Project Title

Sprucing Up the Sequencer

Name

John Kiril Swenson

Contact

- School email: swen0481@umn.edu
- Personal email: kirilswenson@gmail.com
    - I should respond within a day to emails sent to either of these accounts.
- I'm @linen on blender.chat!

Synopsis

Blender's video sequence editor currently contains a number of "papercuts", or isolated frustrations that can degrade the overall experience and dissuade new users. I intend to implement a wide variety of changes to the video sequencer that I believe will improve UX and promote open-source video editing as a whole, including snapping support for the preview window and marker snapping in the sequencer, options to link or unlink strips, and adding "active" channels to make pasting more intuitive.

Benefits

Blender has already done incredible wonders for open-source software, democratizing 3D modeling and animation and reducing barriers for independent creatives. I envision a future where the same benefits could be achieved for video editing. If the video sequencer UX were improved, I could see Blender becoming an even more powerful open-source alternative to paid programs like Premiere Pro or Davinci Resolve.

By being associated with the Blender suite, many existing Blender artists could easily make the switch and effortlessly integrate the VSE into their workflow. Youtube channels that already use Blender for other purposes could be swayed by the improved user experience, encouraging them to also edit their videos in Blender. Many users could have access to a simpler, more convenient NLE, benefiting the creative ecosystem as a whole.

Deliverables

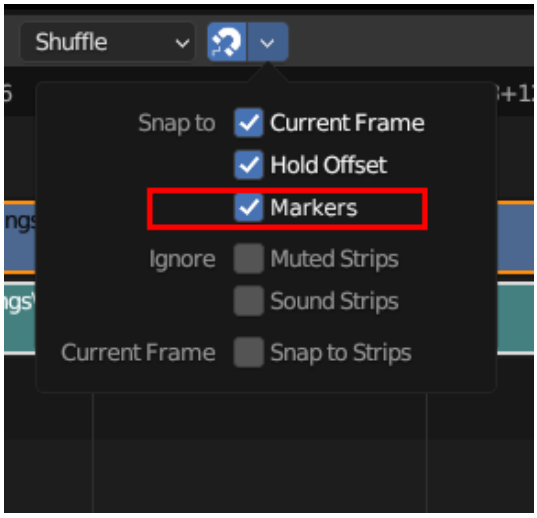
- Improve snapping support in the sequencer by adding the option to snap strips to markers.
- Add a link property to all audio/video strips for a given video file. Make strips linked by default, meaning that changes to one strip affect all other strips from the same file. Allow strips to be unlinked and relinked later.
- Add the ability to select active channels. Channels may be made "active" by clicking on an icon by the mute and lock buttons. By default, these channel(s) and any channels above these channel(s) will be the destination of any pasted or added strip(s), where the desired overlap mode shall determine the ultimate position (either overwriting an old strip or shuffling/expanding upwards). Channel 1 will be made active by default.

- Improve snapping support in the preview by adding the option to snap bounding boxes to all four preview borders (top/bottom/left/right) and horizontal/vertical center.
- End-user documentation

Project Details

Snap strips to markers (suggested by Nebulon1212)

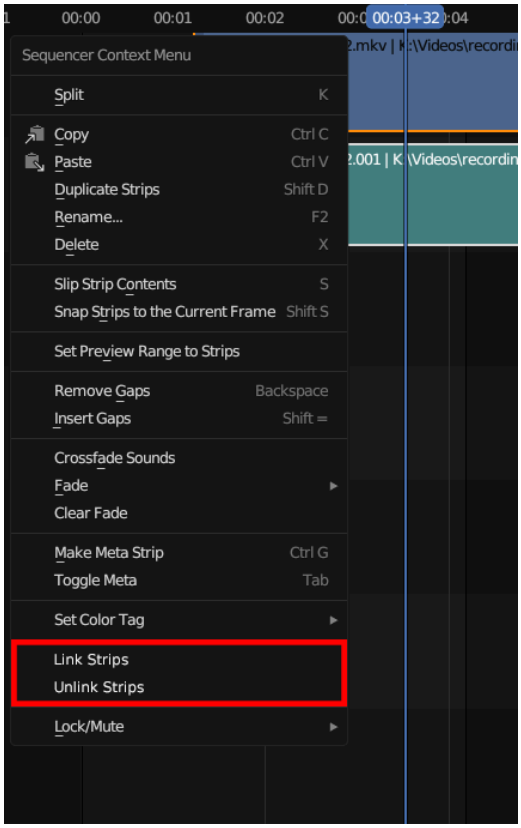
Currently, there is an option to snap strips to strip edges, the current frame, or hold offsets. However, there is no option to snap to markers, which is a common feature seen in other NLEs. As shown below, "Snap to" will also include a "Marker" option which may be toggled on and off. This would function similarly to the other snapping types and therefore the implementation would be adapted from snap_to_current_frame and snap_to_hold_offset RNA. Then seq_snap_target_points_build() and seq_get_snap_target_points_count() in transform_snap_sequencer.cc may be modified to add functionality for marker snapping.



Link and Unlink Strips

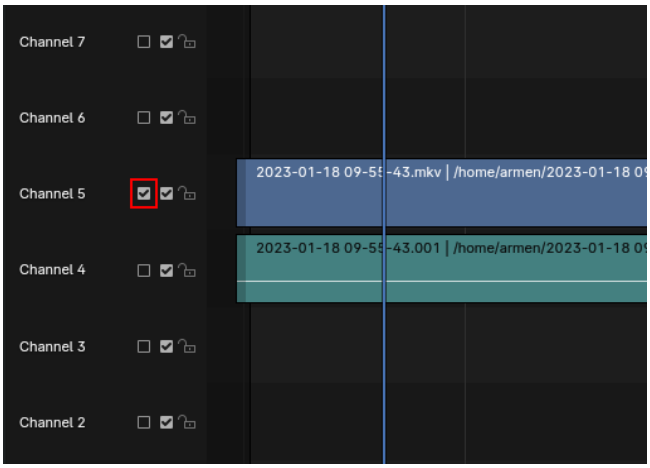
Blender currently treats audio and video strips from a single video file separately, where operations to one only affect operations to the others if both strips are selected at the same time (e.g. by shift-clicking or with box select). I propose an additional option to "link" and "unlink" strips in the sequencer context menu. If a user links together multiple strips, selecting one strip will automatically select all other linked strips, and operations on one strip will also be applied to all other strips. Any newly imported video files will have linked strips by default.

This could be implemented by adding a ListBase of all linked strips (internally represented as "Sequences") to each strip, similar to the data structures for meta-strips. Then, in sequencer_select_set_active(), recurs_sel_seq() may be modified to recursively select linked strips *and* all strips within metastrips. Then all preexisting tools and operators should work as if the user had just selected the strips manually. Linking could have a keybind similar to joining in object mode (ctrl+J), but P is reserved for setting a preview range, so I may correspond with my mentor(s) for further guidance on this matter.



Active channels

One issue with adding or pasting new strips into the sequencer is that the result of these actions are highly scenario-specific and depend on the exact steps taken, which feels random and hard to control from a user's perspective. I propose adding a new "active" property to channels which will give the user the option to explicitly select "destination" channels for operations that create new strips. A mockup of the altered channel look may be seen below, with an additional button to toggle "active" state of a channel:



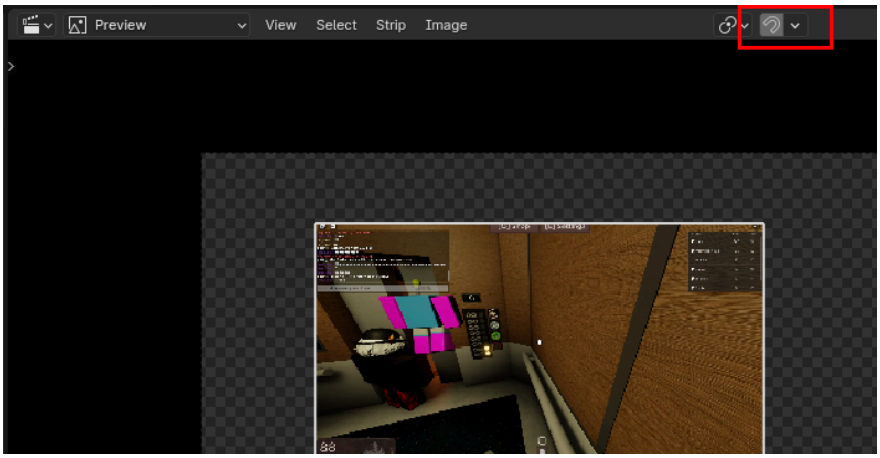
Note that the button used in this mockup is just to showcase the implementation, and I expect to use a different icon within the Blender ecosystem after corresponding with mentors and other developers for suggestions.

The active channel property would work as follows:

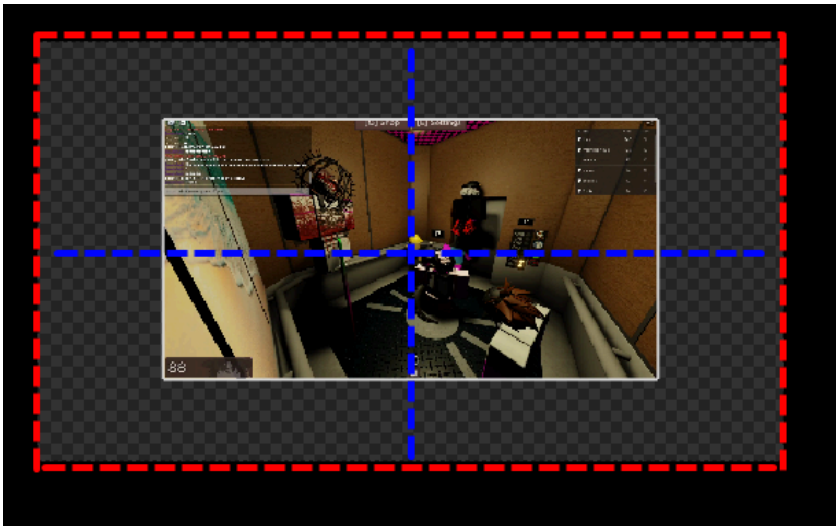
- Channel 1 will be active by default
- All pasted or added strip(s) will attempt to place themselves at the lowest active channel at the current playhead position
    - Overlap mode will decide how the strips behave if there is already a strip present in a given channel at the current frame
    - If strips from multiple channels were copied, the pasted strips will not overlap each other, and will remain in separate channels. The other strips will prioritize filling out other “active” channels but otherwise will just retain their channel offset with respect to the other strips

Preview Snapping Support

Currently, users can grab (G), rotate (R), and scale (S) strip bounds in the preview window. I intend to add a snapping toggle for the preview window which may function similarly to sequencer snapping but with differently defined snap points.



The below image shows the areas of the preview in which I intend to implement snapping support. Bounding boxes of a strip may be snapped to the dotted red lines corresponding to the top, bottom, left, and right render borders. The pivot point of a strip may be snapped to the dotted blue lines corresponding to the horizontal and vertical centers of the render output.



Note that the dotted lines are only meant to clearly illustrate the snap points. For an actual implementation, the drawSnapping() function in transform_snap.cc could be used to draw lines when a valid snap is found.

Although text strips' bounding boxes are larger than they should be to benefit from these snapping changes, there are options to choose text alignment with respect to the render output, so they already benefit from snapping in a sense. I will investigate dynamically sized bounding boxes for text only if I have extra time.

Additional tasks:

If all deliverables are completed and thoroughly tested and there is still time remaining within the project, I will take on additional tasks such as improved pitch management of audio strips. I decided to only tackle this idea if I have enough time at the end, so as to undershoot and deliver on time rather than overshoot and fail to finish all my duties.

Project Schedule

Although GSoC projects are not officially announced until May 1, I have already begun work on fixing bugs and adding functionality to the sequencer, and I intend to spend the entirety of April creating drafts for the features that I wish to implement. This summer, I have no outside time commitments, so I should be able to focus my full attention to the project at hand.

Date	Tasks
April 2-April 31 (4 weeks)	- Regardless of whether or not I am accepted as a contributor, complete PR for "Snap strips to markers" in the Sequencer - Research Blender's codebase further
May 1-26 (3 weeks)	- Solidify details of deliverables with mentor(s) and come up with a plan for each - Finish and test Snap Strips to Markers (Deliverable 1)

May 27 - June 9 (2 weeks)	- Work on, complete, and bugfix Linked Strips (Deliverable 2)
June 10 - June 23 (2 weeks)	- Work on, complete, and bugfix Active Channels (Deliverable 3)
June 24 - July 7 (2 weeks)	- Work on, complete, and bugfix Preview Snapping (Deliverable 4)
July 8 – July 28 (3 weeks)	- Reserve time for a buffer period in case previous features take longer to complete than expected - If all tasks are completed and tested early, spend time on other features - Clean up code where necessary - Write end user documentation
July 29 – Aug 19(3 weeks)	- Perform final bug fixing and testing - Finish up end user documentation - Prepare final submission

The schedule above is subject to change — if I find deliverables to be more complex than I expected, I should have ample buffer time to compensate. On the other hand, if I've completed everything ahead of schedule, I will start additional tasks.

5/25/24 Update: I have spoken with my mentor and have decided to tackle Preview Snapping first, as it seems to be the most unambiguous feature to implement and wouldn't require much input from the UI team.

Bio

My name's John. I'm currently pursuing an M.S. in computer science at the University of Minnesota, and next year will be my second year of the program. In my free time, I draw and animate. I've been using Blender since 2017, using it to model objects for personal projects — it has always felt like forward-thinking software to me, and I don't know what I'd do without it. Python, C, and C++ are my strongest languages -- I helped TA a class on machine architecture for 4 years where we primarily used C, and I've also created many toy engines and image editors for graphics programming classes in C++. As this is my first time meaningfully contributing to open-source development, I am excited for the chance to get involved with the Blender Foundation.

- I fixed a bug where wireframe display did not appear in the sequencer, which was merged to main: https://projects.blender.org/blender/blender/pulls/119811
- A report I created for a regression crash in the sequencer: https://projects.blender.org/blender/blender/issues/120155
- I will also be drafting a PR for marker snapping support in the sequencer between now and May 1st (as mentioned in my schedule).