Activity 9: Interpreting a Musical Score Sheet

Mary Chris Go / 2014 11122

Goal

To be able to extract musical notes from a digitized score sheet and play notes in Scilab

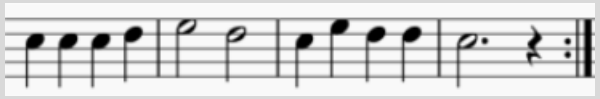
Mary Chris Go / 2014 11122

Chosen Song

Moon Night

I chose this French folk song as it has the easiest to play on the first part bar and given that most notes are whole notes. A beat of 4/4, which is the usual beat for most songs, will help to make this job easy. I cut the music up until the first 4 bars only.

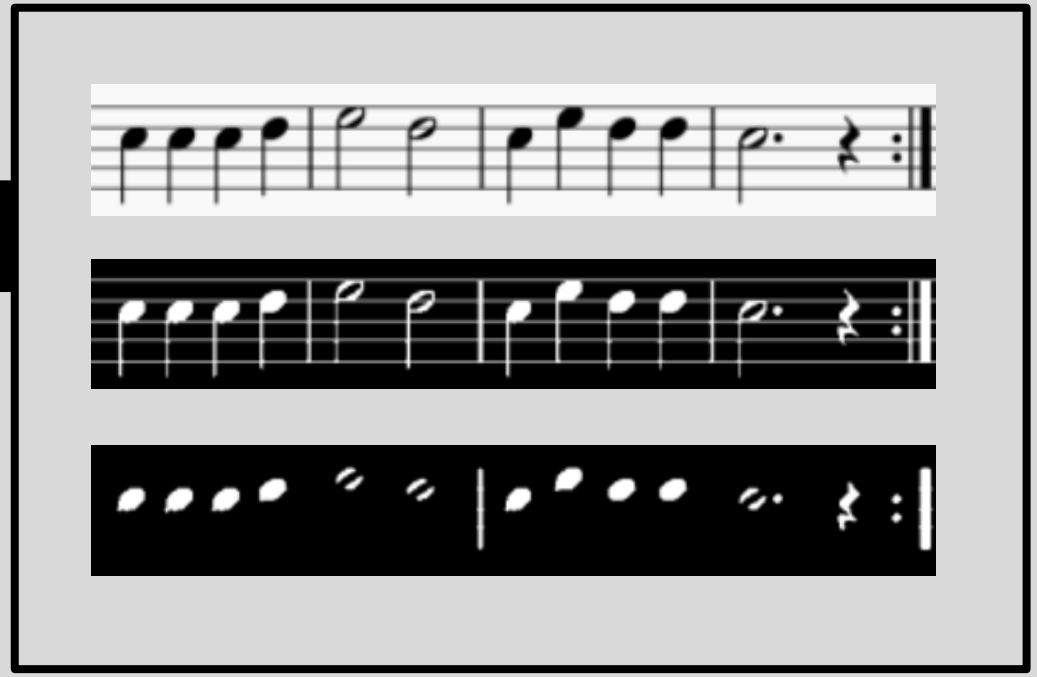
Snippet of music sheet



This is a good chosen song because a blog was written about it already. At least I get to have a basis if I'm doing it right.

Mary Chris Go / 2014 11122

Actual Treatment of images



This was already a cropped image of the whole sheet. As mentioned before, I only took the 4 bars. I removed the G clef so that the code would just recognize the needed parts.

First step is we open the image into grayscale and get the threshold value. Just like how we did from the past activities.

Second step is to remove the bar lines and maintain the notes. We have to be careful not to remove some essential parts such as dotted noted, rest notes, and the repeat sign. We use the past activity by creating a structure element of circle. We located all the blobs that were detected. Some will be very small and therefore, we must consider something greater than a certain threshold (pixels in area).

Given that we have searched the right blobs, we now get the size and its exact location for the right pitch.

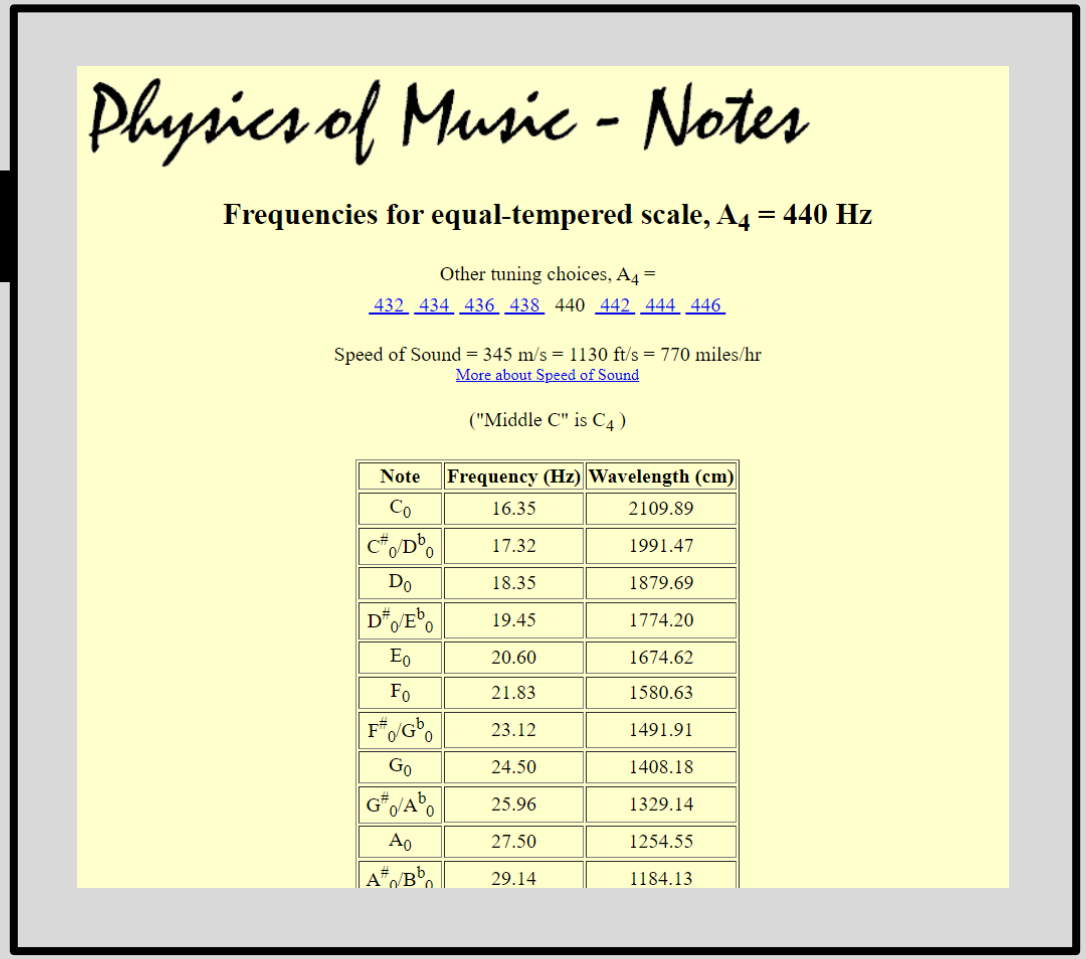
Mary Chris Go / 2014 11122

Scilab Implementation

```
Img=imread('sample.png');
Img=rgb2gray(Img);
[x,y]=size(Img);

circle=CreateStructureElement('circle',1);

B=SearchBlobs(OpenImage(Img<128,circle));
uniquemax=max(unique(B));

xlist = [];
ylist = [];
blist = [];
sizelist = [];
sizes=[];

output=zeros(x,y);

for j=1:uniquemax
sizes=[sizes sum(B==j)]
if sum(B==j)>20 then
[x,y] = find(B==j);
output=output+(B==j);
xlist=[xlist int(mean(x))]
ylist=[ylist int(mean(y))]
blist = [blist j]
sizelist=[sizelist sum(B==j)]
imwrite(B==j,string(size(sizelist,2))+'.'+string(sum(B==j))+'.'+string(int(mean(x)))+'.png');

imwrite(output,'final.png');
```

The snippet code was a summary of how I implemented the last slide. So far, this is doable since we have already done this from the past activities.

Mary Chris Go / 2014 11122



Physics of Music - Notes

Frequencies for equal-tempered scale, $A_4 = 440$ Hz

Other tuning choices, A_4 =

432 434 436 438 440 442 444 446

Speed of Sound = 345 m/s = 1130 ft/s = 770 miles/hr

More about Speed of Sound

("Middle C" is C_4)

Note	Frequency (Hz)	Wavelength (cm)
C_0	16.35	2109.89
$C^{\#}_0/D^{b}_0$	17.32	1991.47
D_0	18.35	1879.69
$D^{\#}_0/E^{b}_0$	19.45	1774.20
E_0	20.60	1674.62
F_0	21.83	1580.63
$F^{\#}_0/G^{b}_0$	23.12	1491.91
G_0	24.50	1408.18
$G^{\#}_0/A^{b}_0$	25.96	1329.14
A_0	27.50	1254.55
$A^{\#}_0/B^{b}_0$	29.14	1184.13

**Screenshot from https://pages.mtu.edu/~suits/notefreqs.html

```
32 flist =[]
33 dlist =[]
34
35 i = 1
36 while i<54
37
38 if sum(sizelist(i)==[180:210])
39 if sum(xlist(i)==[35:38])
40 flist = [flist 261.63*2]
41 dlist =[dlist 1]
42 end
```

A snippet of the code. This block was repeated for several notes (depending on the pixel area and certain conditions). This one is for C_4.

B_3	246.94	139.71
C_4	261.63	131.87
$C^{\#}_4/D^{b}_4$	277.18	124.47

Scilab Implementation

Let's start to segregate each note into its own characteristics. A quarter note has an average area of 190 pixels. For half note, an average area of approximately 45 pixels was found. Dots for the dotted notes are less than 30 pixels. Quarter rests are around 170 pixels. Now, we take these parameters as threshold. Anything beyond and below these values are not considered. Now, using the link, we determine the pitches for each note. The rest was set to a frequency below 20 Hz for us not to hear it.

Results (via Scilab)

By using the sample code from Ma'am Jing, we get to play the notes and save it. See attached file.

This code was my basis after all. So I saved the file first and tried if it worked. And viola, it worked.

Good thing I was able to apply it in a much complicated musical sheet.

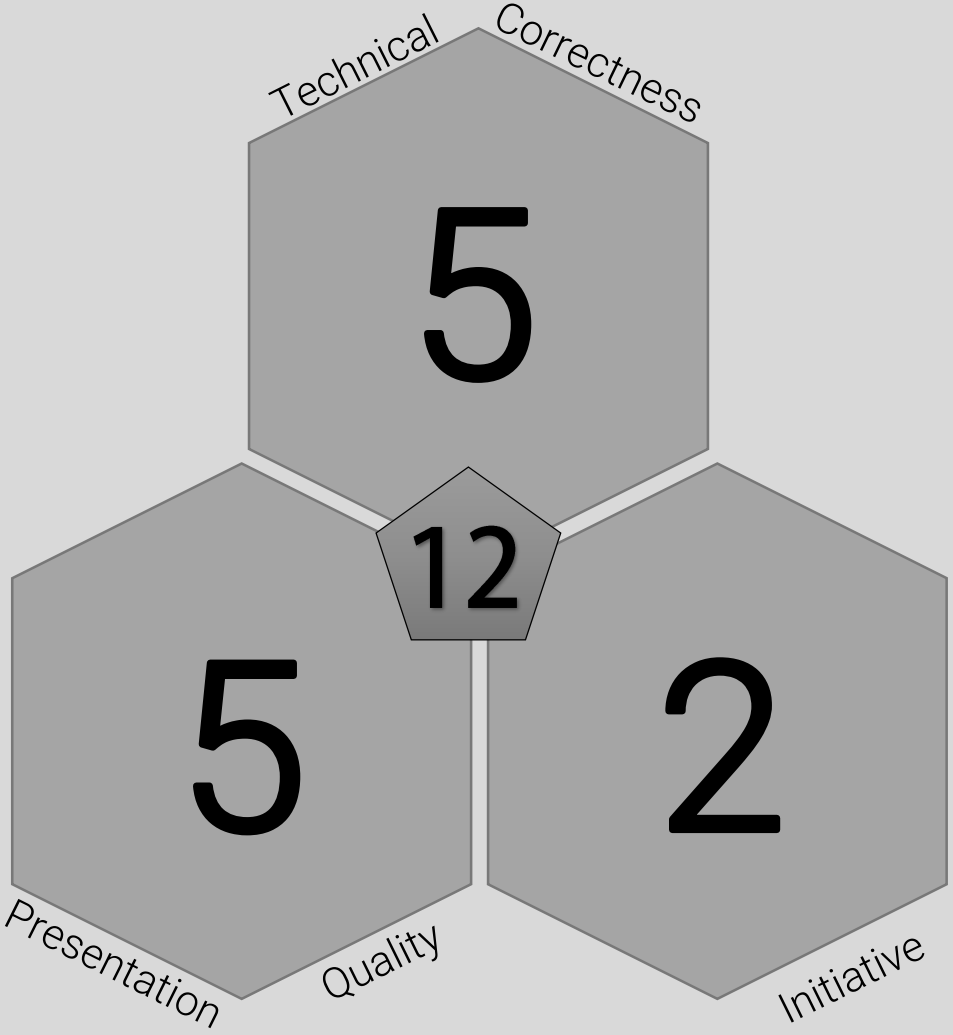
```
function n = note(f, t)
  n = sin (2*%pi*f*t);
endfunction;

C = 261.63*2;
D = 293.66*2;
E = 329.63*2
t=soundsec(0.25);

s  =  [note(E,t),  note(D,t),note(C,t),  note(D,t),
note(E,t) note(E,t) note(E,t)];
sound(s);
```

Mary Chris Go / 2014 11122

Summary

Fun, but frustrating. It took me 2 weeks to do the job. The coding part was hard though it was an integration of everything I've learned. New techniques must be learned to finish the whole activity. But the reward is satisfying. It's amazing to see all the possible capabilities of image processing. This is my first time encountering music with image processing. I have a bunch of complicated SATB musical sheets at home and if I have extra time, I'll try to do this. Another frustrating but fun activity to look forward haha!

Self-Evaluation



References

- Soriano, M., “Intepreting a musical score sheet,” 2019
- https://appphy186haroldco.blogspot.com/2016/12/activity-9-playing-musical-notes.html?fbclid=IwAR3GXDG2Y8Fwm6MlAHGbHcIRLv28qLipAfimvfaeQBdUnxd-0hw9DtlQCVQ
- https://raaromeroap186.wordpress.com/2016/12/04/a9-playing-notes-by-image-processing/
- https://www.image-line.com/support/flstudio_online_manual/html/glossary_envelope.htm