Pico Lütjens 07.02.22

Audio Generation from Image

General Description:

This Project makes use of the Earsketch Website ("https://earsketch.gatech.edu/landing/") and its Audio Library as well as Processing (Java Mode). The Website uses Python or Javascript for the Coding of music.
The Processing sketch in the Project Files is a Generator that generates a finished python skript for you to paste on the website.
It is generated based on the image you give to the sketch. There is no randomness involved, so each image generates a unique music that is always the same for that specific image.

Usage:

The Usage is very simple. You just need to download and safe an image you want to be translated into music and that matches the specifications mentioned in the sketch. The Image needs to be safed in the data folder or the sketch folder itself.
Then in the sketch you simply need to change the Datapath to the image in line 9, and then the safepath in line 12.

Example:

```
source = loadImage("path");
generator.savetofile("path");
```

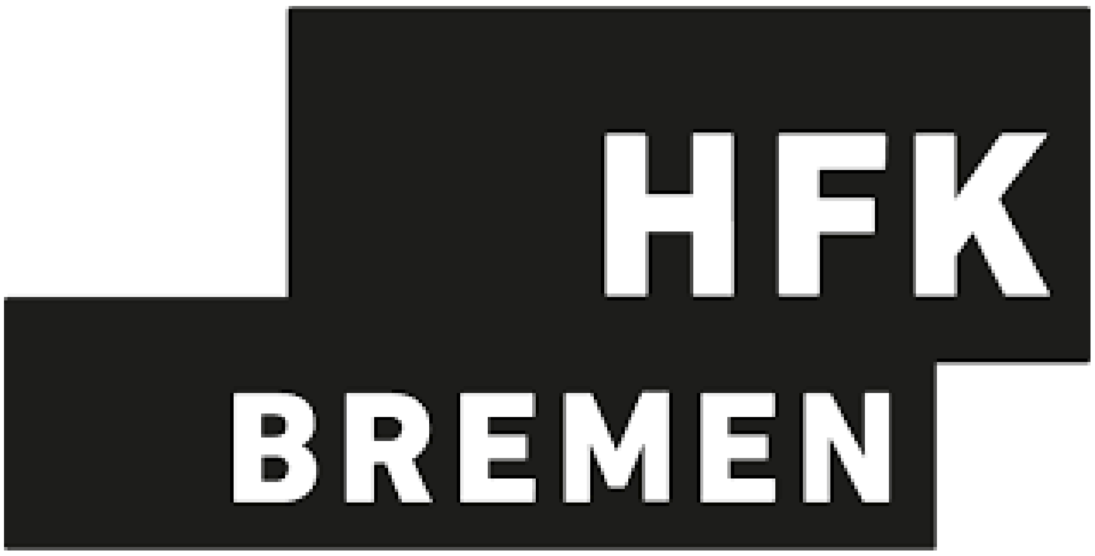
When you run the sketch a little window opens which is automatically done because of the way Processing is programmed. We do not need the window. You can just close it and then go to your specified output directory and copy the generated Code in the File. You can then paste it on the Website after you clicked on "Start Coding" and click on "RUN" in the Top/Right to interpret the Code. Once it is interpreted you can press play on the Top of the Website and listen to your generated music.

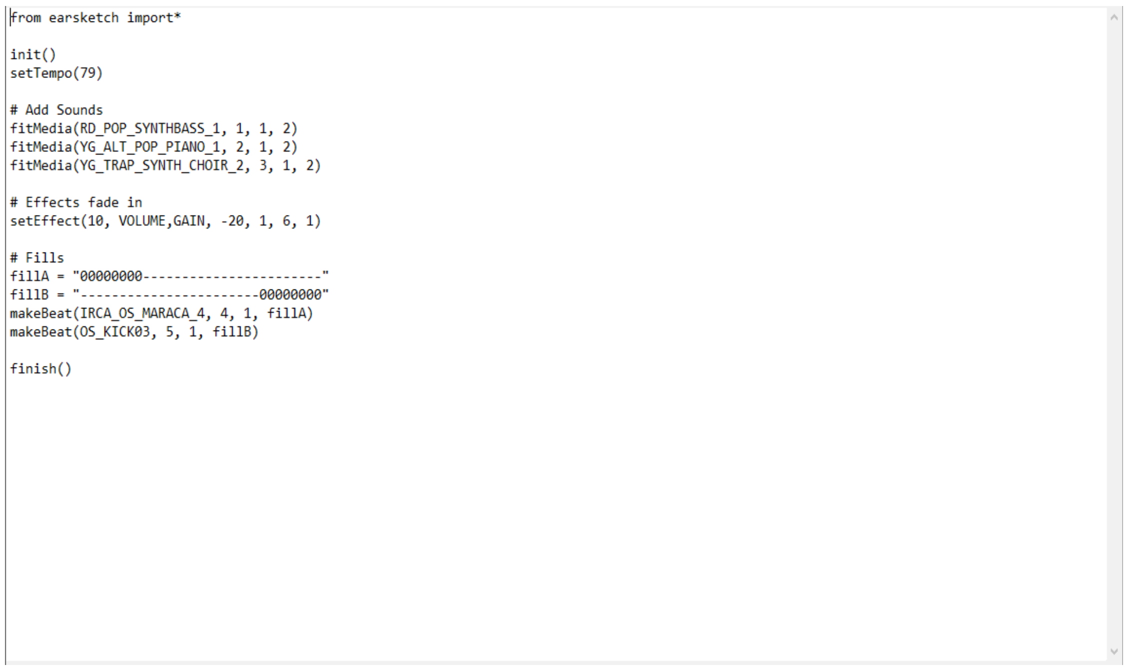
Generation Parameters:

The Code that is being generated is dependent on a few things. These contain for example the resolution of the image from which the Tempo is being generated, as well as the brightness/darkness of pixels, the average values and the brightest row and the brightest column. These generate parameters like the choosen instruments for the bass voice, the middle voice and the high voice. The Bass voice can contain Bass, Brass and Percussion. The middle voice can contain Guitar, Piano and Strings. And the high voice contains mostly single leading voice or choir. The Drumbeat is generated by the brightest row in the image and basically reads through from pixel to pixel.

Example HFK Logo:

This image of the HFK Logo generates the following code:





```
from earsketch import*

init()
setTempo(79)

# Add Sounds
fitMedia(RD_POP_SYNTHBASS_1, 1, 1, 2)
fitMedia(YG_ALT_POP_PIANO_1, 2, 1, 2)
fitMedia(YG_TRAP_SYNTH_CHOIR_2, 3, 1, 2)

# Effects fade in
setEffect(10, VOLUME,GAIN, -20, 1, 6, 1)

# Fills
fillA = "00000000------------------------"
fillB = "------------------------00000000"
makeBeat(IRCA_OS_MARACA_4, 4, 1, fillA)
makeBeat(OS_KICK03, 5, 1, fillB)

finish()
```

This Python Code can now be pasted and then run in a new Earsketch Project on the Website.

Source Code:
(The source Code is also available on github: https://github.com/PicoLuetjens/Image-to-Audio-Code-Generation)

GenerateCode.pde

```
//min image width = 50 max image width = 2000 to work
CodeGenerator generator;
PImage source;

void setup()
{
  source = loadImage("HFK.png");
  generator = new CodeGenerator(source);
  generator.generate();
  generator.savetofile("data/GeneratedCode.txt");
  println("Done generating");
}
```

CodeGenerator.pde

```
class CodeGenerator
{
  //divides the image pixel width and height to get a "normal" amount of takts
  //basically resamples the image
  int short_divident = 10;

  //source image
  PImage source;

  //resampled image
  PImage img;

  //generated code on the fly
  ArrayList<String>gen_code = new ArrayList();

  //list of bass samples
  String[]basssamples;

  //list of middle voices(guitar, strings etc.)
  String[]middlevoicesamples;

  //list of leading voices(vocals)
  String[]leadingvoicessamples;

  //list of beats(percussion and drums)
  String[]beats;

  //calculated length in takts
  int taktslength = 0;

  //calculated length in 16tel notes
  int noteslength = 0;

  //calculated tempo
  int tempo = 0;

  //threshold for generating beat
  int threshold = 200;

  CodeGenerator(PImage img)
  {
    this.source = img;
    this.img = this.source.copy();
    this.img.resize(this.img.width/short_divident, this.img.height/short_divident);
    this.taktslength = floor(this.img.width/16);
    this.noteslength = this.taktslength*16;
    //println(taktslength);
    //println(noteslength);
    this.loadAllSnipets();
  }

  private void loadAllSnipets()
  {
    this.beats = loadStrings("Makebeat.txt");
    this.basssamples = loadStrings("Makebass.txt");
    this.middlevoicesamples = loadStrings("Makemiddlevoice.txt");
    this.leadingvoicessamples = loadStrings("Makeleadingvoice.txt");
  }

  public void generate()
  {
    this.gen_code.add("from earsketch import*");
    this.gen_code.add("");
    this.gen_code.add("init()");
    int tempo = calculateTempo();
    this.gen_code.add("setTempo(" + tempo + ")");
    this.gen_code.add("");
    this.gen_code.add("# Add Sounds");
    this.taktslength+=1;
    String bass = this.basssamples[int(map(getDarkestPixelvalue(), 0, 256, 0,
this.basssamples.length))];
    this.gen_code.add("fitMedia(" + bass + ", 1, 1, " + this.taktslength + ")");
    String middle = this.middlevoicesamples[int(map(getAveragePixelvalue(), 0, 256, 0,
this.middlevoicesamples.length))];
    this.gen_code.add("fitMedia(" + middle + ", 2, 1, " + this.taktslength + ")");
    String high = this.leadingvoicessamples[int(map(getBrightestPixelvalue(), 0, 256, 0,
this.leadingvoicessamples.length))];
    gen_code.add("fitMedia(" + high + ", 3, 1, " + this.taktslength + ")");
    this.gen_code.add("");
    this.gen_code.add("# Effects fade in");
    this.taktslength-=1;
    this.gen_code.add("setEffect(10, VOLUME,GAIN, -20, 1, 6, " + this.taktslength + ")");
    this.gen_code.add("");
    this.gen_code.add("# Fills");
    String fill = this.generateFill(this.getBrightestRow());
    String rev_fill = new StringBuilder(fill).reverse().toString();
    this.gen_code.add("fillA = \"" + fill + "\"");
    this.gen_code.add("fillB = \"" + rev_fill + "\"");
    String beat1 = this.generateBeat1();
    String beat2 = this.generateBeat2();
    this.gen_code.add("makeBeat(" + beat1 + ", 4, 1, fillA)");
    this.gen_code.add("makeBeat(" + beat2 + ", 5, 1, fillB)");
    this.gen_code.add("");
    this.gen_code.add("finish()");
  }

  private int getAveragePixelvalue()
  {
    int divident = this.img.width*this.img.height;
    int sum = 0;
    for (int i = 0; i < this.img.width; i++)
    {
      for (int j = 0; j < this.img.height; j++)
      {
        sum+=brightness(this.img.get(i, j));
      }
    }
    int result = sum/divident;
    return result;
  }

  private int getDarkestPixelvalue()
  {
    int low = 300;
    for (int i = 0; i < this.img.width; i++)
    {
      for (int j = 0; j < this.img.height; j++)
      {
        if (brightness(this.img.get(i, j)) < low)
        {
          low = int(brightness(this.img.get(i, j)));
        }
      }
    }
    //print(low);
    return low;
  }

  private int getBrightestPixelvalue()
  {
    int max = 0;
    for (int i = 0; i < this.img.width; i++)
    {
      for (int j = 0; j < this.img.height; j++)
      {
        if (brightness(this.img.get(i, j)) > max)
        {
          max = int(brightness(this.img.get(i, j)));
        }
      }
    }
    //print(max);
    return max;
  }

  private String generateBeat1()
  {
    int number = this.getBrightestRow();
    number = int(map(number, 0, this.img.width, 0, this.beats.length));
    String out = this.beats[number];
    return out;
  }

  private String generateBeat2()
  {
    int number = this.getBrightestCol();
    number = int(map(number, 0, this.img.width, this.beats.length, 0));
    String out = this.beats[number];
    return out;
  }

  private String generateFill(int row)
  {
    String beat = "";
    for (int i = 0; i < this.img.width; i++)
    {
      if (brightness(this.img.get(row, i)) > this.threshold)
      {
        beat+="0";
      } else
      {
        beat+="-";
      }
    }
    return beat;
  }

  private int calculateTempo()
  {
    int img_value = this.img.width*this.img.height;
    img_value = img_value/short_divident;
    //print(img_value);
    int result = int(map(img_value, 50, 200000/this.short_divident, 80, 150));
    return result;
  }
```

```
 private int getBrightestRow()
 {
  int row = 0;
  int max = 0;
  for (int i = 0; i < this.img.height; i++)
  {
   int sum_brightness = 0;
   for (int j = 0; j < this.img.width; j++)
   {
    sum_brightness+=brightness(this.img.get(i, j));
   }
   if (sum_brightness > max)
   {
    max = sum_brightness;
    row = i;
   }
  }
  return row;
 }

 private int getBrightestCol()
 {
  int col = 0;
  int max = 0;
  for (int i = 0; i < this.img.width; i++)
  {
   int sum_brightness = 0;
   for (int j = 0; j < this.img.height; j++)
   {
    sum_brightness+=brightness(this.img.get(i, j));
   }
   if (sum_brightness > max)
   {
    max = sum_brightness;
    col = i;
   }
  }
  return col;
 }

 public void savetofile(String name)
 {
  String[] code = new String[this.gen_code.size()];
  for (int i = 0; i < this.gen_code.size(); i++)
  {
   code[i] = this.gen_code.get(i);
  }
  saveStrings(name, code);
 }
}
```