Dealing with jit.qt.movie

Basic Operation

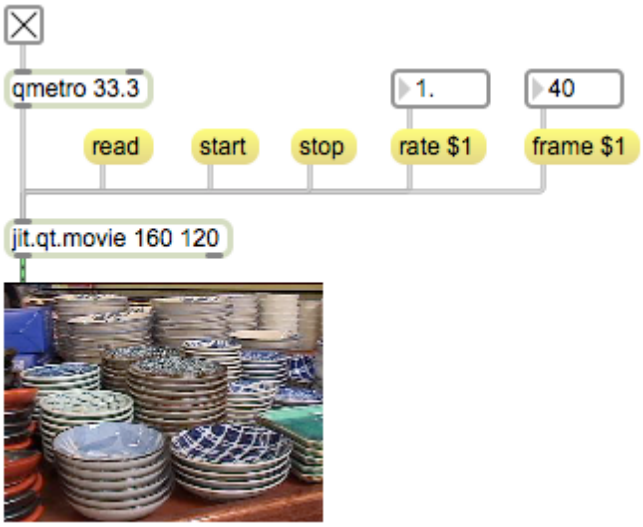


Figure 1.

Jit.qt.movie will be the source of media in most of your patches. In a nutshell, it is the QuickTime movie application embedded in Max. You can get everything most people need from jit.qt.movie with just five commands. Jit.qt.movie is initialized with the dimensions of the output matrix. Media of any size will be interpolated to these dimensions. The type will always be 4 char. (Alpha values are respected, but will almost always be set to 1. This is determined by the program that created the movie.)

Read

Read loads in a movie. It will take anything that the real QuickTime player will show. If you are on a Mac and need windows formats you will want Windows Components for QuickTime from Flip4Mac or Microsoft. This is a free download that lets you view WMVs and so forth. If you pay a bit, you will gain the ability to convert media from one format to another. If you want to clear the movie, use the message *dispose*. Read opens the movie file and loads a couple of frames into memory, but the movie will stream from the disk—if you need to stream several movies at once from something like a flash drive, performance may be poor. The *loadram* command will copy the movie into memory for rapid access. When the copy is completed, either loadram 1 (successful) or loadram 0 (unsuccessful) will be sent out the right outlet.

Start and Stop

Movies begin playing when loaded, unless the attribute *@autostart* is set to 0. Matrices are output when bangs are received, but if bangs are interrupted, the movie continues playing. When the bangs begin again, it will be later in the

movie. The *stop* command will pause the movie and the current frame will be output repeatedly. The *start* command puts the movie in motion again.

Rate

Rate controls the rate of playback. This is a float—1.0 is the normal speed. It is the nature of movies that slowing down results in jerky motion. Negative numbers will show the movie backwards.

Frame

The *frame* command will cue the movie up to a desired frame. This uses a quick algorithm to find the frame, which for obscure reasons occasionally goes wrong with long movies. If the correct frame number is critical, there is a more precise but slower command *frame_true*. The command *jump nn* will jump forward nn frames. Jump with a negative argument will jump back. There is a corresponding *jump_true*.

Looping and the Playbar

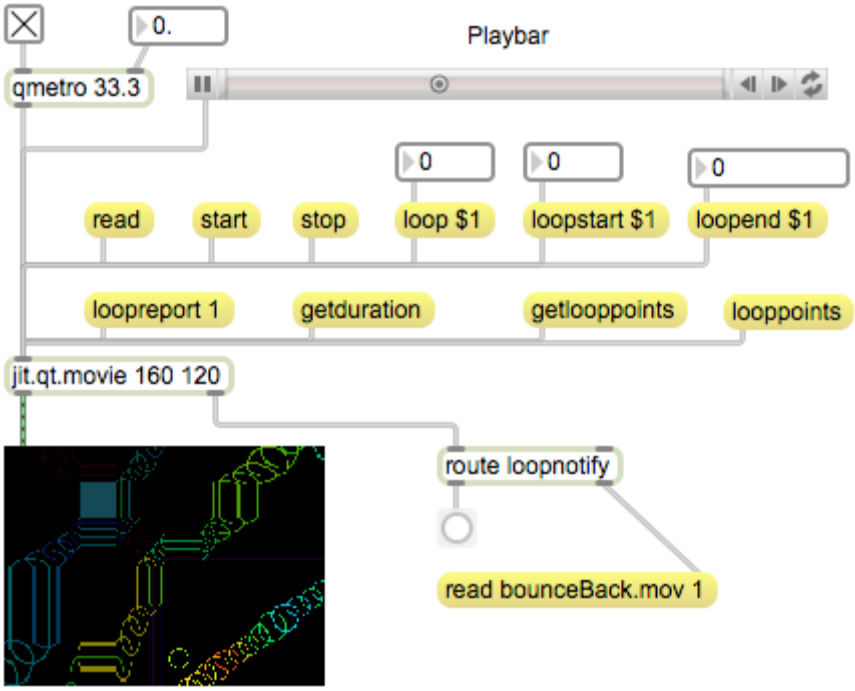


Figure 2.

Movie playback can also be controlled (in Max 6) by a widget called the playbar, which was originally designed to control a primitive movie object called iMovie[1]. Usually we place the playbar below the jit.pwindow, but I have shown it above to clarify the relationship with the jit.qt.movie object. The playbar is patched to the jit.qt.movie inlet as shown. This instructs the two objects to link up behind the scenes so the playbar knows where the movie is and the movie knows what the playbar wants it to do. Actually, all you really need to show a movie is the

[1] This was long before the apple product of the same name.

playbar and a read message, even the metro is unnecessary, because jit.qt.movie will automatically output a matrix for each frame. However, the bangs are required to show movie cueing.

The button at the left end of the playbar controls play/pause. As the movie plays the circle moves and shows progress. When playback is paused, dragging the circle will recue the movie— the metro will then display the current frame. The triangular widgets near the right step back and forward by 10% of the movie duration. The arrows at the right indicate the loop mode. The default mode is simple looping, but single play and back and forth modes are available. The loop start and end can be set by dragging the faint vertical lines. Looping can also be controlled by commands:

Loop setup

The *loop* command can take values of 0 (no loop) 1 (loop) and 2 (loop back and forth). Loop 3 will set single play between the loop points.
Looping occurs between *loopstart* and *loopend* which are specified in QuickTime units. These are arbitrary time units assigned to each movie. The default size of a unit is 600 per second, but a different size can be chosen when a movie is first created. It's good to know the duration of a movie when you set loops, so use the *getduration* command to find out. The message **duration nn** will be sent from the right outlet. Dividing this by 600 will probably get you the length in seconds, but perhaps not. That's covered in more detail in the next section.

The *looppoints* command can be used to change loop start and loop end together. Looppoints with no arguments will set looping to the entire length of the movie.

If the attribute *@loopreport* is set to 1, the object will output the message **loopnotify** from the right outlet when the movie loops. This can be used with a route object to get a bang as shown in figure 2.

More About Time

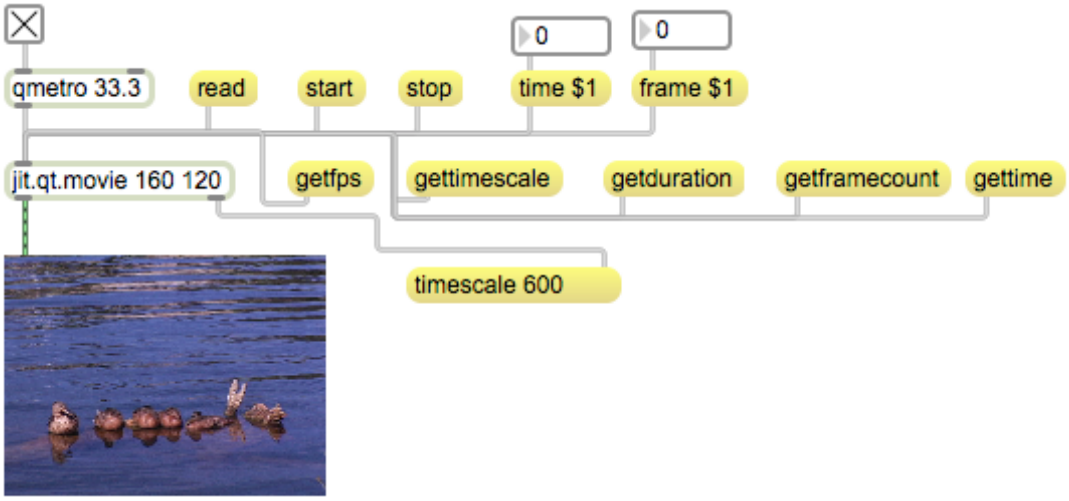


Figure 3.

Time in a QuickTime movie is measured in QuickTime units, which defaults to 600 per second. This is intended to avoid the confusion caused by differing frame rates, sample rates and so forth in the various media that can be in a single movie. I suspect 600 was chosen because the common rates of 24, 25 and 30 fps are simple factors. However, software movie editors allow a choice, and you may encounter other size units. For instance, the countdown movie in the Max distribution has a timescale of 24

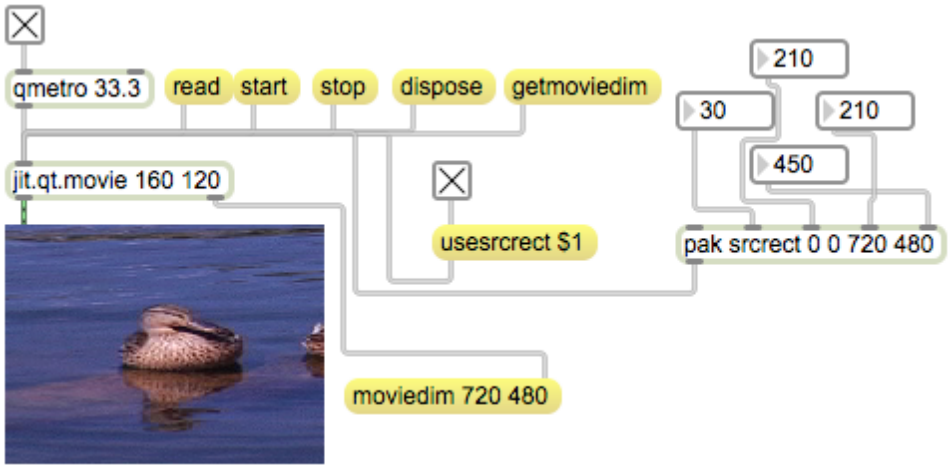
There are several commands that will solve the mystery of time in a movie

- *Gettimescale* will return the number of QT units per second in the movie.
- *Getduration* will return the length of the movie in QT units. Once you know the time scale, you can accurately figure out the length in seconds.
- *Getframecount* will return the number of frames in the movie. You can use this to calculate the number of QT units per frame, but the duration does not have to be an exact multiple of whole frames, so round up.
- *Gettime* will return the current place of the play point in the movie in QT units.
- *Getfps* will return the frame rate of the movie.

Now you can use *time* or *frame* to cue into a movie and know exactly where you are.

You can adjust the playing time with a *scale* command. The command takes four arguments: track number, start of section to scale, end of section to scale, and target value for new end of section. For instance, with a movie that was 240 units[2] long, *scale 1 0 240 120* resulted in a movie that played in half the time. The command scale 1 0 240 480 doubled the length of the movie. This is done by skipping frames or playing frames twice—there is no attempt to smooth the motion.

Zoom in on a Movie



[2] The one I tried had a timescale of 24.

Figure 4.
Jit.qt.movie has source and destination dimension commands, but they are slightly different from the jit.matrix versions. To use these, you need to know the native dimensions of the movie, which are available with the *getmoviedim* command.

Source

Source dimensions are specified with the coordinates we used for rectangles in jit.lcd—Left, Top, Right, Bottom. The command *srcrect* with these arguments will set an area of the movie to bring out in a matrix. *Usesrcrect* turns the feature on. These values are related to the movie's native dimensions, not the arguments in jit.qt.movie. One nice application of this feature is we can get the full resolution of just part of the image in a small window. For comparison, figure 5 shows the result of importing the image at low resolution then zooming in using a matrix:

Figure 5.

Destination

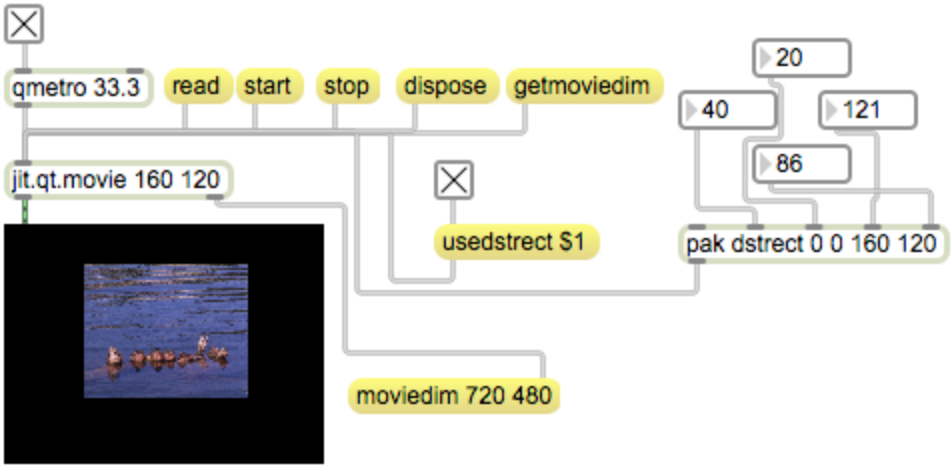


Figure 6.
The destination commands are similar: *usedstrect* and *dstrect* with Left, Top Right Bottom arguments. If you want to get the movie inset in black like figure 6, you need to have the destinations set up before you load the movie. *Dispose* is the only way to get a black matrix after a movie has been shown.

Movie Tracks

QuickTime movies are complicated structures make up of individual clips of media organized in tracks. It's actually much like a pro tools or final cut project, with a master document and a folder full of media files. A QuickTime movie may be self-contained, with all of the media wrapped up in a single file, or it may just have references to files elsewhere on your computer[3]. There's no way to tell just by looking at the file- you just have to make sure to check the correct option when creating the movie.

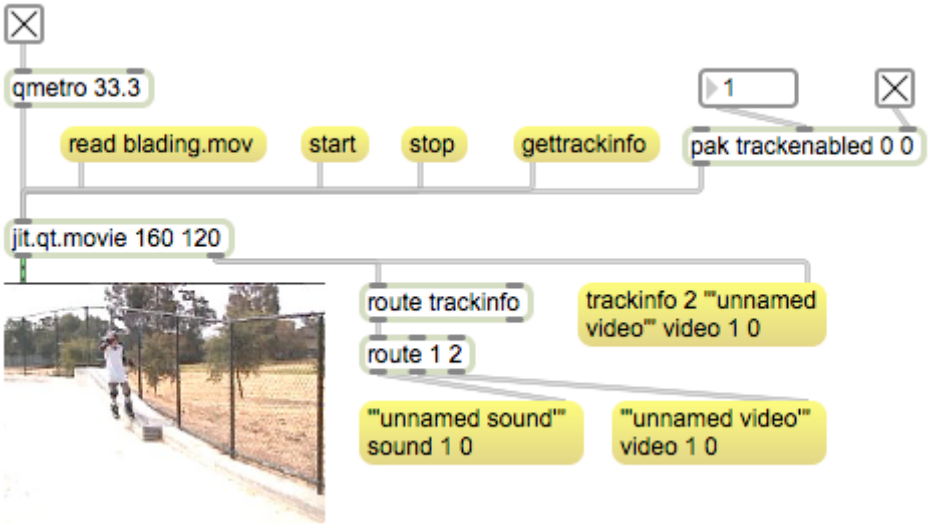


Figure 7.

You can discover what tracks are in a movie with the *gettrackinfo* command. There will be a separate trackinfo message for each track, with the number, name, enabled state, and layer assignment[4] for each. (If you add a number to gettrackinfo, there will only be a message for the appropriate track.) In figure 7, I have obtained trackinfo for the Blading.mov movie from the Max patches folder. This has two tracks, one audio and one video. Usually all we want to do with tracks is enable or disable them, and figure 7 shows a simple way to do that with the *trackenabled* command. The arguments are track number and 1 for enabled or 0 for disabled.

Movie Audio

When a quicktime movie has audio, that audio plays directly to the computer sound output, not Max. The overall audio volume can be controlled with a *vol* command, with a number from 0 to 1. Audio can be controlled for individual tracks by sending commands to the proper track. They take the form command tk nn, where tk is the track and nn is the value.

---

[3] In official documentation, Apple refers to the container file as a movie and the self contained version as a QuickTime movie, a distinction too subtle for me. I often get movies I can't play because they are missing media files.

[4] There can be more than one visible track, say text and video, or partially transparent video and fixed background tracks. The track with the lower layer value is in front.

- *Trackvol* has the range 0 to 1.0.
- *Trackpan* has the range -1 to 1.0. 0.0 is center pan.
- *Tracktreble* is a mile EQ with a range of -1 to 1.
- *Trackbass* has a range of -1 to 1.

There are corresponding get- functions for these.

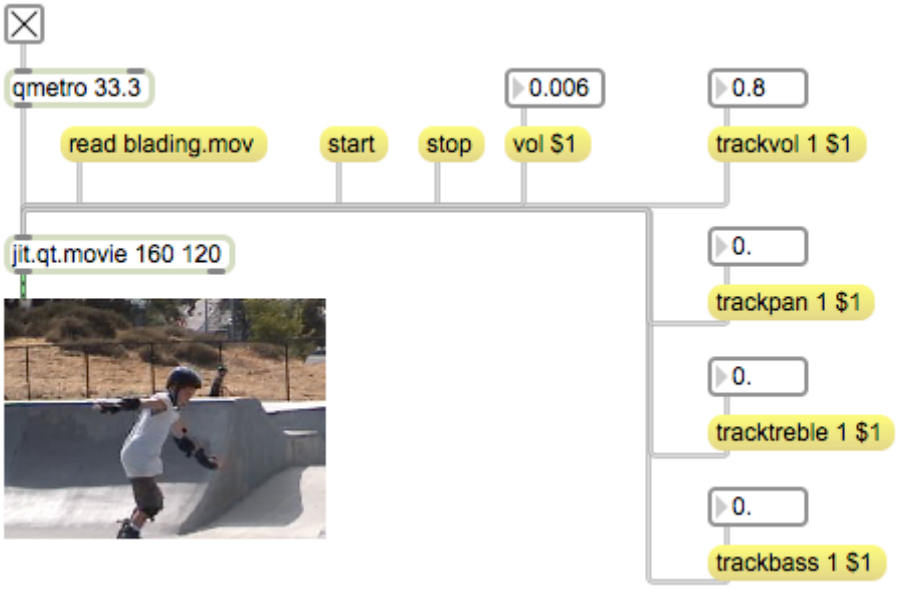


Figure 8.

Figure 9.

Movie audio can be routed into MSP with the spigot~ object. Spigot~ requires a unique name as an argument. The command *soc name* to jit.qt.movie will cause the audio track to be exported to spigot~, which will play it in synchronization with the movie. Once this is done the direct audio will be shut off and the track audio commands to jit.qt.movie will no longer function. The soc command with no name will restore audio to the computer output. Incidentally, jit.qt.movie to spigot~ is the only way to play mp3 files in Max.

Of course many movies do not have any sound. If you want to use jit.qt.movie to add a sound track to a movie, follow this procedure.

- Read the movie.
- Determine the duration with getduration.
- Apply this message: *insert dialog track 1 0* <duration> *-1 0* <duration>
- You should now hear the sound as the movie plays.

Saving and exporting

Once you have modified a movie, you can use *savemovieas* command to make a copy with your changes. This creates a movie with no media—it cannot be moved to another environment, but will work fine as long as the constituent files are undisturbed. To get a self-contained movie use the command *export fulldialog*. This will open a window that lets you pick export options, name the file and save it.

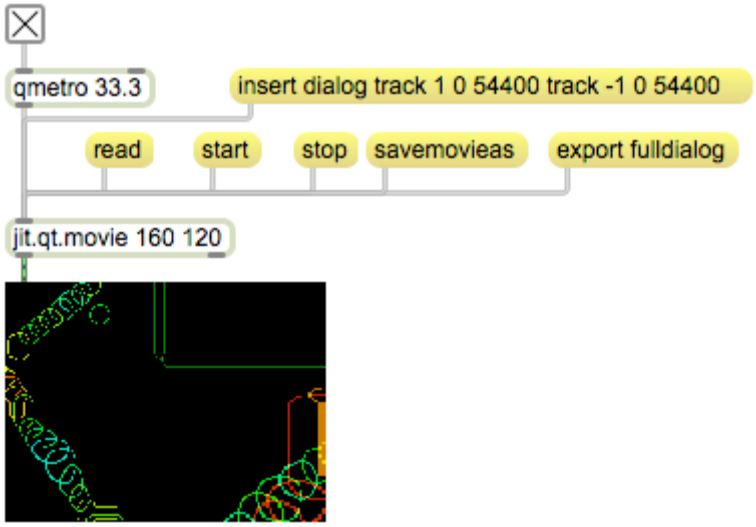


Figure 10.

There's a lot more to jit.qt.movie than is covered here. Study of help and reference files with a fair amount of experimentation will reveal how to add all kinds of tracks, cut and paste within tracks, apply effects, and generally be creative with the quicktime system.