

## Преобразование видео в \*.gif изображение.

Довольно часто возникает весьма специфичная задача быстрого преобразования большого количества коротких видеороликов в изображения .gif формата, для последующего их удобного хранения, использования и загрузки на различные сайты. В этой работе приводится пример программы, способной быстро и удобно преобразовывать большое количество файлов в видео-форматах в .gif изображения с учетом дополнительных, часто возникающих желаний пользователя, таких как изменения разрешения изображения и частоты кадров в гифке.

### Используемые модули.

Программа написана с помощью разных библиотек, таких как:

- OpenCv, для распознавания видео, извлечения и конвертации кадров в формат JPG;
- Pillow, для создания GIF-анимации из кадров, сохраненных в формате JPG;
- Telebot, которая предоставляет необходимые инструменты для создания и настройки ботов в Telegram, обработки сообщений и отправки ответов пользователям.

Также использованы следующие модули:

- Glob - находит все пути, совпадающие с заданным шаблоном в соответствии с правилами, используемыми оболочкой Unix;
- Shutil - содержит набор функций высокого уровня для обработки файлов, групп файлов, и папок;
- Os - это библиотека функций для работы с операционной системой.

### Пояснение к алгоритму.

Программа принимает на вход видео и загружает его с помощью функции `bot.download_file(file_info.file_path)`. Далее, с помощью написанной функции `convert_mp4_to_jpgs(path)` видео раскладывается на массив изображений в формате .jpg. Потом этот массив конвертируется в .gif анимацию через

написанную функцию `make_gif(gif_path, frame_folder)`. Также реализованы функции для конвертации видео с изменением расширения и количества кадров. Отправка анимации пользователю осуществляется с помощью функции `bot.send_animation(message.from_user.id, gif)`.

### Пояснение к телеграмм-боту.

Телеграмм-бот написан на основе библиотеки `telebot` и предоставляемых ею функций для работы с пользователем, например, `bot.register_next_step_handler(msg, f)`, `msg = bot.reply_to(message, '')` и `bot.send_message(message.from_user.id, '')`.

После запуска с помощью команды `/start` телеграмм-бот предлагает пользователю 3 варианта конвертации видео:

- конвертация без изменений,
- конвертация с изменением расширения,
- конвертация с изменением количества кадров в секунду.

При выборе первого варианта бот просит пользователя отправить ему видео и конвертирует его в `*.gif` анимацию. При втором варианте бот просит пользователя отправить размер нового расширения а далее само видео. При конвертации бот учитывает новое расширение, но сохраняет соотношение сторон. При выборе третьего варианта бот просит отправить желаемое количество кадров в секунду и само видео, а при конвертации учитывает нужное число кадров. При возникновении ошибки бот отправляет пользователю соответствующее сообщение и останавливает конвертацию.

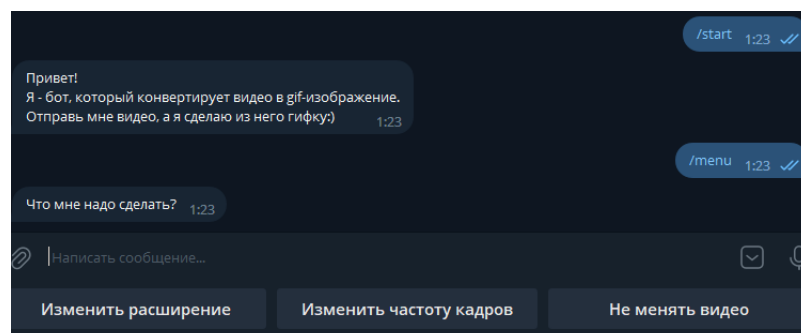
Также реализована команда `/help` с кратким описанием работы бота, а все команды сопровождаются инструкциями и примерами.

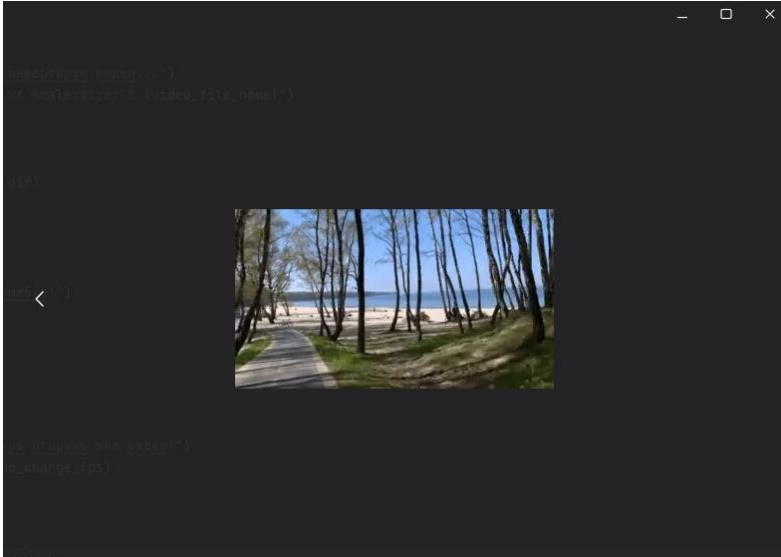
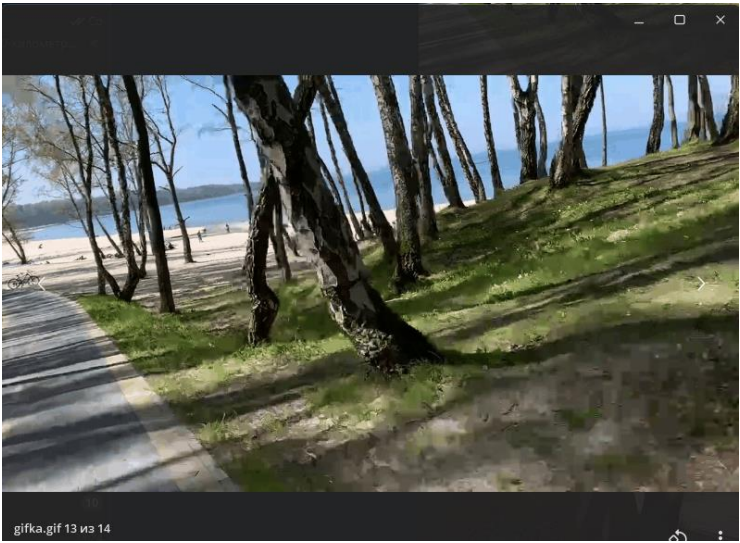
### Таблица функций.

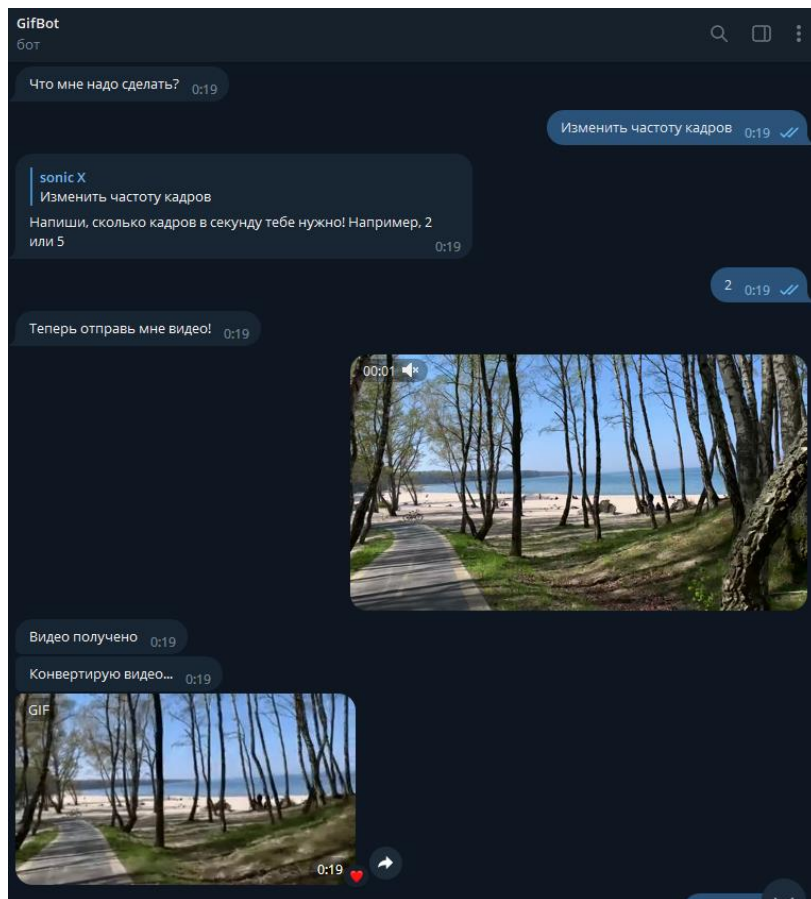
Название функции	Её описание
<code>cv2.VideoCapture(path)</code>	Создает объект захвата видео, который помогает передавать или отображать видео.
<code>video_capture.read()</code>	Возвращает кортеж, где первый элемент — логическое значение, а следующий элемент — фактический видеокادر. Когда первый элемент имеет значение


	True, это означает, что видеопоток содержит кадр для чтения.
<code>video_capture.get(cv2.CAP_PROP_FPS)</code>	Используется для получения важных метаданных, связанных с видеопотоком. Метод принимает единственный аргумент, который соответствуют частоте кадров (CAP_PROP_FPS) или количеству кадров (CAP_PROP_FRAME_COUNT).
<code>cv2.imwrite</code>	Метод используется для сохранения изображения на любое запоминающее устройство. Он сохранит изображение в соответствии с указанным форматом в текущем рабочем каталоге.
<code>glob.glob(f'{frame_folder}/*.jpg')</code>	Возвращает список имен путей, которые находятся в каталоге, который должен быть строкой, содержащей спецификацию пути.
<code>os.remove()</code>	Используется для удаления файла из системы
<code>shutil.rmtree()</code>	Рекурсивно удаляет все дерево каталогов.
<code>bot.polling(non_stop=True)</code>	Запускает бесконечный цикл для получения обновлений из Telegram-бота и передачи их в соответствующие обработчики.
<code>bot.register_next_step_handler(msg, f)</code>	Используется для перехода в другую функцию и ожидания сообщения от пользователя.

## Пример работы.







	gifka (1).gif	Файл "GIF"	12 885 КБ
	gifka.gif	Файл "GIF"	38 269 КБ

## Листинг программы.

```
import os
import telebot
from telebot import types
import cv2
import glob
import shutil
from PIL import Image
```

```
API_TOKEN = '6232705788:AAG0hxSjC928Fq0gJXGGJu5fE7Qy4do6pJM'
```

```
bot = telebot.TeleBot(API_TOKEN)
```

```
size = 0
number = 0
```

```
@bot.message_handler(commands=['start'])
def send_welcome(message: types.Message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    button = types.KeyboardButton("/menu")
    markup.add(button)
    bot.send_message(message.from_user.id, "Привет!\nЯ - бот, который конвертирует видео в gif-изображение."
                                                              "\nОтправь мне видео, а я сделаю из него гифку:", reply_markup=markup)
```

```

@bot.message_handler(commands=['menu'])
def menu(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton('Изменить расширение')
    btn2 = types.KeyboardButton('Изменить частоту кадров')
    btn3 = types.KeyboardButton('Не менять видео')
    markup.add(btn1, btn2, btn3)
    bot.send_message(message.from_user.id, 'Что мне надо сделать?', reply_markup=markup)

@bot.message_handler(commands=['help'])
def send_help(message: types.Message):
    bot.send_message(message.chat.id, "Я конвертирую файлы формата mp4 в файлы формата gif."
                                     "\nДля этого я разбиваю видео на кадры, а потом создаю из них gif"
    анимацию."
                                     "\nЧтобы начать, введи команду '/start' и следуй инструкциям:")

@bot.message_handler(content_types=['text'])
def reply(message: types.Message):
    if message.text == 'Не менять видео':
        msg = bot.reply_to(message, 'Просто отправь мне видео!')
        bot.register_next_step_handler(msg, handle_video)
    elif message.text == 'Изменить расширение':
        msg = bot.reply_to(message, 'Напиши новое расширение! Например, 600 или 200')
        bot.register_next_step_handler(msg, handle_video_with_resize)
    elif message.text == 'Изменить частоту кадров':
        msg = bot.reply_to(message, 'Напиши, сколько кадров в секунду тебе нужно! Например, 2 или
5')
        bot.register_next_step_handler(msg, handle_video_with_fps)
    else:
        bot.send_message(message.from_user.id, 'Ты хочешь от меня слишком многого...')

def handle_video(message):
    try:
        bot.send_message(message.from_user.id, 'Видео получено')
        file_details = message.video
        file_id = file_details.file_id
        file_info = bot.get_file(file_id)
        downloaded_video = bot.download_file(file_info.file_path)
        video_file_name = message.video.file_id + ".mp4"
        with open(video_file_name, 'wb') as saved_file:
            saved_file.write(downloaded_video)
            gif_file_name = 'gifka.gif'

        bot.send_message(message.from_user.id, 'Конвертирую видео...')
        convert_mp4_to_jpgs(video_file_name)
        make_gif(gif_file_name)
        gif = open(gif_file_name, 'rb')
        bot.send_animation(message.from_user.id, gif)
        gif.close()

```

```

    os.remove(video_file_name)
    os.remove(gif_file_name)
except:
    bot.send_message(message.from_user.id, 'Ошибка!')

def convert_mp4_to_jpgs(path):
    video_capture = cv2.VideoCapture(path)
    still_reading, image = video_capture.read()
    frame_count = 0
    if os.path.exists("output"):
        # убираем предыдущие GIF frame файлы
        shutil.rmtree("output")
    try:
        os.mkdir("output")
    except IOError:
        print("Error occurred creating output folder")
        return

    # настраиваемые параметры
    seconds = 0.1
    fps = video_capture.get(cv2.CAP_PROP_FPS) # получаем кадры в секунду
    multiplier = fps * seconds

    while still_reading:
        cv2.imwrite(f'output/frame_{frame_count:05d}.jpg', image)

        # читаем следующее изображение
        frame_id = int(round(video_capture.get(1))) # текущий номер кадра, округленный
        still_reading, image = video_capture.read()

        if frame_id % multiplier == 0:
            still_reading, image = video_capture.read()
            frame_count += 1

def make_gif(gif_path, frame_folder="output"):
    images = glob.glob(f'{frame_folder}/*.jpg')
    images.sort()
    frames = [Image.open(image) for image in images]
    frame_one = frames[0]
    frame_one.save(gif_path, format="GIF", append_images=frames,
                  save_all=True, duration=50, loop=0)

def handle_video_with_resize(message: types.Message):
    global size
    size = message.text.lower()
    msg = bot.send_message(message.chat.id, "Теперь отправь мне видео!")
    bot.register_next_step_handler(msg, handle_and_resize)

def handle_and_resize(message):

```

global size

try:

```
bot.send_message(message.from_user.id, 'Видео получено')
file_details = message.video
file_id = file_details.file_id
file_info = bot.get_file(file_id)
downloaded_video = bot.download_file(file_info.file_path)
video_file_name = message.video.file_id + ".mp4"
with open(video_file_name, 'wb') as saved_file:
    saved_file.write(downloaded_video)
    gif_file_name = 'gifka.gif'

bot.send_message(message.from_user.id, 'Конвертирую видео...')
os.system(f'ffmpeg -i {video_file_name} -vf scale=size:-1 {video_file_name}')
convert_mp4_to_jpgs(video_file_name)
make_gif(gif_file_name)
gif = open(gif_file_name, 'rb')
bot.send_animation(message.from_user.id, gif)
gif.close()
os.remove(video_file_name)
os.remove(gif_file_name)
```

except:

```
bot.send_message(message.from_user.id, 'Ошибка!')
```

def handle\_video\_with\_fps(message):

global number

number = message.text.lower()

msg = bot.send\_message(message.chat.id, "Теперь отправь мне видео!")

bot.register\_next\_step\_handler(msg, handle\_and\_change\_fps)

def handle\_and\_change\_fps(message):

try:

```
bot.send_message(message.from_user.id, 'Видео получено')
file_details = message.video
file_id = file_details.file_id
file_info = bot.get_file(file_id)
downloaded_video = bot.download_file(file_info.file_path)
video_file_name = message.video.file_id + ".mp4"
with open(video_file_name, 'wb') as saved_file:
    saved_file.write(downloaded_video)
    gif_file_name = 'gifka.gif'

bot.send_message(message.from_user.id, 'Конвертирую видео...')
convert_with_fps(video_file_name)
make_gif(gif_file_name)
gif = open(gif_file_name, 'rb')
bot.send_animation(message.from_user.id, gif)
gif.close()
os.remove(video_file_name)
os.remove(gif_file_name)
```

except:



```
bot.send_message(message.from_user.id, 'Ошибка!')
```

```
def convert_with_fps(path):
    global number
    video_capture = cv2.VideoCapture(path)
    still_reading, image = video_capture.read()
    frame_count = 0
    if os.path.exists("output"):
        shutil.rmtree("output")
    try:
        os.mkdir("output")
    except IOError:
        print("Error occurred creating output folder")
        return
    seconds = 0.1
    fps = video_capture.get(number)
    multiplier = fps * seconds

    while still_reading:
        cv2.imwrite(f'output/frame_{frame_count:05d}.jpg', image)

        frame_id = int(round(video_capture.get(1)))
        still_reading, image = video_capture.read()

        if frame_id % multiplier == 0:
            still_reading, image = video_capture.read()
            frame_count += 1

if __name__ == '__main__':
    bot.polling(none_stop=True, interval=0)
```