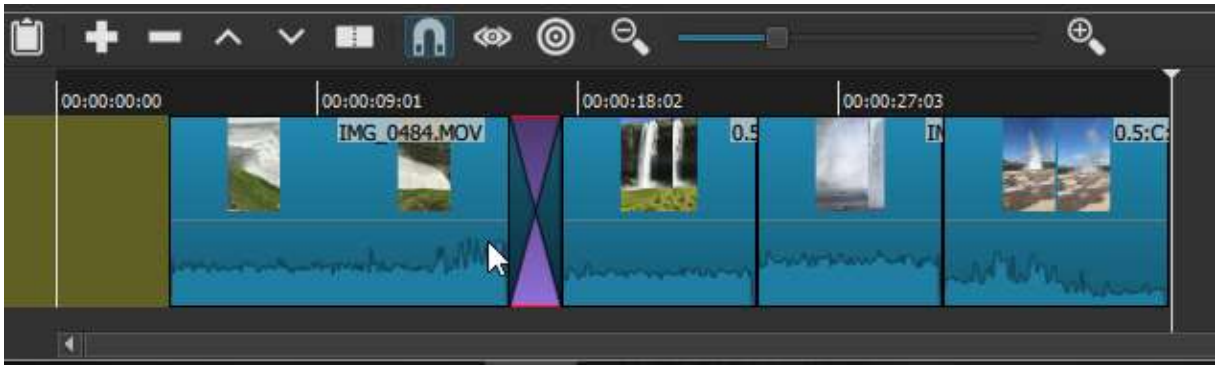
As it happens, I find after playing the result that .333x speed is actually a bit too slow, but .5x speed is very nice, and results in a second segment more proportional in duration to the first and third. Excellent. This still leaves an issue with the final segment, though, which as you may recall showed the Geyser erupting. The entire clip is 9 seconds long, or near enough in duration to the others. But erupting geysers are fleeting things, and the actual eruption is only the first half (or really third) of the clip, after which point the camera lingers briefly on the steam aftermath and then pans down toward the ground. We don't want or need this last footage. So now we will perform our first **Split** operation, to get rid of it. First we will need to select that clip in the track, and zoom out the Timeline considerably so we can see and step through it in much greater detail (with the track selected, the Playhead will move to that spot and will always remain at the center of the zoom, so you don't need to hunt for it again). That expanded Geyser segment now looks like this, with the playhead roughly at its center:



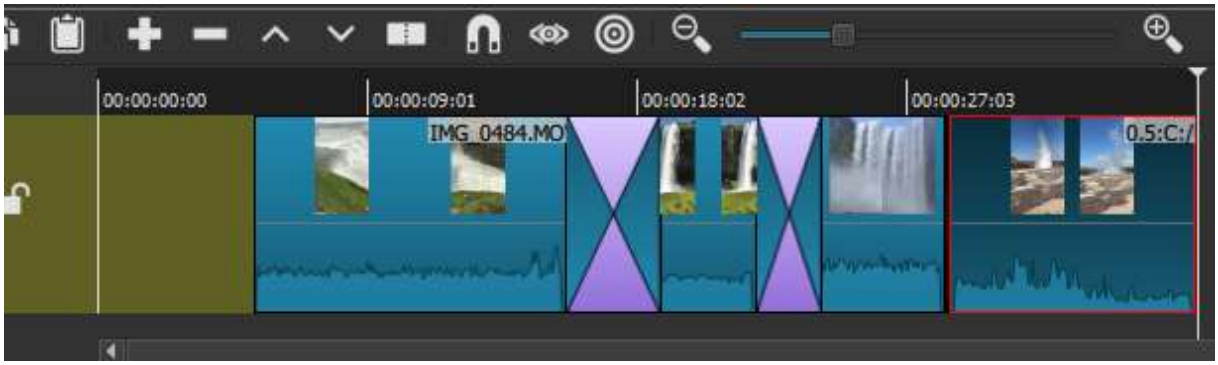
As noted previously, you can move the playhead back and forth either by dragging it at the top, or by 'stepping' with the forward and backward keyboard keys. You will likely find the latter gives the most granular control. I will choose a point a few seconds after the Geyser has performed its show and left the more slow-moving cloud of steam, just slightly left of where the playhead is shown here. Now I will "Split at Playhead" -- this is NOT the "Cut" button that happens to be showing its mouseover label in this screen shot, but rather the icon just to the left of the horse-shoe (mousing over each button in turn will always show you its name and function). That button will create a split in the 0628 (Geyser) clip at the precise spot of the playhead -- after which everything to the right of the playhead becomes a separate segment, and can be Deleted. Better still, the now very-short segment of Geyser clip that remains, to the left of the playhead, can be given the slo-mo treatment as well, which ends up working very well as it did for our 1113 clip. And we're back to four segments of roughly equal duration *and* visual interest, with a nice rhythmic alternation between real-time and slo-mo speeds.

Is this masterpiece finished, then? Of course not. Leaving aside more advanced questions like labeling, an introductory title, and what to do with the audio (all of which will be addressed in subsequent videos), there is still a basic question of the **transitions** from one clip to the next. As it stands that transition is entirely abrupt. **Shotcut** offers a wide range of options for transitioning from one video clip to the next, though most of these are applied through **Filters**, and will be the subject of a more advanced chapter. However, one of the most useful types of transition, the Dissolve (and an accompanying audio cross-fade), is available directly from the Timeline UI. To create a dissolve transition, all you need to do is grab one clip -- your mouse cursor becomes a hand icon when you are appropriately hovered over a clip to drag it -- and drag it so that it

somewhat overlaps an adjacent clip, either to the immediate right or left, then release the clip you're dragging Here, I have dragged the first waterfall clip (our original) slightly to the right, so that it overlaps with the slow-mo second clip:



The new crossed region shows what was the area of overlap, and has now become a **dissolve transition**; if I now step through or play the track, my first waterfall image will dissolve into my second. As it happens, this first effort yielded a transition that was too brief, in my view: probably less than a full second in duration. It is possible to resize such a transition one you've created it; but as this can be tricky I would suggest instead Undoing the transition, and trying again with a larger region of overlap in your drag. I was happiest with a transition about twice the size of the first one (without changing the zoom of the Timeline). After that, I created another transition between the second clip and the third, by dragging the second one to the right by a roughly similar amount. I also previewed a third transition from there to the final, Geyser clip, but decided that with that clip already only catching the second half of the eruption (a function of the videographer's reaction time), it was better to leave that transition abrupt. Thus my V1 track now looks like this:



Here you can see the revised first transition, between the first and seconc clips, and the second transition that is slightly smaller (and we see that there remains no transition between the third and final clips). With any transition between shorter clips, such as these, it's important to keep in mind that you need to preserve *some* sufficient region of each clip outside the transition area, or the viewer won't have an adequate chance to register the image itself (or any accompanying titles or voice narration) before the next transition begins.

You may also notice that the first clip no longer begins at the zero point of the track, because we dragged it to the right to create the first transition. If we now play the track as is, the resulting gap will show up as a black screen for those first few seconds, before the first clip abruptly plays. We *might* want to delete this gap by right-clicking on it, which will bring up the single option to "Remove", after which all content will shift left to the zero point. But this opening gap is also a very nice length for an opening title, something we will cover in the next Chapter. For now we will leave this opening gap where it is.

With that, we are well on our way to creating a quite respectable promo video for the Icelandic Chamber of Commerce -- or at least one 30-second segment of such a video. For it to be really useable, this segment would probably need to be assembled together with other segments featuring different geological features, cultural life in Reykjavik, Icelandic ponies, whale-watching or whatever. Of course the overall promo video would probably begin with an intro title, and would probably feature intercut section titles and/or superimposed text captions. Implementing these will be the subject of our next chapter.

More Editing Basics: Edit-Friendly formats, Intro Titles and Floating Captions

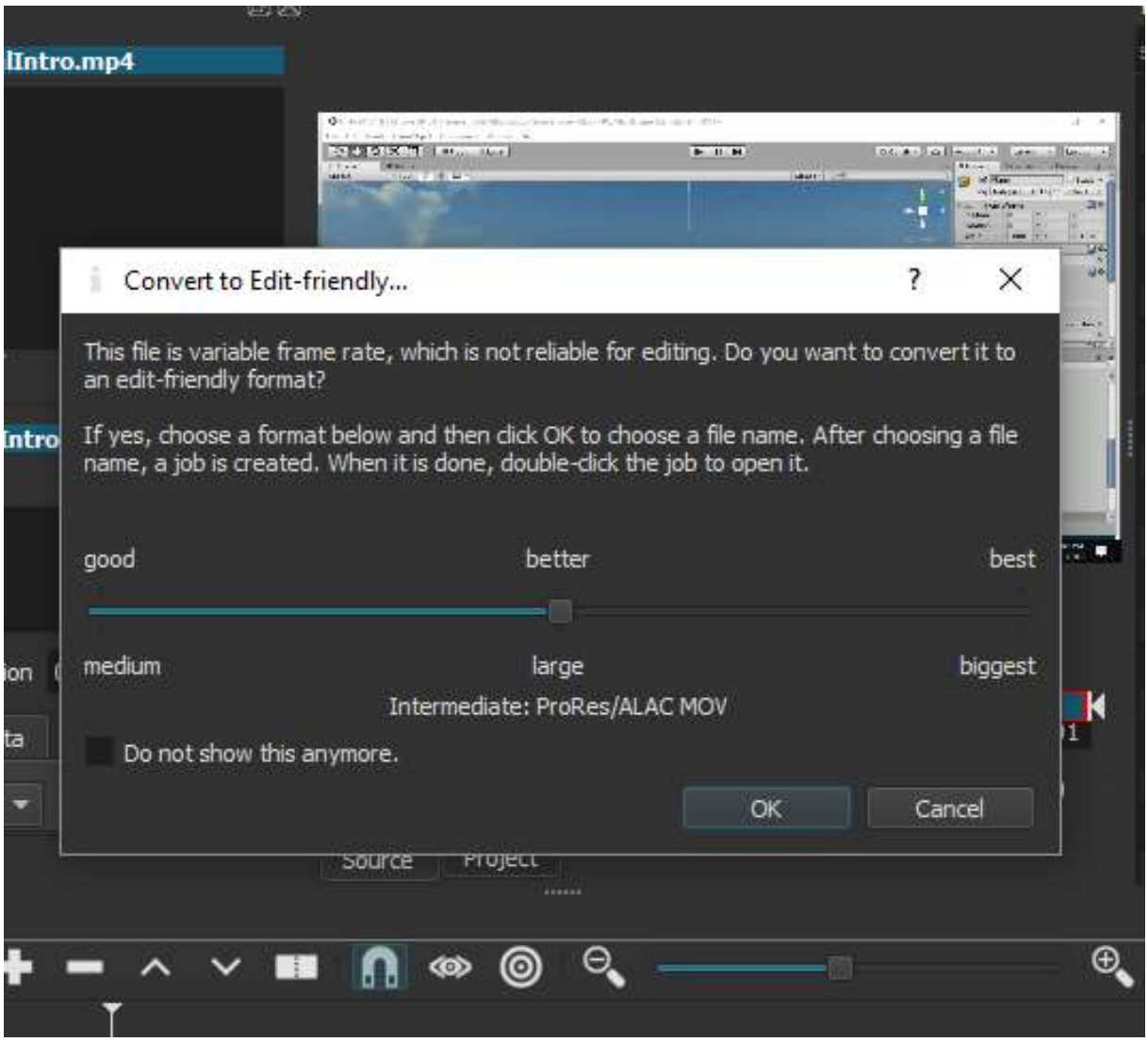
We now consider the tasks of adding an introductory title screen and on-screen captions to our videos. While we could continue working with our Water-Wonders-of-Iceland example from the previous chapter, it might be better to take up a very different kind of video as an example--if only to make the point that most of these basic editing operations will be equally useful to a wide range of video genres. Here I will work with a video screen-capture segment I recorded as the general introduction to a video-tutorial series for building and coding a certain 3D gaming scenario. These details are not particularly important, for our purposes, except by way of explaining the specific titles and captions we'll be looking at; this could equally well have been a tutorial on how to play the guitar intro to "Stairway to Heaven," or how to tie a Windsor knot. Any such tutorial would probably benefit equally from titles and captioning.

We will begin, then, by Opening in **Shotcut** a segment of 'raw' video, of just over a minute duration, which I captured as part of the overall introduction to my tutorial series. To record this and other video for the series, I used one of the countless free video-capture tools now available: in my case the tool was a web-based screen video recorder from Apowersoft, which I chose because it had sufficient features for my purposes but was still extremely light-weight, and thus didn't compete for resources with the very resource-intensive Unity3D engine. This might go without saying, but the best screen recorder out there won't help you much if it bogs down the software you're trying to demonstrate. This also implicitly answers another question that some readers might have: why use two different software tools for screen-video *recording* and video *editing*, when a number of tools promise the ability to do both?

The reality is, first, that very few *free* tools do a truly competent job of both, and free is presumably a consideration if you're read this far in a **Shotcut** manual. Just as importantly, though, the tools that *do* a competent job of both, free or otherwise, are in my experience massively greater resource hogs than something like Apowersoft or even many app-based screen recorders; so unless you have really massive CPU and RAM available, good luck trying to screen-record a demo of a software tool like Unity, or any authoring tool by Adobe, or just about any software made by Microsoft. A major reason Adobe and Microsoft products are such bloated beasts to begin with--and so expensive--is that they try to be all things to all people. Function-targeted, and where possible open-source, software is always going to be easier on your machine as well as your budget.

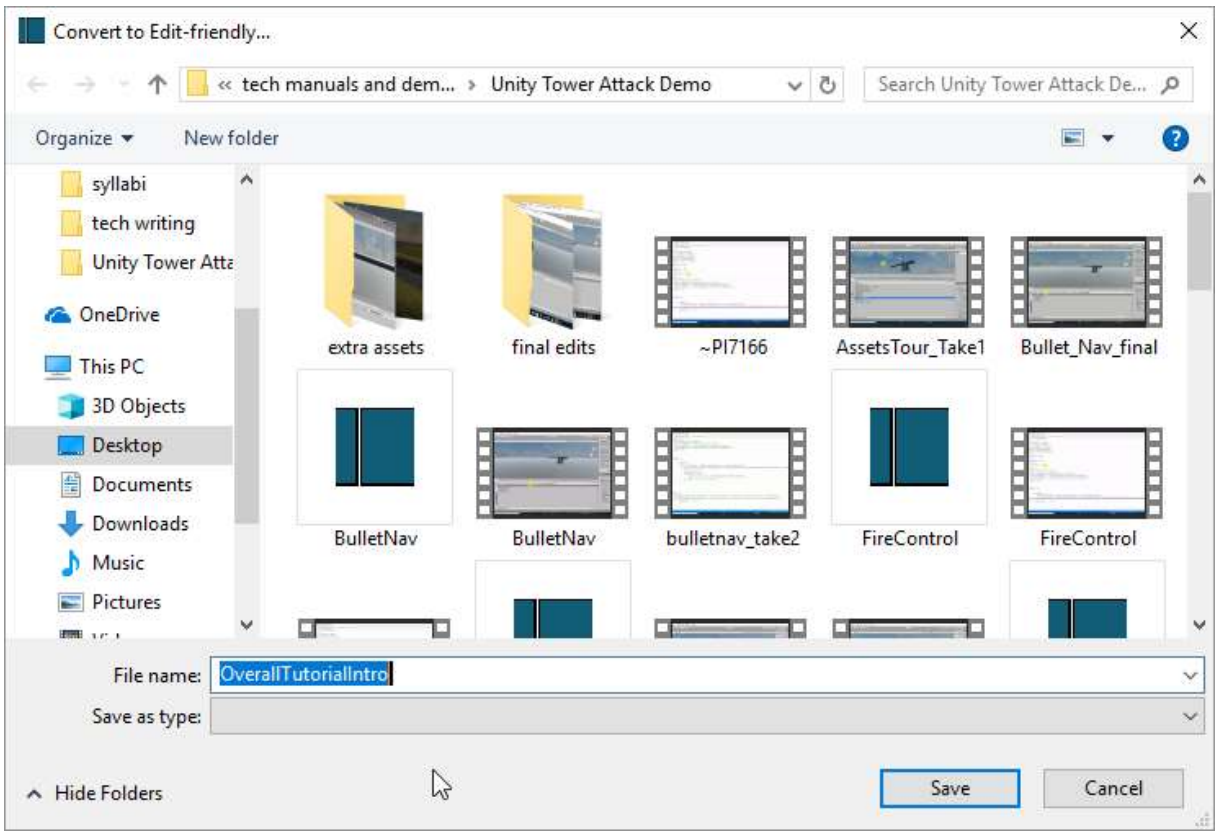
Edit-Friendly formats

Apowersoft, then: this tool outputs its video as mp4, which as we have seen is one of many formats that **Shotcut** works with. However, an interesting thing happens when I go to **Open** my raw "OverallTutorialIntro.mp4" in **Shotcut**. The file opens and begins immediately playing in the main Source window, as we'd expect; but directly in front of that I get a pop-up window:



What has happened here, as the text above indicates, is that **Shotcut** has assessed the file I've opened and concluded that it isn't in the most "edit-friendly" format: specifically, this one was recorded by Apowersoft at a variable rather than fixed frame-rate (unlike our Iceland iPhone video clips, which were all at a consistent 30 frames per second). This does not suggest a shortcoming of Apowersoft but rather a sign of some intelligence: any smart screen-capture software will devote more frames-per-second to *movement* on the screen, rather than moments when the screen is stationary, as a way to minimize the size of raw screen-capture videos.

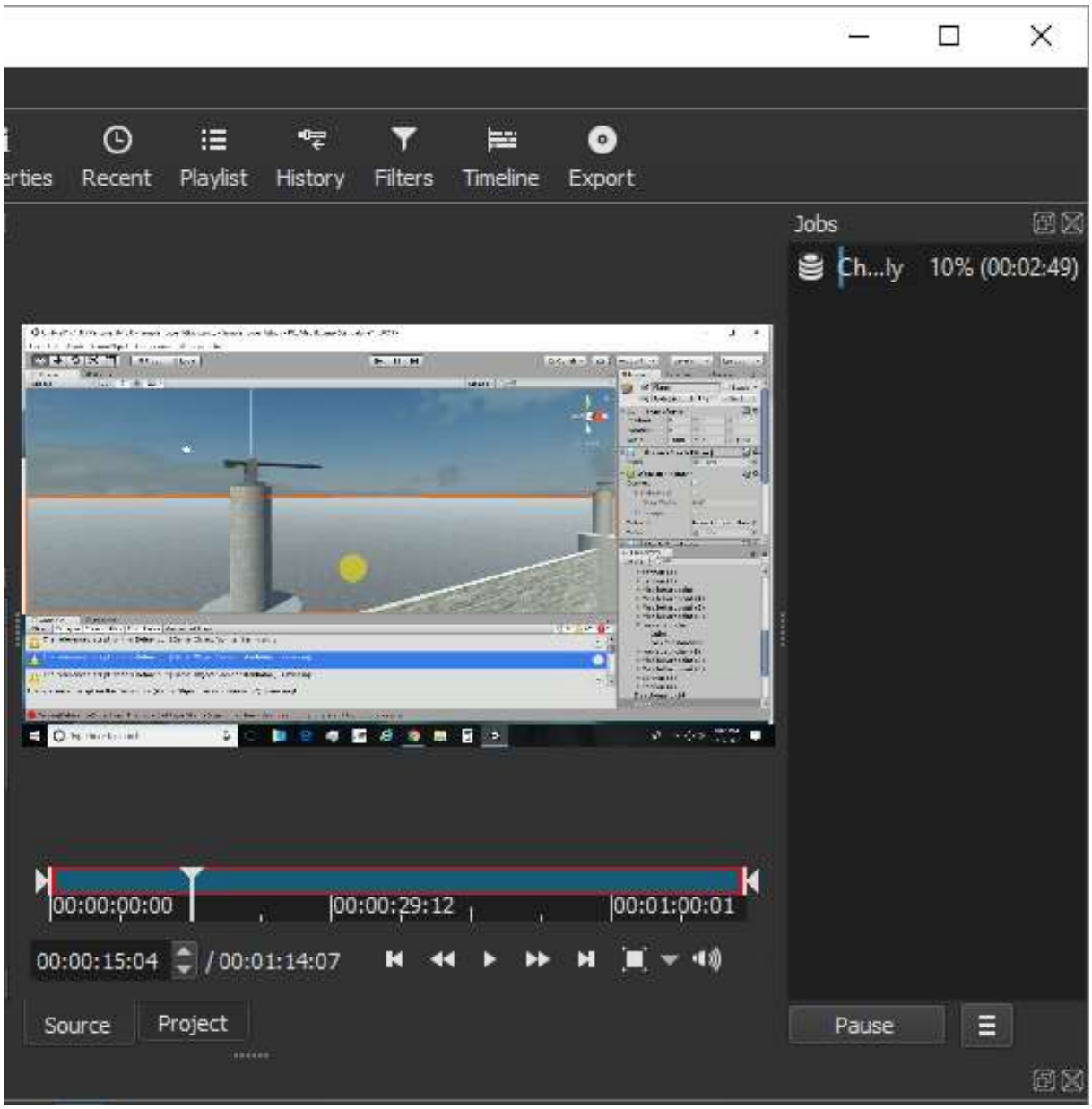
But **Shotcut** prefers to work with segments in a consistent frame-rate. So it has simply offered to convert my video into a particular kind of .MOV file that it can work with more easily. I could refuse the offer, hit "Cancel" and take my chances with the raw mp4; but instead I'll take up **Shotcut**'s offer and hit 'OK'. As soon as I do, a standard explorer dialogue opens, allowing me to choose a name and saving location for the edit-friendly version of the file:



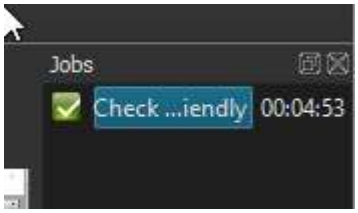
By default, **Shotcut** has selected a name (and location) which is identical to the video file I chose to Open in the first place. Even accepting this default would not result in *overwriting* the original file, since the converted version is in a different format (in this case it would have a .mov rather than .mp4 file extension). Nevertheless, I will append "_friendly" to the file name to avoid any possible confusion with the original.

When I now hit Save, the conversion process will be kicked off; this process is not instantaneous but may take seconds to minutes, depending on the length of the original file. The process will therefore be queued as a **Job** and indicated in the **Jobs Panel**, which will open to the right of the main Source window, showing this job with a progress indicator, as seen here at 10%:

(cont'd next page)



When this conversion Job is finished, it will display the total job duration (this one took 4:53) and the icon will also turn green, as shown:



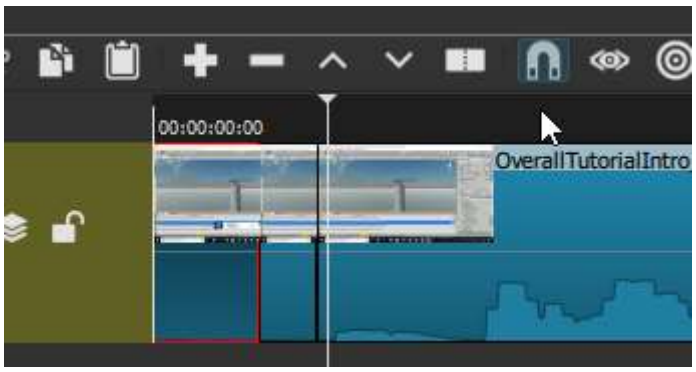
I can now double-click this item in the Jobs panel, and this converted .mov file will replace my original mp4 file in the Source view (what we're seeing there is the UI of the Unity3D tool that I am demonstrating in this tutorial). As we saw two chapters ago, if I now want to incorporate this file into my project as material for editing, I will need to first add it to the **Playlist**, and from there to the **Timeline**. Below, an edit-friendly version of my Tutorial Intro clip now resides on its own track in the Timeline:

As with the waterfall clips we opened in the previous chapter, here too we can see the wave-form of the audio associated with this video. Here, though, that wave-form takes a much more distinctive and articulated shape: this is because what we are seeing is the voiceover narration I recorded along with this raw screen capture. Although this lies outside the focus of the present chapter, I should note here that these distinctive wave-form shapes (as visual representations of the actual audio signal) are extremely useful for purposes of **editing**. If you are going to trim away video segments, or intercut between segments, you'll typically want to do so at some logical pause in your voice narration and not in the middle of a sentence or a word. Moreover, even the most disciplined voiceover narration almost always contains some unwanted pauses, stammers, filler words like "um" and "you know", or simply moments when you backtrack and rephrase a point. These can be precisely edited out of the audio (as I did for all segments of this finished video tutorial) by splitting at precise points around the wave-forms you'd like to omit. We will explore this in our first chapter on Audio Editing.

Introductory Text Titles

For now, we'll assume that the voice narration is already perfect, and this clip is ready to serve as the general Introduction to my tutorial series. Or at least as the beginning of one: this clip is only a bit over one minute duration, and in fact it took me three or four raw 'takes' to cover all of the various points I'd envisioned for the general series Intro. These various clips were then imported into **Shotcut**, assembled onto a single video track, and cut and spliced together until I had a single continous sequence to my satisfaction. But that process was in principal identical to the one we saw in assembling the 'Water Wonders of Iceland' video in the last chapter, so there's no need to reproduce it here. What this segment does still need, however, is an **introductory text title** that will display visually for 4-5 seconds before the screen-capture and narration begins (I could also include a brand logo image with the intro title if that were appropriate).

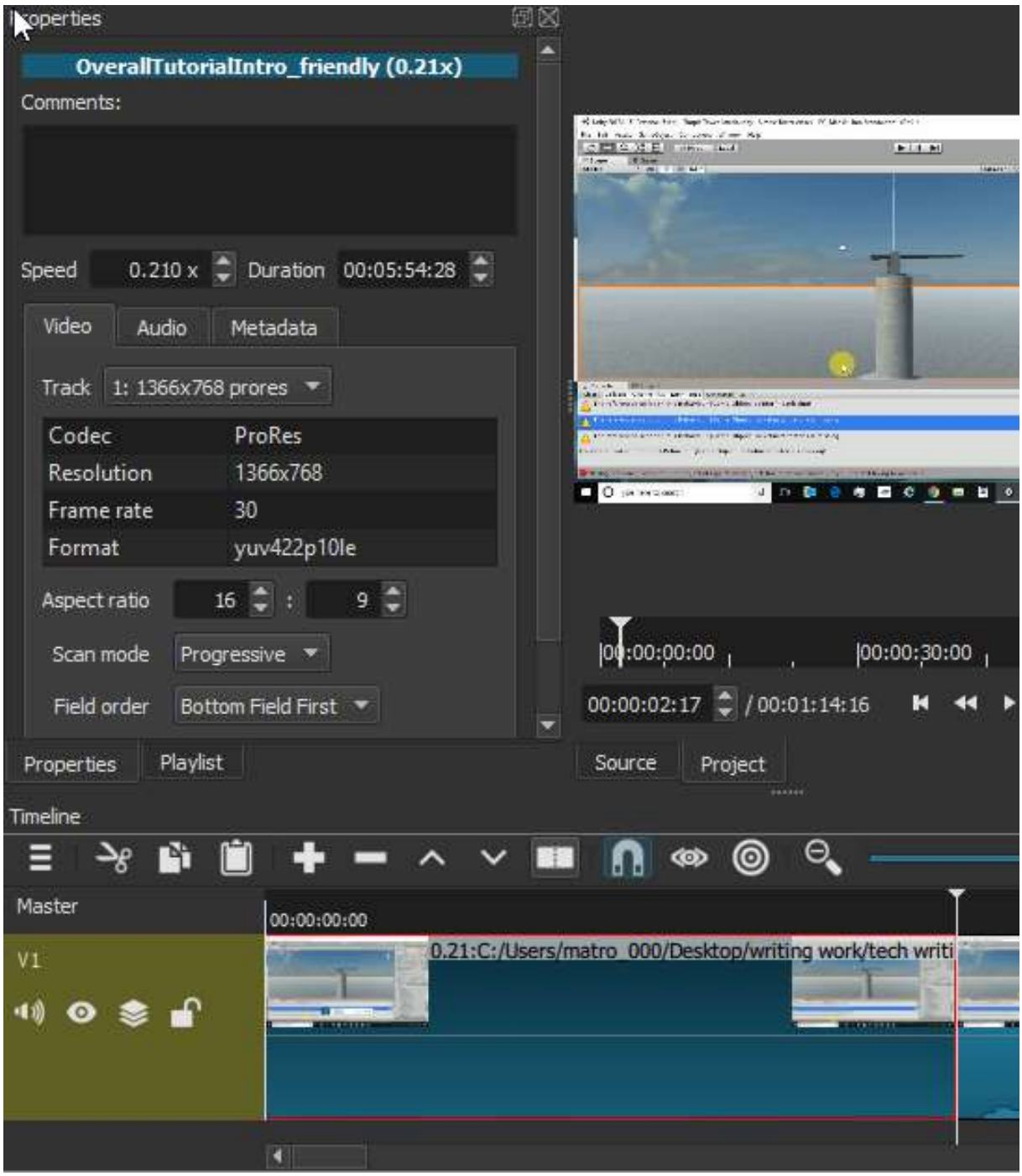
There are several different visual styles one could follow for such an intro title, each of which should be familiar from Youtube videos and other short films. One would be to open with the visual image of the video itself (typically the very start), with an introductory text caption superimposed over it for 4-5 seconds before fading away. Here the video background itself could be moving -- ie, it could be the opening 4-5 seconds of the actual raw video-- IF that much video was available before any voiceover narration or other recorded sound began to play. But if, as in the present case, that much introductory footage isn't available, we can simply use a 'freeze-frame' of the first available video footage, extended to whatever duration is necessary, as the background for our text introduction. If I wanted to implement this approach, I would begin by zooming out my track to see greater detail and then stepping through the opening frames, looking for a moment prior to the start of the voiceover and which shows no visual motion on the screen. Below I have identified a good candidate segment:



Here I have made two splits (then moved the Playhead just to the right of the second split, so that it is visible). Between these two splits is my target segment, with no audio yet (as can be seen in the waveform) and the screen visible but with no cursor or other motion. The red-highlighted segment to the *left* of my target segment, which is the actual beginning of my raw recording, shows a brief glimpse of my screen-capture software, so I will delete that out. To do this I will use the **Lift** control on that segment, not Cut or Delete, because I want to leave a gap in that spot to work with. After trimming away the first segment, I hover my mouse over the left edge of the second (target) segment, and stretch it as far as it will go to the left, filling the gap and making that segment duration longer, like so:

The video now begins with a 'freeze-frame', which actually still comprises several literal frames but for all intents and purposes is static. But it's still less than a second in duration -- not nearly long enough to serve as an appropriate background for our title. We already know a precise way to make that segment longer, though: I can select it, go to **Properties** (which will automatically load the properties of that segment), and change its **Speed** from 1.000x to any fraction thereof. Below, I've changed the speed to .210x, which yields an opening segment of a nice multi-second duration:

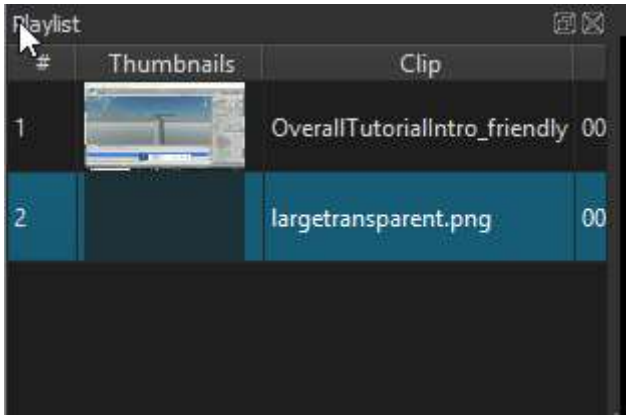
(cont'd next page)



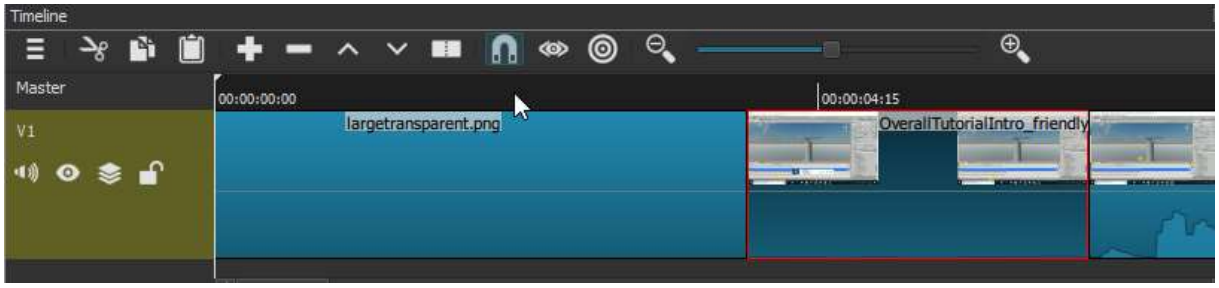
Note, again, that resizing your segment this way, rather than maually by dragging it, means that all following segments are automatically shifted to the right, which can be very useful especially if you have multiple separate segments.

Now, *IF* I wanted to use this superimposed visual style of opening title, what I would do at this point is apply a **Text Filter** to the opening segment, sizing and positioning the text appropriately as it appears on the Source screen. We will see what this kind of operation looks like shortly. As it happens, however, I want to use an alternative visual style: a (white) text intro that appears initially on a *black* screen, sustains that look for 4-5 seconds and then dissolves to the actual moving screenshot and the start of narration. This has a nice dramatic effect and is also less cluttered visually: my white Title text will appear starkly against the black background, whereas withe a superimposed title over the video image you need to choose a text color that will stand out against any and all colors in the screen capture itself.

What we want to do, then, is to replace all (or most) of our stretched-out 'freeze-frame' intro segment here with a static image which will a) show up as black in the finished video and b) accept a text filter for our Intro text. For this we could use a pure-black jpeg or png image, but just as good for our purpose here -- and much better for our next purpose -- will be a *transparent* .png. Here I'm using a 1300 x 700 px transparent png I created in Gimp (an open-source alternative to Photoshop); one could use any number of image-editing tools, or even Powerpoint, to create such an image, or else probably find one in a Google image search. Just make sure the image is entirely transparent across its surface. Here is the image imported into my Playlist:

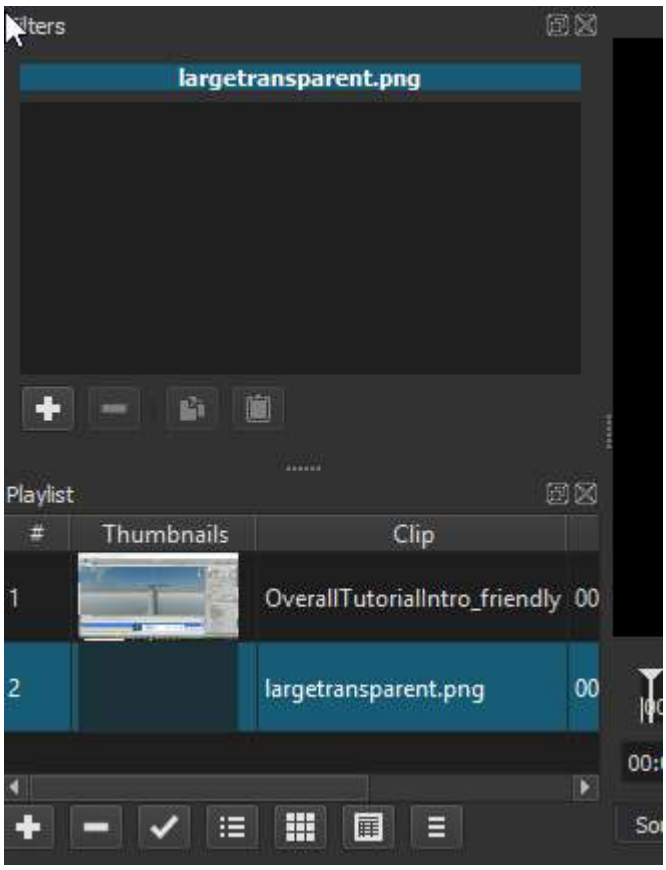


With this item selected in the Playlist (it will be highlighted in blue, as shown above), I can now transfer it to the existing V1 video track in the Timeline. I want it at the very beginning of the track, so I make sure my playhead is at the beginning point, and then use the **Paste** control (the clipboard icon) of the Timeline. My transparent image is now inserted into the V1 track at point 0, with all subsequent segments shifted right:

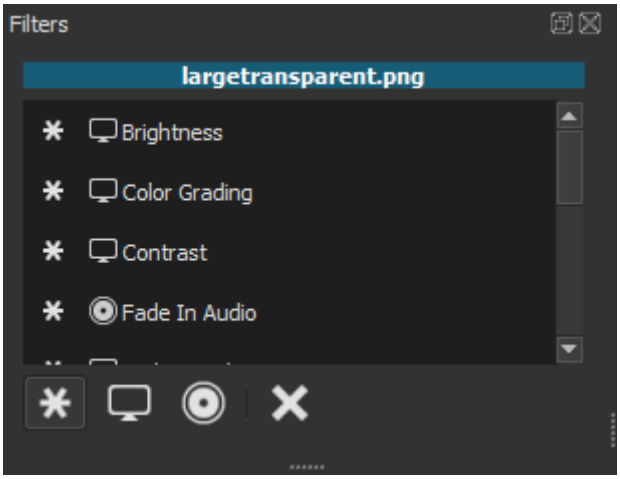


By default, **Shotcut** assigns static images like this a duration of 4 seconds; we can change this to any duration we need in the **Properties** panel for this image, but in fact 4 seconds is very close to what we want. Notice that the 'freeze-frame' segment we created above is still in place, following the image; for now we will keep this, because we'll want to use some of it for the dissolve transition from the intro screen to the video proper.

First, though, it's time to add the text introduction itself, by way of a text filter applied to the transparent image. With that image item still selected in the Playlist, we open the Filters panel:

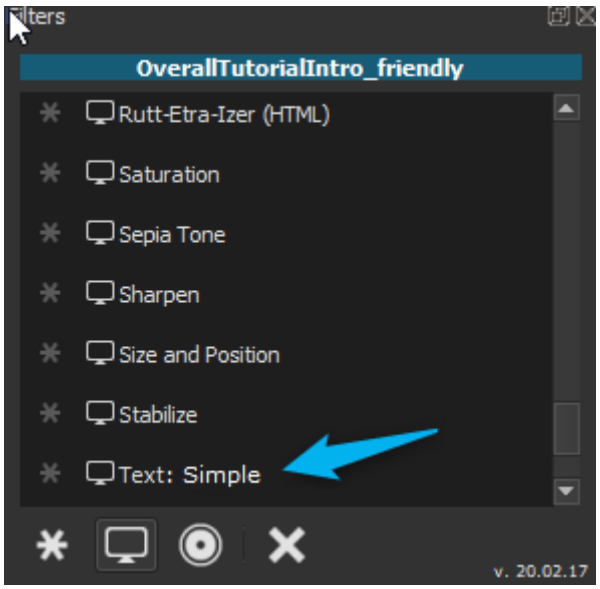


This screenshot shows both the Filters panel (still empty) and the Playlist panel, to make the point that there are "**+**" controls for *both* panels. To add a filter to the selected transparent image, we want the "Add a filter" **+** button from the Filter panel controls. Hitting this 'add' button will, by default, open up a list of "favorite" filters:

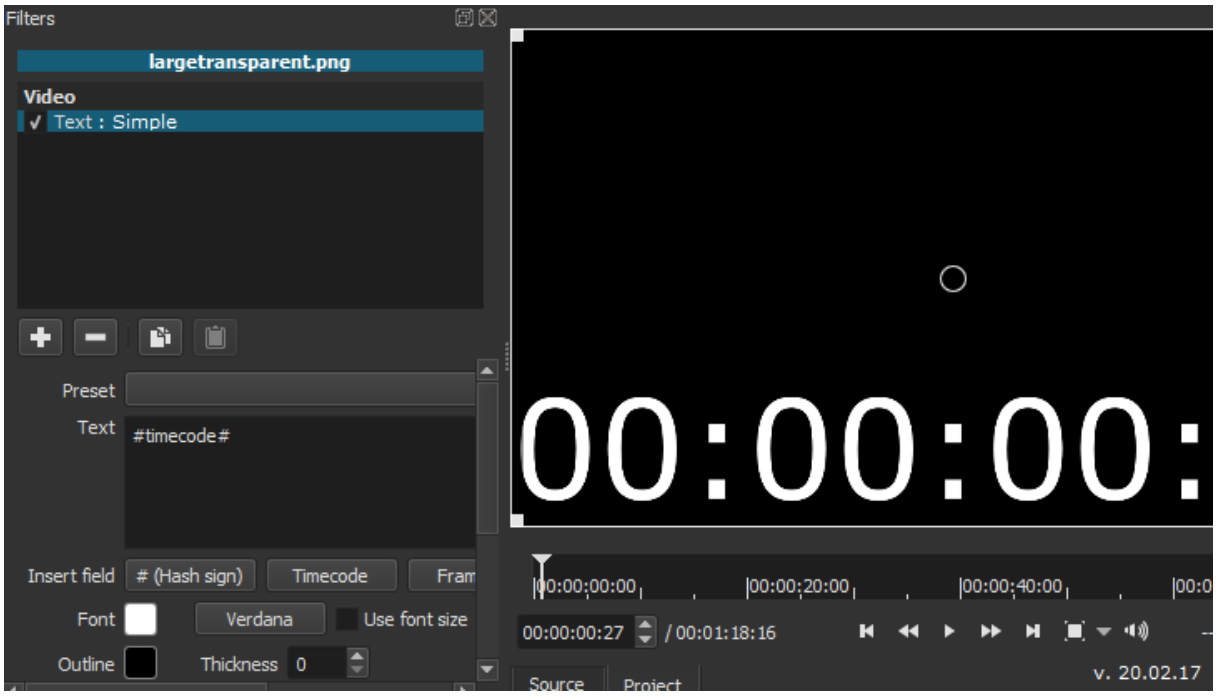


This list of 'favorite' filters includes some popular filters for altering the video image, the audio track, as well as for adding html overlays. But this is only a small sub-set of the vast array of available filters, which can be searched /sorted differently using the buttons now visible at the bottom of the Filters panel. We will select the second (screen) button to get a much longer, alphabetized list of filters related to altering visual aspects of the video display. What we're after is the **Text : Simple**[2] Filter, near the bottom of this list:

[2] In version 18.x, filter name was **Text**. With version 20.02.17 name has changed.

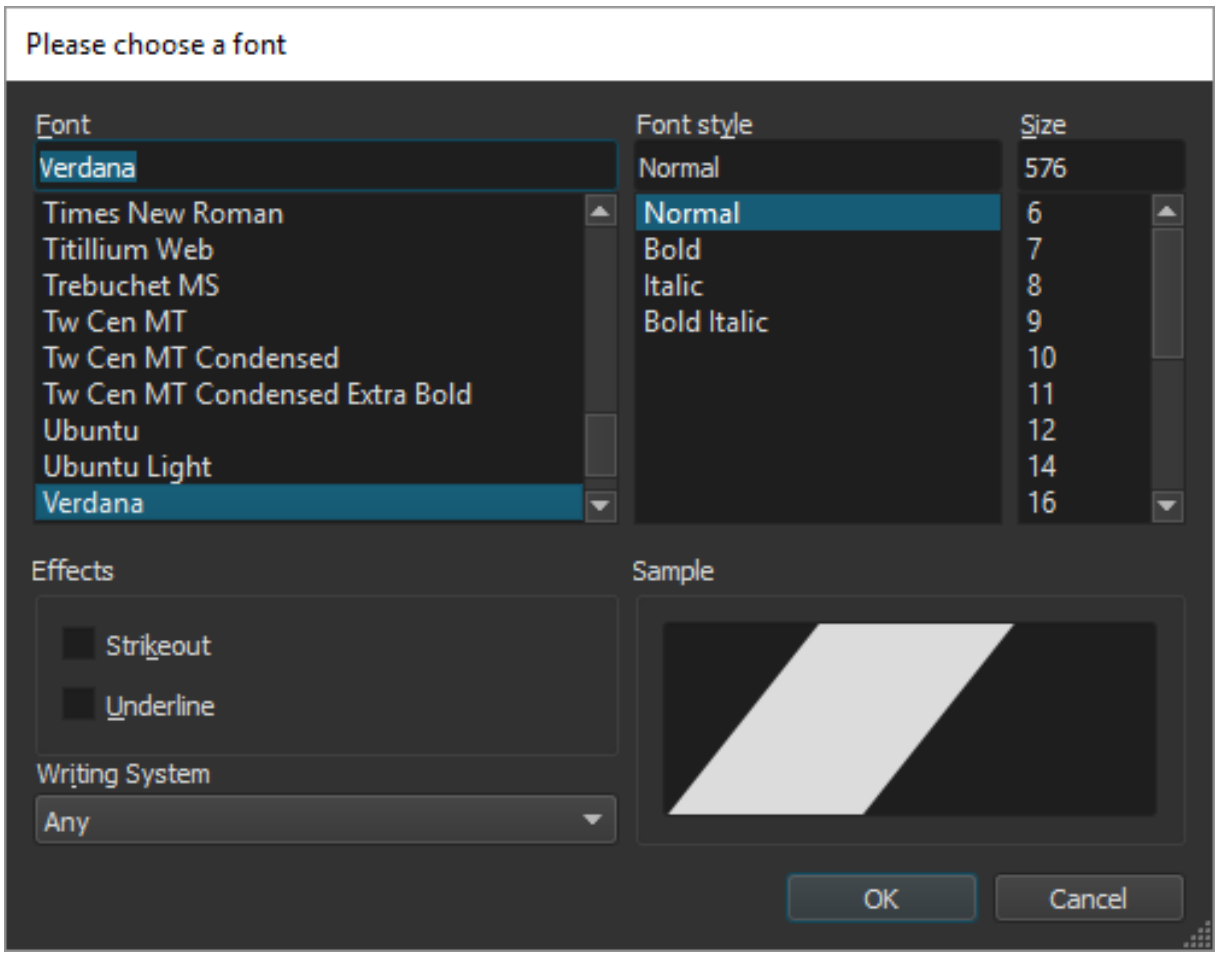


If I now click[3] on the Text : Simple filter in this list, **Shotcut** will add this filter to the selected Playlist item (the transparent image) and will simultaneously open a dialog exposing various important properties of the Text filter:

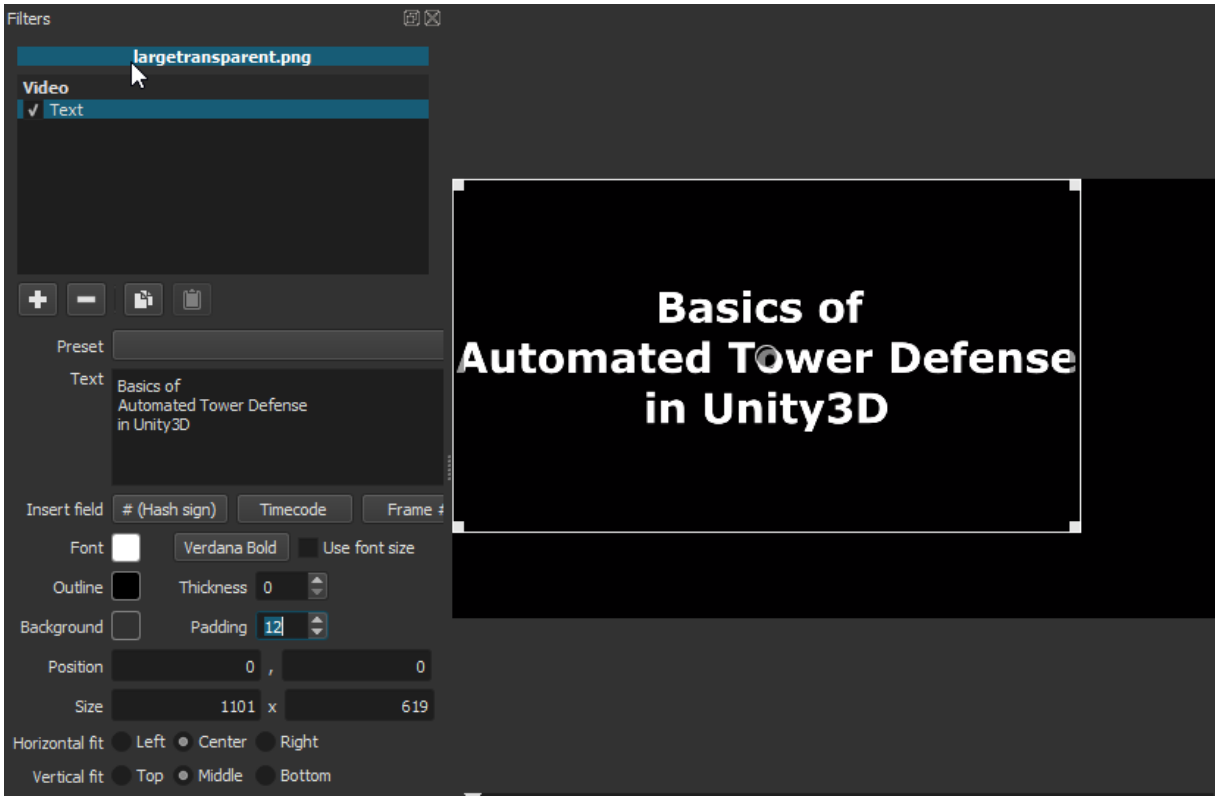


The most important property of the Text filter is, of course, the text itself. By default, the text is set to a timecode field, and to a very large white Verdana font. The effect on the actual video display is now shown in the main Source panel (right now we are at the 00:00 moment of the timeline). We can replace the #timecode# keyword with text of our choice; here we will enter our chosen title: "Basics of Automated Tower Defense in Unity3D". We can then click on the button labeled "Verdana" to invoke a dialogue allowing control of all Font properties:

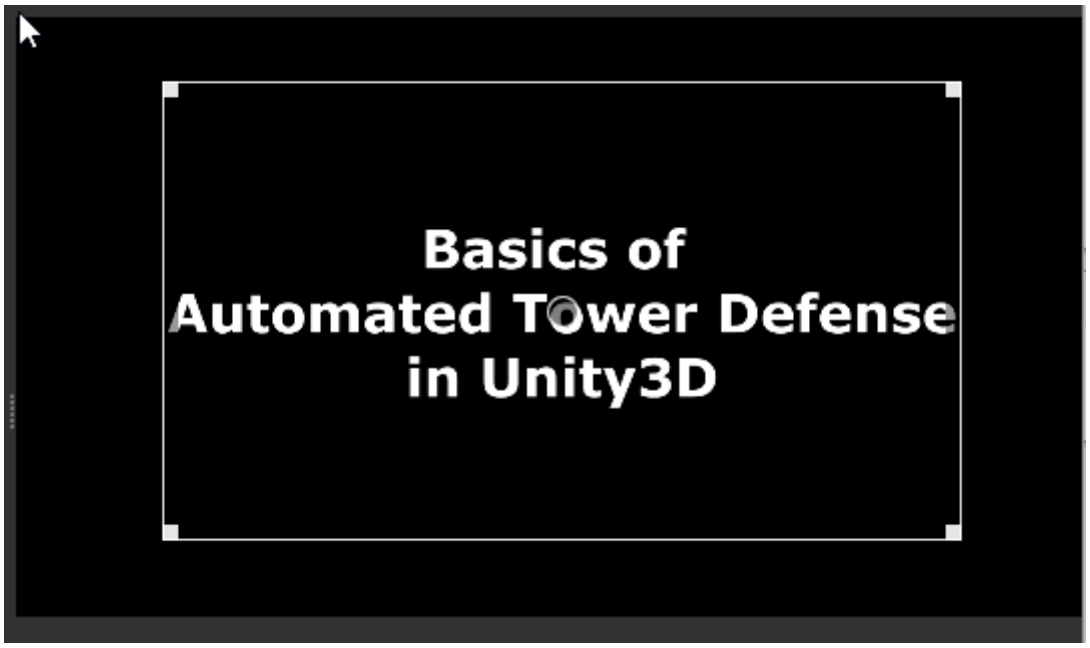
[3] Original reads « double-click », this is wrong, a simple click is enough (v. 20.02.17).



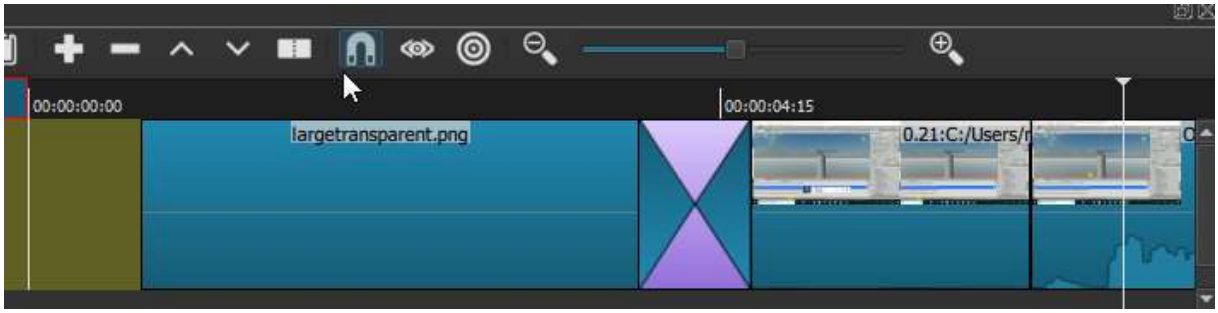
We will keep Verdana font face, but change it to bold, and allow the font size to be regulated by the full size of the text area on the screen. After a bit more manipulation, our result looks like this:



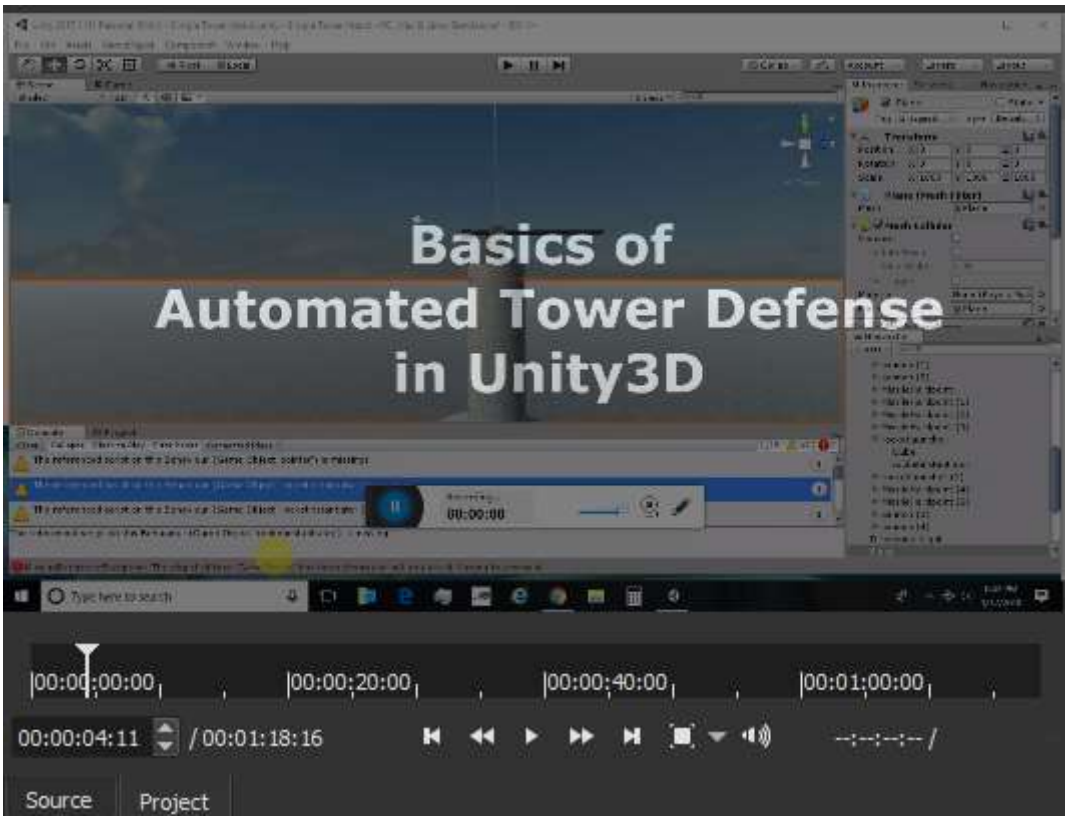
Notice, first, that with a **Shotcut** Text filter it is necessary to add your own line breaks to the text (otherwise it would stretch across as a single line of text). From the controls near the bottom of the Text filter properties I have also chosen a "Horizontal fit" of Center (this really means center-aligned rather than right- or left-justified), and I've chosen my text to be centered vertically as well. But note that this means centered within the **text field**, not the video screen as a whole. Initially these two are identical; however, the text field has a white border and familiar drag/resizing points at each corner, and in this screenshot I have already dragged the lower right corner point inward to make the text field smaller than the full screen, so we can readily see the difference. Of course now the text is no longer centered on the screen, but at this point I can move the entire text field wherever I want on the screen by dragging it from the circle in the center. After a bit of dragging and some more resizing, I have the text positioned on the screen to my satisfaction:



Note that this graphic is only displaying in the center window because, in the Timeline, the playhead remains somewhere over the transparent image segment in the V1 track, which is the segment to which I have applied the text filter. As soon as I move the playhead on from that segment, the displayed image will abruptly change from the white text/black background to whatever is next--in this case the 'freeze frame' static image from the beginning of the video, which is the next segment in the track. This is what would happen when played in real-time as well. So while the 4-second opening Title is now pretty much the way we want it, we need to do something about the abrupt transition. But we already know how to make this a nice dissolve transition instead: using our mouse in the Timeline, we simply need to drag the Intro image segment a bit to the right (or we could drag the second segment a bit to the left), and the overlap between them will create a dissolve, of that duration. The V1 track in the timeline now looks like this:



Here we can see the familiar cross-pattern indicating the dissolve transition between the Intro title and the opening 'shot' of the video proper. As before, my dragging has left a blank segment to the far left of the title image, which I can right-click/'remove', and everything else will slide over to the zero point. Now, if I were to drag the playhead back over the exact center of that cross pattern (ie, the mid-point of the transition), I would see that the intro title is dissolving nicely into that opening shot:



Notice that the screenshot itself (showing the UI of the Unity3D authoring tool) is almost entirely visible at this point, in mid-transition, even while the white title text can still clearly be read as well. This is because, while the white text is now at 50% opacity, its background has no opacity at all--because that image is *transparent*, not truly black. It merely 'reads' as black to the video viewer (and it will in the exported video) in the same way that a gap left in the video track would read as black. This effect is probably not crucial for the opening title--the transition would not look *that* different even if I'd used a black image--but it will be extremely important when we turn to our final task in this chapter: creating a superimposed text caption in a segment of moving video.

Floating Captions

Intermittent **on-screen text captions** might be added to video for any number of reasons. You might want to visually label some particular item or person that is briefly visible in a video scene (remember that **Shotcut** text filters can be positioned anywhere on the screen that we want them); you might want to label or name the scene itself (a landmark or geographic location, event, person, etc); or you might want to supplement or emphasize a point of voice-over commentary with some visual text. I found myself doing the last of these at various points in my Unity3D tutorial. For example, in a later segment of what would become part of the overall Introductory video, I was demonstrating what the game actually looked like from the player's perspective: flying a jet that had to attack the Tower Defenses of the title. Here's a bit of screenshot showing

the Player jet taking off (as seen within **Shotcut**'s Source window), before the jet turns to attack the towers:

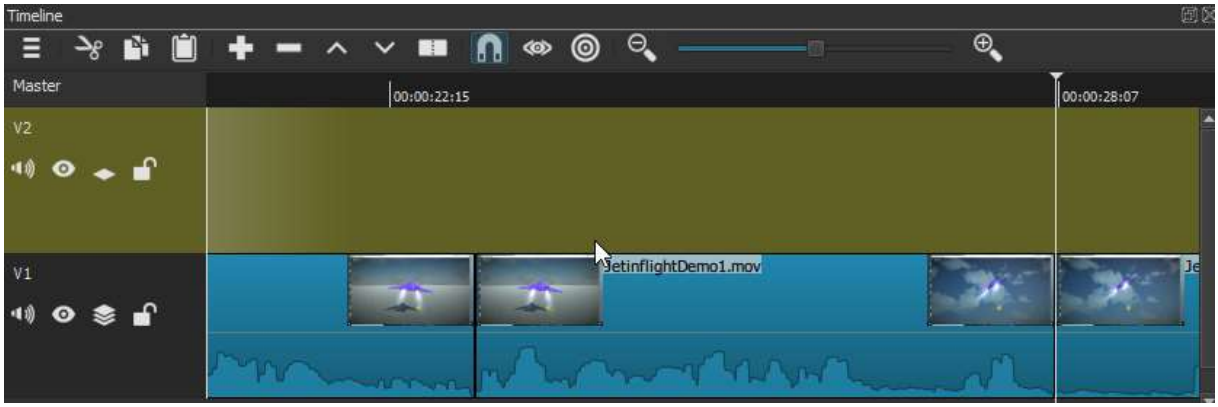
Now, this entire video tutorial series was about creating and coding the anti-aircraft towers themselves so that they function as a formidable automated defense system. For their target I just needed a player-controlled aircraft, and the one I used was largely a stock object, with stock flight-control code, provided with the Unity tool itself. I made this point clearly enough in the voiceover narration, but felt it worth emphasizing further with a text caption that would appear at this same point in the demo. How would we implement this?

In fact we've seen nearly all of the pieces we need already. We know how to add a text filter to a segment of video track, and we know how to define a specific segment through cuts or 'splits' at the start- and end-points. We know how to zoom a video track and 'walk' the playhead slowly along, to choose exactly where to place those start and end splits (based both on the video's visuals and on logical start- and end- points in the accompanying audio track). In this case I've added two splits in my video of the flying jet, creating a brief, 4-second segment of the jet demo exactly where I want my disclaimer caption to appear:

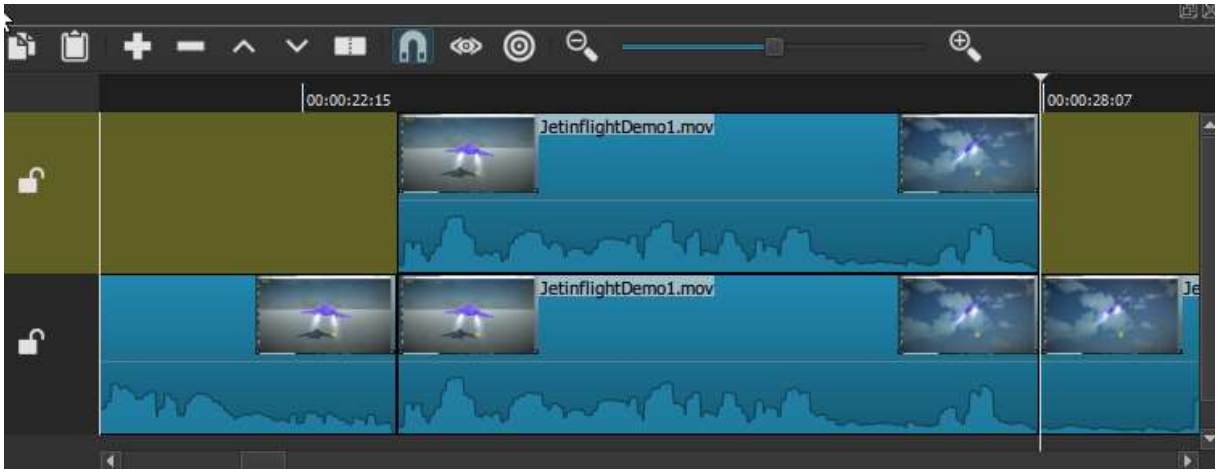
Now, it would be possible at this point to add a text filter directly to this video segment -- as we've seen, we could control the text caption's size, font style and color, as well as its exact position on the screen. But if we did that, the text caption would appear and

disappear abruptly, as the playhead passed into and then out of this specific segment of the track. We all know from watching videos that good on-screen captions fade in and then out, so how to accomplish that effect? Could we solve this with a dissolve-transition at either end of this segment? Not really, because what would be dissolving would be not just the text caption but the *entire audio-video scene*, which would very much disrupt what should be a continuous viewing experience. Instead, we need to finally break away from the single-track focus of our editing so far, and make use of **Shotcut**'s ability to 'composite' or merge together multiple video tracks.

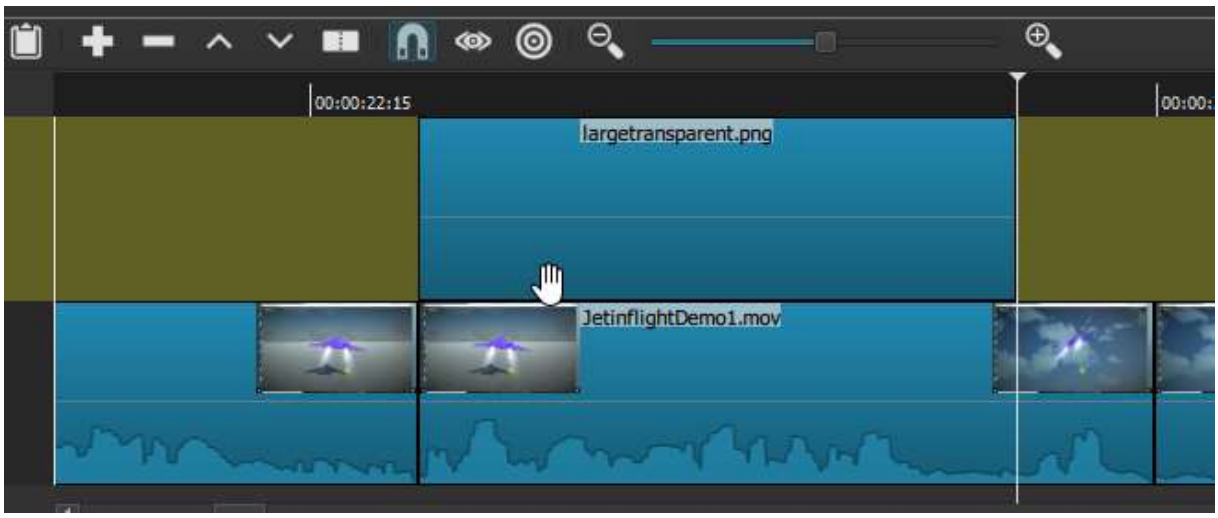
The first thing to do, then, is use the Menu control of the Timeline toolbar (the farthest-left icon seen below) to "Create New Video Track". This new track will now appear, empty, just above the first one:



At this point we could take several routes. One would be to **Copy** the defined segment and **Paste** it in the new V2 track, directly above its original place (but leaving the original in place as well). The result would look like this:



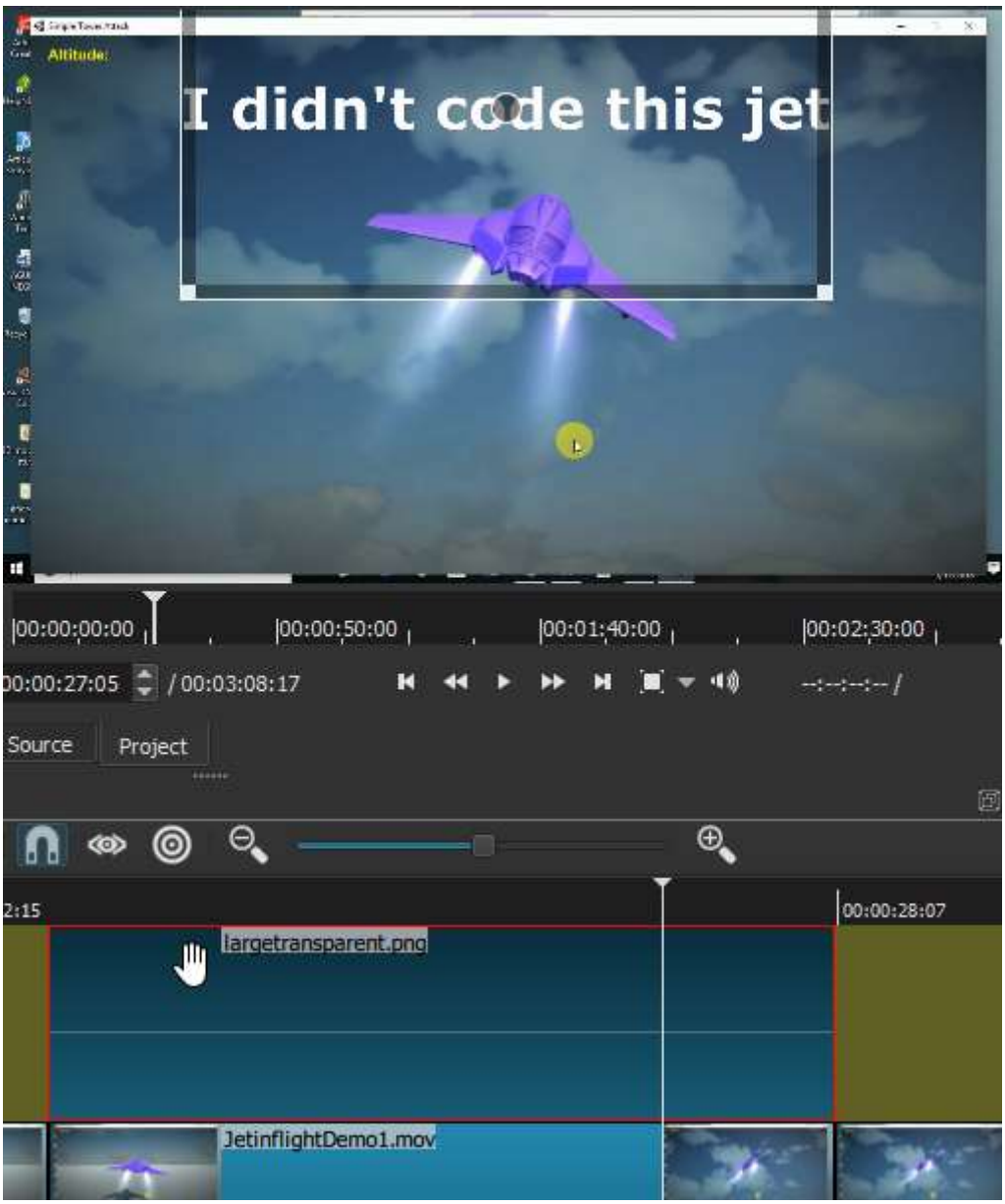
We could then apply a text filter to the duplicate segment in the V2 track, and also apply a **fade in video** and **fade out video** filter. The challenge here (besides getting the two segments exactly synchronized) is that I have seen this method work sometimes, and not other times; the text caption does not always fade, as it should, when attached to an opaque background. But the alternative route, which always works, is not to use an opaque background at all: instead of duplicating the defined segment, we can paste our *blank transparent PNG image* into the space directly above the jet segment. To do this, we need to make sure the V2 track is selected, make sure the transparent image is selected in the Playlist, move our Playhead directly over the left-hand cut (which defines the start of the segment), and hit the **Paste** control. The result will look like this:



It happens that the jet segment we defined is a bit more than 4 seconds in duration -- which is the default duration of a static image added to a project in **Shotcut** --so that, as we can see here, the image segment is initially a bit shorter than the jet segment beneath it. We can remedy that either by going into the Properties of the image and lengthening it to match the jet segment, or 'manually' (which might be easier in this case) simply by dragging the right edge of the image segment until it matches the cut beneath it.

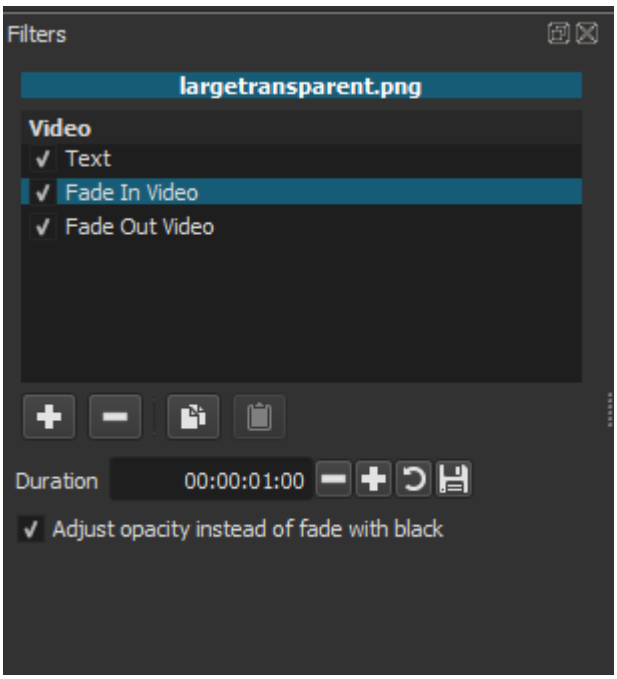
Now, with that image segment still selected, we can open the Filters panel and add a **Text filter** to it, just as we did for the Intro segment. I'm going to stick with Verdana bold and white--it shows up well enough against this background, and it's always good when possible to stay consistent in your fonts. Unlike the title, though, I don't want to use the entire screen for my disclaimer caption; so I will size and position it accordingly:

(cont'd next page)

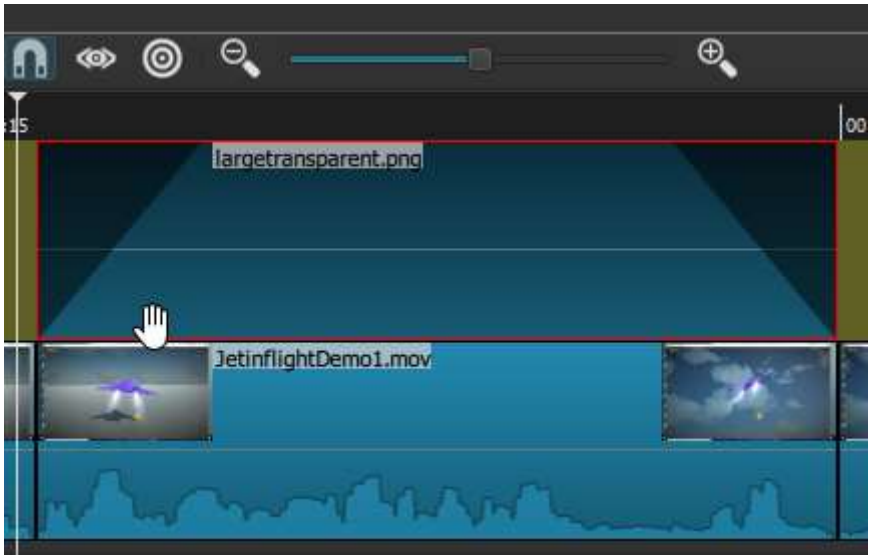


This screenshot is tall enough to let us see both the two Timeline tracks and the resulting output in the viewer: because **Shotcut** is compositing or merging the two video tracks, we see the opaque lettering of the text caption in the top track, but also right through the transparent background of that segment to the scene of the first track beneath it. The caption appears to be simply floating there, over the moving video, for the duration we've chosen. All that remains is to add a **Fade In Video filter** and a **Fade Out Video filter** to the image segment as well:

(cont'd next page)



These two filters can be found in the 'screen' filter list we saw above (along with the Text filter -- they are just much higher in the list as it is alphabetically ordered). Again, we need to be clear that we are applying these fade filters to the 'video' itself--or in this case, to the transparent image segment--and not to the text as such. It is doubtless on the **Shotcut** team's roadmap to add fade-in/fade-out capabiity directly to Text; but in the meantime, using a transparent image as our vehicle provides exactly the same end result. At least it will as long as we remember one final step: we must check "Adjust opacity instead of fade with black", on *both* the fade in and fade out filters, in order to get the effect we want. And we should note the other adjustable property of the Fade filters: **Duration**. By default this is set to 1 second for the complete shift in opacity, from 0 to 100% or back. I find this a very workable fade duration for text captions, but one could choose to make it shorter or longer as well. These fades, incidentally, are visually represented by the slanting shadowed areas we can now see in the Timeline, in the image segment on the upper track:



Were we to move the cursor/Playhead to the center of one of these slanting regions, we would see in the viewer that our disclaimer text caption is faded to 50% opacity. Meanwhile, nothing else about the flow of the video or audio changes: our text caption simply fades in, hangs there for just over 4 seconds and then fades away.

This process could be repeated to add as many text captions as necessary for a given video track--though actually, having implemented the first one, it would be easiest simply to **Copy** this top image segment, move the Playhead to a new location in the V1 track, and **Paste** the copy just above it in V2. It isn't even strictly necessary to demarcate the video segment in the V1 track with splits: I just did that as a convenient way of marking where I wanted the caption to appear and disappear. (There is little downside, though, to using splits for markers this way: upon Exporting, **Shotcut** will merge all track segments, regardless of number, into a single continuous video file). Of course, at this point the copied image will still have the original text caption we created before, but this is easily changed. Having pasted the copy in its new location, one can then open the Filters panel again--making sure that the *copy* image segment is selected, not the original--and double-click on the Text filter to expose its properties dialogue. The text can be changed to an appropriate new caption for this segment, and one can also resize or move the text area now as well.

Simple floating text captions like these are by no means the only objects we can superimpose on running video segment, via filters. **Shotcut** includes a 3D text filter as well, which to my eyes looks about as tasteful as 3D text in any other editor; but other filters allow us to superimpose static html shapes, static images of whatever size and screen position, or even inset windows with other video clips running. Some of these latter possibilities will be explored in one of the Advanced chapters.

Meanwhile, a final point to make here is that, having seen how to create a white-on-black Intro title, dissolving into the opening (initially static) shot of the video proper, we already know how we could add *end credits* to a video as well. First, select a relatively static final segment of the video, and stretch it out enough to use for the transition; then Append to this whole video sequence another copy of your transparent-background image; then add a text filter to it (and/or an html filter if you have, for example, a brand logo image you'd like to display in the end credits); and finally create the dissolve transition, going in the reverse direction from the first one. Or, if you don't want to dissolve *directly* from the video image to the credit text itself, you could add in a shorter blank segment of transparent-background image first, which will just read as a 'fade to black', and then a *separate* one to which you'd apply both a text (and/or html filter) and fade-in and fade-out filters. As a purely stylistic choice, I opted to only create an end-credit effect at the end of my *final* video segment for my Unity tutorial series; but many Youtube tutorial series have an end-credit seqence at the end of each video.

Audio and Video (I): Post-Recording Cleanup, Muting and Separating Audio Tracks

In our chapters to this point, we have mentioned audio tracks largely in passing. All of the examples I have worked with so far -- whether waterfalls filmed by iPhone in Iceland, or screen-capture video demonstrations of software -- have assumed an audio track already present with the raw video clips, captured in the same recording as the video images themselves. There are, indeed, few videography scenarios imaginable these days where *some* kind of audio wouldn't be captured as well. This might be fairly incidental to the relevant images: the rushing of a waterfall, the murmur of spectators at an event, the sound of traffic; or it could be centrally relevant to the video material itself: the audio portion of an interview or oral presentation, for example, or a musical performance, or the voice narration for a video demonstration of any kind.

The nature of the video and its intended purpose (as an edited, final product) will of course determine what we want to *do* with this recorded sound. We might want to leave it exactly as is. If the audio represents speech that is crucial to the scene -- as with an interview or a narrated demonstration of some process -- we might want to boost the volume a bit for clarity, and/or perhaps lightly edit or 'clean up' the audio track to remove pauses, hesitations and filler-words. Alternatively we might choose to *mute* the recorded sound, either partially or completely, either throughout a full video clip or just at certain moments. This might be particularly true if we intended to add some kind of *supplementary* sound to the video: a musical soundtrack, for example, or a literal 'voice-over' narration which wasn't part of the original recording. In the world of professional film-making, sound editing is an entire complex professional discipline, which has to do both with enhancing, balancing and perfecting the recorded sound from the original 'take' as well as adding in sometimes many different layers of supplementary sound: 'sound effects' that weren't actually recorded with the audio, mood music of varying kinds (amounting to a full-scale musical score in a feature-length film), perhaps a literal voice-over for a documentary or a *film noir* genre of fictional film; and, of course, actual 'over-dubbing' of the original vocal sound into a foreign language for audiences who prefer not to read sub-titles, or perhaps in a musical if the lead actor or actress doesn't quite have the necessary vocal chops to pull off the singing parts.

In fact, none of the sound-editing procedures I've mentioned here in relation to professional film-making cannot be *done* now with a good digital video editor (or at least that in conjunction with some other editing software, some reasonable recording equipment, etc). As with editing the visual component of video, so with the sound: the real question is how far we want or need to go down the path toward professional film editing, in order to fulfill the actual purpose of our home-made videos. The present guide will assume that we don't want to go *too* far down that path, at least for the present. Accordingly, we will confine ourselves here to some fairly basic sound-editing procedures, using **Shotcut**, that will be useful for most home-video purposes. These include:

1) 'cleanup' editing of an existing voiceover narration (ie one recorded with the original video)

2) muting all or part of the recorded soundtrack for a given video clip

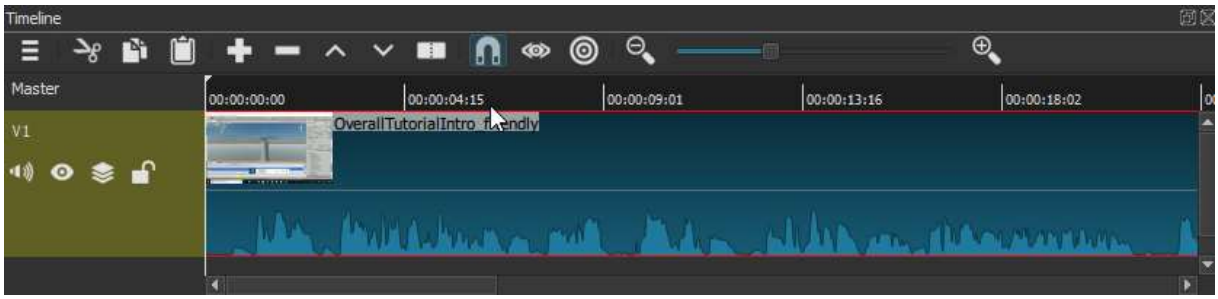
3) adding and mixing in background music as a supplementary soundtrack

4) using **Shotcut** itself to record a supplementary voiceover narration for the video scene

This is still a lot to cover, so we will devote the present chapter largely to (1), though in the course of that we will touch also on (2). In the following chapter we will delve into (3), and finally cover (4) in a final chapter dealing with audio.

'Cleaning Up' a Recorded Voiceover Narration

The following screenshot appeared in the previous chapter; it represents one of my raw 'takes' for the overall Introduction to my Unity video tutorial series, with the clip loaded into **Shotcut**'s Timeline as the V1 video track:

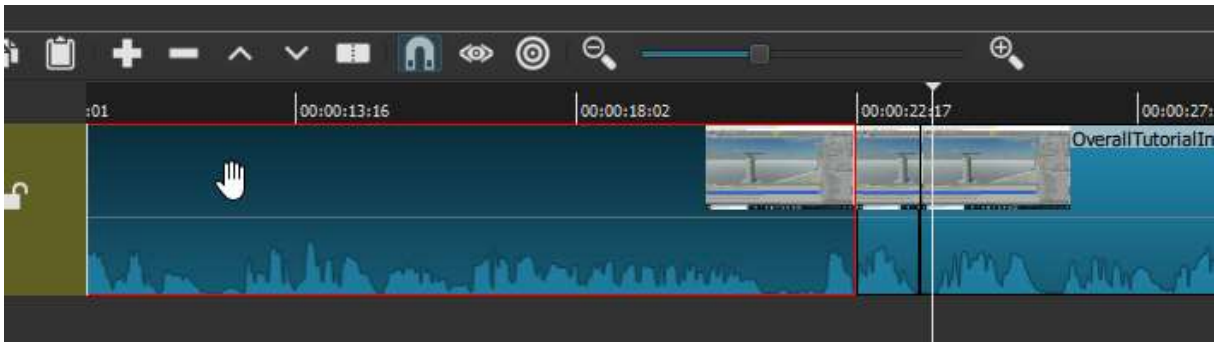


In the last chapter, we focused on the task of adding an introductory white-on-black Title segment to the start of this video track. In reality, however, I first needed to decide how much of this raw 'take' I actually wanted to *use*, and then confirm that it would actually come *first* in the sequence of clips that together, with some editing, would constitute my finished 10-minute introductory video for the series. Then I needed to clean up my captured voiceover narration of what I was displaying on the screen (as I did throughout the entire series) -- removing unwanted pauses and hesitations, stammers and filler-words: 'um', 'you-know', 'like', etc, as well as larger back-tracking in the flow of my ideas or actions on-screen.

Now, we should note that many video demos available on Youtube make no effort at this kind of cleanup, at all; it's a matter of personal preference and (to be sure) how much editing time you have. Some video demonstrators doubtless feel that leaving in the hesitations, filler words, digressions, and even occasional software errors and other unexpected behavior makes the demo more lively or accessable to a newbie audience. Personally, I find that the end product is a more watchable and effective demonstration when it's had a degree of vocal cleanup. Of course, at the other extreme, if I were a professional voice artist, working from a written-out script, my 'raw' voice recording would typically not *need* much cleanup, if any. But I am not; and the odds are decent that readers of this guide will not be either. More importantly, if we *were* doing a professional-sounding, scripted-out voiceover narration of the on-screen activity, it would much more likely be applied *after the fact*, as a supplemental audio track separate from the initial video recording. In other words, 'performing' your professional, scripted out narration while *simultaneously* doing something on the screen -- writing code, fiddling with or gesturing at different UI elements--is extremely difficult to do. And it becomes logically impossible in other video scenarios calling for commentary: if your video is capturing a rare sighting of mink crossing your back yard, or your daughter scoring her first soccer goal, or a political demonstration unfolding unpredictably, then by definition your in-the-moment commentary will be unscripted and perhaps not especially calm or linear. In those cases we really need to *choose* between keeping the raw, in-the-moment reaction--which might be entirely appropriate depending on the effect we want--or replacing any such captured audio with an over-dubbed, well-considered vocal narration. Think about the voiceover in a classic BBC nature documentary, which is most definitely *not* done in the actual moment of filming, rhetorical conventions notwithstanding ("Oh, look! It's a young stoat!"). In the next chapter, we will explore the options, in **Shotcut**,

for an overdubbed, after-the-fact voiceover; as well as adding mood music or other sound layers.

To start, though, we will take the case of a video demonstration; and we will assume that a video demo you've chosen to record, of a process you're enacting yourself and are fairly familar with, will be a case where you *can* record an in-the-moment, blow-by-blow commentary on what you're doing which will be coherent *enough* that it can be cleaned up with a little editing. The next question is: can we do this audio editing *in place*, that is, by trimming out sections of the video track that *contains* the audio wave-form, or will that kind of trimming cause the video itself to be too jumpy? There's really no way to determine this except case-by-case, for any particular moment of audio we'd like to trim. For example, the screenshot below shows the same V1 track we saw above, but now I have defined a short audio region I'd like to trim -- it's an "um...uh" sequence that comes up with some frequency in my narrations. I defined this precise region by, first, zooming in the track a bit to see more detail, then playing at normal speed until I heard the offending filler-noise; then pausing and walking the Playhead back and forth, using the left and right arrow keys, until I could identify precisely what part of the wave-form represented that sound (after a while you'll find that you can recognize the particular shape of these wave-forms pretty easily). I've then done a **Split at Playhead** at the beginning and end points of that precise region; we can see both of these splits here:



So, can I now simply delete (or **Cut**) this segment of the full video track (which would be a ripple-delete, causing all subsequent segments to shift left, leaving no gap)? That depends. In walking the Playhead back and forth, I have also been watching **Shotcut**'s central viewing screen to see how much screen movement actually occurs during this segment. As it happens, in this instance there's none at all, and so this segment is safe to simply ripple-delete from the V1 track. And in fact, it is in the nature of software video demos like this that the *great majority* of audio moments you'd want to delete are probably safe to do in this manner. When it comes down to viewing software demos in terms of very brief moments, of a second or less in duration, we find that very often there is either no screen movement at all -- we are simply talking, explaining a point -- or else there is a very small, trivial movement of the cursor (part of a larger arc from one region of a UI or perhaps code editor to another) that we can skip over in a way that will not really register with the viewer. After all, this is why screen-capture software can get away with variable-frame recording in the first place. Or, there can be much longer pauses, of several or even tens of seconds duration, where we simply lose our train of thought or hesitate over what to say next or how to phrase it -- but these tend to be precisely the places where we're also *not doing anything significant on the screen*, because the motion there is mirroring our thought/speech pattern. Again, these segments are typically safe to delete. And even if such a cut results in the effect of your cursor suddenly jumping halfway across the screen, as long as the resulting *audio* is smooth and the viewer is not *looking* for important visual information that is suddenly missing, they will tend to be quite forgiving of any such jump.

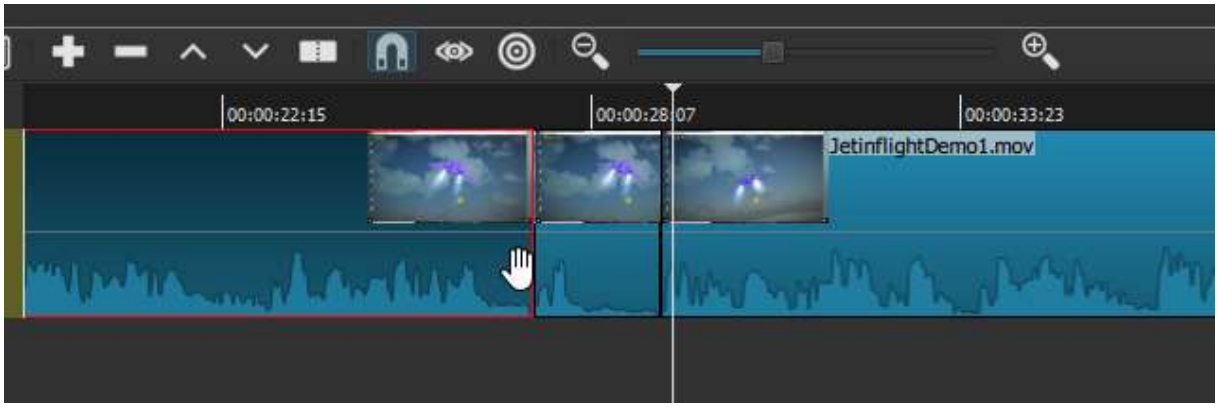
As a practical matter, I would note that over the course of my full, 10-video demo series for the Unity tutorial, I was able to do several hundred such simple cuts to the video track, representing easily 90% of the audio moments I wanted to clean up (and

incidentally, shortening the total viewing time of the series by at least 20%). But what about the remainder? Let's consider the video segment we saw in the last chapter where I was demonstrating how the game looked from the player-jet's perspective:

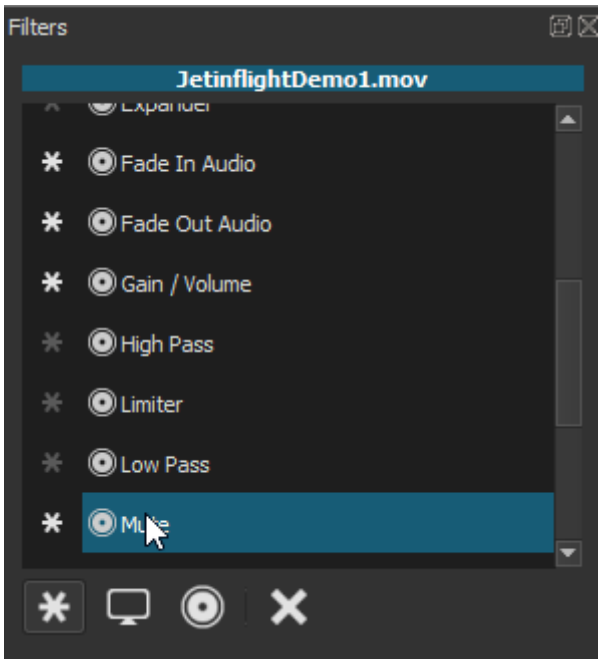
This is a video showing continuous, fluid motion for several full minutes--while the jet took off, turned, made several attacks on the towers and was eventually destroyed by them, all with my running commentary. There were some stammers and filler-words here too -- yet here, any cut in the video long enough to remove even the briefest of these audio blemishes would cause a jarring disruption of the visual flow itself. What to do in this case? One option (which I mostly chose) was simply to leave these alone, and live with some sound blemishes for this particular segment. But I could also have **muted** those particular moments of audio, while leaving the video in place and flowing smoothly -- as long as I was willing to put up with silence for the durations I was muting.

Muting part (or all) of an audio track

Below is the V1 track for this jet-in-flight video, with a segment delineated (through start and end splits) where I had an 'um' followed by a brief pause:

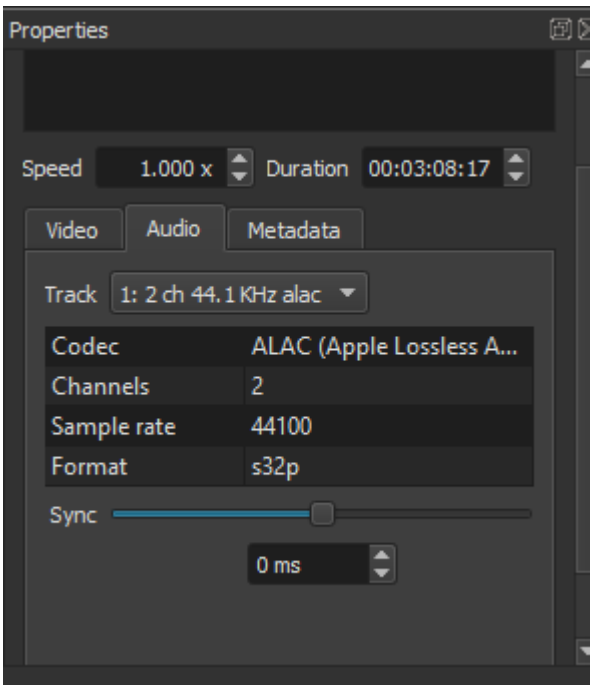


Again, this is *not* a place where we can simply delete the segment, without causing a very noticeable 'skip' in the video motion. In this case the skip would simply be annoying, but one can imagine this was instead a moment of video where I was actually demonstrating some crucial UI operation, on screen, so that the deletion could cause a real disruption in the visual information being presented to the viewer. However, we should recall here that, having defined a brief segment or 'clip' like this, deletion is not the only thing we can do with it. We can also apply filters to this particular segment, and we can change some of its properties. As it happens, both of these provide an option for **muting** this particular segment, while leaving the video playing undisrupted. As we saw in the last chapter, we can apply any filter we want to this clip--in this case the clip will not appear in the Playlist, but if we select it on the Timeline itself (it will have the red border once we select it), then open the **Filters** panel, we can select the **Mute filter** either from the "Favorites" list or from the more extensive Audio filters list:



Double-clicking on this filter will apply it to the selected segment; in this case there is no dialog for setting different options, because there are no options with this filter: we've just muted the audio for that segment, full stop. Now when I play the track, the sound simply goes away for the duration of that segment.

As we can see even in this truncated view of the "favorite" ones, audio filters can do a great deal more than simply muting: we could add a fade in and/or fade out to the audio here, we can adjust the volume, we can also adjust the bass and treble, the balance between left and right channels if the audio is stereo; we can add effects such as compressor, expander, reverb and delay. Most of these effects would make little sense applied to such a brief segment of audio, of course, but might make a lot more sense when applied to longer segments or, indeed, to the entire track. Next chapter we will explore fade filters, in particular, on the topic of mixing audio tracks. Meanwhile, though, we should note that there is another and perhaps more straightforward way to mute a single segment (or an entire track, if it is not segmented) by way of its Properties. With that segment of the V1 track still selected, we can summon the **Properties panel** from the main toolbar, and within it, click on the **Audio** tab:



This tab displays a number of read-only properties of the Audio for this segment, for example the Codec, the number of channels (2: L and R) and the sample rate. These will be the same for all of the audio in this overall V1 track. But notice that "Track" itself is a drop-down control. If we click it we find it has two options: one is the currently-displayed track name (which, again, will be the same for the entire audio track). The other option is "None". If we select "None", all of those read-only attributes will disappear, and the audio itself--just for this segment--will be muted. The **wave-form** for that audio will still be visible in that segment of track in the timeline (as it is when we apply a Mute filter as well), which may seem counter-intuitive but is a way of reminding us that this segment of audio *has not been lost or destroyed*, just muted: as long as this overall Project is open, OR has been saved, we can always go back and un-mute that audio segment should we chose to, by reversing whichever muting option we chose before.

Splitting a pre-recorded audio track from its video track

We'll close this chapter with a couple of final points on this general topic of audio 'cleanup'. First, one might be wondering whether it isn't possible to split the audio track off from a video track and edit it *separately*--thereby, for example, cutting out the unwanted audio blemishes but joining up the remainder, so that there are no resulting silences? The answer is yes, it is certainly possible to do this; and there may be *other* good reasons for spitting off an audio track to work with separately (or even export as an audio file). This is probably NOT a good way to accomplish post-recording cleanup, though. To understand why, let's take a quick look at this kind of splitting. Below is the jet-in-flight video track with the earlier splits undone (I just hit "Undo" a couple of times), which will make it easier to copy V1 as a whole track. I have also used the Menu dropdown (far left control on the Timeline toolbar) to "Add Audio Track". As we see, there is now an empty audio track, labeled A1, directly beneath the V1 video track. I have also selected this track by clicking on it, which is why it's highlighted here:

With this new track highlighted, what will happen if I now hit the **Append** control (the big **+** sign) on the Timeline toolbar? **Shotcut** will attempt to paste the V1 track, which is the only item in the Playlist, into the A1 track. But of course A1 is an audio track, so it cannot contain video content. The result is that only a copy of the *audio* from V1 will be pasted in, as we see below:

We can now see two exact copies of the audio waveform -- and at this point they are also exactly *aligned*, because **Shotcut** will always add (or technically 'Append') content to the zero point of an empty track. If I were to play this project right now, it would sound exactly as the V1 clip sounded by itself: I am actually hearing both audio tracks at once, but I cannot tell this because they are exactly superimposed. If I only wanted to hear the new A1 audio track--as I presumably would if I were doing any editing on it--I could mute the audio of the entire V1 track, either using one of the methods discussed above or, to make it really simple, by clicking the far-left "mute" control on the V1 track (this will *only* work with entire tracks, which is why I didn't mention it above).

At this point I could, if I wanted, listen to the A1 track, hunt for offending audio blemishes, split them into separate segments the same way we did above with the video track, and then delete them. But you can probably already see what would happen. Doing an ordinary Cut on the first of my offending audio segments would cause the remainder of the audio track (everything to the right of the cut segment) to shift slightly left; and since audio and video are now separate tracks, this shift means they will no longer be aligned. The misalignment will be only very slight after this *first* deletion -- perhaps a second or less -- and that alone would probably not be a huge practical issue for a screen-video demo (though it would already be annoying if, instead, this were a video of someone speaking, where their mouth movements were no longer fully aligned with their voice). But every subsequent little deletion would add to this lag. By the time we had gotten to, say, ten of them -- an average number of little audio flaws for a 3-minute 'raw' video like this -- the accumulated delay between the audio and the video would become unwatchable, even for a software demonstration. It would be impossibly distracting for an interview, and even worse for something like a video demonstration which involved playing a musical instrument. It's true that we could avoid this problem by doing a "Lift" delete rather than an ordinary ripple-delete -- this would simply leave a small gap in the audio track wherever we'd removed anything -- but then we've accomplished nothing we couldn't have done by simply muting the audio segment in

place. To review, then: splitting off the audio track from the video is easily done, and *may* be appropriate to accomplish some editing goals. But we should beware of any editing operation that leaves the audio track either shorter OR longer than the video track, because the A/V disconnect is something that most viewers will not tolerate.

This brings up a final point about this kind of post-recording audio 'cleanup': it is extremely difficult to pull off, and often impossible, in certain specific video scenarios where the audio and video are very tightly linked, and both central to the informational substance of the presentation. One example would be a video demonstrating how to play a piece of music (or anything else, like a scale or chord progression) on a particular instrument. If I am demonstrating a typical blues chord progression on a guitar, and both showing and narrating exactly what I'm doing, but I also say "um" or "you know" or similar filler words--or the dog starts barking, or my roommate slams the door--while in the midst of this process, I don't have a lot of options: if I mute out the offending words/noises I also mute out the music for those moments, and if I trim the entire video for those moments (our first option above) I lose the music *and* have a confusingly choppy video. This would also apply, really, to any 'talking head' type presentation, such as an instructor speaking into the camera for a segment of eLearning content. The audience is watching you speak, so even if they can't *hear* you say "um", or "where was I going with this? Oh yeah..." or "sorry, that was just the dog", they can still *see* you say it; and if you cut out both the audio and the video for those moments, the result again is going to look distractingly choppy to an audience accustomed to the normal audio/visual signals of oral communication. Of course, film-makers have struggled with this same challenge since the advent of talking pictures. So in these specific kinds of video scenarios, the solution now is really the same as it's been all along: a combination of 1) rehearsal, 2) multiple takes, and happily 3) the ability to splice together usable segments of multiple takes to form a continuous sequence.

With that said, we will turn in the next chapter from the topic of cleaning up existing audio to the topic of adding in new, supplementary audio, and how these might be mixed together to form a coherent whole.

Audio and Video (II): Adding and Mixing in Musical Tracks

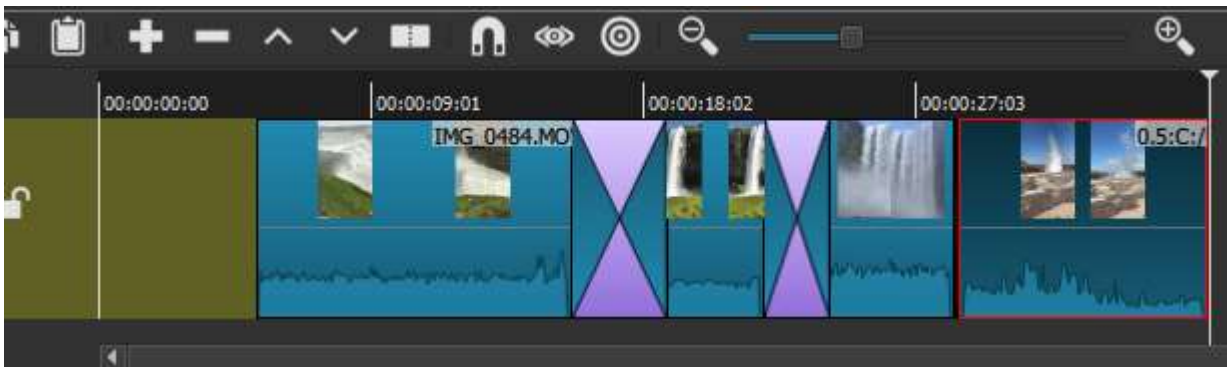
At the beginning of the last chapter, we discussed some of the general reasons one might want to enhance or 'clean up' an audio track which was recorded along with the raw video, as well as add in and mix together supplementary audio, such as a musical soundtrack to help establish mood, or record an after-the-fact voiceover narration for all or parts of the video. We then looked in some detail at ways of cleaning up minor flaws in an existing voiceover track, which also necessitated an understanding of various options for muting parts (or even all) of the audio in a recorded video track. We finally saw how we could spit off the existing audio track from a video recording, so that it could be enhanced or edited separately (with the caution that one generally wanted to keep a pre-recorded audio track *in sync* with its corresponding video).

It's now time to start looking in detail at adding in *supplementary* audio to an existing video track. In the present chapter we will consider the case where we want to add in 'found' audio, for example a stretch of music that we think might make a good accompaniment for the on-screen action in all or part(s) of our video. In the next chapter we'll look at the task of recording our own after-the-fact voiceover narration for on-screen events, which happily can be done directly through **Shotcut**. Of course, for both of these cases we will need to know how to 'mix' together the various audio tracks we end up with, and the video, so that the right sounds are audible at the right moments on screen.

Adding background music (and some benefits of exporting your work in stages)

Nearly all of our concrete examples in the past several chapters have dealt with one particular video genre: the video demonstration or tutorial. This is one genre where adding background music would generally be a very bad idea, except perhaps for a brief swell of music during the introductory title segment, and/or concluding credits segment. At all points in between, background music would simply distract from your vocal narration, and even from the actions you're performing on the screen, when you want your viewers to be paying very close attention to both. This is true for software demos, as in our specific examples, but no less so for any other type of visual process demonstration (and doubly so for a *musical* demonstration: consider how confusing it would be to have a background soundtrack playing while you tried to demonstrate chord progressions on a guitar or piano). More generally, with any and all video genres you will need to consider whether adding background music (throughout or at given moments) will *enhance* the overall viewing experience, by helping to convey a certain mood, or will instead *detract* from the experience by clashing with other audio that is important to the substance of the video, or in some other way distract the viewer from what you are seeking to convey.

For this reason, some of the best candidates for background music are *nature* videos, where there is typically no 'diegetic' (that is, in-scene) human speech at all for the music to distract from; and where any recorded natural sounds can be muted entirely or quieted in volume so they mix nicely with the background music. Let's take as an example our own nature-video sequence, the "Water Wonders of Iceland" sequence which we crafted several chapters ago from a number of brief iPhone clips shot on vacation there. Here is the video track as we last left it:



As we may recall, this video sequence joins together brief clips of three different waterfalls and then a volcanic geyser; the two crossed areas indicate where we created dissolve transitions between the first & second and second & third clips; and we have also slowed the 2nd and 4th clips to half of normal speed, because a) they looked good in slo-mo, and b) this led to more balanced duration among the four clips. Now, as we can see visually from the wave-forms displayed here, each of these clips already has its own captured sound track (technically 'diagetic' or in-scene, and in fact captured at the same time as the video), which for the most part is simply the roaring sound of the water itself. The only exception is the fourth clip, where we can see much more variation in the wave-form: what we have there a combination of the geyser's own sound with the appreciative exclaimations of watching spectators who are not shown in the actual video. But while this last clip contains human vocalizations, we do not hear individual words but simply a collective, emotional response of awe and delight, which very nicely underscores the captured visual image itself. All of this audio has been captured at a pretty healthy volume as well, which we can already see from the height of the wave-forms themselves, but can verify more precisely if we open up the **Peak Meter** panel, and watch it as we play the video sequence:

(cont'd next page)

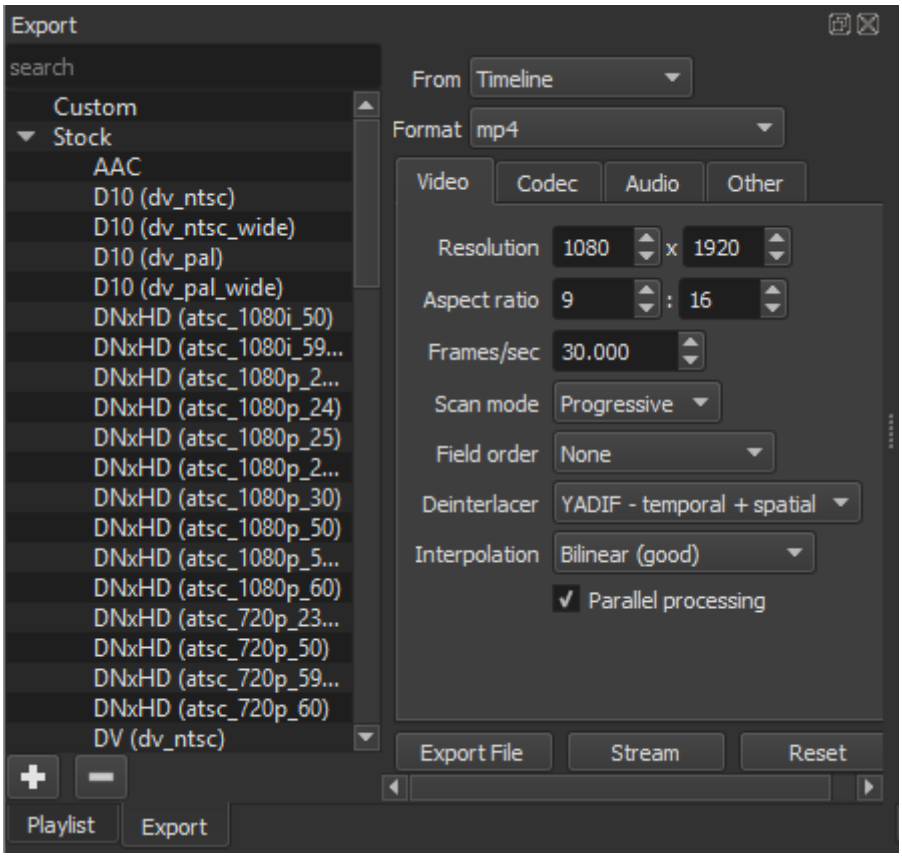
Of course, if the raw captured audio happened to be quieter than this--and we wanted it to be more prominent in the final video product--we could always boost the volume (or gain) by adding a **volume filter** to one or more segments or clips. But as always with sound recording, having a nice healthy volume to start with means that you've captured the audio *detail* -- the texture, nuance and resolution -- very well; this would likely not be the case if your captured audio had half the volume indicated here, and you simply doubled it with a gain control.

Mixing

All in all, then, we might conclude that this particular video sequence already 'speaks for itself', and is perfectly effective in conveying its intended impression without the need for any supplemental sound at all (at least musical accompaniment; we will consider a voice-over narration separately below). Nevertheless, with the ease of digital editing in tools like **Shotcut**, there is no reason not to experiment a bit, and *see* if maybe this sequence could be further enhanced with the right musical background. We would certainly not

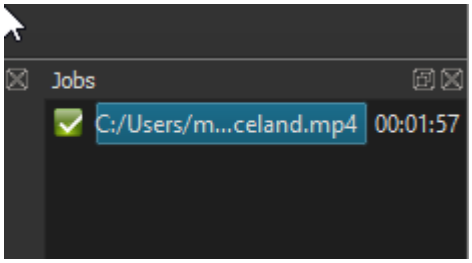
want to lose (that is, mute) the captured 'natural' soundtrack entirely, but we can experiment with mixing it in at various volume levels with an imported musical track.

The first thing we'd probably want to do, though, is to save the video segment as is--I had already saved it as a **Shotcut project file**, but I mean **export** it as a regular-format video such as mp4 or .mov. This accomplishes several things. First, we would then have the video in its current form, with all the sequence-editing we've done but still just the 'natural', raw sound-track, in case we later decide that any audio augmentation wasn't a good idea. Second and more immediately, exporting this video now will yield a single, un-segmented video, which will make it far easier to mix the audio for this entire track all at once, or define new segments if that is appropriate for applying audio filters. Shortly we will see what I mean by this. Here we will hit the "Export" button at the far right end of the main Toolbar, which invokes the following panel:

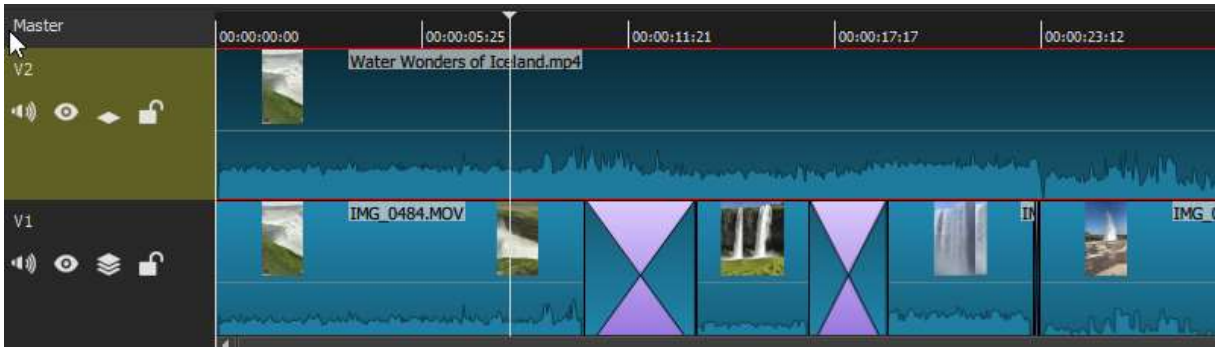


The range of export format options and other choices made available by **Shotcut** is truly enormous, and really requires its own chapter to explore in any depth. For now, suffice it to note that, were we to click the "Format" dropdown shown here, it would open a list of video file formats, in alphabetical order, so long that it extends well above *and* below the entire height of my computer screen--and that only covers the format options with names beginning "i" through "p"! It would be surprising if most **Shotcut** users even knew that most of these formats existed, let alone their various pros and cons. Fortunately, here as everywhere that **Shotcut** presents you with a wide range of often esoteric options, it also *defaults* to a vanilla, middle-of-the-road option that is generally safe to simply accept if you are a beginner. In this case **Shotcut** has suggested an mp4 format for export, with a resolution, aspect ratio, frame-rate and other parameters which are largely inherited from the original raw iPhone clips. if I had instead assembled this sequence out of video clips from disparate sources, with differing format parameters, **Shotcut** would have to make a best-guess 'average' of these in its default suggestions for exporting, and it *could* be worth second-guessing some of those choices. But not in the present case. We will simply accept the defaults, and hit the "Export File" option. This will evoke the standard file explorer allowing us to choose a file name (we'll go with "Water Wonders of

Iceland.mp4") and a saving location. At that point, when we hit Save, it will queue this export as a Job, opening the **Jobs Panel** to show us this job and its progress (much as we saw for the 'Edit-friendly' file conversion a few chapters ago). Because this video sequence is so short, and not especially complex in its parts, the export process takes under two minutes to complete, as shown here:



Now, if I double-click on this completed item in the Jobs panel, it will open it in the Source window and immediately begin playing, looking just like it did in the V1 track above because it is the exported rendition of that track. But remember that it's *not yet part of the project* and available for further editing! To make that happen, as with any other imported asset, we would need to add it to the Playlist and from there to the Timeline. Here, however, we need to decide whether we actually wanted to add this mp4 clip to the *existing* Iceland project, or instead close that project and open the exported file in a wholly new project. In the first route, our Iceland project would become cluttered and possibly very confusing, because we now would have multiple versions of the same media present: in our **Playlist**, the newly-created "Water Wonders of Iceland.mp4" would now be the fifth item, along with all of the individual iPhone .mov clips that went into creating it. And as for the **Timeline**, simply using the Add/Append (**+**) button in the Timeline toolbar would definitely *not* be a good idea, because doing this would append the new mp4 clip to the end of the same sequence as it is already layed out, in segments, in the single V1 timeline--so if we then played it, in essence the same movie would play twice. Instead, we would need to create an *additional* video track (Menu > "Add Video Track") and then, with that new track highlighted, append the mp4 clip into *that* track at the zero point. The result would look something like this:



Really, the only reason I've pursued this route is to highlight the benefits of our **Export** exercise itself. The *content* of these two tracks is identical, as far as the player is concerned, even though they look very different in the Timeline. In case you're wondering, what would actually appear in the play window would be the *top* track (V2 in this case); **Shotcut** will always give precedence to the top-most video track, wherever there is content present in it--which is why, two chapters ago, we put our floating-text-caption segment in a video track *above* the main one. But just as you can mute an entire track, you can also **hide** it, by clicking the little eyeball icon just to the right of the speaker (mute) icon on that track. If we hid the V2 track here, we would see the original V1 track beneath it play instead--yet there would be no visible or audible difference whatsoever: the content is identical. However, from an *editing* perspective the difference is considerable: as you can see, the V2 track is a single, seamless whole, with none of the segmentation, transition markers or different clip names that are present in V1 below

it. Indeed, had we previously added a floating text caption to the V1 waterfall sequence, as we saw several chapters before, the *effect* would be preserved after exporting, but all visible signs of that editing work would have similarly vanished in the V2 track. As noted above, having a single, seamless clip as the V2 track means we can now add audio filters to the entire track, or segment it differently than V1 was segmented.

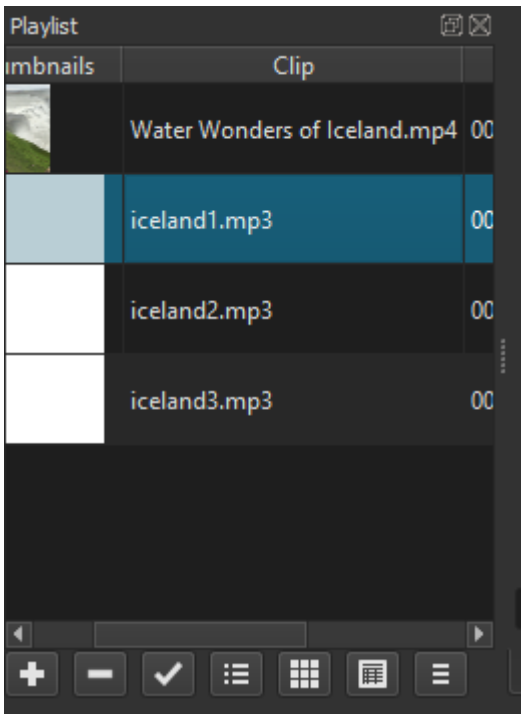
So would we *want* to work with these two different artifacts--which can really be thought of as different stages in the same project--displayed simultaneously in the same project view? One possible benefit of keeping V1 and V2 here side-by-side (or over-and-under) is that V1 *does* preserve the visual markers of exactly where one 'shot' begins, transitions and ends; and that could *potentially* be useful when adding in futher effects (audio effects, for our present purposes). But we would want to make sure we were *applying* those new effects or additions to V2, not to V1. Indeed, we would need to be very aware that all of the edits we've made in V1 are still 'writable', or changeable -- intentionally but also unintentionally -- as long as that track is open in our project and has not been **locked** (using the lock button to the right of the mute, hide and composite buttons in the track controls). Perhaps we'd want to keep V1 unlocked because we *want* to be able to make adjustments to the original editing--move, expand/contract or delete segments or transitions, delete, modify or add filters, etc--but if we did make any such changes, then V2 will no longer be 'current', and we'd need to delete and re-export it anyway. All things considered, keeping V1 open and unlocked concurrently with V2 seems to be a dangerous game. Certainly some of that danger goes away if V1 is locked (and stays locked); but simply displaying it still means that **Shotcut** is consuming a good deal more memory--all of those edits have to remain loaded separately into memory, whereas in V2 they are simply part of a single encoded file. And the UI is simply more cluttered, as well: there is that much less space to add, for example, multiple new audio tracks, as we will want to below.

With all of this taken into consideration, I will simply delete the V1 track (I can right-click in the control portion of the track, and from there select "Remove Track"); and I will also remove the original .mov iPhone clips from the Playlist, leaving it less cluttered too. At this point, **Shotcut** will automatically rename the V2 track to V1, as it's the only remaining video track in the project. The effect now is exactly the same as if I had simply started a new project and opened the mp4 file into it. I will go ahead and **Save** the project, in its new state, with a new name too. My *old* Iceland project, with the still-segmented V1 track, is still available as a separate **Shotcut** project file if I ever decided I needed to revisit it.

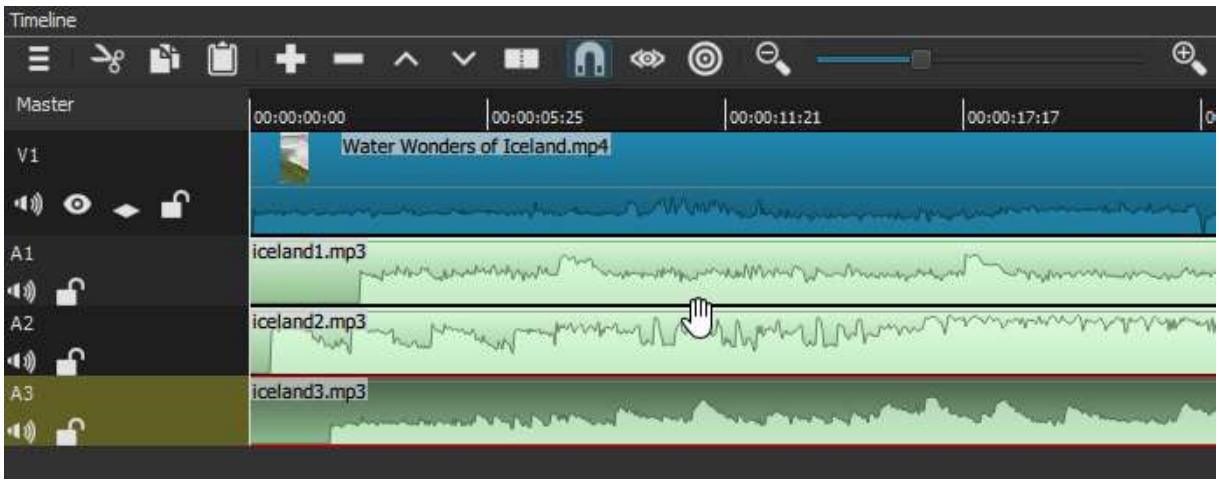
We are finally ready to look for some suitable background music to add to our video. Unless you are an accomplished musician with good digital equipment, you will likely find your audio rather than create it, and these days you will likely find it online. You can do a google search for 'free background music' or similar terms, perhaps specifying a particular mood as well--for our water-wonders video we want instrumental music, certainly, and something a a bit soft and dreamy. Many sites offer music, sometimes in a wide range of genres, available for free download under a Creative Commons licence, as long as it's not ultimately put to a commercial use. You may also have a particular song or song segment in mind already, in which case you can likely find it on Youtube, Spotify etc--where you can also do a general search for 'dreamy instrumental' or whatever style you want. Sites like these are set up for streaming, and typically do *not* facilitate downloads as such, but there are any number of free software tools which let you record directly from your computer's audio output (aka sound card), allowing you to capture, as a high-quality .wav or mp3, really any audio that your computer can play. I find that Audacity is an ideal tool for this kind of recording, as it is for general audio editing. Such technical questions aside, if you take this latter route (grabbing an audio stream online) you should be aware of potential copyright issues, although these are easy enough to avoid. YouTube itself, for example, may not allow you to upload your own finished video (if that's even your intent), or may flag it for removal or simply send you a notice, if it detects you've used copyright-protected audio material; but this typically only happens if

you're using *major-label popular music*. So unless you are specifically setting out to create your own 'music video' rendition of your favorite pop song, you are unlikely to encounter this issue (and even then, millions of Youtubers before you have done precisely this and gotten away with it--either because the music is old enough that it's effectively if not legally in the public domain, or because they've very slightly sped up or slowed down the play-speed of the music and thus gotten around the search algorithm). But the present discussion will assume that we're using music as background for our video, not as the foregrounded subject itself.

As it happens, I have found three different dreamy, instrumental tracks that I think are worth trying out as potential background music for my Iceland video. The video itself is just over 30 seconds long; each of these audio tracks is a minute long or longer, so among other things I'll want to see how well the various segments (bars, measures) of music in each piece map onto the video. There is no good reason why I can't import all three tracks into my project, and audition each one in turn. To do this I follow essentially the same process I would for importing a video clip: "Open" will simply cause **Shotcut** to begin playing the audio file in the Source window (we'll hear the audio but of course see no video, just a blank white screen); I'll then need to add this audio track to my **Playlist**, and from there add it to the **Timeline**. Here is my Playlist with the three audio tracks added (along with the mp4 Iceland video itself). I have also selected the first audio track, so it's blue-highlighted:



If I were simply to hit the Append (**+**) control in the Timeline now, with nothing in the Timeline itself except the V1 video track, **Shotcut** will actually append the "Iceland1" audio segment *to the end of the video*; that's certainly not what we want. Instead, we need to use the dropdown from the Timeline Menu control to "Add Audio Track", then select this new A1 track, and then hit the Append control. "Iceland1.mp3" is now added in to this new A1 track, starting at the zero position. I want to add the other two audio tracks in the Timeline as well, for easy comparison between them, so I will create two more audio tracks and add "iceland2" and "iceland3" to these. In both cases, I need to first make sure the correct clip in Playlist is selected (double-click on it for good measure), and then make sure that the correct *audio track* is selected, so I don't end up appending the audio to the end of another audio track (as always, though, it's easy to make such mistakes at first, and if you do, simply hit "Undo" and try again). One all three are correctly added, my Timeline looks like this:

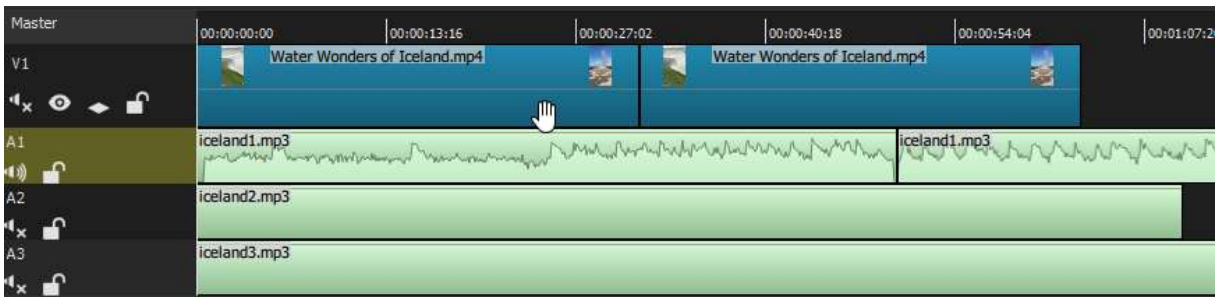


As we can see, the Timeline is getting pretty crowded, between V1 and the three new audio tracks. One way I've eased the crowding a bit was to right-click in the control area of my V1 track and select "Make Track Shorter"--though to be honest the video track only became about 25% shorter when I did this. The audio tracks really cannot be shortened at all, because as it is they are only tall enough for their wave-form to fit at its highest points. To be fair, though, it is rare that you would need to edit with more than four tracks at any one time: once I've chosen the one background-music track I want to use, I can remove the others; this will free up space for, say, a separate video track for captioning purposes, as we've seen previously, or an additional audio track if I wanted to record a voice-over narration (which I won't for this video, but I will for a later example).

Several other things are immediately clear from looking at the Timeline now (or would be if we could see the whole thing). One is that the three audio clips don't all start at the same time: the A2 track begins several seconds before A1 or A3. I might want to trim (delete) a bit of A1 and A3 so they were more comparable --but then it isn't yet clear that I even want to *use* the first bars or segments of any of these audio tracks. It happens that all three music tracks follow a somewhat similar pattern, of stating a certain 'theme' for the first 20-30 seconds and then re-stating that theme with more instrumentation. Perhaps the fuller instrumentation will work better, particularly blended with the 'natural' or diegetic audio I intend to keep. Since the V1 video itself is only about 30 seconds long, right now all three audio tracks extend out far beyond the end of the video (and the above screenshot). In fact, one thing I could do to compensate for this, just for my initial audition purposes, is to paste in a *second* copy of the video right after the first in the V1 track (by selecting that clip in my Playlist, then selecting the V1 track, then hitting 'Append'). Now the running length of video is nearly as long as the full audio tracks.

We are *almost* ready to begin our auditions for Best Soundtrack, though one crucial step remains: if I were to play this project as is, we would hear all three of the musical tracks *at once*, on top of the diegetic audio of V1. That would definitely sound awful! Fortunately there is a **Mute** button for each audio track as well, so I will mute A2 and A3 for now, and just listen to A1. After a brief listen it also becomes apparent that the natural soundtrack of V1 is simply too loud, as recorded, to hear the music well; so for now I will mute the V1 track too (later we can add a volume filter to try to find an appropriate mix). I should note here that, unlike muting through a filter or by switching off the audio track in Properties, when we use the track's own Mute control the audio wave-form *does* disappear for that track. The result at this point looks like this, in the Timeline:

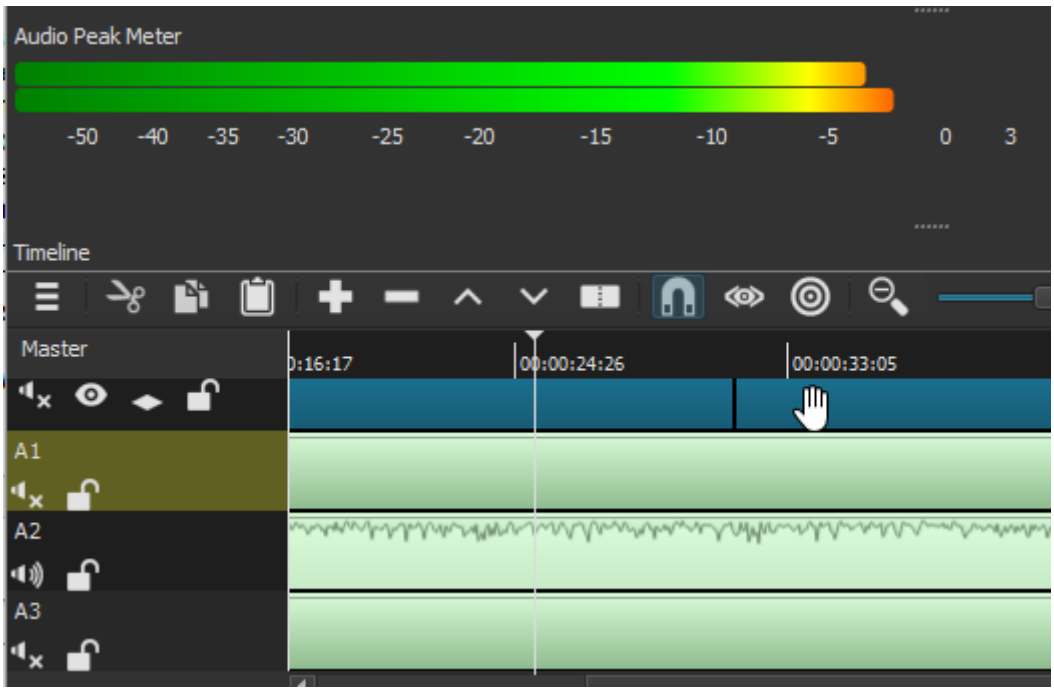
(cont'd next page)



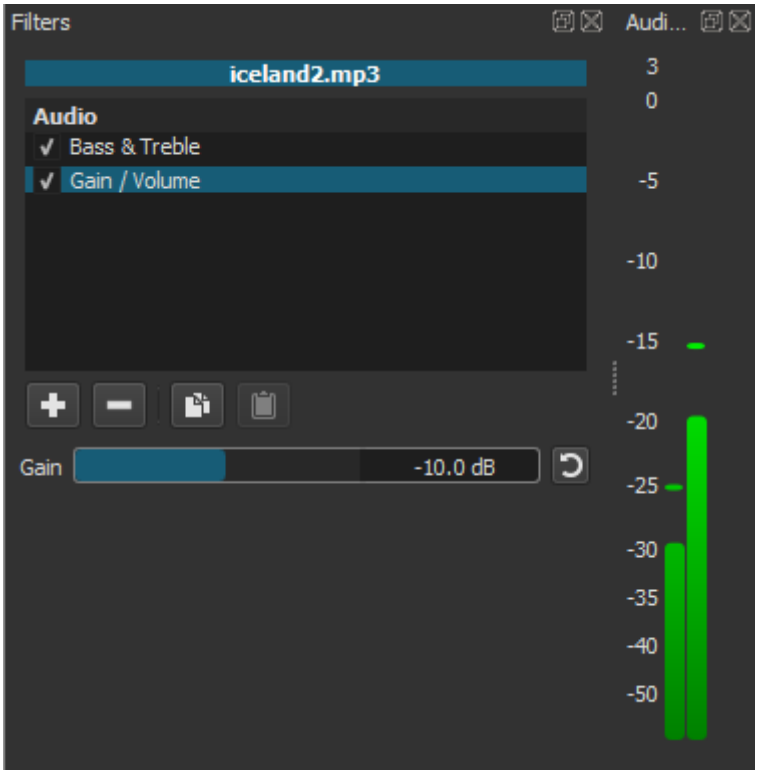
Here we can see that only the wave-form of A1 is visible (and that's all we hear upon playing); notice how the Mute icons for the other three tracks are now different too, further indicating that each of these has been muted. We can also see that I've **zoomed out** the entire Timeline view considerably, so that we can now see both copies of the video, pasted end-to-end, and the end-point of at least the A2 audio track, which is slightly shorter than A1 and A3. Finally, you may notice that I'd made a **split** at a certain point in the A1 track. This is to mark a point of interest in that audio track: after an initial statement of the musical theme (which I believe is four bars long, but in any case about 20 seconds), there is another 4-bar bridge or transition--you can see the distinctly different wave-form shape--and finally a restatement of the original theme, with more instrumentation, beginning where I've placed that split. I actually really like how this first piece works with the Iceland visuals, and especially the restated theme, which combines the opening piano with some guitar and light percussion. Unfortunately, this whole piece is divided into roughly 20-second segments, while the video itself is 30 seconds--too much of a mismatch in length, at least for now. So I will move on to consider the other music tracks in turn.

The A2 track presents an interesting case: I really like this music too, especially the keyboard intro; but this time when the larger instrumentation kicks in, it is *really* heavy, especially at the bass end. The wave-form itself should have clued us into this: you can see below how it's crowded close to the top of the track (compare to the previous screen-shot, of the A1 wave-form), which means that when Audacity recorded it, it was probably already 'clipping' the loudest points so as not to be too distorted. But just to verify this, I've dragged **Shotcut**'s Peak Meter down closer to the Timeline--where it also helpfully assumes a horizontal rather than vertical orientation--and this really tells the tale: whereas our peak-meter reading of the original captured audio in V1 (see the second screenshot in this chapter) showed a near-ideal level of about -17 dB, here the meter goes all the way up past -5, turning a warning yellow and then orange at the very tip. This audio is simply far too loud, and definitely distorted for much of its length:

(cont’d next page)



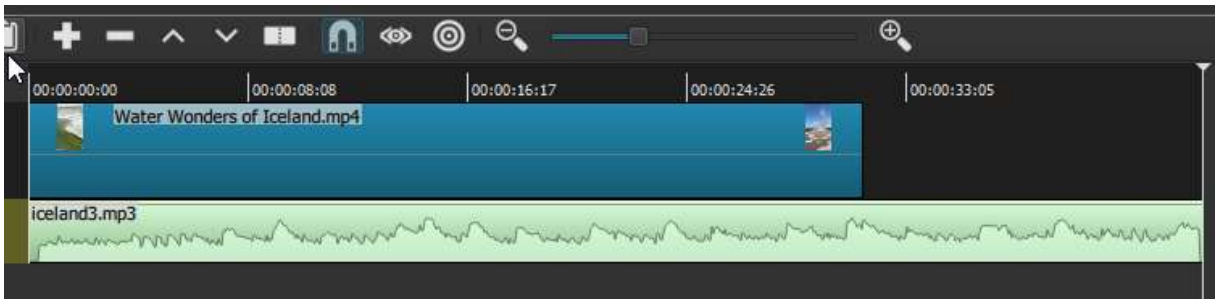
What to do about this? Experimentally, I applied both a **Volume/Gain filter** and a **Bass/Treble filter** to this track (by selecting the A2 track, opening the Filters panel, searching among Audio filters for the right ones, and double-clicking on each). Using the controls provided by these two filters, I dialed down the Bass specifically and the overall Gain more generally: as I did the latter, I could see the Peak Meter move downward in real time, to a level that looked much less scary (notice I have once again re-docked the Peak Meter to make it more convenient for this screen-shot):



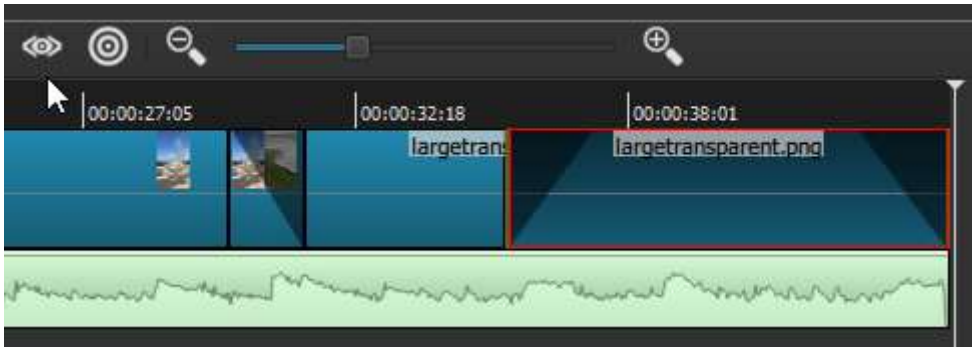
So now the actual peak levels seem ok. Unfortunately, while the *volume* of the A2 track is now acceptable, the *distortion* remains, when I play it: this distortion is an artifact of the recording itself, and can't really be edited out after the fact. Now, if I were really motivated, I could go back to Audacity and try to re-record what I'm calling "Iceland2"

from the original stream online (assuming I could even find it). However, I have my doubts that this will be fruitful: after all, I had recorded it at exactly the same input gain as the other two tracks, and yet it ended up far more 'hot' or distorted; this suggests to me that the original track is much louder than the other two, has far more bass, and is probably already too distorted for me to use, even with up-stream editing in Audacity. Probably not worth the effort. So I move on to consider the A3 track instead.

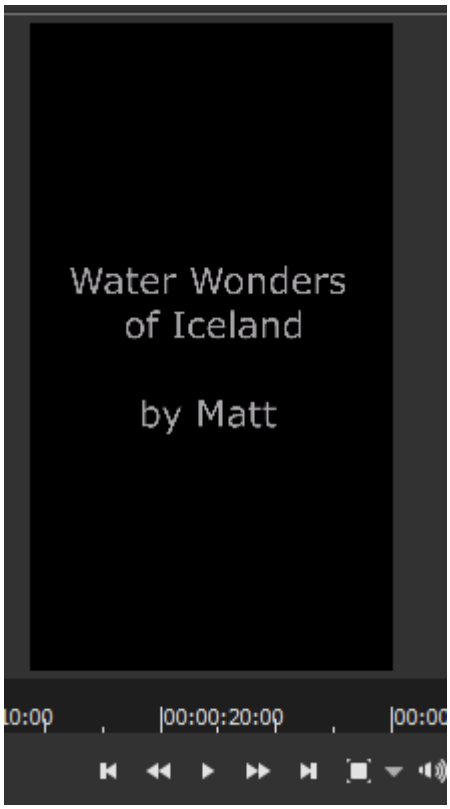
Just as in Goldilocks, it turns out that the *third* track is just right (believe it or not, I didn't plan this in advance: it just worked out that way). "Iceland3" happens to be structured in segments much closer to the same 30-second length as the video clip; and while the second such segment (or perhaps the third) does eventually become too heavy with rock-style instrumentation, so that like the second track it is simply too 'hot' and heavy to use, the opening 30-second section is just about perfect, in both length and mood. So I will delete the other two tracks from the Timeline, as well as the duplicate length of video clip from V1, since I certainly don't want it in the final product. I will also make a split in the A3 track itself at a point about 4-5 seconds after my video ends, and delete the remainder (since I know I won't use it). My timeline now looks like this:



Why don't I cut off the Iceland3 track right at the end of the video itself? Because something about the shape of the music itself, at this point, has given me an idea: particularly with the right audio fade filter applied, the final seconds might be an ideal background for a closing title or credit sequence. It hadn't even occurred to me to add one before, but video and sound editing in general can be driven, to a surprising degree, by such happenstance. Technically speaking, we know from past chapters how we might add a closing credit sequence: We will import our transparent background image into this project, append it the end of the video clip, and add a Text filter to it -- in this case the text will be "Water Wonders of Iceland, by Matt" (we're keeping things very simple, for illustration purposes). After some experimenting I decide that, rather than use a dissolve-transition from the end of the video, I will apply a fade-to-black filter to a short segment of the end of the video (which is already a near-still at the end of the Geyser spout), follow that with a brief blank segment of my transparent-background image (which will read as black); then follow *that* with a longer segment to which I've applied my text filter, appropriately sized and positioned, and my fade-in and fade-out filters. The result in the Timeline now looks like this, zoomed in a bit so we can see more detail in this ending sequence:

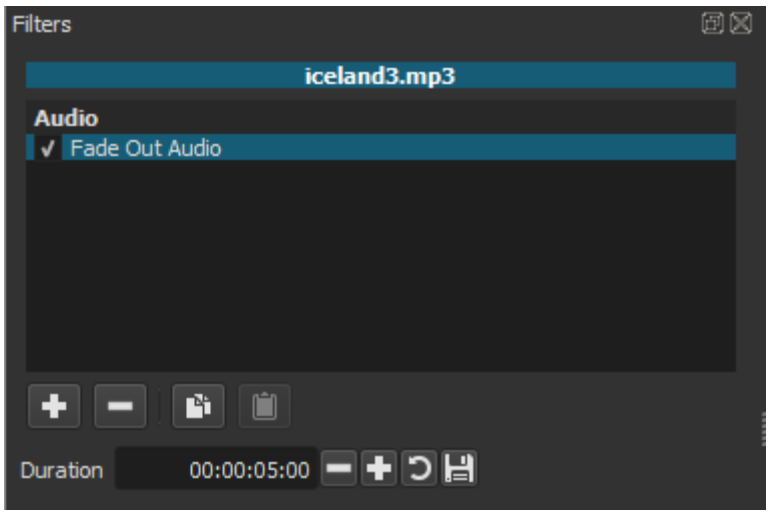


Here we can see a total of three 'fade' filters applied: the first (fade-out) to my brief end-segment of the video, and the second and third applied to the last and longer transparent-image segment, which is the one that carries my text filter. In all three cases I have lengthened the duration of the fade filters, from the default of 1 second to about 1:20 seconds. If I now back up my playhead to the mid-point of the final filter, I can see my end-title/credit text in mid fade-out on the screen itself:



At this point we have a fairly pleasing ending sequence -- except that our audio track just carries the full length of the end-credit sequence and then ends abruptly. We need it to fade out instead. In general, just as with the video track, for audio too a filter must be applied to a defined segment, or else it will apply to the full audio track. But that is not an issue with a **Fade-out Audio** filter, because even if we apply it to the entire A3 audio track, it will only take effect working backward from the end-point. All we need to adjust is the duration of the fade: by default, as with a video fade-out the duration is 1 second, but this is far too short for our purposes (or probably any, for an audio fade). After some experimentation I bump up the duration to a full five seconds, which amounts to a very slow fade across the whole duration of our end-credit visual. For whatever reason, the **Shotcut** team has not yet added a visual correlative to this audio filter in the Timeline, so there is nothing additional to show you except for the fade-out filter itself applied in the Filters panel:

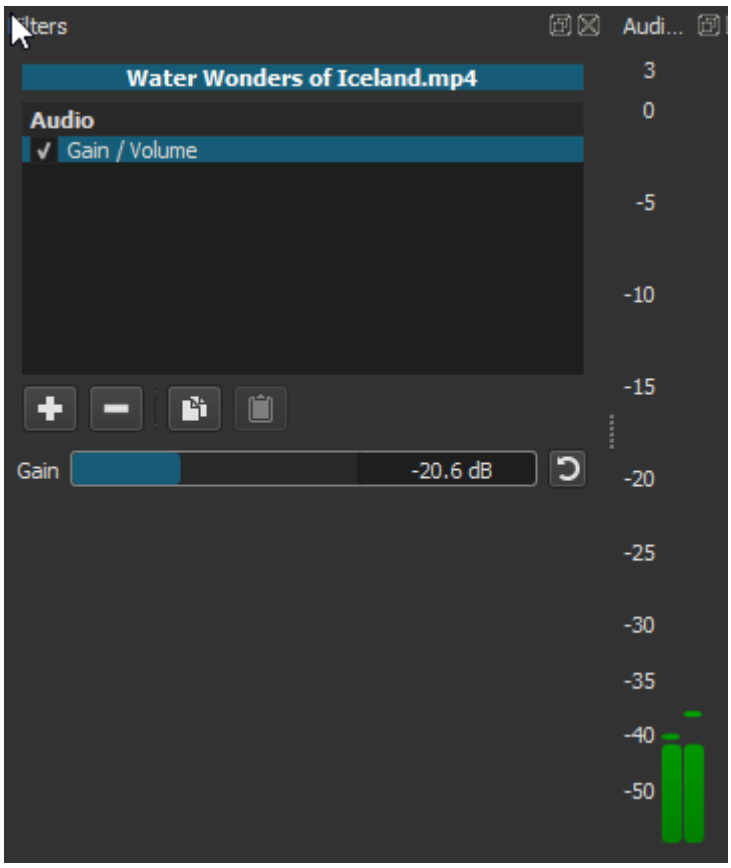
(cont'd next page)



So are we there yet? We *could* be: the whole point of going through this kind of careful audio selection and editing is to end up with a musical sound-track that 'fits' so well with the on-screen images that the whole effect appears 'natural' and complete. Yet we need to recall here that the video clips I sequenced together had their own, *literally* 'natural' sound, which we have simply muted entirely during this soundtrack-audition exercise. What we want to do, then, is make sure the mp4 video clip itself is selected (either in the Timeline or in the Playlist), and apply a **Gain/Volume** audio filter to it--not a Fade, as we just did to the Iceland3 music track, because we want to control the volume along its entire length. As we already saw when we tried this on the Iceland2 sound track, the Gain/Audio filter gives us a slider control which, as we move the blue bar from its default position either left or right, changes the Peak Meter signal in real-time so we can judge the effect of our changes.

If you go back for a minute to review the second screenshot of this chapter, you'll recall that the 'natural' sound for these video clips was right around -20 to -17 dB: very nice when heard on its own, but much too loud in conjunction with the musical soundtrack we've chosen. So I want to back it down to less than half of that volume. Exactly how much will be a matter of experimentation, as with just about every other aspect of audio-video editing. Eventually I settle on a volume (or gain) of a bit under -40 dB, as shown here (bearing in mind that for this screenshot I have muted the separate *music* track, so it doesn't register in the Peak Meter):

(cont'd next page)



When I now unmute the music track, and play the project, I hear both of the sound tracks together in a 'mix' that is really quite nice: the music track is distinctly audible, establishing a serene and gently upbeat mood to the whole piece (and helping to further cement the unity or coherence of the four separate iPhone clips--a not-inconsiderable benefit of a musical soundtrack); while just enough of the natural sound itself is audible as well to give us some sense of the actual force of these natural water wonders.

Of course, as this is a text-and-screenshot document, you're going to have to take my word for the qualities of the actual audio-visual work I have produced; and indeed, even if this were instead a video demonstration so you could see and hear everything for yourself, you might decide at any step of the way -- right back to the original selection and arrangement of clips, let alone the sound selection and mixing -- that you might have made different choices. Which is entirely fine, and more than fine: let a thousand audio-visual flowers bloom! The point is that now, hopefully, you begin to have an idea how to go about it for yourself, using this excellent editing tool.

Audio and Video (III): Recording a voice-over narration

We've now explored both how to clean up a pre-recorded voice narration, when this has been captured along with your original video, and also how to choose, add and mix in a 'found' audio track intended to establish mood, most typically a music track. Of course, if you were a musician you could record your own musical sound-track (doubtless with more specialized sound-recording equipment and software), in which case it would presumably be tailored more closely than 'found' music to the needs of your own video footage; but the resulting audio file would still need to be imported into **Shotcut** and probably mixed a bit for compatibility with other audio within the project, before the whole thing was exported as a finished video file. There is, however, one major audio scenario we have yet to explore: using **Shotcut** to record a voice-over narration for a pre-existing video, which you've either recorded yourself or perhaps even found elsewhere. And because of the *kinds* of video that this would make sense for, it's entirely conceivable that you might also add a musical background track in this case too--albeit one mixed in judiciously with your voice-over, so the two enhanced rather than interfered with each other.

As we noted a few chapters ago, a voice-over narration added in-after-the fact (ie, not recording with the video) is typically a very different animal from voice narration/commentary/reaction captured during the video recording itself. After-the-fact narration is generally much more polished, considered, rehearsed to a certain extent (at least mentally, if not literally) and sometimes even scripted out in advance. In many cases, we are simply choosing one or the other kinds of narration depending on what we want to achieve or emphasize in the video; and of course in some cases the two (after-the-fact and 'on-camera' or diegetic commentary) can be found mixed together in the *same* video: perhaps at a political demonstration, performance or other public event where on-camera reactions, interviews and comments are intermixed with the videographer's after-the-fact informational commentary or narration. For cases like that, the present chapter simply supplies the last missing piece for such an audio mix. Indeed, *whatever* kinds of audio you are mixing together for your video, we still need to look specifically at how you can use **Shotcut** to shift emphasis back and forth from one audio track to another, depending on the needs of the moment.

It has to be noted, though, that there are certain kinds of video where after-the-fact narration works extremely well--much better, at least for informational purposes, than in-the-moment narration--and other kinds where it is simply not practical at all. In earlier chapters we've looked in some detail at video demonstrations, and seen that there can be quite complex and tedious work involved in 'cleaning up' the captured narration, with all of its pauses, filler-words, back-tracking, side-tracking, random external noises, etc (unless we choose, as many YouTubers do, to simply leave all that in). So why not simply forgo all that (or mute it), and replace it with a nice, fluent, after-the-fact narration? The answer is that this is nearly impossible to pull off. This is probably easiest to grasp in the case of a demonstration involving a musical instrument: how exactly would you mix together the audio of your in-the-moment playing (even if you weren't 'talking though' your fingers' placement and motions at the same time), with the audio of an-after-the fact commentary? This is why, when people absolutely *have* to correct or enhance the informational flow of their music demos, they do it with floating text captions, not with over-dubbed commentary. But this difficulty actually applies much more broadly to video demos of almost *any* kind, at least where you (the videographer) are the one performing the action being demonstrated. It is simply much easier to 'talk through,' to narrate, actions that you yourself are performing, *in the moment*, than it is to come back after the fact and try to narrate those same actions in perfect sync with what the viewer sees on the screen--even if that's just a cursor moving over a computer screen, typing out code or fiddling the controls in a UI. If you are

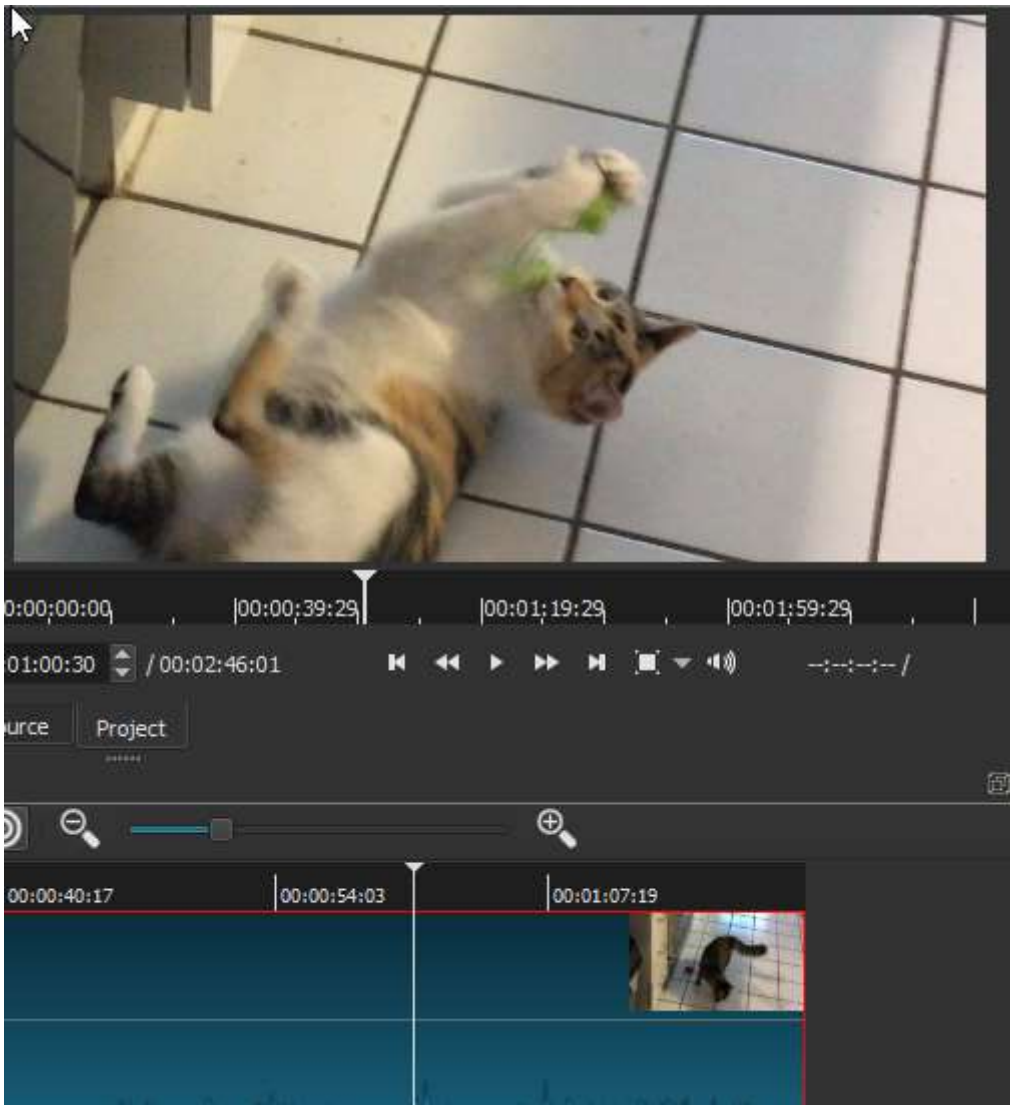
making your own video demos, *of anything*, you simply have to be as focused and deliberate as you can be, while you talk through what you are trying to demonstrate and/or explain. And/or do what you can to clean up the verbal flow in post-recording editing. Or else simply not worry about it. In any case, if you are exclusively a video demonstrator of this kind, then the present chapter is not for you, any more than the previous chapter on adding mood music.

But there are other kinds of video scenarios where focused, deliberate, flawless in-the-moment narration is itself difficult or impossible, and these are precisely the ones where after-the-fact voiceover narration is most useful. If we are video-recording the actions of other people or things (animals, moving objects, violent weather, etc), unless we are in some way capable of *directing* those actions, then pretty much by definition we are witnessing something unpredictable, and we can almost never comment on it with the kind of fluency that comes from rehearsal and review of the material before we speak. If you *can* do this, you are probably a professional sports commentator and are not reading this guide. For the rest of us, there is after-the-fact voiceover narration--which in theory could be recorded using any recording device, simply while *watching* the video footage in question; but is generally far easier when it can be recorded using your video editor itself.

As it happens, **Shotcut**'s method for recording after-the-fact narration for existing video is not the most intuitive you may encounter. There is no button or control on the main toolbar labeled 'Record'; no Record panel; or even a 'Record...' option named as such in any of the menu dropdowns. You need to follow a series of steps which you would very likely never stumble on by accident. Nevertheless, if you follow these steps, as we'll outline them below, you can produce a high-quality voice recording from your computer's built-in or external mic, as you would wih any other recording software. This audio clip can then be positioned, edited and mixed at will with other elements of the video project.

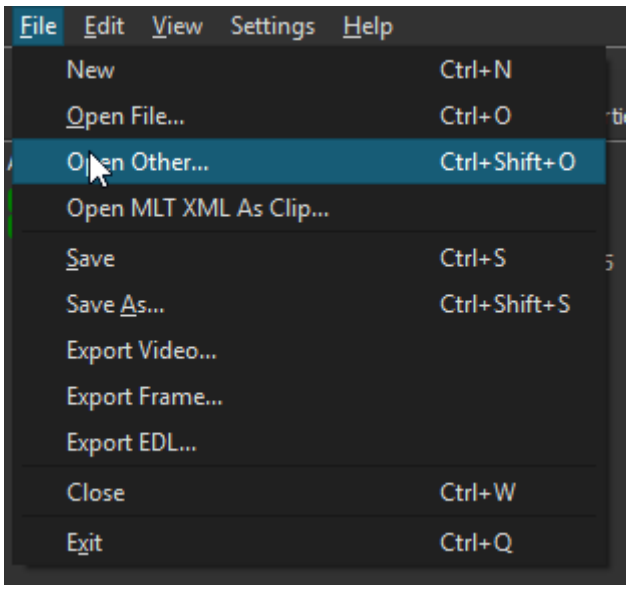
We'll want to begin with some kind of video clip to comment on, and for this exercise I've chosen a 1:20 long iPhone clip of my girlfriend's cat playing. This video offers good material for the exercise because it's long enough for the cat (a 7-month female kitten named Strummer) to do a range of different activities meriting commentary: she follows her videographer/owner around the room, eats some food, plays with her new bird-mouse toy, and fights briefly with her sister. In the following screenshot we see her playing with the bird-mouse, around a minute into the video:

(cont’d next page)

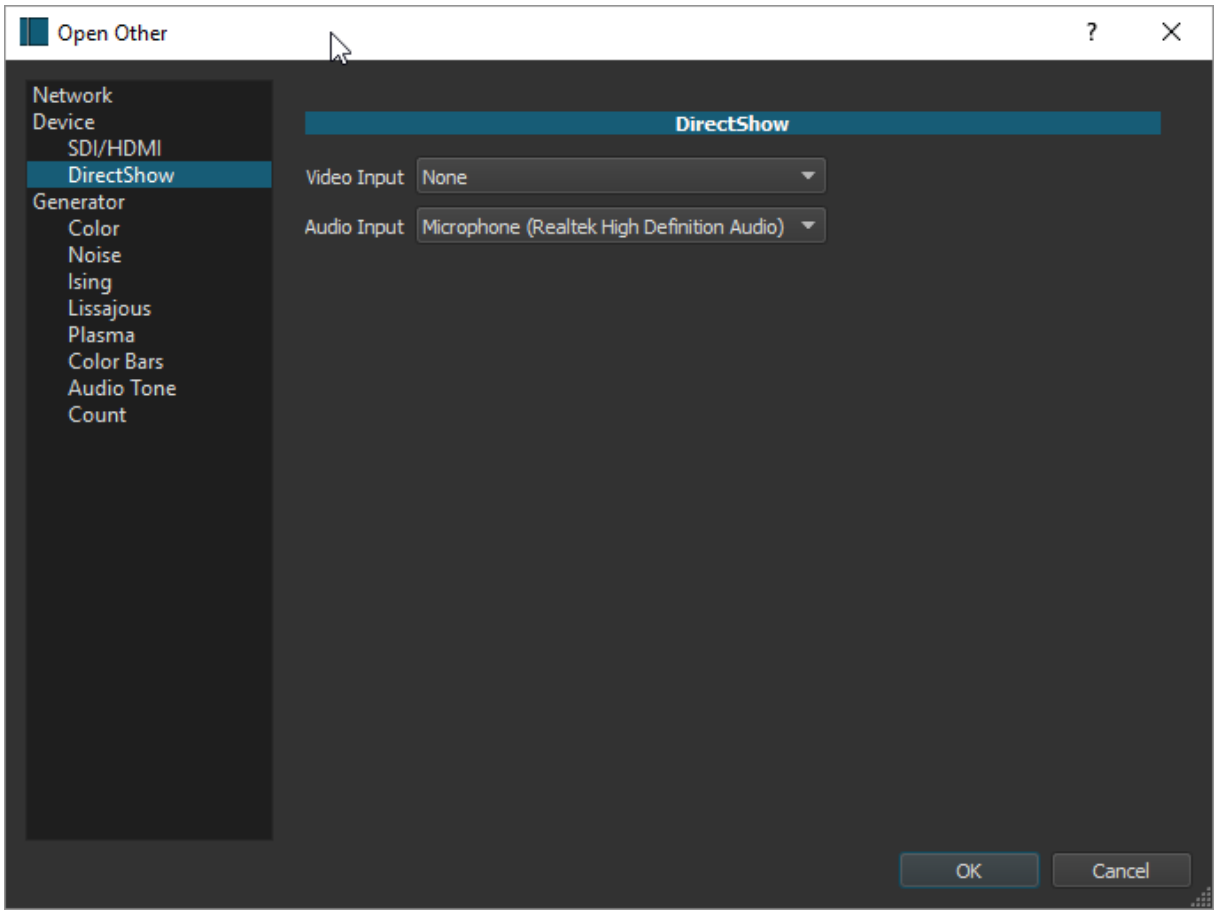


At the very bottom of the video clip, which is already loaded into the Timeline of a new project, we can just see a few audio wave-form spikes: there is very little 'diagetic' audio captured with this video, mostly just some brief vocalizations by Strummer (perhaps her own commentary on her activities). In short, we will have plenty to comment on and little exisiting audio to contend with. The first thing I will do, then, is use my rewind control to move the Playhead back to the start of the track. Now we're ready to begin the audio capture process itself. The first thing we'll do is go the **File menu** at the top of the UI, and from this dropdown choose "Open Other...":

(cont'd next page)



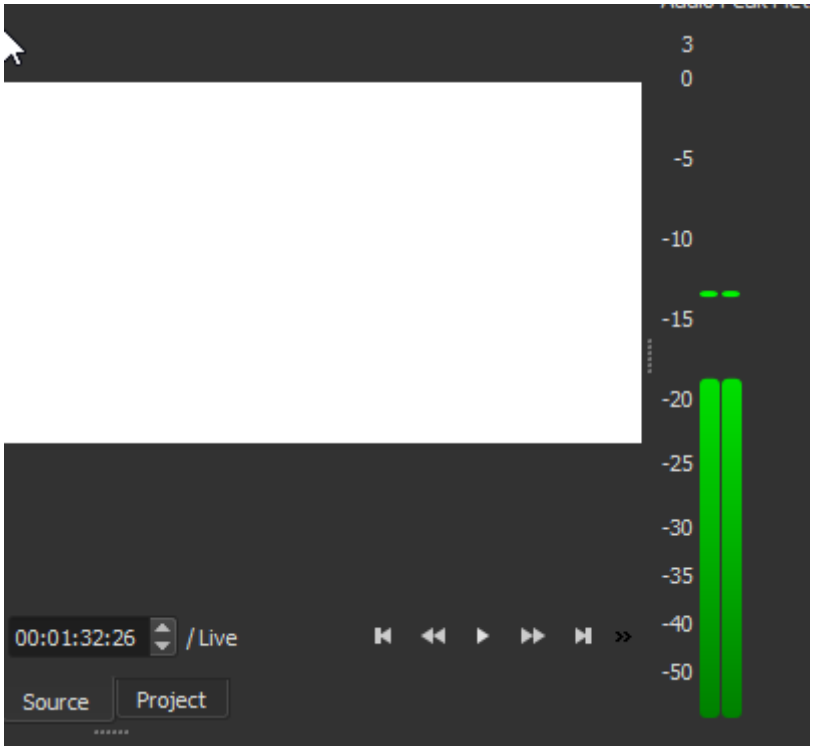
Many of the options in this File dropdown are either familiar already from previous chapters -- Open File, Save, Save As..., Export Video, Close, Exit -- or can probably be guessed: 'Export Frame' will export a single frame of a video as a static image; 'Export EDL' means export a '**E**dit **D**ecision **L**ist' file, which is a video-editing file something like a **Shotcut** project file but shared across a number of different editors, including Adobe Premiere, Avid Composer and Apple Final Cut Pro. But what is 'Open Other...'? When we select it, we get a dialog with what seem at first like a number of quite miscellaneous options:



In terms of the left-hand options, '**Network**' appears to allow capture of video from a remote URL, though I haven't experimented with it. The list of '**Generators**' refers to an

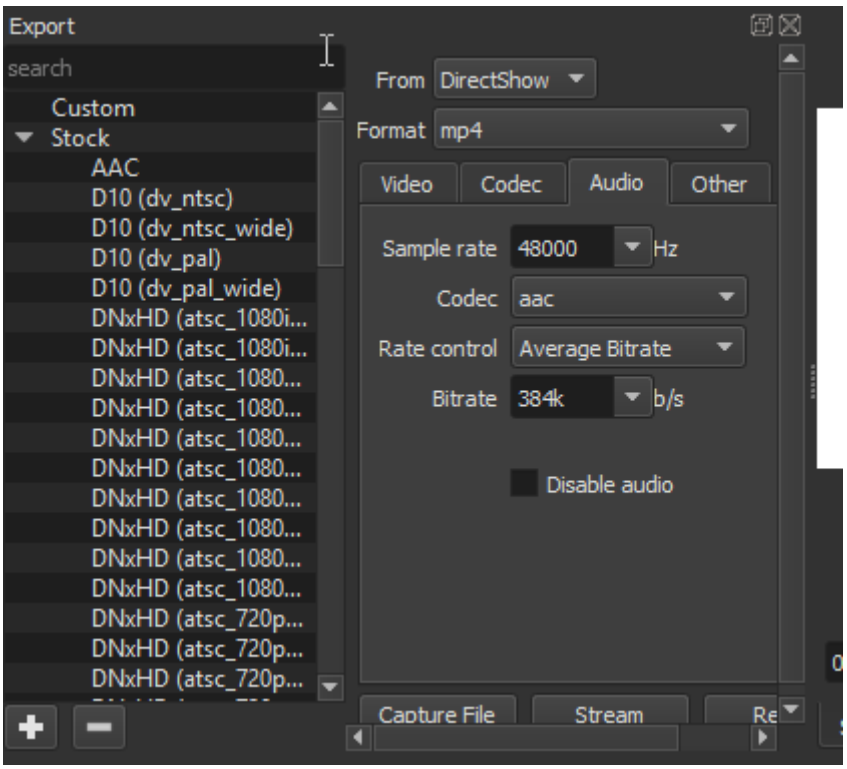
array of what seem related to audio or video filters, except that rather than transforming an existing segment of audio or video track, they *create* new audio or video segments with various properties. These too will be left to the reader to explore. We are interested in the '**Devices**' section, and specifically (on a Windows machine like mine) in 'DirectShow', which is the current built-in Windows multimedia framework and API. SDI/HDMI refer to input protocols for external devices. All of these disparate options, then, turn out to be types of *Input*, or ways of generating new content. By choosing DirectShow, I'm telling **Shotcut** that I want to use something on my own computer as an input device. In the main region of this dialog there now appear dropdown selectors for both Audio and Video recording input. I am not interested in recording new video here, so I leave that selector with its default of "None"; but I am interested in recording audio, so I change that selector to the only other option available in my case, my built-in RealTek mic and the associated sound card (if I had an external mic installed, that option would presumably show up as well). Having asked **Shotcut** to accept audio input from my Mic, I can now hit "OK".

When I do so, I am presented with a blank white video screen (since I've chosen no video input), but you can see from the Peak Meter that **Shotcut** is now monitoring my audio input (I'm speaking aloud here in the shot):



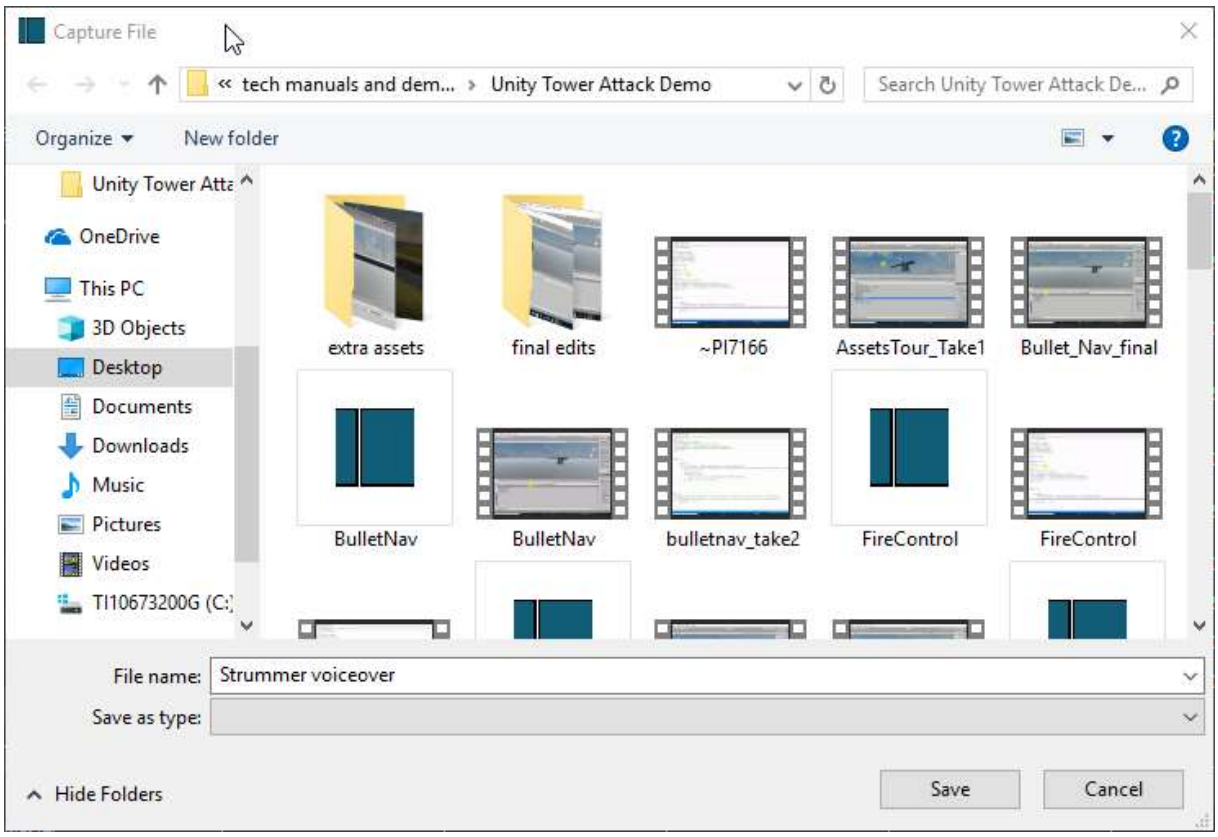
But while **Shotcut** is now *monitoring* my input mic, it isn't yet *capturing* any sound. To do that, I must open the **Export panel**, which shows the following when I flip from the default Video tab to the Audio tab:

(cont'd next page)



We've seen this Export panel before, but now some particulars are different: most notably, two--in the 'From' dropdown at the top, **DirectShow** is specified here as the source, rather than a track or playlist item as in a more typical Export; and at the bottom we can (just) see a "Capture File" button rather than the "Export" button we would normally expect. If I now hit this Capture File button, it will open a standard file browser allowing me to name the audio file I'd like to capture and specify a saving location. Before I do this, though, there's one more crucial step: in the 'Format' dropdown, it has defaulted to mp4--because I've been exporting finished video projects as mp4 files--but in this case we want a more suitabe *audio* file format. From either that dropdown itself (which will give me a very long alphabetized list of formats), or from the search box in the upper right region, I can choose .WAV, which is still probaby the most desirable format for raw audio (although mp3 would work fine as well). Having specified .wav for my format, now I hit "Capture File", and get the expected file-saving dialogue:

(cont'd next page)



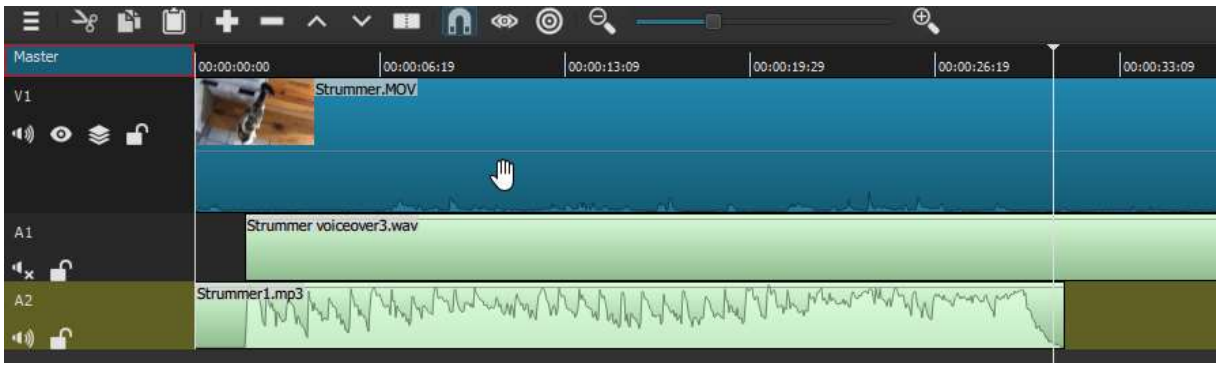
I will call this new audio file "Strummer voiceover"; and while it's unrelated to most other files there, I'll keep the default location which is the folder where I saved my Unity3D demo assets (I have been saving my project files for the present manual in this location too). As usual, it really makes no difference where you save your ouput, as long as you can remember it afterwards! Now, here is a really critical and perhaps non-intuitive point: as soon as I hit "Save" from this dialog screen, I have actually *started the capture process* for my 'Strummer voiceover' audio file; in other words, from this moment on, that file will be a recording of everything I speak into the Mic, until the point where I hit the button in the Export panel which previously read "Capture File", but will now read "End Capture." So as soon as I've hit Save here, I'll want to flip back to my Project view in **Shotcut**, start up the Strummer video, and commence my narration of what I'm seeing on the screen. Even with some practice there will doubtless be a few seconds before I'm ready to start narrating, but we can always trim that out of the captured audio afterwards.

Once we've hit "End Capture", the resulting .wav file will either be queued as a **Job**, if its long enough, or (as in the present case) will be more or less immediately processed and will then begin playing in the Source window. That window will still show a blank white (this is only an audio file, not a video), but we should be able to hear a playback of the audio track we've just recorded. The audio file will also now exist, physically, at the location and with the name we've specified above. As with any other asset, though, to make any actual use of it within the project, we'll need to add the audio clip to the Playlist and from there to the Timeline--and, just as we did with 'found' audio in the last chapter, again we'll need to first create a new audio track and then **Append** this audio clip to that track. Here is the resulting Timeline, with my original Strummer .mov video in the V1 track, and my newly-recorded "Strummer voiceover" in the A1 audio track:

As you can see, the audio file is actually named "Strummer voiceover3.wav" -- even for this exercise it took me three tries to get something satisfactory, each time requiring a new wav file name (I could have simply over-written the same file, but it's generally wise to save separate takes, since you never know if 'voiceover2' will turn out to be better than 'voiceover3' after all). In any case, in the recorded audio track we can see that the wave-form is much more prominent and distinctive than it is in the original video clip: this is a human voice rendering commentary on the video action, recorded at a healthy -20 dB or so, just as the Peak Meter registered above. We can also see that I have already scooted the audio clip a bit to the left of the video's zero point. In fact (as predicted above), the audio actually began some seconds earlier, before I had the video running at all; so after assessing that, I have clipped that first portion out; and while I was at it I clipped a bit more until I had some commentary on the actual action at the start of Strummer's video. As it is now positioned, the audio track is in sync real-time with the actions on the screen, and plays more or less to the end. Strummer's video clip, originally recorded in near-silence, is now enhanced (or we'll pretend its an enhancement, at least) by my faux-BBC nature- documentary-style voiceover commentary on Strummer's actions throughout the short film.

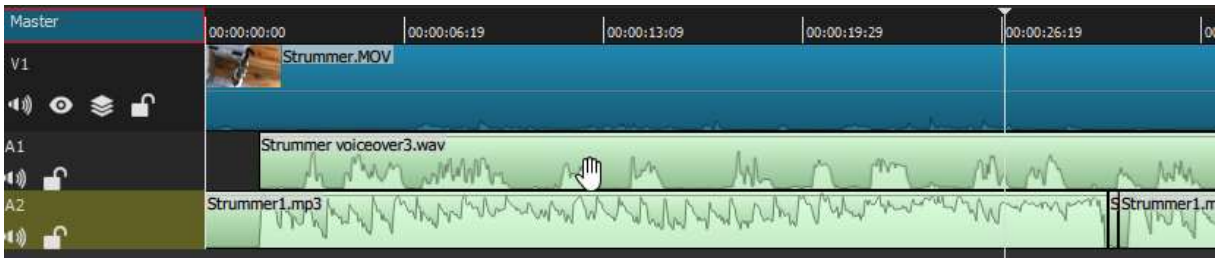
Is this project ready to be re-exported, then, as a finished mp4 video? Here I will set aside questions of length: a proper Strummer documentary would probably want to run a good 10 minutes, and thus would need to contain several such clips, sutured together with transitions (and, in fairness, should probably also be intercut with some footage of Strummer's sister Freyja). But we've already seen how we'd do this, from a technical standpoint. More interesting for the present exercise will be the addition of a supplementary musical soundtrack, which most classic nature documentaries also feature.

To find a good musical track for this video, I went through much the same process we saw in the previous chapter, except that this time I searched for 'jaunty' instrumental music as more fitting for the visual material here. At one music site I found more than a hundred jaunty and/or cheeky orchestrated pieces, and really the challenge was narrowing down to a single one that seemed to best fit. The catch was that this site wanted to charge for a download, even for non-commercial purposes; I politely declined their offer, however, and simply recorded the audition stream using Audacity. This method would not be legal, or recommended, if I intended to put my end-product video to any remotely commercial use, but I don't. So 'strummer1' it is. As we did in the last chapter, I'll **Open** that audio mp3 file (which will immediately begin playing in my Source), then add it to the Playlist, and from there to the Timeline. Remember that in order to do this, I'll first need to create a new, second audio track, then make sure this A2 track is *selected* before I use the 'Append' Timeline control to add in the clip:



You'll notice two things immediately here: one is that the music clip is quite a bit shorter than the V1 video clip--as a unit, the music theme was only about 30 seconds long. But that is typically not a problem with instrumental music: we can just append one or more additional copies to the end of the first one, to get the required length; the repetition will not typically be noticed. You'll also see that I've muted, for now, my A1 voiceover track, just so I could see/hear the music track by itself with the Strummer movie. It's really a perfect match, except for the length, which I'll correct next. With the A2 audio track still selected, and the Strummer1.mp3 clip selected in the Playlist, I hit "Append" twice more, and this gets me to a music-track length just over the total length of the video (I also trim out the gaps at the beginning of my second and third copies, before the actual wave-form begins again, though I can leave it in for the first iteration). In fact, just as we saw in the last chapter, it happens that in this case too the 3rd iteration of the music track ends about 5 seconds after the end of the video. It could trim that away, OR I could once again leverage that happy chance to add a 5-second end-credit sequence to the end of the video, just as we did before.

But for now we have a different issue: look what happens when I un-mute my (A1) voice-over commentary track, which once again makes that wave-form visible:

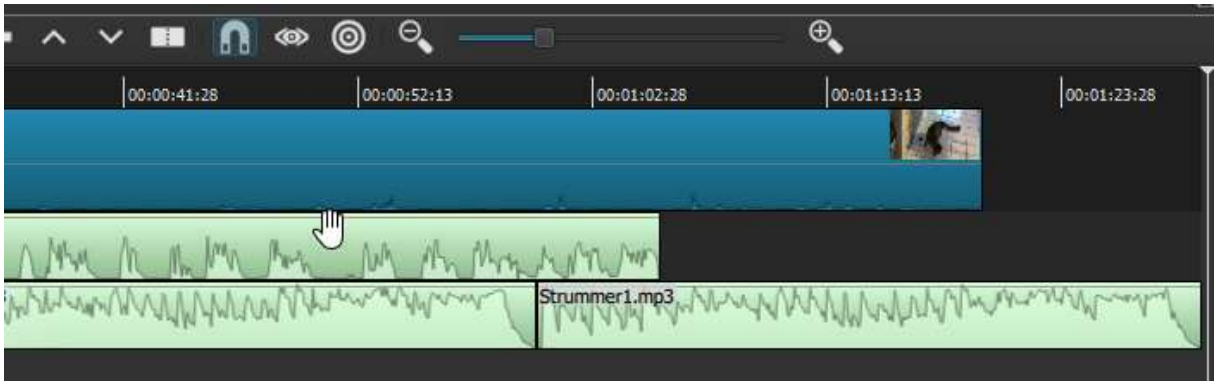


Even before playing the project, we can already see the problem just by looking at the relative size of the wave-forms, which correspond well to the recorded volume of the audio: the music track is much louder than my voiceover, except at a bare few points where they are roughly the same volume. That isn't going to work. In the last chapter, when we mixed together the musical soundtrack with the natural (diegetic) sound of the Iceland waterfalls, it was completely appropriate for the music to be foregrounded. But now it is not; the jaunty orchestral music entirely drowns out my BBC commentary, which is important to the actual informational load of the video. We need a different 'mix' of the two audio tracks.

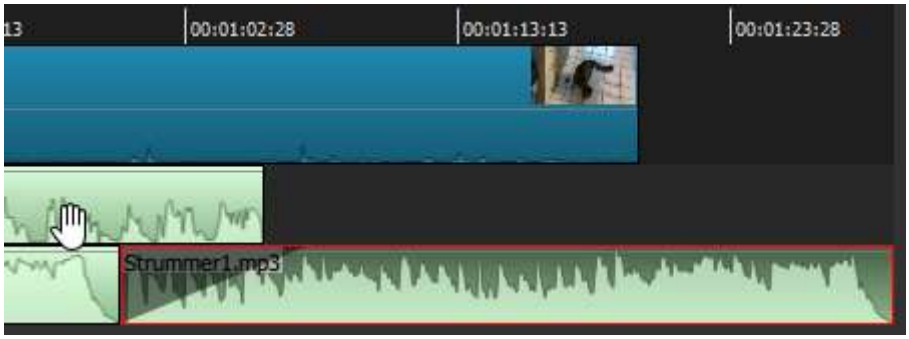
To start with, we'll simply apply a Volume/gain filter to the A2 musical audio, and try turning down the gain to about half (as we did with the natural waterfall sound in the last chapter--please refer to that for visual details). Actually, you'll note that in the A2 track we're actually dealing now with three *separate* segments (three copies of the original music clip), so I must apply the same gain filter three times. I could get around this by Exporting the A2 track by itself (the easiest way to do this, without disrupting the overall project, would be simply muting A1, and hiding and muting V1), and then re-importing it as a single whole track; but instead I use the opportunity to experiment with slightly different gain settings for the three segments. A setting of slightly less than half the

original volume for A2 seems just about perfect, with the jaunty background music adding nicely to the tone but nowhere overwhelming my plummy faux-BBC voice.

Now, we observed early in this chapter that audio 'mixing' often involves not merely adjusting the gain along entire *lengths* of your different audio tracks, but also often adjusting the relative gain of different tracks at different moments. If my voiceover narration had been more sporadic than it was, leaving long enough moments of silence between comments, it would probably have made sense to 'swell up' the background music during these moments, using Fade In and Fade Out audio filters. As it stands, there are really no gaps in the flow of my narration long enough to do this--except in a way that would sound nonsensical; but there is an opportunity to show this operation at the end of the whole piece. Let's take a look:



As we can see, the voice narration in the A1 audio track actually ends a good 10 seconds before the video does--with the A2 music track extending for an additional 10 seconds or so, where I would contemplate putting in a closing title/end-credit segment (probably first with a freeze-frame of the final video shot, fading to black before the end text appeared -- all of which we know how to do now). But is there a reason why the music needs to stay so muted, after the voiceover ends? It might be more effective if it swelled up at the end, partly as a way to simply emphasize the ending. How might this be achieved? The final segment of A2 is already separate, so it can (and indeed must) be filtered separately. I will start by raising the gain of this segment back up close to where it was originally recorded. I will then add a Fade-in Audio filter, and set its duration for so long -- 7 seconds -- that the swelling music doesn't step on the end of my narration at all. The result now looks like this:



Again, I will ask you to imagine that an end-credit sequence has been applied to the end of the V1 video track, with its requisite segments and filters (there is no need to go through that exercise again), so that V1 now extends precisely as far as A2. We can see the lengthy audio fade-in applied at the beginning of the final music segment, tapering up so as not to interfere with the end of the A1 voiceover. I could apply a corresponding fade-out filter to the end of this segment, but there is absolutely no need: the music itself ends very organically, just where we wanted to.

It might appear that much in this exercise has relied on random good fortune, such as the chosen musical piece happening to be just the 'right' length. But of course it wasn't the right length: it was much too short to start with, then significantly too long when tripled -- I just happened to find a good use for the extra length, which would probably appear in the finished video as if it were totally designed, particularly as we've adjusted the music. If I really didn't want an end-credit segment in the video, I would probably have used a fade-out audio filter to end my music in a (relatively) graceful way; or else chosen a different musical piece with a different structure that ended slightly sooner. Or if I were really wedded to using the Strummer1 piece, I could adjust the length of the video itself--for example by making the rolling-over-with-toy bit a separate, slow-motion sequence, which would have the effect of lengthening the overall video by however many seconds (cats at play turn out to be almost as good candidates for slo-mo as star athletes do--at least when shot in the kind of high-quality video that iPhones produce these days).

The point I'm making here, as I did at the end of the last chapter, is that in video editing of any complexity at all, you will be faced with a galaxy of potential choices, driven in part by a certain overall vision or effect you started out with, but driven as well by choices that open up along the way as you begin to assemble separate pieces, which in some cases you'll have created yourself, in other cases you'll have 'found'. But in either case, you'll never be able to imagine the different ways that your pieces can fit together until you try some actual editing. Save your work -- as project files and as exported video--at different stages, so that you can revisit any version if you want; and in between, just resolve in advance that you'll make as much use of the "Undo" and "Redo" buttons as you need to.

Release Notes

From https://www.Shotcut.org/download/releasenotes

Release 20.02.17

- Added **Settings > Preview Scaling**!
- Added **Export > Advanced > Video >Use preview scaling**.
- Added **Views > Scopes > Video Vector**.
- Added **Filters > Audio > Pitch**.
- Added the ability to rename clips in **Properties**.
- Added support for using a video clip in Transition **Properties > Video**.
- Added a few more export presets:
  - audio/ALAC
  - audio/FLAC
  - intermediate/DNxHR HQ
  - intermediate/ProRes HQ
  - intermediate/ProRes was changed to ProRes 422
- Added **Arabic** translation.
- Fixed dropping multiple files to **Playlist** in a new project (broken in v19.12.31).
- Fixed some broken keyboard shortcuts in the Turkish translation.
- Fixed **Properties > Speed** not working after a project file repair.
- Fixed clip selection after **Insert Track** or **Remove Track**.
- Fixed **Playlist > Add Selected to Timeline** creates corrupt clips (broken in v19.12.31).
- Fixed **Settings > Display Method > Software (Mesa)** on Windows (broken in v19.12.31).
- Fixed creating a **Project folder** with leading or trailing spaces.
- Fixed saving the length property in MLT XML as a time value independent of frame rate.
- Fixed starting **Text: Simple** video filter with "@" shows "0".
- Fixed seeking previous & next on the first track in **Keyframes** where you trim a filter or use simple keyframes.
- Fixed an unexpected transition is created when moving a clip rightward adjacent to the next clip in **Timeline** (regression in v19.12.16).
- Fixed drag-n-drop from **Source** player to **Timeline** left player in an inconsistent state (broken in v19.09.14).
- Fixed an inconsistent colorspace conversion when accessing a cached image.
- Fixed **Playlist > Copy** followed by a change in **Properties** incorrectly changes the playlist item.
- Fixed clicking on the rectangle control may change its size.
- Fixed using the **LUT (3D)** filter with file with extended characters in its file path on Windows.
- Fixed a crash when using a transition on every track at the same time.
- Improved the reliability of **Timeline > Select None**.
- Changed **Timeline > Master > Properties > Frame rate** to show 6 decimal digits.
- Reduced the latency of scrubbing (regression in v19.12.31).
- Changed the video-overlay rectangle control used in some filters to allow changing the position by dragging from anywhere inside the rectangle.
- Changed the **Filters** panel on macOS to prevent floating to avoid a frequently reported problem of the **Filters** window appearing blank/black.
- Changed **Timeline > clip context menu > Detach Audio** to not seek afterwards.
- Improved mouse wheel and trackpad behavior in **Timeline**.
- Upgraded MLT to version 6.20.0 and WebVfx to version 1.2.0.

Release 19.12.31

- Fixed a **Timeline** clip corruption bug in version 19.12.16 when moving clips with **Ripple** on.

- Added support for moving multiple clips on the **Timeline**.
- Added two new video scopes were added: **RGB Parade** and **RGB Waveform**.
- Allow reducing track height to very short in **Timeline** and **Keyframes**.
- Fixed crackly audio playback for some Windows users.
- Fixed loading project files made before v19.06 when not using a period for a decimal point.
- Fixed **Lift** and **Ripple Delete** on **Timeline** not reliably removing all selected.
- Fixed checking **Properties > Image sequence** may not update duration in **Playlist**.
- Fixed the intra-VLC and non-linear-quantizer options in the XDCAM-422 and D10 export presets.

Release 19.12.16

- Fixed **Scale** animation not linear in the **Rotate & Scale** filter.
- Fixed audio crackles in first couple of seconds of export.
- Fixed Ctrl+A selects all in **Playlist** as well as Timeline.
- Fixed the size and position of the **Text: HTML** editor with **Settings > External Monitor** enabled.
- Fixed drag-drop from **Playlist** to **Timeline** sometimes reorders the **Playlist**.
- Fixed **Color Grading** and **Contrast** creates a weird color after deleting a keyframe.
- Fixed updating x265-params in **Export > Other** after making changes in **Codec**.
- Fixed opening another project in the same session breaks master track filters.
- Fixed saving a preset with a slash in the name.
- Fixed export fails if the system temporary directory is not writable.
- Fixed removing some temporary files on exit.
- Fixed the timeline **Ripple All Tracks** option was not saved with history (before the current setting used during undo and redo).
- Fixed an image sequence in **Export > From > Playlist** may show "INVALID" on export.
- Fixed changing **Properties** (for example, image duration) not updating the **Playlist**.
- Fixed snapping to horizontal grid lines.
- Fixed **Timeline** clip context menu > **Properties** broken in v19.09.
- Fixed bad transitions created by trimming beyond media beginning or end.
- Fixed audio does not follow default device changes on Windows.
- Fixed changing the speed of the clip on the right side of a transition creates INVALID transition.
- Fixed track filters were not restored when undoing a **Remove Track**.
- Changed the minimum of the **Gain/Volume** filter to -70 dB.
- Removed **Settings > Deinterlacer > YADIF**. (This was causing crashes, and this option only affects preview, not export.)
- Added support for adding a transition in **Timeline** when dragging over a gap.
- Added support for free-form movement of clips on the **Timeline** (no more snapping back).
- Change the **Timeline** and **Keyframes** toolbars respond to **View > Small** Icons.
- Upgraded Mesa software OpenGL in Windows build to version 19.2.7.
- Upgraded SDL audio output library in Windows build to version 2.0.10.
- Added a limit to undo **History** configurable to new configuration key undoLimit that defaults to 1000.
- Added 3 new filters:
  - **Gradient** video filter
  - **Scan Lines** video filter
  - **Noise Gate** audio filter
- Added a new color gradient control to the following filters:
  - **Audio Light Visualization**
  - **Audio Spectrum Visualization**
  - **Audio Waveform Visualization**
- Added **View > Scopes > Video Zoom**.
- Added **Reverse** checkbox to the **Mask: From File** filter.
- Added **Remove Finished** to the **Jobs** menu.
- Added **Playlist > Update Thumbnails**.

- Added **Update Thumbnails** to the timeline video clip menu.
- Added keyboard shortcut Shift+Escape to give the player focus (take focus away from certain widgets).
- Added a **Two Column Scroll** template to the **Text: HTML** filter.

Release 19.10.20

- Fixed **Open Other > Audio/Video Device** capture (broken in v19.09).
- Fixed a crash in **Timeline** when you **Lift** the first clip in a track (broken in v19.09).
- Fixed automatic configuration of VA-API for **Export > Use hardware encoder** (broken in v19.09).
- Fixed **Blend Mode** filter affects other clips on the track (if the track **Properties > Blend mode** was not changed).
- Fixed adding keyframes to some video filters into area after extending the clip:
  - **Text: Simple**
  - **Rotate and Scale**
  - **Size and Position**
- Fixed color picker not automatically changing alpha from 0 to 255.
- Fixed a restarted job reports stopped when completed successfully.
- Fixed HTML file names with extended UTF-8 chars on Linux.
- Fixed **Timeline** audio waveform after changing **Properties > Audio > Track**.
- Fixed a crash opening a project that includes itself by making a self-repair on open.
- Fixed saving the length property in MLT XML as a time value independent of frame rate.
- Fixed changing **Video Mode** of an opened project breaks timing of edits.
- Fixed **Timeline** ruler not synchronized with the tracks' scroll after resizing panel or window.
- Fixed **Timeline > Undo** after splitting the second clip of a transition corrupts the timeline.
- Use the main video stream by default when there is an embedded album/poster/thumbnail.
- Minor improvements to the **ProRes Export** preset.
- Improved performance of some video filters when using parallel processing:
  - **Blur: Exponential, Gaussian, Low Pass**
  - **Glow**
  - **Mask: Simple Shape**
  - **Reduce Noise: HQDN3D**
  - **Sharpen**
- Improved performance of track blending in areas with transparency.
- Reduced the free disk space check to 25 GB.
- Opening Matroska files containing HuffYUV or Ut Video is much faster.
- **Timeline** waveforms are dimmed instead of hidden when track is muted.
- Changed RGB video clip **Properties > Color range** to **Full** and disabled.
- Changed **Properties > Reverse** and **Convert to Edit-friendly** best option (MKV) to use PCM for audio.
- Changed the **Export > lossless > Ut Video** preset to the **matroska** format.
- The signed macOS app is now notarized.
- Added new video **Filters**:
  - **Choppy**
  - **Nervous** (a random selection of previous and current frame)
  - **No Sync**
  - **Trails**
  - **Vertigo**
- Added **Keyframes** toolbar buttons and keyboard shortcuts for filter-trimming and simple keyframes:
  - [= set the filter start
  -] = set the filter end
  - { = set the first simple keyframe
  - } = set the second simple keyframe
  - Alt+[= seek previous simple keyframe

- Alt+] = seek next simple keyframe
- Added keyboard shortcuts for **Filters**:
- F = open the Filters panel and filter chooser. (At this point search has focus and the first filter is selected…)
- Up/Down - select the previous or next filter in the list
- Enter = add the selected filter to the list
- Shift+F = remove the selected filter from the list
- Added Thai translation.

Release 19.09.14

- Added the option **Play After Open** (default on) to the playlist menu to control above behavior.
- Added multi-select to **Playlist** as welll as **Select All** (Ctrl+Shift+A) and **Select None** (Ctrl+Shift+D) to its menu.
- Added multi-select to **Timeline**, which is currently limited to the remove/delete and lift operations.
- Added **Select All** (Ctrl+A) and **Select None** (Ctrl+D) to the **Timeline** menu.
- Added keyboard shortcuts for some existing **Timeline** menu actions:
- **Insert Track** (Ctrl+Alt+I)
- **Remove Track** (Ctrl+Alt+U)
- **Copy Timeline to Source** (Ctrl+Alt+C)
- Added new video filters:
- **Dither**
- **Halftone**
- **Posterize**
- **Threshold**
- **Elastic Scale** (non-linear horizontal scaling)
- **Blend Mode** (overrides the track **Properties > Blend mode** for that clip)
- Added a **Galician** translation.
- Fixed a crash in some audio filters when using 1 or 6 channels.
- Fixed showing language **English (Great Britain)** when **English (United States)** is chosen.
- Fixed a crash bug in v19.06 when changing image sequence **Repeat** in **Properties**.
- Fixed a bug in v19.08 where dropping a video into **Playlist** on a new project does not update the **Automatic Video Mode**.
- Fixed a bug in v19.08 in the on-screen rectangle control (as used in **Text: Simple** and **Size and Position** filters among a few others).
- Fixed changing speed of a clip with a colon in the file name.
- Fixed reading MLT XML with a colon in the file name of a relative path.
- Fixed the playlist menu button disabled after removing all clips.
- Fixed reloading **Fade In Video** or **Fade Out Video** using opacity may alter the colors.
- Fixed **Convert to Edit-friendly** failing on GoPro videos.
- Fixed filters during a transition are truncated after a **Split** on the timeline.
- Fixed a bug in v19.08 where **Keyframes** becomes broken after trimming on the timeline.
- Reduced the size of the installation by 255 MiB
- Upgraded FFmpeg to v4.2.
- Increased export process priority on Windows from Low (idle) to Below Normal.
- Changed default HEVC quality to 45% so the x265 crf matches its default of 28.
- Added the clip's name to the end of a clip in **Timeline** if its block is wide enough.
- No longer seek after dropping a clip from the player to the **Timeline**.

Release 19.08.16

- Changed **Playlist > Open As Clip** to simply **Open**. This action now opens the playlist item directly in the **Source** player, and all changes made in **Source** (trim in/out), **Properties**, **Filters**, and **Keyframes** apply to the playlist item immediately without an explicit update.

- Added **Playlist > Copy** that opens a copy of the playlist item in **Source** just like the old behavior. This is useful if you want to trim out another shot from the same source clip or create a different sub-clip with different filtering.
- Changed double-click on a playlist item to **Open** the clip instead of **Copy** it.
- Added keyboard shortcut Shift+C to **Copy** a playlist item.
- Now, when you drag a clip from **Playlist** to **Timeline** the timeline shows an appropriately-size box on a track.
- Fixed a performance regression (since v19.06) in the following filters: **Chroma Hold, Flip, LUT 3D, Mirror, Noise: Fast, Reduce Noise: Smart Blur**.
- Fixed reloading the filter UI for **Rutt-Etra-Izer**, **Text: 3D**, and **Text: HTML** resets the filter trimming in **Keyframes**.
- Added support for keyframes to the **Lens Correction** and **Mosaic** video filters.
- Fixed **Swirl** when maximum = 0%
- Changed the minimum values for **Mosaic** to 0%.
- Removed the scrolling animation from the **Blank Web Animations** HTML template.
- Fixed pasting filters changes the trim and keyframes of the existing filters.
- Fixed **Crop: Circle** and **Crop: Rectangle** not clearing the canvas resulting in trails in some situations.
- Fixed color incorrect when using the **LUT 3D** filter with some other filters following it.
- Fixed reliability of the **Stabilize** video filter to write its results (.stab) file.
- Fixed showing vidstab.trf as a missing file.
- Fixed updating **Stabilize** and **Normalize: Two Pass** results to clips copied between **Source**, **Playlist**, and **Timeline**.
- Added the ability for the **Stabilize** and **Normalize: Two Pass** filters' analysis jobs to update pending export jobs.
- Added the option to run pending **Stabilize** and **Normalize: Two Pass** filters' analysis jobs on export. This only works for **Stabilize** if you are using the project folder feature. Or, if not using the project folder feature, you must click **Analyze** to assign a results file name, but you can stop the analysis job.
- Added support for interlace output to **Properties > Reverse** and **Convert to Edit-friendly** including overrides for **Scan mode** and field order.
- Improved detection of interlaced video in some files such as Ut Video in Matroska.
- Added **Comments** to image properties along with a menu button with: Copy Full File Path, Show in Folder, and Set Creation Time…
- Add resolution and refresh rates to the screens in **Settings > External Monitor** to make them easier to differentiate.
- Fixed switching between different external screens on the same GPU.
- Fixed external screen not showing on correct screen in some arrangements.
- Changed the default video quality to 55% for the **Default** and **YouTube** presets. This aligns with the x264 default crf of 23 and produces a smaller file that most people desire for upload without significant quality loss.
- Added text after **Export > Advanced > Codec > Quality** to show the generated codec-specific quality level (e.g. crf for x264).
- Fixed **Stream** broken by check for writable file.
- Fixed double-click in **Recent Projects** loading twice.
- Fixed disabling meters in the **Audio Loudness** scope not shrinking space.
- Added version metadata to the AppImage for Linux.
- Added md5sums.txt and sha256sums.txt to the GitHub releases page.
- Added saving **Timeline** track height to configuration, not only a project file.
- Fixed trimming an unselected clip in **Timeline** does not correctly adjust its filters.
- Changed **Settings > Interpolation > Nearest Neighbor** to no longer relax seek accuracy. Instead, seek accuracy is now relaxed only during trick playback (reverse, rewind, fast forward).
- Added a **Korean** translation.

Release 19.07.15

- Fixed **Timer** video filter shows incorrect decimals.
- Fixed clips that become INVALID were saved as uneditable text clips.

- Fixed a crash in **Wave** video filter.
- Fixed a crash when changing clip **Speed** to something very low.
- Fixed long **Projects folder** path in the **New Project** view.
- Fixed the temporary backup file is empty when saving MLT XML.
- Fixed saving existing project on Dropbox gives an error.
- Fixed advanced keyframes removed when trimming.
- Fixed loading the frame rate from an export preset.
- Fixed multiple **Stabilize** video filter analysis jobs may try to write to the same .stab file.
- Fixed reported timecode of failed **Export** job if (**Frames/sec** is not 25 or not changed in **Advanced > Video**).
- Fixed incorrect audio waveform after **Undo** after insert/paste into **Timeline**.
- Fixed filters added to a clip with a transition are not the correct length and not active during the transition.
- Fixed **Blur: Gaussian** filter makes bottom 3 rows of the image with black or garbage.
- Fixed help text in the **New Project** view may be truncated without scroll bar.
- Fixed **Crop: Circle** > Radius = 100% not completely extended.
- Fixed loading image sequences that do not put leading zeroes into their number (bug introduced in v19.06).
- Changed **Average Bitrate** for libopus audio codec to set vbr to constrained.
- Changed **Crop: Source** to use the source clip's resolution as maximum for parameters.
- Changed the **Convert to Edit-friendly** and **Reverse** "better" option from ProRes to DNxHR, which is faster.
- Changed the **Convert to Edit-friendly** and **Reverse** "best" option from FFV1 to Ut Video, which is faster.
- Improved full range handling in **Convert to Edit-friendly** and **Reverse** by respecting an override in **Properties > Color Range**.
- Improved detection of full range color in video clips.
- Changed ripple move on the Timeline to push the clips when the drop zone is a gap.
- Added " - Converted" into the suggested file name when using **Properties > Convert to Edit-friendly...**
- Added a status message at the start of opening a project.
- Added a drop-down of common frame rates to **Export** and **Custom Video Mode**.
- Added a dialog to ask if the standard, fractional frame rate was intended in **Export** and **Custom Video Mode**.
- Added a **HD 1080p 50 fps** video mode.

Release 19.06.16

- Fixed deleting the project file if there was a save error.
- Fixed reliability of **Settings > Display Method > Software** on Windows.
- Fixed **Crop: Source** filter not working with **Color** clip.
- Fixed using filters on **Color Bars** and other generator clips.
- Fixed audio filters (**Compressor, Expander, Limiter, Notch, Reverb**) broken\ on comma for decimal.
- Fixed alpha video opaque on gaps in **Timeline**.
- Fixed **Convert/Reverse** if there no audio track.
- Fixed **Measure Video Quality** broken.
- Fixed saving the app directory in XML.
- Fixed **Alpha: Adjust > Invert** checkbox on reload.
- Fixed color eye-dropper (picker) error.
- Fixed audio **Pan** filter channel resets on reload.
- Fixed a crash using **Mirror** filter before **Rotate and Scale** or **Size and Position**.
- Fixed poor reverse audio quality for mp4 and mkv options.
- Fixed Simple Scroll HTML template may not scroll Up or Left completely.
- Changed project file to use period for decimal point regardless of OS locale (region/language setting).
- Changed **Export > From** to show **Source** instead of base file name.
- Improved Export Job progress and estimated time remaining.
- Changed **Timeline** ruler interval to 5 seconds.

- Renamed video filter **Circular Frame** to **Crop: Circle**.
- Renamed video filter **Crop** to **Crop: Source**.
- Renamed video filter **Text** to **Text: Simple**.
- Renamed video filter **3D Text** to **Text: 3D**.
- Renamed video filter **Overlay HTML** to **Text: HTML**.
- Renamed video filter **Blur** to **Blur: Box**.
- Renamed **Reduce Noise** video filter to **Reduce Noise: Smart Blur**.
- Changed the default for **Settings > Display Method** back to **DirectX** on Windows.
- Changed maximum duration of **Color**, **Text**, and **Color Bars** clips to 4 hours.
- Added **Jobs** to the main toolbar.
- Reordered panel buttons on main toolbar to match **View** menu.
- Increased maximum value of Timer filter's Start Delay, Duration, and Offset to 24 hours.
- Added **View > Show Text Under Icons** to menu.
- Added **View > Show Small Icons** to menu.
- Added support for alpha channel to **Crop: Circle**.
- Added **Crop: Rectangle** video filter with support for alpha channel.
- Added **Add Keyframe** button in Keyframes (only on parameters that show a curve UI).
- Added **Ripple All** button to **Timeline** toolbar.
- Added keyboard shortcuts Ctrl+0-9 to toggle the panels.
- Added Alt 0/+/- shortcuts to adjust the zoom in **Keyframes**.
- Added a vertical **Flip** video filter.
- Added **Blur: Exponential** video filter (fast and bleeds to edges).
- Added **Blur: Low Pass** video filter (fast and bleeds to edges).
- Added **Blur: Gaussian** video filter (slow and bleeds to edges).
- Added **Reduce Noise: HQDN3D** video filter.
- Added **Noise: Fast** video filter.
- Added **Noise: Keyframes** video filter.
- Added Swedish translation.

Release 19.04.30

- Fixed reading some AVCHD files after a camcorder splits.
- Cosmetic fixes for timecode spinner and toolbar icons on high DPI systems.
- Bundled libnsl for Linux to fix Fedora.
- Fixed **Video Waveform** scope graticule not showing on non-dark theme.
- Fixed incorrect compositing z-order after **Insert Track**.
- Fixed cosmetic problem with main toolbar using **System** theme on macOS.
- Fixed current track changes after inserting or overwriting a clip in Timeline.
- Fixed absolute paths can be introduced on Windows when using a project folder.
- Fixed sometimes showing forward slashes for file paths on Windows.
- Fixed first clip does not start at beginning when drop to Timeline after **File > New** or **File > Close**.
- Fixed **Properties > Convert** and **Reverse** on files with album art or embedded thumbnail.
- Fixed a possible crash on **File > New** or **File > Close**.
- Fixed filters on clips not extended into transition when adding a transition by trimming or resizing a transition.
- Fixed **Open Other > ALSA Audio** on toolbar on Linux.
- Fixed changing **Properties > Speed** or **Color Range** on a timeline clip may crash.
- Fixed timeline correction when drag clip to another track and then back to original.
- Fixed reloading the **Mask: From File** filter resets **Threshold** keyframes.
- Fixed the **Mask: Simple Shape** filter does not work when **Width** or **Height** is 0%.
- Increased parameter ranges in the **Rotate and Scale** filter.
- Added verification that saved MLT XML is well-formed XML before (over-)writing the target file.
- Changed default **Outline Thickness** (3) and alpha of **Text** filter to match **Open Other > Text** and prevent a problem where the text outline is aliased when the clip is transparent.
- Changed the version check to use HTTPS for increased privacy.

- Added DLL redirection files for other .exe programs on Windows to prioritize **Shotcut**-provided DLLs over those in system or $path.
- Removed the buggy **Merge with next clip** from Timeline clip context menu.
- Added (creation) **Date** column to Playlist **Details** view.
- Added **Set File Date…** to the Playlist menu and item context menu.
- Added **Sort By Name** and **Sort By Date** to the Playlist menu.
  These are single-shot sorting commands. The playlist is still fixed-ordering. The new options do not automatically sort new playlist entries.
- Added new video filters:
  - **Grid**
  - **Audio Dance Visualization**
  - **Audio Light Visualization**
  - **RGB Shift**
  - **Glitch**
  - **Distort**
- Added **Zoom 300%, 400%, 500%, 750%, and 1000%** to the player's zoom menu.
- Added **Settings > Drawing Method > Software (Mesa)** on Windows.
  This is not good for performance, but it improves compatibility. Use only as a last resort.
- Added **Display Method > OpenGL** or **Software (Mesa)** on Linux.
- Added **Norwegian Nynorsk** translation.

Release 19.02.28

- Fixed opening image sequence on Windows with extended/special characters in file path/name.
- Fixed crash on some video clips, particularly those with Google Pixel 3 and likely others.
- Fixed field order with interlaced export.
- Fixed an image artifact when using keyframes with the **Mask: Simple Shape** filter.
- Fixed image and alpha channel integrity with transitions on clips with a non-opaque alpha channel.
- Fixed a crash when changing clip **Properties > Audio > Track**.
- Fixed **Properties > Video > Color Range** inaccurate after changing it.
- Fixed **Shake One Second** presets in regions that use comma for decimal point.
- Fixed moving a **Playlist** item to the end.
- Fixed more dialogs to be modal to prevent them from going behind the main window.
- Fixed **Overlay HTML** webvfx templates when using a project folder.
- Fixed volume slider appears before main window at launch on macOS.
- Fixed more dialogs to use sheet style on macOS.
- Fixed a few small memory leaks in MLT.
- Upgraded Qt to version 5.9.7.
- Changed **Width** and **Height** minimum to 0 for **Blur** filter.
- Set the **Save as type** list on all file save dialogs (convert, reverse, text, EDL, image).
- Improved the quality of Export Frame > WebP.
- Show "Not Seekable" instead of "Live" when opening a non-seeable clip (or device or stream).
- Show a status message when trying to drag from Project player.
- Prefer loading DLLs in **Shotcut**'s install folder over those in System32.
- Default the out point of the **Color**, **Count**, and **Text** generator clips to the same as image duration (default 4 seconds).
- Added **Offset** to **Timer** video filter.
- Added **Vertical HD 30 fps** and **Vertical HD 60 fps** video modes.
- Added support for HTTPS.
- Added `--QT_SCALE_FACTOR` and `--QT_SCREEN_SCALE_FACTORS` command line options.
- Added **English (Great Britain)** translation.

Release 19.01.27

- Fixed **Text** animation/keyframes not working in v19.01.24.
- Fixed distortion when a **Mirror** filter is placed before a **Size and Position** or **Rotate and Scale** filter.

- Fixed **Levels** filter in locales with comma for decimal point.

Release 19.01.24

- Fixed launch crash on Linux by excluding libdrm libraries.
- Fixed audio level changed by the **Mask : From File** filter.
- Fixed **Text** filter may have glitches with **Export > Parallel processing** (regression in v18.12.x).
- Fixed missing file detected when **Stabilize** filter added without clicking **Analyze**.
- Fixed external monitoring on Linux screen.
- Fixed unable to detect hardware encoders with spaces in the installation folder path.
- Fixed changing **Speed** can move a clip.
- Fixed changing **Speed** has no effect when the system region and language decimal separators are different.
- Fixed filter duration not adjusted when trimming a transition.
- Fixed ripple trim when clip has a transition.
- Fixed removing a transition by trimming adds some frames.
- Fixed aspect ratio for **File > Export Frame** with non-square pixel video.
- Fixed handling relative paths to external resources in the **Overlay HTML** editor.
- Fixed a crash opening a project after removing the bottom video track.
- Fixed the lossless/H.264 preset to be completely lossless.
- Fixed dropping a file from a file manager whose name has special/extended characters (e.g. [,]).
- Fixed the state of the enabled checkbox for the **Overlay HTML** filter and when using a color picker.
- Fixed distortion after changing the **Keyframe Type** of a keyframe for the **Scale** parameter in the **Rotate and Scale** filter.
- Fixed right-clicking a keyframe to open context menu may change its position.
- Upgraded FFmpeg to v4.1
- Improved **Color Grading** filter by letting all parameters go from -100% to 100%.
- Added an automatic retry without **Parallel processing** when **Export** job fails.
- Set the **Save as type** list on the **Export File** dialog on Windows when a preset defines a filename extension.
- Added **Center Playhead** option to the **Timeline** and **Keyframes** menus.
- Added **Slow Zoom __, *Hold* __** presets to the **Size and Position** filter.
- Added a **Chroma Hold** video filter.
- Added a **Swirl (HTML)** video filter.
- Added a simple **Templates** framework to the **Overlay HTML** filter (see path **Shotcut**\share**Shotcut**\qml\filters\webvfx\templates to add your own) with the following templates:
  - Blank HTML
  - Blue Middle Bar
  - Creative Commons Music
  - Blank with Web Animations
  - Simple Scroll

Release 18.12.23

- Fixed color accuracy of **lossless/Ut Video** preset and use yuv422p format.
- Fixed number of digits for seconds in the **Timer** video filter when using **Format > MM:SS.SS**.
- Fixed compositing completely transparent areas of alpha channel darkens output.
- Fixed some generators (**Color Bars, Ising, Lissajous, Plasma**) not working correctly in **Timeline**.
- Fixed **Settings > External Monitor > DeckLink/Intensity** can stall when **Settings > Realtime** is enabled.
- Fixed making project folder if a parent folder does not exist.
- Fixed **New Project > Start** does not save a .mlt file until you open media and save.
- Fixed launch on some Linux (e.g. gentoo) by including libselinux .
- Fixed crash when undo **Playlist > Remove All**.

- Fixed auto-saved file not removed when save and exit with no changes.
- Fixed the color accuracy of the **Color** generator.
- Fixed numeric locale bug in the **Audio Spectrum Visualization** and **Audio Waveform Visualization** filters.
- Fixed clip reloaded when leaving **Properties > Speed** with no change.
- Fixed **Settings > External Monitor** on Windows screen.
- Fixed launch crash on Linux with (Csound) csladspa < 6.11.1 installed.
- Improved color accuracy of internal RGB-to-YUV conversions.
- Changed **Quality** to 100% when using the **intermediate/ProRes** preset.
- Added 16:9, 9:16, and 1:1 aspect ratios for transition wipes.
- Moved **Properties > Reverse…** from the overflow menu to a button.
- Changed **Properties > Reverse…** to use project folder if used.
- When **Reverse** job completes, automatically open it and add it to the **Playlist**
- Start playback from the beginning if you play when the play head is at the end.
- Improved speed of the **Text** and **Timer** video filters.
- Added **View > Scopes > Video Histogram**
- Added **Preset** to some more filters:
  - **Audio Gain/Volume**
  - **Blur**
  - **Brightness**
  - **Opacity**
- Added **Preset** and simple and advanced keyframes to the **Balance** and **Pan** audio filters.
- Added a **Levels** video filter with simple and advanced keyframes!
- Added **Properties > Color Range** for video clips.
- Replace the **Mask** video filter with 3 new filters:
  - **Mask: Simple Shape**
  - **Mask: From File**
  - **Mask: Apply**

Release 18.11.18

- Fixed crash in Export (bug in v18.11.13).
- Fixed NVENC hardware encoders on Windows and Linux.
- Fixed VA-API hardware encoders on Linux. As a result, the Linux build is now based on Ubuntu 16.04 (glibc 2.23), which may reduce compatibility with older Linux systems.
- Fixed hardware encoder detection on Windows.
- Added **Audio Waveform Visualization** video filter.
- Added **MM:SS.SS** to **Timer** filter.
- Added IRE graticule and tooltips to the **Video Waveform** scope.
- Added support for the mouse wheel to the **Color Grading** circles.
- Added configuration setting player/warnGPU, which is a boolean that defaults true.

Release 18.11.13

- Added an **Advanced** mode to **Export**.
- Added **Use hardware encoder** checkbox to **Export**.
- Added VA-API hardware encoding for Linux.
- Added videotoolbox hardware encoding for macOS.
- Added **New Project** / **Recent Projects** screen.
- Added **10 Pixel Grid** and **20 Pixel Grid** options to the player grid button menu.
- Added **Spot Remover** video filter.
- Added **View > Scopes > Video Waveform**.
- Added **Settings > Video Mode > Non-Broadcast > Square 1080p 30 fps** and 60 fps.
- Added **Ut Video** presets to **Export**.
- Added signed app bundle for macOS.
- Fixed support for macOS 10.10 and 10.11.
- Fixed clearing export preset search collapses categories.
- Fixed searching export presets in categories.
- Fixed initial rectangle size for **Size and Position** filter.
- Fixed reopening **Timeline** changes zoom level.

- Fixed exit sometimes hangs.
- Fixed some filters' presets do not save any values:
  - **Key Spill: Advanced**
  - **Chroma Key: Advanced**
  - **Reduce Noise**
- Fixed A/V synchronization on some files.
- Fixed seeking on audio files with album art.
- Fixed saving multiple lines of text in preset for **Text** generator.
- Fixed crash when undoing split and transition on Timeline.
- Fixed filters not applied correctly when using **Export > From > Each Playlist Item**.
- Improved reliability of **Audio/Video Device** capture.
- Fixed **Color** generator did not signal colorspace.
- Fixed transfer characteristic conversion and full range output in **Export**. (Add `mlt_image_format=rgb24`, `color_range=jpeg`, and `pix_fmt=yuvj420p` in **Other** for full range output.)
- Made **GPU Effects** hidden and discouraged.
- Added support for project folder to **Stabilize** and **Overlay HTML** filters.
- Increased **Scale** maximum to 500% for **Rotate and Scale** filter.
- Improved support for DDS, ICO, and WebP images.
- Bundle more library dependencies on Linux.
- Converted macOS build to standard app bundle layout.

Release 18.10.08

- Added support for Intel Quick Sync Video hardware-accelerated video encoders to the Windows build (in **Export > Codec** choose h264_qsv or hevc_qsv).
- Added **Grid** and **Safe Area** overlays with a toggle/menu button to the player.
- Added snapping to the grid and safe areas for the VUI rectangle control as used by **Text**, **Size and Position**, and more filters.
- Added **Open Other** to the main toolbar with a drop-down menu.
- Added the ability to drag-n-drop folders from a file manager into **Shotcut**.
- Added the ability to supply multiple file and folder name arguments to the **Shotcut** command line executable.
- Added the ability to make a temporary **Custom Video Mode** (leave **Name** blank).
- Added **Settings > Video Mode > Custom > Remove…**.
- Added **View > Layout > Remove…**.
- Added **Settings > Clear Recent on Exit** checkbox to prevent saving data on a shared computer account.
- Added command line option `--clear-recent` to enable **Clear Recent on Exit** and hide that option in the **Settings** menu.
- Added a dialog when you click to check for an update that asks if you want to check for update automatically (at startup only) with the option to suppress the dialog indefinitely.
- Fixed audio preview distortion on Windows (regression in v18.09).
- Fixed some AAC MP4 files start in the middle.
- Fixed un-mute a track may not draw its waveforms.
- Fixed whitespace in **Text** filter removed in **Export**.
- Fixed crash adding clip to **Timeline** after removing all tracks.
- Fixed simple keyframes go missing or not all the way to 00:00.
- Fixed switching from simple to advanced keyframes in **Text**, **Rotate and Scale**, **Timer**, and **Size and Position** filters.
- Fixed a possible crash when adding a transition by trimming.
- Fixed crash on macOS after the app restarts itself when some **Settings** are changed.
- Fixed moving a clip to the left where the right edge is not a blank.
- Fixed some **Timeline** actions do not work correctly after a **Ripple** move.
- Fixed undo/redo form trim-to-transition over a blank/gap.
- Fixed **Ripple** moving a clip to the end of a track was not extending the hidden black background.
- Hide the **Text** generator if **Settings > GPU Effects** is on (incompatible).
- Fixed the **Rotate and Scale** filter preset not saving keyframes for the **Scale** parameter.

- Fixed a crash opening multiple files at once either through **File > Open** or drag-n-drop from a file manager.
- Fixed a crash closing a playlist-only project with **Automatic Video Mode**.
- Fixed changing position or removing advanced keyframes for the **Scale** parameter of the **Rotate and Scale** filter distorting the aspect ratio of the image.
- Fixed **Timeline > Split** not working if the current track is empty. (It should split the topmost clip under the playhead.)
- Fixed clicking the reset button of the **Center** checkbox of the **Crop** filter does not re-enable the other controls.
- Fixed the **Timeline** and **Keyframes** timeline rulers are incorrect after changing **Settings > Video Mode**.
- Fixed a crash when the project frame rate is very low (< 6 fps).
- Fixed a crash when switching keyframes on and off for position and size parameters in video filter such as **Size and Position**, **Text**, and **Timer**.
- Fixed trimming multiple track filters hides them.
- Fixed making a **Text** preset does not save the text (only all other parameters).
- Changed resolution restriction from a multiple of 8 to a multiple of 2.
- Improved the layout of the filter chooser in **Filters**.
- Changed **Timeline** fade controls to behave the same as **Keyframes** simple keyframes.
- Changed the **Noise** generator from opening as a live source to a clip with a duration.
- Changed drag-n-drop to **Playlist** to not automatically open the first file unless the project is empty.
- Changed the **Rotation** parameter of the **Mask** filter to use degrees, and fixed its reset button.
- Added more library dependencies to the Linux portable tar, AppImage, and snap builds including the SWH LADSPA plugins.

Release 18.09.16

Fixed broken color selection in the following (non-GPU) video filters:

- Chroma Key: Simple
- Chroma Key: Advanced
- Key Spill: Advanced
- White Balance

Release 18.09.15

- Fixed image transform regression in **Rotate and Scale**, **Size and Position**, and **Text** filters.
- Fixed image scaling interpolation method when not using a transform filter.
- Fixed a crash applying an image transform filter to an image with alpha channel (PNG, SVG).
- Added a highlight for the buttons in the **Filters** chooser.
- Fixed new **Ripple** and **Snapping** keyboard shortcuts not always working.
- Fixed moving a clip to another track and back not working correctly.
- Fixed dragging clips to the **Timeline** stops working.
- Fixed redo trimming the in point of clip on **Timeline** not working correctly.

Release 18.09.13

- Added **Ripple** support for **moving** clips including **Ripple All Tracks**.
- When **Ripple** is on and you move a clip to the right, it now ripples (pushes) all the clips to the right instead of making a transition (including the **Ripple All Tracks** option).
- Added **Ctrl+R** keyboard shortcut to toggle **Timeline Ripple**,
  Ctrl+Alt+R to toggle **Ripple All Tracks**, and
  Ctrl+Shift+R to toggle both.
- Improved **Snapping** behavior on the **Timeline**.
- Added **Ctrl+P** keyboard shortcut to toggle **Snapping**.
- Added **Reset Track Height** (**Ctrl+0**) to the **Timeline** menu.

- Added a **Timer** video filter.
- Added support for negative rotation to the **Rotate and Scale** filter.
- Added **Shake 1 Second - Scaled** and **Shake 1 Second - Unscaled** presets to the Size and Position filter.
- Added a **Text** clip to the **Open Other** dialog.
  This is a convenience item that creates a transparent **Color** clip with a **Text** filter.
- Added **Extract Sub-clip** to **Properties** menu for audio/video clip.
- Added **Export** presets for WebM VP8 and VP9 with alpha channel.
- Fixed a crash when quickly changing clip selections in the **Timeline**.
- Improved closing **Shotcut** more reliably and the behavior of multiple **Shotcut** processes.
- Fixed **Undo** beyond **Remove Track** may crash.
- Fixed a crash when moving a clip after undoing a transition.
- Fixed a crash loading project after some sequence of inserting and removing tracks.
- Fixed a crash loading an image with the wrong filename extension.
- Fixed OS audio device changes may cause **Shotcut** to hang.
- Fixed crash dragging simple keyframes beyond the edges of the clip.
- The 32-bit Windows build can use more memory to reduce risk of crash:
  3 GiB on 32-bit Windows and 4 GiB on 64-bit Windows.
- Improved behavior of opening WebP images.
- Fixed advanced keyframes for the **Size and Position** filter.
- Fixed inserted and then hidden video tracks become audio tracks after re-opening the project.
- Fixed changing **Video Mode** from **Automatic** to something else with nothing opened.
- Fixed moving a transition to another track leaves a hidden clip on the old track.
- Fixed trimming the out point of a timeline clip may change the in point of the following clip.
- Fixed **Properties** > **Speed**, **Convert**, **Reverse**, and **More Information** does not work sometimes after reloading a project.
- Fixed reliability of changing size and position fields with keyframes in **Size and Position** and **Text** filters.
- Fixed trimming outward with **Ripple All Tracks** moved clips on other tracks in the wrong direction.
- Fixed audio waveforms still appear on muted tracks after adding a track.
- Fixed edges of text with the **Text** filter may get cut off (clipped).
- Fixed **Size and Position** and **Rotate and Scale** filters may leave black right and bottom edge of frame.
- Fixed trimming a timeline clip with **Fade In/Out Video** filters with "Adjust opacity instead of fade with black" enabled.
- Fixed **Export** may change the actual frame rate and cause timing errors.
- Fixed reducing the **Frame/sec** in **Export** causes timing errors in the output.
- Removed the **Compositing** toggle icon from the **Timeline** track head. Now, it is located in the track **Properties** as **Blend mode** with a **None** option.
- In the **3D Text** filter, replaced the Droid fonts with Liberation due to an incompatible license.
- Improved the performance of changing filter parameters and scrubbing.
- Limit the number of background thumbnail and waveform generation threads to 4.
- Prevent trying to generate audio levels for waveforms for still images and other silent sources.

Release 18.08.14

- Fixed regression in 18.08.11 that can put corrupt character in the .mlt XML project file.
- Fixed the size of an existing Text filter with animation breaks if you select the clip and the filter.

Release 18.08.11

- Fixed new crash in v18.08 changing **Settings > Video Mode** with nothing opened.
- Fixed crash while adjusting position or size in **Text** and **Size and Position** filters.

- Fixed position and size information incorrect if resolution or aspect ratio changed in **Export**.
- Fixed **Text** animation (keyframes) when not the first clip in the timeline.
- Fixed track name not editable after the track head had been selected at least once.
- Fixed right-clicking a timeline clip breaks automatic timeline scrolling during playback.
- Added Quicktime Animation **Export** preset to export video with alpha channel.
- In **Export**, let option values in **Other** tab override values generated by other fields.

Release 18.08.01

- Added the timecode of failure to the **Jobs** panel when an export job fails (makes locating problem areas in the project easier).
- Added an **Unpremultiply Alpha** video filter (handy to fix compositing for video clips with an alpha channel that has had its color pre-multiplied with its alpha).
- Fixed various crash regressions since v18.05.
- Fixed audio distortion during preview (regression in v18.07).
- Fixed custom **Export** presets broken if name contains parentheses (regression in v18.07).
- Fixed **Properties > Reverse...** broken if numeric region setting uses comma for decimal (regression in v18.07).
- Fixed **Overlay HTML** editor easily destroys default scripts (introduced in v18.07) if **WebVfx JavaScript extension** enabled.
- Fixed custom interlaced **Export** presets loading as progressive (regression in v18.07)
- Fixed **Timeline > Copy Timeline to Source > Export** fails on unsaved (Untitled) project.
- Fixed **Text** filter has aliased edges (regression in v18.07).
- Fixed **Stabilize** video filter not available on Linux (regression since v18.06).
- Fixed changing **Speed** in **Properties** breaks all filters on that clip (regression since v18.05).
- Fixed **Fade Out Video** (and keyframes in general) broken on still images whose in point is > 0 (regression since v18.03).
- Fixed accuracy of **Properties > Duration** for image clip on the timeline.
- Some fixes for changing **Settings > Video Mode** after starting a project.
- Fixed compositing of upper video tracks becomes broken if bottom video track is deleted.
- Fixed images with alpha channel (e.g. PNG) on upper video tracks have dark edges after compositing if the **Size and Position** or **Rotate and Scale** filters are not used.

Release 18.07

- Numerous fixes as usual.
- Changed shortcut for **Add Video Track** to Ctrl+I.
- Changed shortcut for **Redo** to Ctrl+Y on Windows.
- Added **10%** to the player zoom menu.
- Hide the VUI (video user interface) when the play head is not over the clip with the current filter.
- Renamed the Rotate filter to **Rotate and Scale**.
- Keyframes are now saved in the project file using time clock values instead of frame numbers to make them adaptive to frame rate.
- Reverted memory manager change from v18.05 pending further testing to improve stability.
- Added advanced keyframes to the **Size and Position** filter.
- Added simple and advanced keyframes to the following video filters:
  - **Rotate and Scale**
  - **Text**
  - **Glow***
  - **Contrast***
  - **Sharpen***
  - **Vignette***

  * = including the (still experimental) GPU filter
- Added **Copy Timeline to Source** to Timeline menu.

- Changed the audio codec to AC-3 for the edit-friendly and reverse MP4 file format.
- Changed **J** and **K** key behavior to change speed up or down before changing direction.
- Added categories to the **Export** presets (custom presets can start their name with "category)" to use a category).
- Added **View > Layout** menu for custom and stock layouts:
  - Timeline Project
  - Playlist Project
  - Clip-only Project
  - Player
- Changed the default, first-time user UI layout to **Timeline Project**.
- Changed **Properties > menu > Reverse...** to work on a trimmed clip from either **Source** or **Timeline**.
- Added **Properties** menu items to the context menus for **Timeline** and timeline clips.
- Added logic to sort GoPro files when multiple files are opened or dropped.
- Added support for setting project file name (using **File > Save**) for empty project.
- Added new HTML template for WebVfx JavaScript extensions enabled in **Overlay HTML** video filter.
- Added search field to the filter chooser in **Filters**.
- Upgraded MLT to v6.10.0

Release 18.06

- Many bug fixes due to introduction of keyframes and change to memory management in v18.05.
- Added simple and advanced **Keyframes** to the **Blur**, **Mask**, and **Saturation** filters.
- Added seek buttons for simple **Keyframes**.
- Added ability to add and remove advanced **Keyframes** using double-click.
- Added ability to drag advanced **Keyframes** to adjust both value and position. (When dragging, hold down Ctrl key to adjust only value or Alt key to adjust only position.)
- Added double-click to toggle simple **Keyframes** controls (circles).
- Added double-click to toggle fade in/out controls (circles) on **Timeline** clip.
- Added many animated (keyframes) presets to the **Size and Position** filter.
- Added **Hue/Lightness/Saturation** video filter.
- Added **Reverse** to clip **Properties** menu.
- Added **Detach Audio** to timeline clip's context menu.
- Added 5.1 surround support to the **Copy Channel** and **Swap Channels** audio filters.
- Added caution message to GPU Effects confirmation dialog.
- Added a **Keyboard Shortcuts** link to the **Help** menu.
- Changed presets file format to YAML.
- Changed **Settings > GPU Processing** to **GPU Effects**.
- Reduced memory usage on 32-bit builds (by constraining multi-threading).
- Upgraded **FFmpeg** to v4.0.
- Integrated **AMD AMF** hardware-accelerated H.264 and HEVC encoders on Windows (Set **Export > Codec** to **h264_amf** or **hevc_amf**. Requires recent Radeon or AMD APU.)
- Upgraded MLT to git master (v6.8.0 minimum required to build).

Release 18.05

- Added **Keyframes** for Filters:
  - Gain / Volume
  - Brightness
  - Circular Frame
  - Color Grading (no simple)
  - Opacity
  - Size and Position (simple only)
- Added support for mono and 5.1 surround sound: **Settings > Audio Channels** and **Export >Audio > Channels**.
- Added **Finnish** translation.
- Restored **GIF Animation** for Export.

- Reduced memory footprint (especially for Rotate and Size and Position filters).
- Changed **Export** default settings to reduce output size by increasing GOP and number of B frames.

Release 18.03

- Added a **Sketch** filter.
- Added numeric fields to the **Color Grading** filter.
- Added **All** option to **Properties > Audio > Track**.
- Added **Estonian** translation.
- Improved image loading speed on Windows.
- Improved mouse-wheel & trackpad scrolling in **Timeline**.
- Improved support for JACK Audio (Linux and macOS only).
- Upgraded FFmpeg to version 3.4 and latest vpx for much faster VP9 encoding.
- Upgrade MLT to v6.6.0.
- Upgraded SDL to version 2.0.

Release 18.01

- Added **Audio Spectrum Visualization** filter.
- Added support for font size and italics to **Text** filter.
- Added a **Mask** filter.
- Another important fix for accuracy of XML time values for non-integer frame rates.

[end of document]