Usada Pekora BGM animation synchronization

by Steven “Baltakatei” Sandoval

Email: `baltakatei@gmail.com`
Web: `reboil.com`

2023-06-09

v0.0.3

1 Summary

Pekora Usada (ja: 兎田ぺこら; hep: *usada pekora*) is a VTuber who, as of 2020, frequently uses a looping background audio loop titled「たぬきちの冒険」while broadcasting their livestream on YouTube. On 2020-08-16, an animator publishing under the Twitter usename `@kaynimatic`, published an animation showing a walk animation of Pekora’s character set to the aforementioned audio loop. Later, other versions of this video animation were uploaded to YouTube[1] but which did not attempt to synchronize the walking animation with the music tempo.

This paper documents such a synchronization attempt.

2 Background

2.1 Givens

The music clip of たぬきちの冒険 has a tempo of 132.0 bpm ($b = 132.0\,\frac{\text{beats}}{\text{min}}$).[2]

The Kaynimatic animation shows Pekora in a 10-frame walk cycle ($c'_{\text{in}} = 10\,\frac{\text{frames}}{\text{cycle}}$) padded in a staggered fashion in the Twitter video into a 27-frame loop.[3] During each walk cycle, Pekora takes two steps, meaning two beats occur during each 10-frame cycle (i.e. $b' = 2\,\frac{\text{beats}}{\text{cycle}}$).

3 Methodology

3.1 Frame extraction

FFmpeg was used to extract the frames from the Twitter video (`input.mp4`).

```
$ ffmpeg -i input.mp4 ./frames/%04d.png
```

Then, the resulting image files (`0001.png`, `0002.png`, …, `0432.png`) were then manually pruned to include only the 10 unique frames.

3.2 Synchronization strategy

Assuming the music is to be played as-is, in order to create a video animation set to the 132.0 bpm music in which each 10 frame walk cycle is mapped across two beats, two strategies may be used:

- Adjust the video framerate.
- Pad the video frames.

[1] GKen. (2020-10-06). *Youtube*. “兎田ぺこら Usada Pekora BGM (1 HOUR) [60 FPS]”. ID: `RYhKUKzD6IQ`.

[2] MAKOOTO. (2019-01-15). *DOVA-SYNDROME*. https://dova-s.jp/bgm/play10441.html .

[3] Yu, Kay. (2020-08-16). *Twitter*. https://twitter.com/kaynimatic/status/1294982862710611968 .

The time required for two beats (one walk cycle) is:

$$
\begin{aligned}
t' &= \frac{b'}{b} \\
&= (b') \cdot \left(\frac{1}{b}\right) \\
&= \left(\frac{2\,\text{beats}}{\text{cycle}}\right) \cdot \left(\frac{\text{min}}{132\,\text{beats}}\right) \cdot \left[\left(\frac{60\,\text{sec}}{\text{min}}\right)\right] \\
&= \left(\frac{2 \cdot 60}{132}\right) \cdot \left(\frac{\text{sec}}{\text{cycle}}\right) \\
t' &= \left(\frac{10}{11}\right) \frac{\text{sec}}{\text{cycle}} \\
t' &= 0.\overline{9090}\,\frac{\text{sec}}{\text{cycle}}
\end{aligned}
$$

Where:

$$
\begin{aligned}
t' &: \text{ time per animation cycle} \left[\frac{\text{sec}}{\text{cycle}}\right] \\
b' &: \text{ beats per cycle} \left[\frac{\text{beats}}{\text{cycle}}\right] \\
b &: \text{ tempo} \left[\frac{\text{beats}}{\text{min}}\right]
\end{aligned}
$$

If video framerate were completely adjustable, the minimum framerate (r_{min}) for a 10 frame looping animation ($c'_{\text{in}} = 10\,\frac{\text{frames}}{\text{cycle}}$) would be:

$$
\begin{aligned}
r_{\text{min}} &= \frac{c'}{t'} \\
&= (c'_{\text{in}}) \cdot \left(\frac{1}{t'}\right) \\
&= \left(\frac{10\,\text{frames}}{\text{cycle}}\right) \cdot \left(\frac{\text{cycle}}{\frac{10}{11}\,\text{sec}}\right) \\
&= \frac{(10\,\text{frames})}{\left(\frac{10}{11}\,\text{sec}\right)} = \left(\frac{10 \cdot 11}{10}\right) \frac{\text{frames}}{\text{sec}} \\
r_{\text{min}} &= 11\,\frac{\text{frames}}{\text{sec}}
\end{aligned}
$$

Where:

$$
\begin{aligned}
c'_{\text{in}} &: \text{ input frames per cycle} \left[\frac{\text{frames}}{\text{cycle}}\right] \\
r_{\text{min}} &: \text{ minimum framerate} \left[\frac{\text{frames}}{\text{sec}}\right]
\end{aligned}
$$

If a more conventional output framerate such as 60 fps were to be used (i.e., $r_{\text{out}} = 60\,\frac{\text{frames}}{\text{sec}}$), each input frame could be displayed multiple times as duplicate frames. In order to calculate how many times an input frame would need to be duplicated in the output video of a given frame rate (r_{out}), a first step would be calculating how many output frames (c'_{out}) would be displayed during the time (t') of a single animation cycle. The next step would involve dividing this output frame count by the total number of available input frames (c'_{in}), yielding the number of times each input frame must be duplicated in the output.

The first step calculation of c'_{out} using the example is as follows:

$$\begin{aligned} c'_{\text{out}} &= t' \cdot r_{\text{out}} && (1)\\ &= \left(\frac{10}{11}\frac{\text{sec}}{\text{cycle}}\right)\cdot\left(60\,\frac{\text{frames}}{\text{sec}}\right) && (2)\\ c'_{\text{out}} &= \frac{600}{11}\frac{\text{frames}}{\text{cycle}} && (3)\\ c'_{\text{out}} &= 54.\overline{5454}\,\frac{\text{frames}}{\text{cycle}} && (4) \end{aligned}$$

Where:

$$c'_{\text{out}} \;:\; \text{output frames per cycle}\left[\frac{\text{frames}}{\text{cycle}}\right]$$

The given values of t' and r_{out} do not permit c'_{out} to take on an integer value. Therefore, if the second step of dividing by $10\,\frac{\text{frames}}{\text{cycle}}$ were executed here, the result would alos be a non-integer value. Since output frames cannot be displayed for fractional amounts of time some approximation or change to assumptions must be made; in order of decreasing severity, these changes may include:

- Change the number of input frames. Requires reanimating the original work.
- Change the given tempo. Requires adjusting music tempo.
- Dynamically duplicate input frames to output frames. Requires giving up perfect frame-to-beat timing but preserves tempo and overall rhythm.

3.3 Procedure

In this example, dynamically adjusting the number of times each input frame is duplicated in the output will be the method used. Therefore, the second step is of calculating the number of duplicate frames becomes a dynamic procedure that varies depending upon a running sum of fractional remainders of $\frac{c'_{\text{out}}}{c'_{\text{in}}}$ evaluations. This procedure can be described as follows:

- Calculate $\frac{c'_{\text{out}}}{c'_{\text{in}}}$.
- Store the quotient and fractional remainder separately. (e.g. for $\frac{60}{11} = 5 + \frac{5}{11}$, "5" is the quotient, "$\frac{5}{11}$" is the fractional remainder).
- Use the quotient as the integer number of duplicates of the current input frame to append to the growing output set of frames.
- Add the remainder to a running sum.
- If the sum is greater than 1.0, then add an additional frame and decrement the sum by 1.0.
- Repeat for the next input frame.

In practice, some additional considerations make for cleaner output (i.e. loops cleanly).

- Assemble a set of output frames containing d cycles in which each input frame is duplicated, on average, $d\cdot\frac{c'_{\text{out}}}{c'_{\text{in}}}$ times.
- Adjust $d(n)$, so that $d\cdot\frac{c'_{\text{out}}}{c'_{\text{in}}}$ yields a value sufficiently near an integer value to an acceptable margin.

Adjustments to $d(n)$ may be calculated by varying n using equation 5:

$$\begin{aligned} d(n) &= n \cdot \frac{c'_{\text{out}}}{c'_{\text{in}}} \qquad (5) \\ &= \frac{(11\,\text{dupes}) \cdot c'_{\text{out}}}{c'_{\text{in}}} \qquad (6) \\ &= \frac{(11\,\text{dupes}) \cdot \left(\frac{600}{11}\right) \frac{\text{frames}}{\text{cycle}}}{\left(10\,\frac{\text{frames}}{\text{cycle}}\right)} \\ d(n) &= 60\,\text{dupes} \end{aligned}$$

Where:

d : number of times an input frame is duplicated in the output [dupes]
n : duplication factor

In this example, an appropriate value for n is 11 as shown in equation 6. This is because "11" since it rests in the denominator of $c'_{\text{out}} = \frac{600}{11} \frac{\text{frames}}{\text{cycle}}$ which is a clean fraction.

In practice, every input frame is not guaranteed to be duplicated the same number of times in the output as each other input frame (i.e. when the remainder sum grows above 1.0 and an additional frame is added). In this example, the minimum number of duplicates is $\frac{c'_{\text{out}}}{c'_{\text{in}}} = 5.\overline{4545} \cong 5$ with some input frames occasionally needing to be duplicated 6 times to cover the remainder in order to maintain synchronization of the animation to the music tempo.

In the example, 11 full cycles requires precisely 600 output frames. Therefore, a clean loop will last $(11\,\text{cycles}) \cdot t' = \left[\left(\frac{10}{11}\right) \frac{\text{sec}}{\text{cycle}}\right] \cdot (11\,\text{cycles}) = 10\,\text{seconds}$. Using a different output video framerate will result in a different selection of n that may change the clean loop length, especially with non-integer frame rates such as $59.94\,\frac{\text{frames}}{\text{sec}}$.

3.4 Program implementation

A Bash script was written to execute the procedure described earlier. A copy of the program may be found at `./exec/assemble_frames.sh` in the repository containing this document.[4]

The script is basically a function defined to take the following parameters into account in order to transform c_{in} sequential input image files into a set of the minimum count of looping sequential image files (c_{out}) that could be used to construct a video in which music rhythm matches the animation rhythm. The script output is a set of PNG files within a specified output directory.

b : tempo $\left[\frac{\text{beats}}{\text{min}}\right]$
r_{out} : output video framerate $\left[\frac{\text{frames}}{\text{sec}}\right]$
c'_{in} : total input frame count [frames]
b' : beats per cycle $\left[\frac{\text{beats}}{\text{cycle}}\right]$
R_{cycle} : input frames per beat $\left[\frac{\text{frames}}{\text{beat}}\right]$
p_{in} : path to directory containing input frame files
p_{out} : path to directory for output frame files
n : duplication factor

[4] See https://reboil.com/mediawiki/BK-2023-04 .

Executing the program in a BASH shell with arguments relevant to the example looks like:

```
$ mkdir ./data/oframes
$ /bin/bash ./exec/assemble_frames.sh 132.0 60 10 2 5 \
  ./WKRM/iframes/ ./WKRM/oframes/ 11; ls ./oframes/
```

Arguments are:

- `132.0` : tempo $\left[\frac{\text{beats}}{\text{minute}}\right]$
- `60` : output video framerate $\left[\frac{\text{frames}}{\text{sec}}\right]$
- `10` : total input frame count [frames]
- `2` : beats per animation cycle $\left[\frac{\text{beats}}{\text{cycle}}\right]$
- `5` : input frames per beat $\left[\frac{\text{frames}}{\text{beat}}\right]$
- `./WKRM/iframes/` : path to directory containing input frame files
- `./WKRM/oframes/` : path to directory for output frame flies
- `11` : duplication factor (how many animation loops to output)

Once the output directory has been populated with the output frames (in this example, 11 loops produce 600 output frames), video editing software such as KDENLIVE[5] may be used to import the PNG files as an image sequence in a video configured for 60 $\frac{\text{frames}}{\text{sec}}$. The 132.0 $\frac{\text{beats}}{\text{min}}$ looping audio track may also be added at this stage.

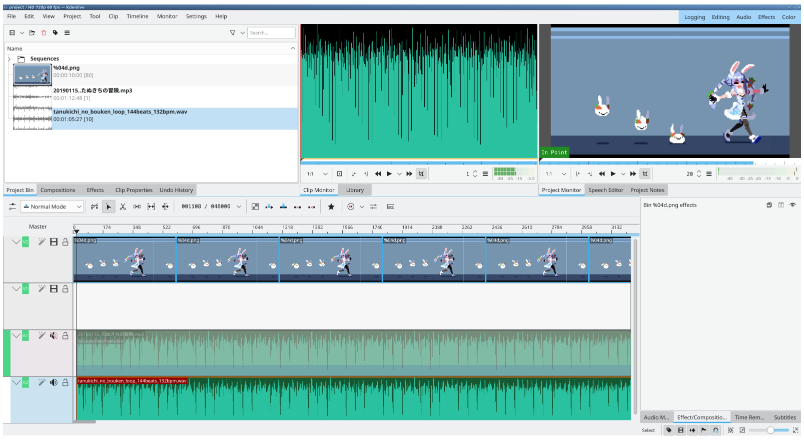
project / HD 720p 60 fps — Kdenlive
File Edit View Project Tool Clip Timeline Monitor Settings Help
Logging Editing Audio Effects Color
Name
Sequences
%04d.png
00:00:10:00 [80]
20190115..たぬきちの冒険.mp3
00:01:12:48 [1]
tanukichi_no_bouken_loop_144beats_132bpm.wav
00:01:05:27 [10]
In Point
Project Bin Compositions Effects Clip Properties Undo History
Clip Monitor Library
Project Monitor Speech Editor Project Notes
Normal Mode
001108 / 048000
Bin %04d.png effects
Master
%04d.png
tanukichi_no_bouken_loop_144beats_132bpm.wav
Audio M... Effect/Compositio... Time Rem... Subtitles
Select

Figure 1. A screenshot of the editing GUI in KDENLIVE `v23.04.1`.

[5]. "Kdenlive Free and Open Source Video Editor". *kdenlive.org* https://kdenlive.org/ .