python-twitch-stream Documentation

Release 0.0.dev1

Jonas Degrave

October 26, 2017

Contents

Python-twitch-stream is a simple lightweight library, which you can use to send your python video to twitch and react with the chat in real time. Its main features are:

- Supports sending of audio and video in a thread safe way to your twitch channel.

- Allows to interact with the chat of your channel by sending chat messages and reading what other users post.

There is a complete tutorial available at Tutorial.

Installation

In short, you can install the latest stable version over pip.

```
pip install python-twitch-stream
```

Make sure to also install a recent ffmpeg version:

```
sudo add-apt-repository ppa:mc3man/trusty-media
sudo apt-get update && sudo apt-get install ffmpeg
```

The ffmpeg library needs to be very recent (written in october 2015). There are plenty of bugs when running a stream using older versions of ffmpeg or avconv, including but not limited to 6GB of memory use, problems with the audio and synchronization of the audio and the video.

Or alternatively, install the latest python-twitch-stream development version via:

```
pip install git+https://github.com/317070/python-twitch-stream
```

Support

For support, please use the github issues on the repository.

Contents

twitchstream.chat

This file contains the python code used to interface with the Twitch chat. Twitch chat is IRC-based, so it is basically an IRC-bot, but with special features for Twitch, such as congestion control built in.

class twitchstream.chat.**TwitchChatStream**(*username*, *oauth*, *verbose=False*)

The TwitchChatStream is used for interfacing with the Twitch chat of a channel. To use this, an oauth-account (of the user chatting) should be created. At the moment of writing, this can be done here: https://twitchapps.com/tmi/

> **Parameters**
>
> - **username** (*string*) – Twitch username
>
> - **oauth** (*string*) – oauth for logging in (see https://twitchapps.com/tmi/)
>
> - **verbose** (*boolean*) – show all stream messages on stdout (for debugging)

connect()

Connect to Twitch

join_channel(*channel*)

Join a different chat channel on Twitch. Note, this function returns immediately, but the switch might take a moment

> **Parameters** **channel** – name of the channel (without #)

send_chat_message(*message*)

Send a chat message to the server.

> **Parameters** **message** – String to send (don't use n)

twitch_receive_messages()

Call this function to process everything received by the socket This needs to be called frequently enough (~10s) Twitch logs off users not replying to ping commands.

> **Returns** list of chat messages received. Each message is a dict with the keys ['channel', 'user-name', 'message']

twitchstream.outputvideo

This file contains the classes used to send videostreams to Twitch

class `twitchstream.outputvideo.`**`TwitchBufferedOutputStream`**(**args*, ***kwargs*)

 This stream makes sure a steady framerate is kept by buffering frames. Make sure not to have too many frames in buffer, since it will increase the memory load considerably!

 Adding frames is thread safe.

 `get_audio_buffer_state`()

  Find out how many audio fragments are left in the buffer. The buffer should never run dry, or audio and video will go out of sync. Likewise, the more filled the buffer, the higher the memory use and the delay between you putting your frame in the stream and the frame showing up on Twitch.

  :return integer estimate of the number of audio fragments left.

 `get_video_frame_buffer_state`()

  Find out how many video frames are left in the buffer. The buffer should never run dry, or audio and video will go out of sync. Likewise, the more filled the buffer, the higher the memory use and the delay between you putting your frame in the stream and the frame showing up on Twitch.

  :return integer estimate of the number of video frames left.

 `send_audio`(*left_channel*, *right_channel*, *frame_counter=None*)

  Add the audio samples to the stream. The left and the right channel should have the same shape.

  Parameters

   • **left_channel** (*numpy array with shape (k,) containing values between -1.0 and 1.0. l can be any integer*) – array containing the audio signal.

   • **right_channel** (*numpy array with shape (k,) containing values between -1.0 and 1.0. l can be any integer*) – array containing the audio signal.

   • **frame_counter** (*int*) – frame position number within stream. Provide this when multi-threading to make sure frames don't switch position

 `send_video_frame`(*frame*, *frame_counter=None*)

  send frame of shape (height, width, 3) with values between 0 and 1

  Parameters

   • **frame** (*numpy array with shape (height, width, 3) containing values between 0.0 and 1.0*) – array containing the frame.

   • **frame_counter** (*int*) – frame position number within stream. Provide this when multi-threading to make sure frames don't switch position

class `twitchstream.outputvideo.`**`TwitchOutputStream`**(*twitch_stream_key*, *width=640*, *height=480*, *fps=30.0*, *ffmpeg_binary='ffmpeg'*, *enable_audio=False*, *verbose=False*)

 Initialize a TwitchOutputStream object and starts the pipe. The stream is only started on the first frame.

  Parameters

   • **twitch_stream_key** –

   • **width** (*int*) – the width of the videostream (in pixels)

   • **height** (*int*) – the height of the videostream (in pixels)

   • **fps** (*float*) – the number of frames per second of the videostream

   • **enable_audio** (*boolean*) – whether there will be sound or not

   • **ffmpeg_binary** (*String*) – the binary to use to create a videostream This is usually ffmpeg, but avconv on some (older) platforms

- **verbose** (*boolean*) – show ffmpeg output in stdout

reset()
 Reset the videostream by restarting ffmpeg

send_audio(*left_channel*, *right_channel*)
 Add the audio samples to the stream. The left and the right channel should have the same shape. Raises an OSError when the stream is closed.

> **Parameters**
>
> - **left_channel** (*numpy array with shape (k,) containing values between -1.0 and 1.0. k can be any integer*) – array containing the audio signal.
>
> - **right_channel** (*numpy array with shape (k,) containing values between -1.0 and 1.0. k can be any integer*) – array containing the audio signal.

send_video_frame(*frame*)
 Send frame of shape (height, width, 3) with values between 0 and 1. Raises an OSError when the stream is closed.

> **Parameters frame** (*numpy array with shape (height, width, 3) containing values between 0.0 and 1.0*) – array containing the frame.

class twitchstream.outputvideo.**TwitchOutputStreamRepeater**(**args*, ***kwargs*)
 This stream makes sure a steady framerate is kept by repeating the last frame when needed.

> **Note: this will not generate a stable, stutter-less stream!** It does not keep a buffer and you cannot synchronize using this stream. Use TwitchBufferedOutputStream for this.

send_audio(*left_channel*, *right_channel*)
 Add the audio samples to the stream. The left and the right channel should have the same shape.

> **Parameters**
>
> - **left_channel** (*numpy array with shape (k,) containing values between -1.0 and 1.0. k can be any integer*) – array containing the audio signal.
>
> - **right_channel** (*numpy array with shape (k,) containing values between -1.0 and 1.0. k can be any integer*) – array containing the audio signal.

send_video_frame(*frame*)
 Send frame of shape (height, width, 3) with values between 0 and 1.

> **Parameters frame** (*numpy array with shape (height, width, 3) containing values between 0.0 and 1.0*) – array containing the frame.