УДК 004.94
DOI: 10.18384/2949-5067-2024-1-33-47

ПРЕОБРАЗОВАНИЕ ПИКСЕЛЬНОЙ СТРУКТУРЫ В ЗВУКОВЫЕ ОТОБРАЖЕНИЯ. Часть 2

Ключников С. А., Калашников Е.В.

Государственный университет просвещения
141014, Московская область, г. Мытищи, ул. Веры Волошиной, д. 24,
Российская Федерация

Аннотация
Цель. Выявить связь между визуальным и звуковым восприятием.
Процедура и методы. При помощи объектно-ориентированного программирования (ООП) языка Python ищется способ преобразования визуального (пиксельного) отображения в звуковое отображение. Применяется ряд современных и функциональных библиотек. Используется современные способы «упаковки» всех программных компонентов в один файл для удобной развёртки программы на электронно-вычислительном устройстве (ЭВМ) с любой современной операционной системой (ОС).
Результаты. Создан программный продукт на основе современного ООП языка программирования и функциональных библиотек, позволяющий представить пиксельную структуру визуального изображения в звуковое отображение.
Теоретическая и/или практическая значимость заключается в раскрытии современного способа «упаковки» всех программных компонентов в один файл для удобной развёртки программы на электронно-вычислительном устройстве (ЭВМ) с любой современной операционной системой (ОС). Это позволяет преобразовать цветовое визуальное изображение со множеством оттенков в звуковое отображение.
Ключевые слова: преобразование, программирование, Python, библиотеки, музыка, октавы, фортепиано, цветовая модель, пиксели

CONVERTING A PIXEL STRUCTURE INTO SOUND IMAGINATIONS. Part 2

S. Klyuchnikov, E. Kalashnikov

Federal State University of Education
ulitsa Very Voloshinoi 24, Mytishchi 141014, Moscow Region, Russian Federation

Abstract
Aim. To identify the connection between visual and sound perception.
Methodology. Using Object-Oriented Programming (OOP) of the Python language, we are looking for a way to convert a visual (pixel) display into an audio display. A number of modern and functional libraries are used. Modern methods of “packing” all software components into a

single file are used for convenient program deployment on an electronic computing device (computer) with any modern operating system (OS).

Results. A software product based on a modern OOP programming language and functional libraries has been created that allows you to present the pixel structure of a visual image into an audio display.

Research implications. The significance lies in the disclosure of a modern way of "packing" all software components into a single file for convenient program deployment on an electronic computing device (computer) with any modern operating system (OS).

Keywords: conversion, programming, Python, libraries, music, octaves, piano, color model, pixels

Введение

Рассматривается возможность внесения большего разнообразия в настройку мелодии, полученной с изображения. В программе использовалась одна гармоника Ля минор. Это приводило к тому, что возникало множество подходящих и однотипных композиций [2]. Чтобы разнообразить такую однотипность, необходимо внести изменения в программный компонент кода. Для этого следует разнообразить начальную ноту Ля минор и расширить функционал интервалов. Благодаря этому разнообразие в генерации мелодии станет гораздо шире. Это положительно скажется на уникальности и насыщенности каждой сгенерированной композиции.

1. Расширение возможностей генерации мелодий

Необходимо изменить начальную ноту до Ля минор и расширить диапазон интервалов. Это увеличит возможности для создания мелодии, что сделает каждую сгенерированную композицию более уникальной и насыщенной.

Дня начала настроим процедурную генерацию частоты для любых из вариантов тональности. Для этого сопоставим ноты музыкального инструмента фортепиано с частотами[2].

```
def get_piano_notes():
    octave = ['C', 'c', 'D', 'd', 'E', 'F', 'f', 'G', 'g', 'A', 'a', 'B']
    base_freq = 440 #Частота A4
    keys = np.array([x+str(y) for y in range(0,9) for x in octave])
    start = np.where(keys == 'A0')[0][0]
    end = np.where(keys == 'C8')[0][0]
    keys = keys[start:end+1]
    note_freqs = dict(zip(keys, [2**((n+1-49)/12)*base_freq for n in range(len(keys))]))
    note_freqs[''] = 0.0 # stop
    return note_freqs
```

[2] См.: Генерация музыки из изображений с помощью Python (перевод Д. Брайта) [Электронный ресурс] // Хабр : [сайт]. URL: https://habr.com/ru/companies/ruvds/articles/708890/ (дата обращения: 04.06.2023).

Функция, представленная выше, является основной в данном разделе программы, поскольку при помощи функции get_piano_notes можно сопоставлять ноты, которым соответствует набор манипулятивных инструментов (клавиш) на фортепиано [1; 2; 3]. Частота воспроизведения при этом используется в герцах.

note_freqs = get_piano_notes()# загрузка словаря с нотами

Определение интервалов между тонами в гаммах, для индексации нот:

scale_intervals = ['A','a','B','C','c','D','d','E','F','f','G','g']

Благодаря этому стало возможным находить индекс гаммы в массиве тонов. Данное действие является необходимым, поскольку в дальнейшем потребуется реиндексировать массив, чтобы он начинался с необходимого элемента.

index = scale_intervals.index(whichKey) # определение индекса необходимой клавиши.

new_scale = scale_intervals[index:12] + scale_intervals[:index] # реиндексация элемента для начала с необходимой клавиши.

Таким образом, становится возможным определение гамм и массивов, в котором, в свою очередь, каждому элементу сопоставляется переопределённый массив элементов с индексом:

```
if whichScale == 'AEOLIAN':
    scale = [0, 2, 3, 5, 7, 8, 10]
elif whichScale == 'BLUES':
    scale = [0, 2, 3, 4, 5, 7, 9, 10, 11]
elif whichScale == 'PHYRIGIAN':
    scale = [0, 1, 3, 5, 7, 8, 10]
elif whichScale == 'CHROMATIC':
    scale = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
elif whichScale == 'DORIAN':
    scale = [0, 2, 3, 5, 7, 9, 10]
elif whichScale == 'HARMONIC_MINOR':
    scale = [0, 2, 3, 5, 7, 8, 11]
elif whichScale == 'LYDIAN':
    scale = [0, 2, 4, 6, 7, 9, 11]
elif whichScale == 'MAJOR':
    scale = [0, 2, 4, 5, 7, 9, 11]
elif whichScale == 'MELODIC_MINOR':
    scale = [0, 2, 3, 5, 7, 8, 9, 10, 11]
elif whichScale == 'MINOR':
    scale = [0, 2, 3, 5, 7, 8, 10]
elif whichScale == 'MIXOLYDIAN':
    scale = [0, 2, 4, 5, 7, 9, 10]
elif whichScale == 'NATURAL_MINOR':
    scale = [0, 2, 3, 5, 7, 8, 10]
elif whichScale == 'PENTATONIC':
```

```
        scale = [0, 2, 4, 7, 9]
    else:
        print('Недопустимое имя )
```

Также следующим шагом можно определить интервалы для их использования при создании композиций.

```
harmony_select = {'U0' : 1,
                  'ST' : 16/15,
                  'M2' : 9/8,
                  'm3' : 6/5,
                  'M3' : 5/4,
                  'P4' : 4/3,
                  'DT' : 45/32,
                  'P5' : 3/2,
                  'm6': 8/5,
                  'M6': 5/3,
                  'm7': 9/5,
                  'M7': 15/8,
                  'O8': 2
                 }
nNotes = len(scale)#длина гаммы в нотах
freqs = []#инициализация массива
for i in range(nNotes):
    note = new_scale[scale[i]] + str(whichOctave)
    freqToAdd = note_freqs[note]
    freqs.append(freqToAdd)
```

2. Модернизация программного компонента с использованием OpenSourse-библиотек языка программирования Python

С учётом всех проведённых операций и привлечением на этом этапе OpenSourse-библиотек[3] приходим к конечной форме программы, позволяющей на сетке пикселей преобразование «цвет-звук» [2; 3]:

```
#Import necessary libraries
import streamlit as st
import pandas as pd
import numpy as np
import cv2
import random
from pedalboard import Pedalboard, Chorus, Reverb, Gain, LadderFilter,Phaser,
Delay, PitchShift, Distortion
from pedalboard.io import AudioFile
from PIL import Image
```

[3] См.: Генерация музыки из изображений с помощью Python (перевод Д. Брайта) [Электронный ресурс] // Хабр : [сайт]. URL: https://habr.com/ru/companies/ruvds/articles/708890/ (дата обращения: 04.06.2023).

```
from scipy.io import wavfile
import librosa
import glob

#Функция генерации частоты в герцах посредством нот
def get_piano_notes():
    # Верхние и черные клавиши
    octave = ['C', 'c', 'D', 'd', 'E', 'F', 'f', 'G', 'g', 'A', 'a', 'B']
    base_freq = 440 #Частота A4
    keys = np.array([x+str(y) for y in range(0,9) for x in octave])
    # Trim to standard 88 keys
    start = np.where(keys == 'A0')[0][0]
    end = np.where(keys == 'C8')[0][0]
    keys = keys[start:end+1]

    note_freqs = dict(zip(keys, [2**((n+1-49)/12)*base_freq for n in range(len(keys))]))
    note_freqs[''] = 0.0 # Остановка
    return note_freqs
#Масштаб выбранный пользователем
def makeScale(whichOctave, whichKey, whichScale):
    #Загрузка словаря
    note_freqs = get_piano_notes()
    # Определение тональности. Верхний регистр – белые клавиши, нижний регистр – чёрные клавиши.
    scale_intervals = ['A','a','B','C','c','D','d','E','F','f','G','g']
    #Поиск индекса ключа
    index = scale_intervals.index(whichKey)
    #Предопределение интервалов шкалы, интервалы должны начинаться с whichKey
    new_scale = scale_intervals[index:12] + scale_intervals[:index]
    #Выбор масштаба
    if whichScale == 'AEOLIAN':
        scale = [0, 2, 3, 5, 7, 8, 10]
    elif whichScale == 'BLUES':
        scale = [0, 2, 3, 4, 5, 7, 9, 10, 11]
    elif whichScale == 'PHYRIGIAN':
        scale = [0, 1, 3, 5, 7, 8, 10]
    elif whichScale == 'CHROMATIC':
        scale = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
    elif whichScale == 'DORIAN':
        scale = [0, 2, 3, 5, 7, 9, 10]
    elif whichScale == 'HARMONIC_MINOR':
```

```
        scale = [0, 2, 3, 5, 7, 8, 11]
    elif whichScale == 'LYDIAN':
        scale = [0, 2, 4, 6, 7, 9, 11]
    elif whichScale == 'MAJOR':
        scale = [0, 2, 4, 5, 7, 9, 11]
    elif whichScale == 'MELODIC_MINOR':
        scale = [0, 2, 3, 5, 7, 8, 9, 10, 11]
    elif whichScale == 'MINOR':
        scale = [0, 2, 3, 5, 7, 8, 10]
    elif whichScale == 'MIXOLYDIAN':
        scale = [0, 2, 4, 5, 7, 9, 10]
    elif whichScale == 'NATURAL_MINOR':
        scale = [0, 2, 3, 5, 7, 8, 10]
    elif whichScale == 'PENTATONIC':
        scale = [0, 2, 4, 7, 9]
    else:
        print('Invalid scale name')
    #Инициализации массива
    freqs = []
    for i in range(len(scale)):
        note = new_scale[scale[i]] + str(whichOctave)
        freqToAdd = note_freqs[note]
        freqs.append(freqToAdd)
    return freqs
#Преобразование HUE оттенка в частоту
def hue2freq(h,scale_freqs):
    thresholds = [26 , 52 , 78 , 104, 128, 154, 180]
    #note = scale_freqs[0]
    if (h <= thresholds[0]):
        note = scale_freqs[0]
    elif (h > thresholds[0]) & (h <= thresholds[1]):
        note = scale_freqs[1]
    elif (h > thresholds[1]) & (h <= thresholds[2]):
        note = scale_freqs[2]
    elif (h > thresholds[2]) & (h <= thresholds[3]):
        note = scale_freqs[3]
    elif (h > thresholds[3]) & (h <= thresholds[4]):
        note = scale_freqs[4]
    elif (h > thresholds[4]) & (h <= thresholds[5]):
        note = scale_freqs[5]
    elif (h > thresholds[5]) & (h <= thresholds[6]):
        note = scale_freqs[6]
    else:
```

```
        note = scale_freqs[0]
    return note
#Создание музыки по изображению
def img2music(img, scale = [220.00, 246.94 ,261.63, 293.66, 329.63, 349.23, 415.30],
          sr = 22050, T = 0.1, nPixels = 60, useOctaves = True, randomPixels = False,
          harmonize = 'U0'):
```

Аргументы:

Аргумент img: (массив) изображение для обработки

Аргумент scale: (массив) массив, содержащий частоты, на которые нужно сопоставить значения H

Аргумент sr: (int) частота дискретизации, используемая для результирующей композиции

Аргумент T : (int) время в секундах для пересчёта каждой ноты в песне

Аргумент nPixels: (int) количество пикселей, используемых для создания композиции

Возвращает:

Аргумент song : (array) Numpy массив частот. Может быть воспроизведён с помощью ipd.Audio(song, rate = sr)

```
    #Пробование графического изображения в  HSV
    hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
    #Получение масштаба изображения
    height, width, depth = img.shape
    i=0 ; j=0 ; k=0
    # Инициализация массива, который включает в себя оттенки для каждого пикселя изображения
    hues = []
    if randomPixels == False:
        for val in range(nPixels):
            hue = abs(hsv[i][j][0]) #This is the hue value at pixel coordinate (i,j)
            hues.append(hue)
            i+=1
            j+=1
    else:
        for val in range(nPixels):
            i = random.randint(0, height-1)
            j = random.randint(0, width-1)
            hue = abs(hsv[i][j][0]) #Координаты оттенка пикселя (i,j)
            hues.append(hue)

    #Создание DataFrame  оттенками и частотами
    pixels_df = pd.DataFrame(hues, columns=['hues'])
```

```
    pixels_df['frequencies'] = pixels_df.apply(lambda row :
hue2freq(row['hues'],scale), axis = 1)
    frequencies = pixels_df['frequencies'].to_numpy()
    #Преобразование частоты в ноту
    pixels_df['notes'] = pixels_df.apply(lambda row :
librosa.hz_to_note(row['frequencies']), axis = 1)
    pixels_df['midi_number'] = pixels_df.apply(lambda row :
librosa.note_to_midi(row['notes']), axis = 1)
    #Создание словаря гармонии
    #Унисон = U0 ; semitone = ST ; major second = M2
    #Минор 3 = m3 ; major third = M3 ; perfect fourth = P4
    #Диатонический тритон = DT ; perfect fifth = P5 ; minor sixth = m6
    #Мажор 6 = M6 ; minor seventh = m7 ; major seventh = M7
    #Октава = O8
    harmony_select = {'U0' : 1,
                'ST' : 16/15,
                'M2' : 9/8,
                'm3' : 6/5,
                'M3' : 5/4,
                'P4' : 4/3,
                'DT' : 45/32,
                'P5' : 3/2,
                'm6': 8/5,
                'M6': 5/3,
                'm7': 9/5,
                'M7': 15/8,
                'O8': 2
               }
    harmony = np.array([]) #Массив с содержанием гармонии песни
    harmony_val = harmony_select[harmonize] #Соотношение гармонии

    #song_freqs = np.array([]) #Массив с частотами
    song = np.array([])     #Сигналы
    octaves = np.array([0.5,1,2])#Переход на октаву выше или ниже
    t = np.linspace(0, T, int(T*sr), endpoint=False) # Переменная – время
    #Создание песни с массивом numpy :]
    #nPixels = int(len(frequencies))#Пиксели в изображении
    for k in range(nPixels):
      if useOctaves:
        octave = random.choice(octaves)
      else:
        octave = 1
```

```
            if randomPixels == False:
                val =  octave * frequencies[k]
            else:
                val = octave * random.choice(frequencies)
            #Создание заметок
            note   = 0.5*np.sin(2*np.pi*val*t)
            h_note = 0.5*np.sin(2*np.pi*harmony_val*val*t)
            #Размещение заметок в массивы
            song     = np.concatenate([song, note])
            harmony  = np.concatenate([harmony, h_note])
            #song_freqs = np.concatenate([song_freqs, val])
        return song, pixels_df, harmony
    # Создание заголовка визуализации программы на сайте
    st.title("Цвета Ноты")
    st.markdown("Это web-приложение превратит изображение в музыкальную композицию")
    #Создание выплывающего окна
    df1 = pd.DataFrame({'Scale_Choice': ['AEOLIAN', 'BLUES', 'PHYRIGIAN', 'CHROMATIC', 'DORIAN', 'HARMONIC_MINOR', 'LYDIAN', 'MAJOR', 'MELODIC_MINOR', 'MINOR', 'MIXOLYDIAN', 'NATURAL_MINOR', 'PENTATONIC']})
    df2 = pd.DataFrame({'Keys': ['A','a','B','C','c','D','d','E','F','f','G','g']})
    df3 = pd.DataFrame({'Octaves': [1,2,3]})
    df4 = pd.DataFrame ({'Harmonies': ['U0','ST','M2','m3','M3','P4','DT','P5','m6','M6','m7','M7','O8']})

    st.sidebar.markdown("Выберите образец изображения, если вы хотите использовать одно из предварительно загруженных изображений. Выберите Изображение пользователя, если вы хотите использовать своё собственное изображение.")
    _radio = st.sidebar.radio("",("Использовать готовое изображение", "Загрузить собственное изображение"))
    sample_images = glob.glob('*.jpg')
    samp_imgs_df = pd.DataFrame(sample_images,columns=['Images'])
    samp_img = st.sidebar.selectbox('Выберите образец изображения', samp_imgs_df['Images'])
    #Загрузка изображения
    user_data = st.sidebar.file_uploader(label="Загрузите собственное изображение")
    if _radio == "Использовать готовое изображение":
        img2load = samp_img
    elif _radio == "Загрузить собственное изображение":
        img2load = user_data
```

```
#Отображение картинки
st.sidebar.image(img2load)
col1, col2, col3, col4 = st.columns(4)
with col1:
    scale = st.selectbox('Какой жанр вы хотели бы использовать?', df1['Scale_Choice'])
    'Вы выбрали ' + scale + ' жанр'
with col2:
    key = st.selectbox('Какой ключ вы хотели бы использовать?', df2['Keys'])
    'Вы выбрали:', key
with col3:
    octave = st.selectbox('Какую октаву вы бы хотели использовать', df3['Octaves'])
    'Вы выбрали:', octave
with col4:
    harmony = st.selectbox('Какую гармонию вы хотели бы использовать?', df4['Harmonies'])
    'Вы выбрали:', harmony
col5, col6 = st.columns(2)
with col5:
    #Выбор пользователем случайных пикселей
    random_pixels = st.checkbox('Использовать случайные пиксели для создания песни?', value=True)
with col6:
    #Продолжительность песни
    use_octaves = st.checkbox('Использовать случайные октавы нот при создании песни?', value=True)
col7, col8 = st.columns(2)
with col7:
    t_value = st.slider('Длительность ноты', min_value=0.01, max_value=1.0, step = 0.01, value=0.2)
with col8:
    n_pixels = st.slider('Сколько пикселей вы используете? (Больше пикселей занимает больше времени)', min_value=12, max_value=320, step=1, value=60)
st.markdown("## Педалборд")
col9, col10,col11,col12 = st.columns(4)
#Параметры хоруса
with col9:
    st.markdown("### Параметры хоруса")
    rate_hz_chorus = st.slider('Рейдовый фрейм', min_value=0.0, max_value=100.0, step=0.1, value=0.0)
#Параметры искажения
with col10:
```

```
        st.markdown("### Параметры искажения")
        delay_seconds = st.slider('Задержка в секундах', min_value=0.0, max_value=2.0,
step=0.1, value=0.0)
    with col11:
        st.markdown("### Параметры искажения")
        drive_db = st.slider('Праметры искажения(дб)', min_value=0.0,
max_value=100.0, step=1.0, value=0.0)

    # Параметры усиления
    with col12:
        st.markdown("### Параметры усиления")
        gain_db = st.slider('Коэффицент усиления(дб)', min_value=0.0,
max_value=100.0, step=1.0, value=0.0)
    st.markdown("### Параметры реверберации")
    rev1, rev2, rev3, rev4, rev5= st.columns(5)
    # Параметры реверберации
    with rev1:
        room_size = st.slider('Комната', min_value=0.0, max_value=1.0, step=0.1,
value=0.0)
    room_size1 = st.markdown("Комната – Реверберация небольшого помещения.
Подходит для применения к акустическим инструментам в камерной
атмосфере.")
    with rev2:
        damping = st.slider('демпинг', min_value=0.0, max_value=1.0, step=0.1,
value=0.0)
    with rev3:
        wet_level = st.slider('wet_level', min_value=0.0, max_value=1.0, step=0.1,
value=0.0)
    with rev4:
        dry_level = st.slider('dry_level', min_value=0.1, max_value=1.0, step=0.1,
value=0.1)
    with rev5:
        width = st.slider('длительность', min_value=0.0, max_value=1.0, step=0.1,
value=0.0)
    st.markdown("### Параметры фильтра")
    lf1,lf2,lf3 = st.columns(3)
    #Параметры фильтра
    with lf1:
        cutoff_hz = st.slider('частота', min_value=0.0, max_value=1000.0, step=1.0,
value=0.0)
    with lf2:
        resonance_lad = st.slider('резонанс', min_value=0.0, max_value=1.0, step=0.1,
value=0.0)
```

```
   with lf3:
      drive_lad        = st.slider('драйв', min_value=1.0, max_value=100.0, step=0.1,
value=1.0)
   #st.markdown("### Параметры фазы ")
   ch1,ps1 = st.columns(2)
   #Параметры фазы
   with ch1:
      st.markdown("### Параметры фазы")
      rate_hz_phaser = st.slider('рейтинг фазы', min_value=0.0, max_value=100.0,
step=0.1, value=0.0)
      depth_phaser   = st.slider('глубина', min_value=0.0, max_value=1.0, step=0.1,
value=0.0)

   with ps1:
      st.markdown("### Параметры сдвига высоты тона")
      semitones    = st.slider('semitones', min_value=0.0, max_value=12.0, step=1.0,
value=0.0)
   if img2load is not None:
      # Сохранение
      img = Image.open(img2load)
      img = img.save("img.jpg")
      # Чтение файла OpenCv
      img = cv2.imread("img.jpg")
      img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
      #Картинка на экране
      #st.image(img)
      #Создание шкалы звука
      scale_to_use = makeScale(octave, key, scale)
      #Создание песни
      song, song_df,harmony = img2music(img, scale = scale_to_use, T = t_value,
randomPixels = random_pixels, useOctaves = use_octaves, nPixels =
n_pixels,harmonize = harmony)
      #Записать звука в файл
      song_combined = np.vstack((song, harmony))
      wavfile.write('song.wav', rate = 22050, data =
song_combined.T.astype(np.float32))
      audio_file = open('song.wav', 'rb')
      audio_bytes = audio_file.read()
      # Чтение аудиофайла
      with AudioFile('song.wav', 'r') as f:
         audio = f.read(f.frames)
         samplerate = f.samplerate
      # Создание объекта Pedalboard с плагинами:
```

```
    board = Pedalboard([
      Gain(gain_db=gain_db),
      Distortion(drive_db=drive_db),
      LadderFilter(mode=LadderFilter.Mode.HPF12,
cutoff_hz=cutoff_hz,resonance = resonance_lad,drive=drive_lad),
      Delay(delay_seconds = delay_seconds),
      Reverb(room_size = room_size, wet_level = wet_level, dry_level = dry_level,
width = width),
      Phaser(rate_hz = rate_hz_phaser, depth = depth_phaser),
      PitchShift(semitones = semitones),
      Chorus(rate_hz = rate_hz_chorus)
      ])
    # Пропуск звука через петалборд!
    effected = board(audio, samplerate)
    # Записать аудиофайл в формате  -wav:
    with AudioFile('processed_song.wav', 'w', samplerate, effected.shape[0]) as f:
      f.write(effected)
    #Прочитать обработанную песню
    audio_file2 = open('processed_song.wav', 'rb')
    audio_bytes2 = audio_file2.read()
    #Воспроизвести песню
    st.audio(audio_bytes2, format='audio/wav')
    #@st.cache
    def convert_df_to_csv(df):
      # Необходимо кэшировать преобразование, чтобы избежать вычислений
при каждом запуске.
      return df.to_csv().encode('utf-8')
    #csv = song_df.to_csv('song.csv')
    st.download_button('Загузить музыку', data=convert_df_to_csv(song_df),
file_name="song.csv",mime='text/csv',key='download-csv')
   # Изображение не разгружено
  else:
    st.write("Ожидание загрузки изображения...")
  #st.markdown("# Main page ")
  #st.sidebar.markdown("# Main page ")
```

Заключение

В работе найден и построен путь преобразования цветового изображения в звуковое отображение. Это позволяет услышать, как звучит, например, портрет или какое-либо другое изображение. Для реализации такой возможности было выполнено преобразование цветового пространства объекта в пиксельную структуру, в которой каждому пикселю соответствовал свой цвет, яркость и насыщенность. Также было выполнено преобразование звукового поля в

соответствии со звучанием ноты, её тональности, её частотой. Построен словарь соответствия нот частотам и номерам клавиш на фортепиано. Далее для создания и обработки изображений выбрали цветовую модель HSV, был использован ряд библиотек, позволяющих визуализировать объект. На основе построенного цветового пространства построен код для считывания значений тона пикселя. Построен алгоритм преобразования цветового кода, определённого на пиксельной сетке изображения в звуковое отображение этого изображения.

Статья поступила в редакцию 01.11.2023 г.

ЛИТЕРАТУРА

1. Color Calibration on Human Skin Images / Amani M., Falk H., Jensen O. D., Vartdal G., Aune A., Lindseth F. // International Conference on Computer Vision Systems. ICVS 2019 / eds. Tzovaras D., Giakoumis D., Vincze M., Argyros A. Cham: Springer, 2019. P. 211–223 (Series: Lecture Notes in Computer Science. Vol. 11754). DOI: 10.1007/978-3-030-34995-0_20.
2. Ключников С. А., Калашников Е. В. Преобразование пиксельной структуры в звуковые отображения. Часть 1 // Вестник Государственного университета просвещения. Серия: Физика-Математика. 2023. № 4. С. 64–80.
3. Чижов С. А. Управление цветом в широкоформатной печати: RGB VS CMYK // Science Time. 2018. № 9 (57). С. 43–47.

REFERENCES

1. Amani M., Falk H., Jensen O. D., Vartdal G., Aune A., Lindseth F. Color Calibration on Human Skin Images. In: Tzovaras D., Giakoumis D., Vincze M., Argyros A., eds. *International Conference on Computer Vision Systems. ICVS 2019.* Cham, Springer, 2019, pp. 211–223 (Series: Lecture Notes in Computer Science. Vol. 11754). DOI: 10.1007/978-3-030-34995-0_20.
2. Klyuchnikov S. A., Kalashnikov E. V. [Converting a pixel structure into sound imaginations. Part 1]. In: *Vestnik Gosudarstvennogo universiteta prosveshcheniya. Seriya: Fizika-Matematika* [Bulletin of Federal State University of Education. Series: Physics and Mathematics], 2023. no. 4, pp. 64–80.
3. Chizhov S. A. [Color management in large-format printing: RGB VS CMYK]. In: *Science Time*, 2018, no. 9 (57), pp. 43–47.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Ключников Семён Александрович – студент магистратуры физико-математического факультета Государственного университета просвещения;
e-mail: semen.klyuchnikov@mail.ru;

Калашников Евгений Владимирович – доктор физико-математических наук, профессор кафедры вычислительной математики и информационных технологий Государственного университета просвещения;
e-mail: ekevkalashnikov1@gmail.com.

INFORMATION ABOUT THE AUTHORS

Semen A. Klyuchnikov – Master's Degree Student, Faculty of Physics and Mathematics, Federal State University of Education;
e-mail: semen.klyuchnikov@mail.ru;

Evgenii V. Kalashnikov – Dr. Sci. (Phys.-Math.), Prof., Department of Computational Mathematics and Information Technology, Federal State University of Education;
e-mail: ekevkalashnikov1@gmail.com.

ПРАВИЛЬНАЯ ССЫЛКА НА СТАТЬЮ

Ключников С. А., Калашников Е. В. Преобразование пиксельной структуры в звуковые отображения. Часть 2 // Вестник Государственного университета просвещения. Серия: Физика-Математика. 2024. № 1. С. 33–47.
DOI: 10.18384/2949-5067-2024-1-33-47

FOR CITATION

Klyuchnikov S. A., Kalashnikov E. V. Converting a pixel structure into sound imaginations. Part 2. In: *Bulletin of Federal State University of Education. Series: Physics and Mathematics*, 2024, no. 1, pp. 33–47.
DOI: 10.18384/2949-5067-2024-1-33-47