УДК 004

Разработка программы регулирование скорости звука с помощью языка программирования python

Вихляев Дмитрий Романович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Данная статья содержит описание программы, которая способна регулировать скорость воспроизведения звука. Программа написана на языке программирования python, с использованием модуля wave для обработки звука. Результатом исследования станет готовая программа с подробным описанием её реализации.
Ключевые слова: python , модуль wave, обработка звука.

Development of the sound speed control program using the python programming language

Vikhlyaev Dmitry Romanovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article contains a description of a program that is able to adjust the speed of sound reproduction. This program is written in the python programming language, using the wave module for sound processing. The result of the study will be a ready-made program with a detailed description of its implementation.
Keywords: python, wave module, sound processing.

1 Введение

1.1 Актуальность

Уже долгое время звук служит, переносчиком информации и с каждым годом люди находят к нему новые применения. Однако не всегда он соответствует ожиданиям. Иногда тот, кто передаёт звук, может говорить слишком быстро и невнятно, в таком случае человек может не успеть осмыслить всю предоставляемую информацию. В другом случае тот, кто передаёт информацию, может долго растягивать свой монолог, суть которого можно было бы уложить в два или в три раза быстрее, в этом случае встаёт проблема о временных затратах. На помощь приходят современные технологии обработки звука, с помощью которых каждый может подстраивать звуковую передачу информации под себя. Модуль wave языка программирования python имеет большое количество методов для работы со звуковыми файлами.

1.2Обзор исследований

Н.В. Савченко привёл пример использование архитектуры CUDA для решения задач обработки и анализа звука [1]. А.С. Маринчук создал аудио проигрыватель с помощью microsoft visual studio и библиотеки naudio [2]. Р.В. Семченко и П.А. Еровлев создали аудио плеер на андроид смартфон [3]. Л.Г. Кирьянова провела исследование по теме цифровой обработки аудио- и видеоинформации [4]. С.Д. Киселева, О.С. Недильченко, Ф.А. Линев рассмотрели некоторые методы анализа музыкальных фрагментов [5]. Р.В. Добровольский, Т.А. Ким, А.А. Сотников провели исследования цифровой фильтрации, аудио сигналов в режиме реального времени на базе процессора tms320c5515 [6].

1.3Цель исследования

Цель исследования – используя модуль wave и язык программирования python, реализовать программу способную, изменять скорость воспроизведения звука. На входе программа принимает аудио файл, на выходе создаёт редактированную копию входного файла.

2 Материалы и методы

Для создания программы используется модуль wave, и язык программирования python. В качестве IDE используется Pycharm.

3 Результаты и обсуждения

Для достижения цели необходимо импортировать модуль wave. Данный модуль установлен в python, поэтому его не нужно скачивать. Для регулирования скорости звука будет использоваться переменная CHANGE_RATE, её начальное состояние равно единице. В процессе изменения её в большую сторону звуковой сигнал будет ускоряться пропорционально изменению данной переменной.

Далее, с помощью метода wave.open будет получен исходный файл, для этого в параметры данного метода водятся путь к файлу и режим чтения(rb). Чтобы получить необходимые данные для обработки файла, необходимо присвоить данный метод переменной.

Метод getframes() возвращает частоту дискретизации звука, чаще всего она составляет 44100 герц.

Метод readframes() считывает и возвращает не более n фреймов звука, как объект в байтах.

Метод getnchannels() возвращает количество звуковых каналов, если моно, то 1, если стерео, то 2.

Метод sampwiidth() вращает ширину выборки в байтах (рис. 1).

```
import wave

Change_RATE = 0.5

spf = wave.open('ff.wav', 'rb')
RATE=spf.getframerate()
signal = spf.readframes(-1)
channels=spf.getnchannels()
width=spf.getsampwidth()
```

Рис. 1. Получение данных из файла

После того как данные были получены, нужно их изменить и добавить в новый файл. Для добавления данных в новый файл используется метод wave.open, в качестве параметров указывается путь к файлу, который нужно изменить, если такого файла нет, создаётся новый. Вторым параметром данного метода служит (wb). Для создания нового файла ему необходимо задать параметры. Используя метод setnchannels(), можно задать новому файлу количество каналов.

Метод setsampwidth() задаёт ширину выборки в байтах. С помощью данного метода можно с точностью скопировать звук из другого файла.

Метод setframerate() задаёт частоту дискретизации, в качестве параметров используется значение исходного файла, умноженное на переменную, которая отвечает за скорость воспроизведения звука.

Метод writeframes() задаёт фреймы звукового сигнала, полученные от исходного файла.

Метод close() закрывает все программы открытые на чтение или запись (рис. 2).

```
wf = wave.open('gg.wav', 'wb')
wf.setnchannels(channels)
wf.setsampwidth(swidth)
wf.setframerate(RATE*Change_RATE)
wf.writeframes(signal)
wf.close()
```

Рис. 2. Добавление данных в новый файл

Чтобы проверить программу можно запустить её в аудио плеере и узнать, сколько времени будет играть песня. Сначала необходимо узнать, временную длину оригинального файла.

Рис. 3. Время действия оригинального файла составляет 4:17

Теперь можно установить значения CHANGE_RATE на 0.5, после запуска программы появится новый файл скорость воспроизведения, которой должно быть в 2 раза медленнее. Следовательно, время воспроизведения должно, увеличится вдвое (рис.4).

Рис. 4. Время воспроизведения замедленного в 2 раза фала составляет 8:34

Также можно установить значения CHANGE_RATE на 2, тем самым ускорив воспроизведение в 2 раза. Время воспроизведения должно, уменьшится вдвое (рис. 5).

Рис. 5. Время воспроизведения ускоренного в 2 раза фала составляет 2:08

Таким образом, была написана программа для обработки звукового файла, способная изменять скорость воспроизведения звука без потерь данных.

Библиографический список

1. Савченко Н.В. Использование архитектуры CUDA для решения задач обработки и анализа звука // Вестник научных конференций. 2017. № 1-5 (17). С. 151-152.
2. Маринчук А.С. Создание аудио проигрывателя с помощью microsoft visual studio и библиотеки naudio// Постулат. 2020. № 1 (51). С. 132.
3. Семченко Р.В., Еровлев П.А. Создание аудио плеера на андроид смартфон// Постулат. 2019. № 12 (50). С. 14.
4. Кирьянова Л.Г. Цифровая обработка аудио- и видеоинформации // Информационные технологии. Радиоэлектроника. Телекоммуникации. 2012. № 2-2. С. 225-231.
5. Киселева С.Д., Недильченко О.С., Линев Ф.А. Некоторые методы анализа музыкальных фрагментов // Молодежный научно-технический вестник. 2017. № 3. С. 19.
6. Добровольский Р.В., Ким Т.А., Сотников А.А. Цифровая фильтрация аудио сигналов в режиме реального времени на базе процессора tms320c5515 // Технологии инженерных и информационных систем. 2019. № 1. С. 11-20.
7. Егоров Д.П., Кутуза Б.Г. Распознавание нотной партитуры по цифровому аудио сигналу //Актуальные проблемы современного образования. 2018. Т. 2. С. 70-73.
8. Кузькин В.М., Матвиенко Ю.В., Переселков С.А. Применение интерферометрической обработки для локализации малошумных источников звука //Подводные исследования и робототехника. 2019. № 4 (30). С. 49-57.
9. Мокрецов А.В. Погрешность определения угла, местоположения источника звука микрофонной системой с алгоритмом пространственно-временной обработки сигнала // Инженерный вестник Дона. 2012. № 3 (21). С. 52-54.